FEATURE-ALIGNED, SEMIREGULAR, QUAD-ONLY MESH GENERATION

by

Joel Daniels

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

School of Computing

The University of Utah

August 2010

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

SUPERVISORY COMMITTEE APPROVAL

of a dissertation submitted by

Joel Daniels

This dissertation has been read by each member of the following supervisory committee and by majority vote has been found to be satisfactory.

	Chair: Elaine Cohen
	Claudio Silva
	Michael Kirby
	Rich Riesenfeld
	Valerio Pascucci

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

FINAL READING APPROVAL

To the Graduate Council of the University of Utah:

I have read the dissertation of Joel Daniels in its final form and have found that (1) its format, citations, and bibliographic style are consistent and acceptable; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the Supervisory Committee and is ready for submission to The Graduate School.

Date

Elaine Cohen
Chair, Supervisory Committee

Approved for the Major Department

Martin Berzins
Chair/Dean

Approved for the Graduate Council

Charles A. Wigat
Dean of The Graduate School

ABSTRACT

This research focuses on the generation of representational structure from unstructured inputs for model reconstruction to facilitate subsequent geometry processing. In particular, the results include spline-based curve reconstruction of sharp features in point-sampled geometry, and quadrilateral-only mesh processing and generation from polygonal models. The challenges of model reconstruction, as well as the relationship between continuous and discrete geometry, are important thematic elements of this research.

Virtual surface models are ubiquitous in computer graphics, representing fundamental entities in visualization and modeling toolkits. Applications of surface geometry are as diverse as the representational formats constructed to achieve these ends, each designed with different characteristics and advantages. Translation between the assorted forms is often necessitated by the allowable inputs of system implementations and desired algorithms, and requires techniques to reconstruct the model with new geometry. Consequently, model reconstruction algorithms are prevalent in most modeling pipelines, used to augment the availability of applicable algorithms and to improve the robustness and efficiency of subsequent geometry processing. In this work, my simplification and reconstruction results are demonstrated on a variety of organic and mechanical models whose original representations range from point clouds to unstructured and structured triangle, quadrilateral, and hybrid meshes.

For my family, friends and teachers, who have been there throughout my journey.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Many hands must be acknowledged in aiding the completion of this body of work. First and foremost, Elaine Cohen and Claudio Silva played instrumental roles, guiding and advising my research. Thanks to my committee members for their patience, cooperation and signatures. Special thanks to my collaborators and coauthors of the related publications: Elaine Cohen, Linh Ha, Tilo Ochotta, Jason Shepherd, and Claudio Silva. I would also like to thank the many people who have provided constructive critiscisms, reviews and comments: Matt Berger, Peer-Timo Bremer, Tiago Etiene, Miriah Meyer, Valerio Pascucci, Justin Polchlopek, Kristi Potter, Carlos Scheidegger, John Schreiner and the anonymous reviewers of my paper submissions who helped sculpt this work. It is important to thank Karen Feinauer for her support and organization, without which I do not doubt that none of this would have been a possibility. Additionally, I would recognize the AIM@Shape repository, John Schriener and Jason Shepherd, for access to the models used throughout my dissertation. This work was funded by many sources along the way, but it is important to acknowledge the National Science Foundation, in particular CCF0541402, for their support. Finally, I would be remiss if I did not thank my family and friends for their support through this monstrous ordeal.

CHAPTER 1

INTRODUCTION

Surface remeshing is motivated by many computer-aided geometric design (CAGD) challenges, including point- and polygonal based modeling and visualization, reverse engineering, manufacturing and design, and other computer graphics applications. The approximation of an input virtual model, potentially composed of heterogeneous representational formats, as a single meshed structure is a fundamental component of the geometry processing pipeline. The individual element quality of a mesh often determines the robustness and efficiency of many subsequent geometric processing algorithms, necessitating a surface remeshing algorithm to augment the quality of the original model. Similarly, surface remeshing generates watertight representations from scattered point samples to allow reconstruction, processing and visualization of scanned data sets.

In the past, mesh based research has been dominated by triangle based surface representations. As computing power increases, however, quadrilateral meshes are becoming an attractive alternative. This popularity can be partially attributed to the quadrilateral's unique ability to naturally describe orthogonal vector fields, i.e., principal curvature directions, as well as sharing a common domain with surface parametrization solutions. These characteristics enable a number of important applications, including texturing, certain types of finite element analysis, and modeling with Catmull-Clark subdivision or splines. Furthermore, as support for tessellating tensor product parameteric surfaces in GPU hardware becomes available [104], many graphical applications, i.e., visualization systems currently relying on triangle based models, will prefer the use of quadrilateral meshes.

While there exist many geometric processing techniques for triangle meshes, there are few research results on automated algorithms targeting quadrilaterals, in part due to the unique challenges associated with these elements. A potentially nonplanar, nonconvex, bilinear element, the quadrilateral is fundamentally more complex than the triangle, where both planarity and convexity are intrinsic. Unlike triangle based methods, the

structural organization of the quadrilateral mesh imposes global constraints on the mesh connectivity. This can be illustrated by noting that it is impossible to quadrangulate a planar region bounded by a polyline with an odd number of vertices.

1.1 Quadrilateral Mesh Structure

Polygonal models are categorized as *(semi) regular* or *irregular*, according to structural properties of the mesh connectivity. A regular mesh imposes very strict neighborhood constraints, requiring that all internal vertices have valence 4 for quadrilateral meshes. Defined as an ideal vertex, this connectivity facilitates the generation of orthogonal angles and well shaped quadrilaterals. The regularity of the mesh structure facilitates many algorithms that are often subsequently applied to the model. In particular, surface parametrization including texturing [160] and spline based modeling [169], mesh subdivision [28], Fourier- [122] and wavelet-based [166] computations, mesh compression [86] and comparison [125] algorithms exploit the structure of the regular connectivity.

The onerous structure enforced by a regular mesh complicates the construction of such models. In addition to challenges related to element distortions, this constraint is often impossible to satisfy, for only genus 1 (toroidal) models can by described as a regular quadrilateral mesh. In contrast, an irregular model sacrifices the strict connectivity related constraints by allowing any number of extraordinary vertices (nonideal valence 4). This degree of freedom better accommodates the description of complex geometric features, model deformations and tracking topological changes.

A semiregular mesh balances the advantages of the regular and irregular models by allowing a limited number of extraordinary vertices. Tracing the mesh edges that emminate from the extraordinary vertices forms boundary curves between them that define a coarse segmentation of the model. Internally, each of the segmented regions is described by a regular mesh. Semiregular meshes can describe surfaces of arbitrary genus while exhibiting the structural regularity that facilitates many gemoetric processing algorithms.

This research focuses on faithful reconstruction of *manifold* surface models with *arbitrary topology* by automatically generating *feature aligned*, semiregular, quadrilateral-only meshes. A manifold of arbitrary topology is an orientable surface without self-intersections that can contain any number of *handle* elements. Analogous to a coffee mug, a handle is a component of the model that forms a hole through the geometry.

Feature alignment refers to the preservation of sharp edges on the original surface by *aligning* mesh edges to these structures.

As illustrated in Figure 1.1, the feature aligned, semiregular, quadrilateral-only mesh is a powerful representation. The extraordinary vertices define graph cuts that facilitate surface parameterization. As a byproduct of the algorithms proposed in this research, the surface parameterizations allow texturing, displacement mapping and spline based modeling.

1.2 Research Tasks

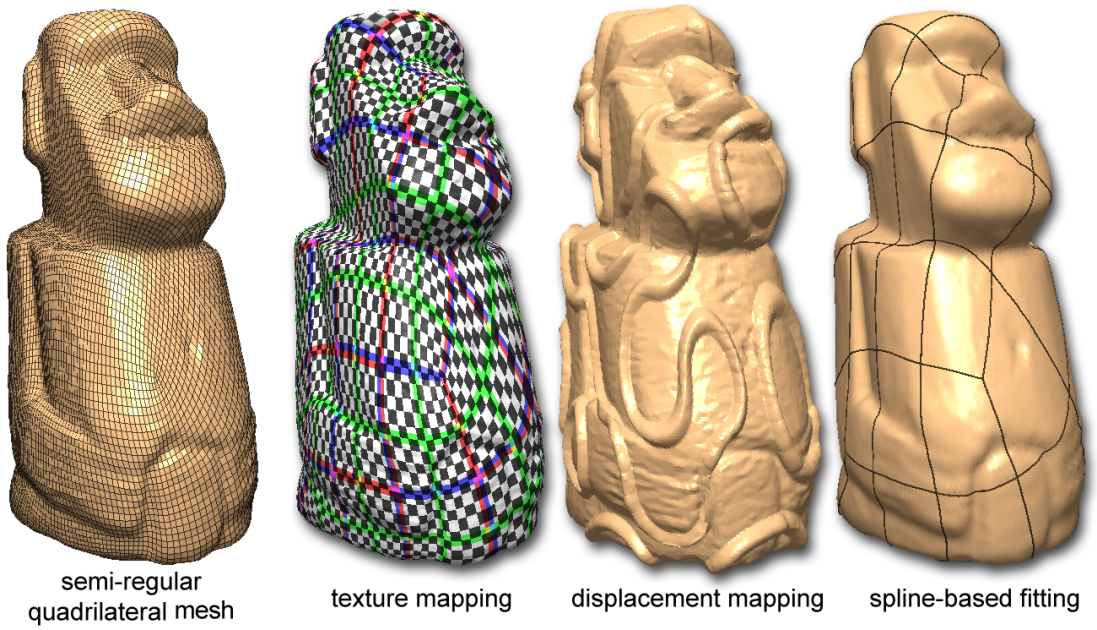
The motivation of this research, an automated surface remeshing algorithm illustrated in Figure 1.2, involves multiple subtasks to generate a *faithful* reconstruction of input geometry. A faithful surface reconstruction is one that respects the input geometry, recreating important feature information and surface geometry with low approximation error. Developing semiregular, quadrilateral-only meshes requires a understanding of the structural challenges associated with quadrilateral elements, as well as a understanding of related challenges associated with surface parameterization.

The reconstruction pipeline depicted in Figure 1.2 shows the development of a feature aligned, semiregular, quadrilateral-only mesh from a point-based model. The input data may come from a variety of sources, including digital scanners or samplings of polygonal meshes, implicit surfaces, and isosurfaces of volumetric data sets. This pipeline is able to process a model that is composed of heterogeneous inputs, frequently addressed by reverse engineering applications, by sampling each component to define a coherent point-based representation.

This dissertation describes techniques that bridge the gap between discrete and continuous geometry by introducing structure to unstructured data. The work stresses robust, connectivity based methods for quadrilateral-only mesh processing. The following subsections outline a series of subtasks affiliated with the overarching research goal, providing insight for the specific challenges addressed by each chapter, as well as presenting the layout of the dissertation.

1.2.1 Important Geometry

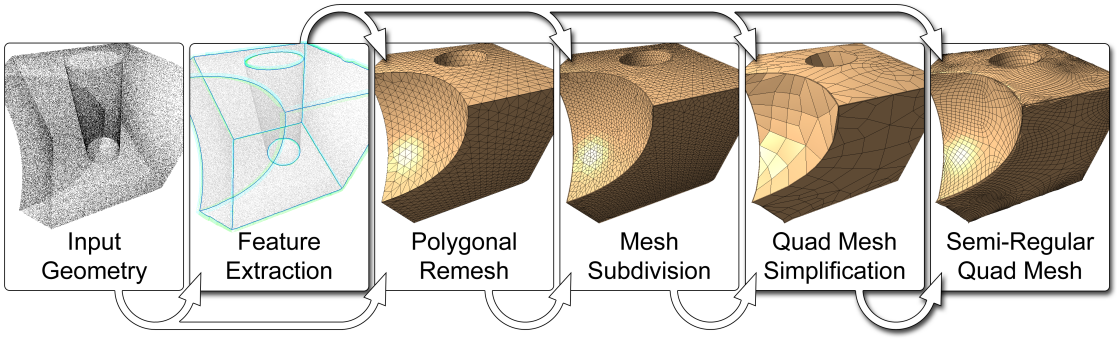
Defining important structural components over a three-dimensional (3D) model enhances the understanding of the surface and aids geometric processing and computer graphics applications. In the conventional sense, this refers to geometry based details,



1.1: A semiregular quadrilateral mesh contains a small number of extraordinary (nonvalence 4) vertices, aligned to describe a coarse segmentation that facilitates texturing, displacement mapping and spline based modeling applications, in addition to other geometry processing.

such as principal curvature information and sharp feature edges; however, it may also extend to attribute based information, including color and material properties. It has been well articulated in previous research and further addressed in Chapter 2, that respecting the model's geometry and associated attributes is important during surface processing. Consequently, identifying and preserving prominent structures on the model is a reoccurring theme throughout this work.

Chapter 3 addresses the reconstruction of sharp features from point-based geometry. Point sets, a dense collection of 3D point locations, typically do not include normals and connectivity information, so geometry processing of these structures may not be straightforward. Robust to noise common in point-based geometry, the associated algorithms reconstruct neighborhood information while generating an enhanced understanding of the point set surface and the affiliated important structures. Feature curves improve an existing surface reconstruction algorithm by introducing structural information to the unstructured data sets that leads attribute aware remeshes.



1.2: This model reconstruction pipeline motivates this research. Given a dense collection of points representing an input surface, the pipeline processes the unconnected discrete data identifying important features, remeshes the input geometry with enhanced knowledge gained from the structural feature entities, then generates quadrilaterals from the polygonal model through subdivision and simplification to construct a semiregular, quadrilateral-only representation of the original surface. The contributions of this dissertation are highlighted by the drop shadow.

1.2.2 Surface Simplification

Surface simplification, used as a building block by many higher-level algorithms, is an instrumental operator within computer graphics and geometric modeling. The goal of mesh simplification, analogous to down-sampling in digital signal processing, is to gracefully remove elements while maintaining mesh fidelity. Surface simplification methods describe a top-down approach in the construction of a mesh hierarchy of varying levels-of-detail, reducing the number of elements during each iteration.

Chapter 4 investigates methods for quadrilateral-only mesh coarsening. This simplification research addresses challenges associated with the structural constraints of quadrilateral elements while generating high quality, attribute aware mesh hierarchies. The utility of mesh simplification is further motivated in this chapter by applying the developed techniques to robustly generate quadrilateral-only meshes.

1.2.3 Surface Reconstruction

The design of surface reconstruction algorithms, specifically polygonal based remeshing, is motivated by two prominent factors, namely model conversion and mesh improvement. Consisting of the controlled resampling and generation of new connectivity over the original surface, model conversion produces useful polygonal meshes from other virtual model representations, i.e., point-based geometry acquired from range scanners. Mesh

improvement does not change the representational format of the original surface, instead modifying existing connectivity information to augment the mesh quality. In either case, the purpose of surface reconstruction is to generate high quality polygonal elements and mesh structure that is critical for the efficiency and robustness of subsequent geometry processing algorithms.

Chapter 5 addresses surface reconstruction and parameterization by constructing semiregular, quadrilateral-only meshes. The approach relies on connectivity based operators, simplification and refinement techniques, to produce quality reconstructions. This chapter includes discussion of approximation error and statistical evaluations aimed at measuring quadrilateral mesh quality. Lastly, Chapter 6 further motivates the use of connectivity based operators introduced throughout the dissertation, by building an interactive framework for quadrilateral mesh improvement.

1.3 Thesis Statement

Feature-aligned, semiregular, quadrilateral-only meshes that simultaneously define discrete, piecewise bilinear surfaces and higher order, continuous geometry offer many advantages for modeling systems and facilitate the error-bounded conversion between representational formats for surface models of arbitrary topology. This work develops automated processing and generation methods of the aforementioned quadrilateral meshes that explores the faithful reconstruction of important surface geometry and attribute data.

CHAPTER 2

BACKGROUND RESEARCH

The focus of this research is quadrilateral mesh reconstruction that is intended for spline-based segmentation and parameterization of two-manifold surface models. An n-dimensional manifold is a topological space in which the neighborhood surrounding every point is homeomorphic to the open unit ball in $\mathcal{R}^n$. In other words, a two-manifold is a well-behaved surface in which every point has a local neighborhood topollogically equivalent to a disc so that the surface model does not contain boundaries or self-intersections. The remainder of this chapter discusses two-manifold geometry processing algorithms, including surface feature extraction, geometric- and attribute-based simplification, generation and parameterization, and describes their releveance to faithful surface reconstruction.

2.1 Surface Features

An important aspect of surface reconstruction is the faithful recovery or preservation of surface features, defined as any prominent geometry or attribute component on the original model. For example, manufactured models, especially machined parts, typically exhibit sharp edges formed by the intersection of two surfaces that describes an important discontinuity in the surface normals. In addition to geometric feature extraction, on which a majority of related algorithms focus, this section discusses attributes, like color and material properties, which may be related to the surface geometry, as well as attribute conscience geometric processing and image processing algorithms.

2.1.1 Mesh-based Feature Extraction

Several techniques have investigated the identification of feature edges of polygonal models. Hubeli et al. [75] create a multiresolution framework with normal-based classification operators to define a set of edges from which a thinning process extracts feature lines. Watanabe et al. [171] use discrete differential geometry operators to compute

approximations of the mean and Gaussian curvatures. Whereas, Hildebrandt et al. [69] use anisotropic filtering on discrete differential geometric approximations of third order derivatives of the surface mesh. Both techniques [171, 69] build a set of feature edges from the extrema triangles. Attene et al. [13] identify chamfer triangles to reconstruct sharp features on meshes. They insert new vertices on the edges and faces of these triangles and project the points to the intersection of planes fit to the surrounding surfaces.

The underlying assumption of connectivity and normals associated with the vertices of the mesh is not appropriate for point-based models. To extract feature lines from point clouds using these techniques, a connectivity construction method (surface reconstruction) must be applied in a preprocessing step. The construction of connectivity is nontrivial, computationally expensive, and moreover, the success of feature extraction relies on the ability of the polygonal meshing procedure to accurately build the sharp edges.

2.1.2 Point-Based Feature Extraction

Geometric processing of point-based models is not straightforward in the absence of connectivity and normal information. For geometric reconstruction of surfaces defined by point cloud data, statistical-based methods describe more accurate continuous surfaces than averaging techniques. *Moving Least Squares* (MLS) surfaces [97, 4], locally computes a weighted least squares fit polynomial function to a finite neighborhood of points surrounding a query point. The *Robust MLS* (RMLS) variant [52] employs principles of robust statistics to segment the neighborhood into distinct groups to define multiple surface fits in regions of sharp corners. The radial basis function approach [26] assigns a finite region of influence to each point over which it affects the surface computation with a smoothly dissipating strength as a function of distance. Statistical point geometry method [80] introduces a hierarchical method of point-based modeling relying on a principal component analysis (PCA) neighborhood grouping scheme that is based on higher dimensional data for attribute information.

The generation of sharp geometric features from point-based geometry typically results from trees built over a subset of points flagged as potentially lying on the model's features. Pauly et al. [121] use covariance analysis of the distance-driven local neighborhoods to flag potential feature points. By varying the radius of the neighborhoods, they develop a multiresolution scheme capable of processing noisy input data. Gumhold et al. [66]

construct a Riemann graph over local neighborhoods and use covariance analysis to compute weights that flag points as potential creases, boundaries or corners. Both techniques [66, 121] connect the flagged points using a minimum spanning tree and fit curves to approximate sharp edges. Demarsin et al. [36] compute point normals using principal component analysis (PCA) and segment the points into groups based on the normal variation in local neighborhoods. A minimum spanning tree is constructed between the boundary points of the assorted clusters, which is used to build the final feature curves.

While these techniques are capable of extracting features on point clouds by connecting existing points, their accuracy depends on the sampling quality of the input model. In contrast to their work, the feature extraction algorithm proposed in this document leverages projection methods to accurately sample the sharp edges and yield more accurate approximations of geometric surface features.

Jenke et al. [76] identify sharp features to improve their Bayesian statistic-based reconstruction algorithm. Their method uses a curvature test to identify potential edge regions, but their classification method limits identification to complete edges and is unable to extract *darts* or dissipating edges.

There are many related algorithms and techniques based on feature edge detection in the realm of image processing [25] that may extend to geometry processing. Examples of this cross-pollination exist that uses fuzzy clustering analysis [176] and explores ways in which image processing [23, 72] can be applied to geometry processing for the decomposition of the given geometry [83, 155]. As image processing methods have focused on the extraction of important features and edge detection their extension to geometry-based methods may be fruitful. Locally (or globally) flattening the input geometry and applying traditional image-based methods to the *MLS*-based residuals or other associated attributes of the point-based model can extract feature information from the model. For a survey of image segmentation techniques, including thresholding [128], watershedding [139], and edge-based [82, 132] approaches, I refer the reader to related surveys [119, 106].

2.2 Surface Reconstruction

Surface reconstruction algorithms are dominanted by triangular-based techniques; however, there is a growing number of quadrilateral-based remeshing methods. This dis-

cussion focuses the review of related techniques for polygonal remeshing to quadrilateral-based techniques and their overlap with surface parameterization problems. In the following section, the assorted methods are categorized as connectivity operators, computational shape, advancing front, resampling, deformations, voxelization, and numerical integration techniques. For each genre, the basic approach is outlined as well as detailing the associated advantages and disadvantages.

Arguably the most straightforward approach to mesh generation is to convert existing mesh connectivity to describe triangles from quadrilaterals or vice-versa. The triangulation of each element of a polygonal mesh will intuitively generate triangular-based surfaces from other polygonal inputs. Similarly, a single iteration of Catmull-Clark subdivision [28] converts arbitrary polygonal surfaces into a quadrilateral-only meshes. The $\sqrt{3}$-subdivision scheme [88] and complete pairing methods [92] that merge two adjacent triangles into a single quadrilateral to generate quadrilateral meshes from triangles. Conversion schemes that rely on subdivision or other splitting schemes generally result in an explosion of mesh elements. Additionally, these techniques have little control over the nodal valence of the final meshes that results in many extraordinary, nonideal valence vertices, and may lose important feature edges and alignment.

Mesh improvement techniques employ localized operators to modify the existing vertex locations and mesh connectivity. For instance, mesh smoothing [177, 78, 50, 181, 154] and fairing [38] describe iterative algorithms that relax vertex locations to improve local element shape while smoothing high frequency noise from the surface. Edge-sharpening [13], similarly modifies vertex location, but for the purpose of reconstructing sharp geometric features on the mesh. While these methods may greatly improve mean quality of the triangle and/or quadrilateral elements, their power is limited by the connectivity of the input mesh, i.e., the nodal valence, that places a ceilings on worst case results. To further improve the mesh quality, reconfiguration methods introduce modifications to the connectivity. Local reconfiguration methods, i.e., edge swap methods, for triangle meshes [71] and quadrilateral meshes [87, 154, 118], sometimes considering semiglobal methods such as wandering edges [168, 156, 157, 22], can greatly improve vertex valences to augment the effects of smoothing methods.

The relationship between the Voronoi diagram and Delaunay triangulations has been exploited by many geometry processing algorithms. For instance, both Voronoi-based techniques [8, 44, 11, 9, 10, 110], and Delaunay triangulations [61, 29, 127, 113, 30, 19]

can be used to reconstruct mesh geometry from unstructured point sets. Additional computational shape-based processing techniques include Voronoi- [50, 181] and Lloyd-based [103, 156, 157, 22] smoothing methods to improve element quality. Typically these techniques successfully apply to triangles; however, their generalization for quadrilateral meshes is not necessarily straightforward. Instead, examples of quadrilateral remeshing built on computational geometry methods are often coupled with conversion schemes, i.e., complete pairing, to generate quadrilateral meshes [92, 146, 148, 167].

Wavefront propagation methods expand a *front* over the model while defining a faceted surface mesh in its wake. Many techniques exist for triangle surfaces [14, 163], including ball pivoting [16], Delaunay-based methods [61, 30] and projection schemes [134], from which conversion schemes can generate quadrilateral models. In constrast, the Q-morph algorithm [118] and the paving method [18] with its variants [173, 27] construct quadrilateral elements directly.

Advancing wavefronts are advantageous because they can place guarantees on topological correctness and are highly parallizable. By controlling the size of elements along the wavefront-based on geodesic [123, 149] or curvature guidance fields [136, 137], these methods attain high quality, attribute-aware elements and construct error bounded remeshes. However, tracking and resolution of wavefront collisions can be difficult and interfere with the threshold element quality [60].

Surface resampling methods are globalized techniques that evenly distribute points over the input surface model. Attraction and repulsion functions [164] represented as charged particles [112, 111, 174, 158] or elliptical bubble packing [145, 147], and computational techniques, including farthest point method [49], Delaunay-based methods [29, 127, 113], and sink insertion [48], are coupled with Delaunay triangulation algorithms [41] to generate triangle meshes. Extensions of bubble packing methods, using rectangular bundled force equations [167, 146, 148], has led to quad-dominant reconstructions with adaptive elements. These techniques generate similar high quality elements, in comparison to advancing wavefronts, without the degeneracies related to wavefront collisions.

Deformable models, based on image processing edge-detection schemes [82, 109], use *shrink wrapping* [77, 89] or *ballooning* [140] schemes to evolve a shape to the original geometry. These techniques have proven useful in describing complex geometry and arbitrary genus models using connectivity-based operators to generate quality elements. However, their application to quadrilateral meshing is complicated by a reliance on

localized refinement schemes. Without allowing T-junctions [100, 138, 68] on the mesh, refinement of a single element may propogate to many elements over the model.

Voxel-based techniques, targeting triangular- and quadrilateral-based reconstructions, constitute some of the most robust remeshing algorithms. The widely popular *Marching Cubes* algorithm [105], and its variants [116, 143], evaluate functional values at the nodes of a grid, then generate triangular-based representations of an iso-surface by applying special case triangulations to the voxels containing the desired iso-value. While some efforts have experimented with the control of the final element sizes [15], the resulting triangles are often poorly shaped.

Surface grafting [151, 144] is a similar voxel-based technique for quadrilateral meshes. A surface model is embedded inside a gridded region and all nodes of the grid identified as *outside* the surface model are culled. The remaining boundary nodes of the grid are projected to the surface to create a structured hexahedral model whose boundary corresponds to a quadrilateral surface mesh. This technique generates many (in practice near 50%) extraordinary vertices, nonvalence four nodes, especially where the mesh is not axis-aligned by creating aliased boundaries after the culling procedure, observed by [24]. Furthermore, similar to Marching Cubes, reconstruction of nonaxis aligned features is nontrivial, the output mesh is dependent on the input mesh alignment, and topological correctness is not guaranteed.

Quadrilateral reconstruction schemes leverage orthogonal frames computed over the surface [81], stemming from the observation that streamlines traced through such vector fields constructs quadrilateral elements. Numerical integration methods trace curves through the two fields seeded at critical points of the vector fields, defining the edges of a quadrilateral mesh. While any orthogonal vector field is applicable, principal curvature directions have proven a very useful means of defining orthogonal frames [7, 107, 126]. Their attractive attributes include high quality quadrilaterals and, naturally, curvature aligned elements. However, in isotropic or flat regions these methods are not well defined.

2.3 Surface Parameterization

To this point, the reviewed polygonal reconstruction algorithms have not focused on the construction of structured meshes. A structured mesh is defined as having an ideal valence at all interior vertices; for instance, triangle meshes desire valence six vertices and valence four vertices are preferred for quadrilaterals. A semistructured,

or semiregular, mesh, is one that contains a minimal number of extraordinary vertices which are characterized has having nonideal valences. As illustrated in Figure 1.1, it is preferable to have the extraordinary vertices align so that they describe the boundaries of a coarse segmentation of the model that facilitates parameterization solutions.

Given a parameterization of a model, it is straightforward to develop a structured quadrilateral remesh of the original model by appropriately sampling the well-known parametric domain. It is in this manner that spline-based models are automatically generated; potentially segmenting a model, parameterizing the individual regions, then resampling structured quadrilateral meshes to guide spline interpolation or fitting algorithms. Due to the relationship with quadrilateral meshing, I review multiple surface parameterization challenges and algorithms.

2.3.1 Parametric Distortion

Parametric distortions are inherent in surface parameterization solutions, typically mapping a polygonal mesh to a well-known domain, except in special cases. A parametric solution will focus on minimizing the error associated with either angle or area distortion depending on the driving application. *Angle preservation*, a conformal mapping, is achieved if the angle between any two intersecting curves on the surface model have an identical angle to their preimage in the parameter domain. *Area preservation*, an equiareal mapping, implies that any region of the surface will have an equal area as its image on the parameter domain. *Length preservation*, an isometric mapping, is both conformal and equiareal, where any curve traced on the surface is identical in length to its mapped image on the well-known parameter domain.

2.3.2 Planar Parameterization

Early surface parameterization techniques map the 3D surface model homeomorphic to the disc to a planar domain, useful for manifolds with a boundary or requiring graph cuts to segment the model into one or more patches. Typically, the boundary vertices of a mesh are mapped to the boundary of a 2D convex region, typically circular, then the remaining interior vertices obtained by minimizing an error metric in the form of a linear system. Planar-based parameterization methods have been addressed in many research papers, originally introduced using uniform weights by [165], extended via new weighting schemes for angle preservation in [45, 53, 54].

However, planar parameterization techniques that require a fixed boundary only per-

form well when the boundary of the model is convex and does not vary greatly from that of the planar domain. In comparison, unconstrained boundary solutions further reduce parametric distortions by allowing the boundary vertices to float to a minimal energy state. These techniques are achieved through insertion of virtual boundary elements [96], explicit computation of the free-boundary [99, 37], relaxation of a fixed boundary map [73, 179], and other techniques [141, 130, 84].

2.3.3 Nonplanar Base Domains

Models without boundaries require graph cuts to segment the surface so that it is homeomorphic to a disc. Instead, many algorithms have opted to map genus-zero models to a spherical domain [2, 62, 124, 102, 93], for a seamless parameterization. For higher genus models or surfaces with large undulations, more complex base domains are generated [45, 67] potentially derived through simplification [46, 159, 94], or using ray casting to shrink wrap semiregular meshes onto the original surface [90, 77].

Many of these techniques are developed for triangle-based reconstructions; however, their extension to quadrilateral remeshing is straightforward, as observed by geometry images [63, 131], polycubes maps [160, 101] and polycube splines [169]. The complexity of surface mapping approaches lies in the construction of a base domain and developing the mapping between it and the original surface; however, typically the related parametric distortions are significantly reduced in comparison to planar mappings.

2.3.4 Chart Segmentation

To reduce parametric distortions and to handle arbitrary topological models, charting methods segment a model into multiple regions homeomorphic to the disc. Following the success of user defined segmentation boundaries [91, 162], automated techniques implement Morse boundary cuts [115, 43, 42, 74] or use variational shape approximation [155, 59, 108] segmentation methods. Chart segmentation has been particularly useful in defining continuous manifold spline surfaces [65, 64, 68] and the extensions to polycube-based splines [160, 169]. For further discussion on the complexities of surface parameterizations and minimization of parametric distortions, I direct the reader to the survey [55, 142].

CHAPTER 3

ROBUST POINT BASED FEATURE EXTRACTION

Digital scanner technology has become more affordable and accurate, increasing its popularity and utility. These scanners collect a dense sampling of points, with mechanical probes or lasers, to generate a virtual representation of a physical form. Consequently, geometric processing of point clouds is becoming increasingly important.

The preservation of sharp features is a primary concern for many geometric computations and modeling applications. In this context, a feature is described as the discontinuity in the surface normals evaluated on the model, where the G^1 continuity is not maintained. Identification of these edges facilitates a better understanding of the model improving filtering [39, 177], reconstruction [134, 33], resampling [13], simplification [58, 70], smoothing [52, 78], and visualization [5] methods.

Features can be computed through the investigation of changes in the normals of neighboring discrete surface data, i.e., polygonal facets or vertices. The most simple method is to implement a threshold test that identifies potential feature edges where normals differ above some tolerance level across adjacent samples. However, the implementation of such detection techniques is not straightforward, since in most cases, normals and connectivity are not provided in point data sets and their reconstruction is a nontrivial problem, especially when sharp features must be preserved.

Digital scanner technology has become more affordable and accurate, increasing its popularity and utility. These scanners collect a dense sampling of points, with mechanical probes or lasers, to generate a virtual representation of a physical form. Consequently, geometric processing of point clouds is becoming increasingly important. However, the noise inherent in scanned models presents a major challenge to feature edge extraction in point clouds. The identification of feature lines is complicated in the presence of noise due to the fact that the large surface gradients appearing in regions with sharp features are similar to those found in noisy areas.

This chapter presents an algorithm to define a set of curves that are aligned along the feature edges of a point cloud [34]. The pipeline, illustrated in Figure 3.1, extends an earlier technique [33] by improving accuracy and reducing noise within the extracted curves near acute feature edges and on low quality point clouds. Additionally, the updated approach uses an iterative least square fitting of B-spline curves to perform smoothing in place of the previous principal component analysis (PCA) based relaxation, that describes continuous curves rather than discrete, piecewise linear polylines.

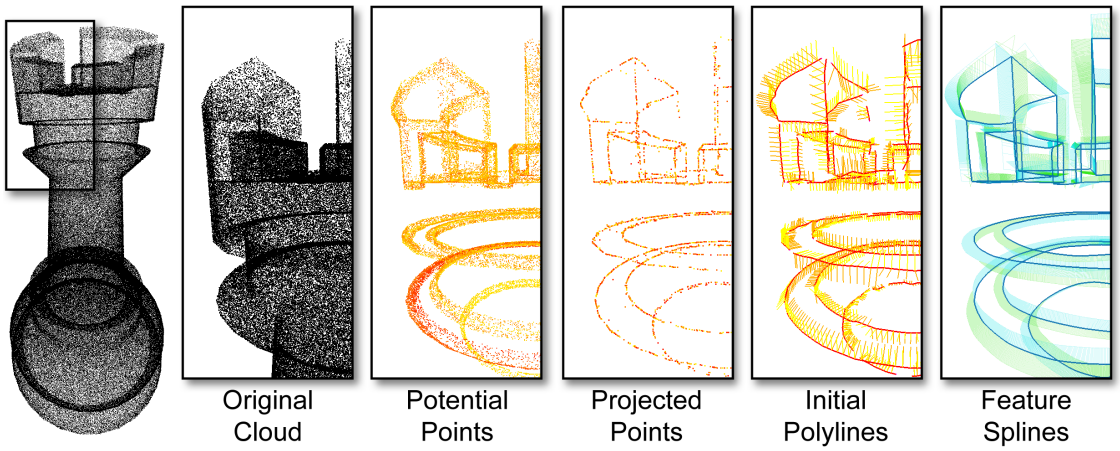
The algorithm is based upon the framework of *Robust Moving Least Squares* (RMLS) surfaces [52]. A number of potential feature points are selected, identified by the RMLS operator to be near to possible features. One issue with the RMLS is the inability to reconstruct smooth features. The projection method produces jagged edges that cannot be used without further processing to construct smooth feature curves. Therefore, a feature growing strategy constructs polylines used to parameterize and guide the creation of spline curves smoothly fit to the RMLS projected edge points. This research accounts for poor sampling quality, due to noise or low point densities, by connecting the created polylines across gaps to construct a set of complete feature splines.

The advantage of the following feature extraction research is that, in contrast to other feature extraction work [69], the input data consist of a set of unorganized points (possibly affected by noise) that approximate the original surface. No assumptions are made of associated attribute information, e.g., normal vectors, given apriori. In the presence of closed loops, the feature edges define a segmentation of the point based model based on complete feature loops. Leveraging this segmentation scheme improves the performance of previously developed point based algorithms, including model compression and surface meshing.

3.1 Point Based Computing

The input data are an unorganized set of points $\mathcal{P} = \{(x, y, z)^T\} \subset \mathcal{R}^3$ that sample some original surface $\mathcal{S}$. The algorithm does not rely on attributes, such as associated point normal vectors or connectivity information. Given $\mathcal{P}$, one computes $\mathcal{S}$ to be the *Moving Least Squares* (MLS) surface defined by a projection operator Ψ that is applied to an arbitrary point $p \in \mathcal{R}^3$ in order to project it onto the surface, $\Psi(p) \in \mathcal{S}$. Each point on the surface projects onto itself, $\mathcal{S} = \{p \mid p = \Psi(p)\}$.

There are several options of selecting the projection operator Ψ, originally proposed



3.1: The feature extraction pipeline (left to right) for a given point cloud, projects candidate points, identified near potential features, to the intersection of locally fit RMLS surfaces. Approximating polylines are grown through the projected data and guide the creation of arc length parameterized B-spline feature curves.

by Levin [98] and more recently modified by [3, 6, 12, 52]. Most variants implement the projection of a point $p \in \mathcal{R}^3$ through finding a plane H that approximates a local neighborhood around p in the set $\mathcal{P}$. Specifically, the plane H is defined by a point $q \in \mathcal{R}^3$ and a normal n, $H = H(n, q) = \{x \in \mathcal{R}^3 \mid \langle x - q, n \rangle = 0, \|n\| = 1\}$, and found, such that the sum of weighted squared distances of points in $\mathcal{P}$ to the plane H, is minimized:

$$(n, q) = \arg\min_{(n,q)} \sum_{p \in \mathcal{P}} \langle n, p - q \rangle^2 \theta(\|p - q\|), \tag{3.1}$$

where $\theta(\cdot)$ is an exponentially decreasing weighting function, e.g., $\theta(s) = e^{-s^2/h^2}$, which assigns larger weights to points near q. The parameter h defines the spatial scale of θ. Once H is found, it serves as a reference for a local Cartesian coordinate system with origin in q and two span vectors that are orthogonal to n. Then, a bivariate polynomial g of a given degree is fit in order to find the projected point $\Psi(x)$, which is above q and on the surface $\mathcal{S}$. This polynomial fitting corresponds to finding

$$\arg\min_{g} \sum_{p \in \mathcal{P}} \|p - p_g\|^2 \theta(\|p - q\|), \tag{3.2}$$

where p_g denotes the projection of p onto the polynomial g in direction of the plane normal n, and $\theta(\cdot)$ is the previously defined weighting function.

The MLS projection relies on the minimization of Equations 3.1 and 3.2, which includes Gaussian weighting of points in the local neighborhoods. This weighting leads

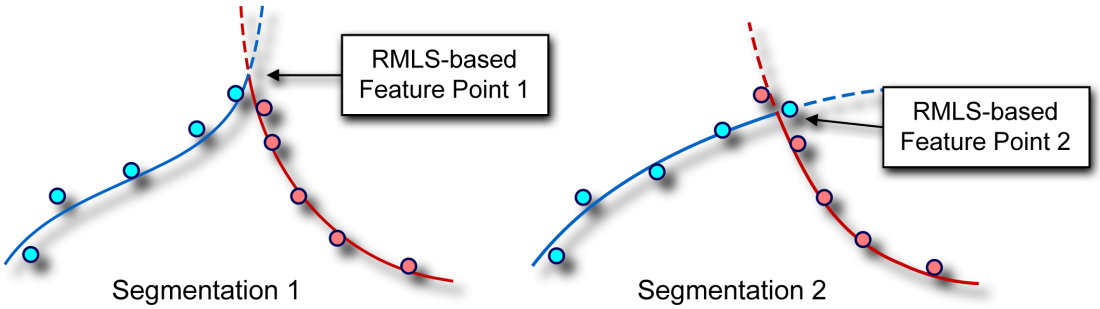
to a smoothing effect that corrects noise and outliers in the set $\mathcal{P}$, but simultaneously removes sharp features. The RMLS variant [52] was developed to solve this problem by considering iteratively constructed neighborhoods based on statistical analyzes of the corresponding point distributions. In particular, they use least median of squares, which is a regression method to minimize the median of absolute residuals between the point set and the fit. Their forward search algorithm grows multiple MLS neighborhoods in order to find smooth, flat, and outlier free regions (*subneighborhoods*) for the final MLS projection.

This approach is inherently capable of detecting noise by disregarding outliers during the surface fitting phase, and defines sharp features. Moreover, it reduces the smoothing effect of the traditional MLS projection and improves numerical stability. However, the drawback is that the clustering is computed independently for all points, which may lead to situations where nearby points do not grow to identical subneighborhoods, a two-dimensional example is diagrammed in Figure 3.2, that results in different configurations of the surface feature. Figure 3.3 highlights the RMLS generated jagged edges while upsampling a scalloped cube model.

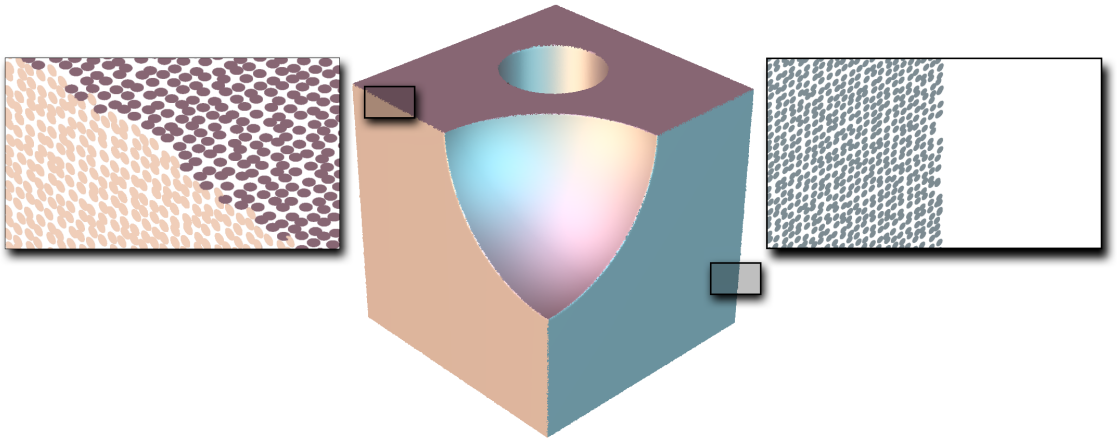
To overcome this problem, this extraction research injects additional information into the point based model by constructing a set of feature curves that *accurately* approximate RMLS created feature edges. The goal is to generate continuous and complete spline curves to *smoothly* annotate the important features on the model. It is important for subsequent geometric processing algorithms that the curves faithfully, within tolerable distances, reconstruct the geometric features of the point set surface while ironing out noise acquired during model creation and RMLS projection.

3.2 Feature Extraction Algorithm

The following chapter describes in detail the five-step algorithm illustrated in Figure 3.1. First, Section 3.2.1 extracts points from $\mathcal{P}$ that are near potential features, offering user tuning inputs to taylor the sensitivity of the algorithm to define the feature detail to be reconstructed. Next, in Section 3.2.2, RMLS fits multiple surfaces to the local neighborhoods of each point and projects the candidate points to the nearest intersection between the RMLS surfaces, approximating initial feature information. Third, a set of feature polylines are constructed through the projected point cloud in Section 3.2.3; that guide the reconstruction of corners and the completion of gaps in Section 3.2.4. Lastly,



3.2: RMLS segmentation may construct different subneighborhoods for nearby points, causing disparities in the intersecting surfaces and feature locations.



3.3: Disparities in the RMLS subneighborhoods of nearby points results in jagged edges when projecting individual points to the point set surface.

Section 3.2.5 defines an arc-length parameterization over the projected points based on the initial feature polylines to which spline curves can be fit. The remainder of this section further discusses the algorithmic details of each of the five steps.

3.2.1 Identifying Potential Edge Regions

The identification of potential edge regions, culling away points from the model to reveal candidate feature points, is an extension of the RMLS based residual computation [52]. Given a point $p \in \mathcal{P}$, they consider the neighborhood $\mathcal{N}(p) \subset \mathcal{P}$ that is used for the polynomial fit of the MLS projection. Considering the polynomial f over the reference plane H that is defined by point q and normal n, they evaluate the maximum polynomial

residual r in a neighborhood around p,

$$r(p) = \max_{x \in \mathcal{N}(p)} \|x - x_f\|,$$

where x_f corresponds to the projection of x onto f in direction of the plane normal n of H, see the diagram in Figure 3.4.

For sharp features, r becomes reasonably large because many point sampled along the sharp edge lift away from the polynomial. Based on this observation, they straightforwardly define any point $p \in \mathcal{P}$ to be a potential feature point, if $r(p) > \tau$ for some user defined threshold τ. Note that strong outliers will also record large r; however, they are eliminated during the RMLS projection stage (discussed in Section 3.2.2).

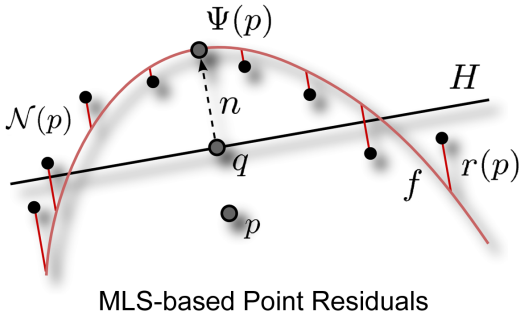
The identification of potential feature points relies on an appropriate selection of the threshold τ. This research extends the RMLS residual based method by using an automatic computation method, which produces an adaptive threshold. In particular, setting τ to the mean residual value taken over all points in $\mathcal{P}$, $\tau = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} r(p)$, leads to good results. Moreover, the user is able to modify τ manually, using an interactive tool. Figure 3.5 shows the Fandisk model after identifying potential feature points using different values for the threshold τ. The output of the identification procedure is a set of potential feature points, i.e., $\mathcal{F} = \{p \in \mathcal{P} \mid r(p) > \tau\}$.

3.2.2 Projecting Points to Edges

By coupling the RMLS projection procedure with a Newton's root finding method, the points $p \in \mathcal{F}$ are projected onto the surface features, returning a feature edge cloud $\mathcal{E}$. The RMLS projection method fits a number of surfaces to the points in the neighborhood around p. This number of surfaces is used to classify the feature type of the projected points in $\mathcal{E}$.

If RMLS returns a single surface, then this implies that no feature was detected and the high residual is assumed to be the byproduct of outliers due to noisy sampling data. In contrast, two or more surfaces indicate the presence of a sharp feature. Newton's method, as used in [121], is able to iteratively project a point p onto the intersection of the two RMLS surfaces closest to p. This process yields an *edge point* e that may be inserted into the edge cloud $\mathcal{E}$ corresponding to $p \in \mathcal{F}$.

If the distance between the original point p and its candidate projection point e, $\|p - e\|$, is less than the radius of the RMLS neighborhood, then e is added to $\mathcal{E}$. The culling process reduces noise within the set of projected points $\mathcal{E}$, especially near acute



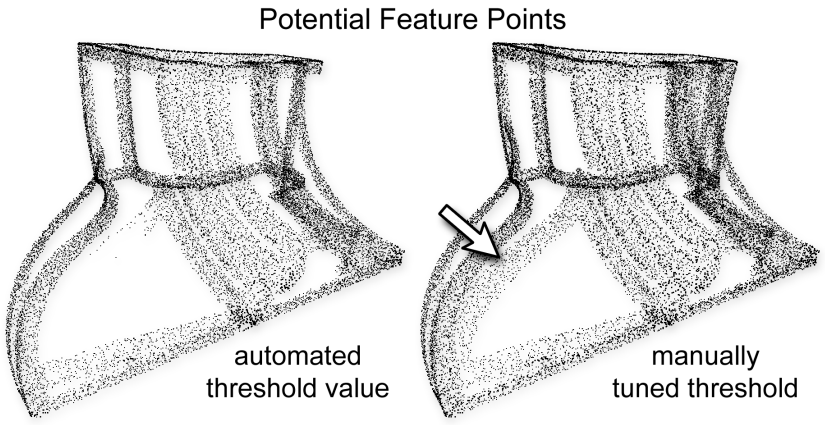
3.4: The residual $r(p)$ is computed as the maximum distance between the points $\mathcal{N}(p)$ in the neighborhood around p and the polynomial fit f in direction of the reference plane, H, normal.

features or on noisy models. The purpose is to keep only the projected points whose feature is well defined by the neighborhood of the original point.

However, in practice, due to irregular sampling in $\mathcal{F}$ and the conditional point insertions, there often exist regions in $\mathcal{E}$ with low sample densities. This becomes problematic during the feature approximation stage, Section 3.2.3, by creating large gaps between extracted polylines. To accommodate for the lost points additional points are inserted into $\mathcal{E}$, upsampling the projected feature edge point cloud. This upsampling occurs by adding points along the eigenvector v_0 that corresponds to the largest eigenvalue of the neighborhood of e, $\mathcal{N} \in \mathcal{E}$, $e \pm \alpha \cdot v_0$. The scalar value α is based on the growth step size defined in Section 3.2.3. Upsampled points are computed only if the computed eigenvectors v_0, v_1 and v_2, with associated eigenvalues $\lambda_0 \geq \lambda_1 \geq \lambda_2$, are highly correlated, $\lambda_0/(\lambda_0 + \lambda_1 + \lambda_2) > \beta$. This analysis is adapted from a surface variation computational technique [120], used to approximate the local curvature information. The models and results presented in this paper experienced good results with $\alpha = 0.75$ and $\beta = 0.7$.

3.2.3 Propagating Feature Polylines

The set of projected feature points $\mathcal{E}$ is unorganized and contains no topological information; however, in many practical applications continuous feature curves are desired. The goal of the feature polyline propagation is to approximate $\mathcal{E}$ with a set of polylines. Because these polylines will guide the future parameterization of $\mathcal{E}$ and iterative spline curve fitting, they are not expected to be smooth; instead, relying on the spline curves to smooth noise in $\mathcal{E}$.



3.5: Potential feature points are identified based on $r(p)$, using the threshold τ, for the Fandisk model with automatically computed ($\tau = 0.057$) and manually adjusted thresholds ($\tau = 0.014$). The arrow indicates a region with an increased number of potential feature points due to a lower τ value.

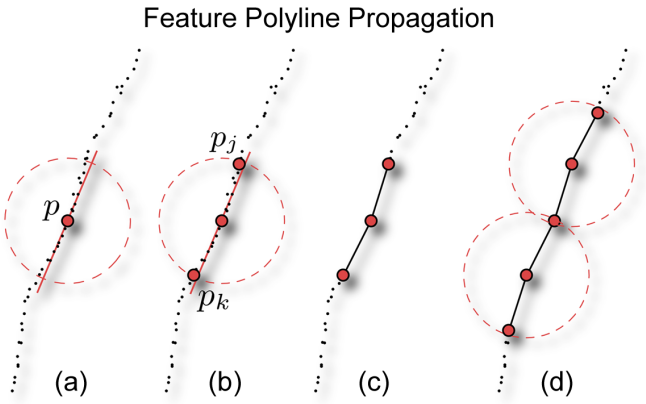
Polylines are generated through $\mathcal{E}$ by adapting a curve reconstruction technique from unstructured point sets [95]. A user-defined maximum allowable segment length s_{max} controls the coarseness of the feature polylines. The propagation is initialized at a random seed point p, where the PCA for points in its neighborhood $\mathcal{N}(p) \subset \mathcal{E}$ with radius s_{max} is computed, illustrated in Figure 3.6(a). All points $p_i \in \mathcal{N}(p)$ are projected onto the line defined by the eigenvector of $\mathcal{N}(p)$ with the largest eigenvalue through p, yielding points p'_i. The growing scheme attaches the two points p_j, $p_k \in \mathcal{N}(p)$ with the furthest corresponding projections p'_j and p'_k in opposite directions from p.

Extending this curve growing scheme in order to produce smoother approximating polylines, the propagation method computes a step vector v_c as the weighted average of neighborhood points:

$$v_c = \sum_{i=0} \theta(\|p'_i - p\|) \cdot \frac{p_i - p}{\|p_i - p\|}.$$

The points are weighted inversely based on their projected distance along the eigenvector, $\theta(s) = e^{-s^2/s^2_{max}}$, and the final step vector v_c is normalized. A new polyline vertex is added to the feature at $p_j = p + s_{max} \cdot v_c$, illustrated in Figure 3.6(b). The procedure is repeated at the p_j, Figure 3.6(c), until no points remain in the growth direction. The process can be simultaneously executed in the opposite growth direction 3.6(d).

An additional termination condition is introduced to further reduce noise in the grown polylines and prevent corner rounding, occuring when a single polyline wraps over a corner



3.6: Feature edge propagation is seeded at a point, projects all neighborhood points within a user specified radius to the major eigenvector, computes the step direction and adds a new point to the approximating polyline. The processed points are removed and the method is repeated until no further points remain in the growth direction.

region to define multiple feature edges. Considering historical information, a growth vector v_g stores the average direction of previous step vectors. To commit a new point p_j, the angle between the current step vector v_c and the growth vector v_g must be below a defined threshold, $\langle v_g, v_c \rangle \leq cos(\alpha)$. In practice, $\alpha = 30°$ provides an adequate angle threshold. After the point is committed to the feature polyline, the growth vector v_g is updated, accumulating the step vectors $v_g = \frac{1}{2}(v_g + v_c)$.

In practice, starting the feature growing procedure at edge points with highly correlated neighborhood eigenvectors achieves the best results. The implementation maintains a priority queue that sorts all edge points $p \in \mathcal{E}$ according to their eigenvector correlation. Given eigenvalues $\lambda_0 \geq \lambda_1 \geq \lambda_2$, the correlation term for a point is computed as $\lambda_0/(\lambda_0 + \lambda_1 + \lambda_2)$. This sorting term seeds the feature growth at points furthest away from corner regions in $\mathcal{E}$ where the correlation term will be computed as a low value.

A feature line is created by seeding the propagation algorithm described above at the point on the top of the queue. After the feature points are connected and the propagation terminates, the elements that correspond to points within the neighborhoods $\mathcal{N}(p)$ of processed points p are removed from the queue. As long as the queue is not empty, there are unvisited feature points remaining thus continuing feature line creation. The polyline growing scheme approximates the projected cloud $\mathcal{E}$; however, it is not yet complete due to gaps that occur in regions with poor sample quality. The fourth stage of the feature extraction pipeline will clean up the grown polylines to produce a set of complete feature

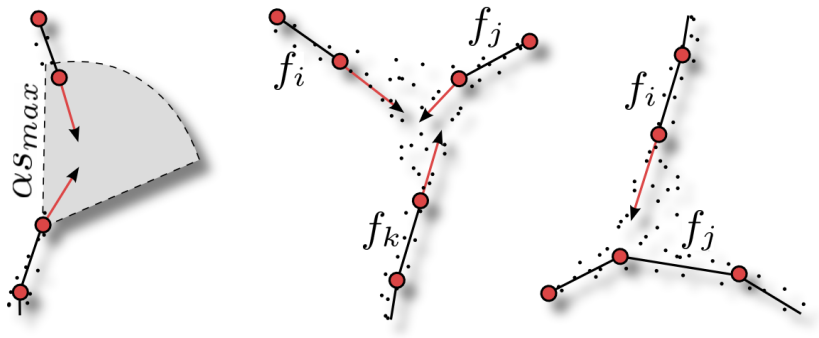
curves.

3.2.4 Completing Feature Curves

In this stage, clean up operations merge and split the feature polylines to describe a complete set of approximating line segments through the RMLS project cloud $\mathcal{E}$. The feature completion method is driven by a directional search approach, illustrated in Figure 3.7, that inspects the regions between the polylines to determine if multiple features can be combined or new corners are necessary. The operations are guided by tangent vectors evaluated at the end points of each feature, computed by fitting a cubic polynomial to the last four points of the polyline. An alternative approach evaluates the direction of the last segment of the curve; however, in practice the cubic fit smooths perturbations for better results.

For each end point p, the method searches for other feature points, within the cone formed by the tangent vector at p and a predefined aperture angle, that satisfy one of the three completion operations illustrated in Figure 3.7: *gap completion*, *corner creation*, and *edge splitting*. Gap completion is necessitated by poor sampling quality, i.e. the distance between two end points is larger than s_{max}. To resolve these configurations, any two polylines with end points that are within αs_{max} distance of each other and have corresponding tangent vectors that point to opposing directions within the aperture angle are merged into a single feature polyline. The corner creation results when multiple feature polylines are detected within the search cone. For three feature lines $f_{i,j,k}$ with end points $p_{i,j,k}$ within the search cone and converging tangents, a corner point is evaluated using the RMLS surfaces fit to the neighborhood of the midpoint, $(p_i + p_j + p_k)/3$. This midpoint is projected to the corner using Newton's method to evaluate the intersection of three RMLS defined surfaces. Each feature polyline $f_{i,j,k}$ is connected to the newly computed corner point. The final completion operator, edge splitting occurs if the end point of a feature polyline f_i projects to an interior point p of another feature line f_j. The feature f_j is split into two feature polylines, f_k and f_l, at p and a corner point c is identified within the region using Newton's method on the RMLS defined surfaces. The three feature polylines f_i, f_k, and f_l are joined at the corner point c. This case resolves situations where a single polyline grew onto multiple features despite the growth vector termination case.

The search algorithm finds the closest vertices of the feature polylines that exists



3.7: A final pass completes the feature edge propogation, constucting a set of *complete* feature polylines by bridging gaps (left), creating new corners (middle) and splitting existing polylines (right).

within the volume of the search cone bounded by the distance αs_{max}. This search is repeated multiple times for every feature endpoint, iteratively increasing α over the range $(0, 1.5]$ to accommodate for the poor sampling quality in $\mathcal{E}$. If no vertices of feature lines are found during the this time, then the endpoint is assumed to belong to a dissipating feature edge (or *dart*), and no connection operation is performed. After the end points have been resolved, the feature curves are considered complete and ready to guide the spline based fitting procedure.

3.2.5 Spline based Fitting and Smoothing

The output of the feature extraction algorithm is a set of continuous spline curves fit to $\mathcal{E}$ guided by the approximating feature polylines. To this point a set of feature polylines $\mathcal{F} = \{f_i\}$ approximate RMLS feature edges through a projected point cloud $\mathcal{E}$. The spline fitting procedure segments $\mathcal{E}$ based on proximity of feature polylines, assigns parameter values to the segmented points, then uses a iterative least squares fitting procedure to define the spline curves.

The segmentation of $\mathcal{E}$ into multiple subclouds is based on the user defined neighborhood size s_{max}, used earlier as a step size for the polyline growth. A point $e \in \mathcal{E}$ is projected to each feature polyline f_i, yielding the point e'_i. The point e is inserted into the subcloud $\tilde{\mathcal{E}}_i$ if:

1. the distance between e and e'_i is less than the specified distance, $\|e - e'_i\| < s_{max}$,
2. e'_i is the nearest projected point to e, $\forall j, \|e'_j - e\| > \|e'_i - e\|$.

After, the points within each segmented subcloud $\tilde{\mathcal{E}}_i$ are assigned parameter values in order to fit spline feature curves. A point $e \in \tilde{\mathcal{E}}_i$ is assigned a parameter value based on the ratio of the arc-length distance of its projection on f_i, e'_i, along the feature polyline and the overall length of f_i. In this manner, all computed parameter values are bounded between $[0, 1]$.

The spline curves are created for each segmented subcloud using an iterative approach that refines the defining knot vector until the curve is within bounded distance tolerances. For a subcloud $\tilde{\mathcal{E}}_i$ with associated parameter values $\mathcal{U}_i$, a spline curve $\mathcal{S}$ with the knot vector $k = \{k_i\}_{i=0}^{N}$ is defined using a least squares fit. The fitting procedure is initialized with the minimal uniform closed knot vector describing a cubic B-spline curve, $k = \{0, 0, 0, 0, 1, 1, 1, 1\}$. Iterative refinement of the final curve is determined by comparing measured distances between $\tilde{\mathcal{E}}_i$ and $\mathcal{S}$, made at the nodal values. For a cubic spline curve, the nodal vector is defined as $\tilde{k} = \{\tilde{k}_i\}_{i=0}^{N-3}$ and a node values are computed as $\tilde{k}_i = (k_i + k_{i+1} + k_{i+2} + k_{i+3})/4$.

Distance comparisons are made by evaluating the two points p_e and p_s at the parameter values between each consecutive pair of nodal values, $u = (\tilde{k}_i + \tilde{k}_{i+1})/2$ on $\tilde{\mathcal{E}}_i$ and $\mathcal{S}$ respectively. While evaluation of p_s is well defined through B-spline evaluation, the approximation of $\tilde{\mathcal{E}}_i(u)$ is the weighted average of the subcloud points,

$$p_e = \frac{\sum_j \theta(|u_{ij} - u|) \cdot e_{ij}}{\sum_j \theta(|u_{ij} - u|)},$$

where, the weighting function is $\theta(x) = e^{-x^2/\alpha^2}$ and α is a user defined fall-off term. If the distance between p_e and p_s is greater than a defined tolerance, $\|p_e - p_s\| \geq s_{max}$, the step size used during the polyline growth, then the nodal value is inserted into a new knot vector k_{new}. The spline curve is re-fit to the subcloud using the new knot vector after all distance comparisons are evaluated. This iterative refinement is terminated when no new parameters are inserted into k_{new}. The refinement of the spline curve converges towards a locally weighted average of the RMLS projected point cloud $\mathcal{E}$; which, in practice accurately annotates the feature edges.

3.3 Applications of Feature Curves

The feature extraction algorithm outputs a set of continuous geometric curves that identify the sharp features of a point set surface. The generated curves highlight important components of the unstructured cloud, facilitating a better understanding of the

underlying model. This section illustrates the benefits of applying the feature extraction algorithm as a preprocess for several subsequent point based geometry processing techniques, including surface segmentation, compression and reconstruction. The effectiveness of the feature extraction algorithm is demonstrated on machine parts, where the sharp edges are well-defined, and all data sets consist only of a set of 3D points, without associated normal vectors and connectivity information.

3.3.1 Surface Segmentation

Surface segmentation algorithms dissect a model into multiple regions that each describe a common component section, and describe a useful geometric processing tool. While several techniques exist, specifically for point based models [175, 178], this approach differs in that it does not require the computation of surface approximations, normals, nor other differential properties. In contrast, this segmentation algorithm divides the model into multiple feature-aligned components using only Euclidean distance computations. The method operates in three stages: (1) identify boundary points based on proximity to input feature curves, (2) iteratively run a region growing algorithm to segment the cloud into multiple groups, (3) merge boundary points into the nearest segmented groups.

For a given point cloud $\mathcal{P}$, the algorithm begins by flagging all points $p \in \mathcal{P}$ within a given radius r of a feature spline curve. The radius r can be user specified or automatically computed for each model based on the sample density, i.e., the average sphere radius needed to describe k points. The points of $\mathcal{P}$ can be divided into two groups classified as boundary (flagged) $\mathcal{B}$ and unvisited $\mathcal{U}$ points. The following algorithm will organize $\mathcal{B}$ and $\mathcal{U}$ into multiple feature-aligned visited point groups $\mathcal{V}_i$ that define the segmentation of the model.

The segmentation process seeds an advancing front region growing technique at a random point $p \in \mathcal{U} \subset \mathcal{P}$. A queue, $\mathcal{W}$, representing the advancing wavefront, is initialized with the neighborhood points of p within the distance r. The growth of the region occurs by processing the top point of $\mathcal{W}$, p_t, until no more points remain within the wavefront. If the point p_t is classified as unvisited, p_t is removed from $\mathcal{U}$ and inserted into the new segmenting region $\mathcal{V}_i$. The unvisited neighborhood points of p_t, within distance r, $\mathcal{N}(p_t) \in \mathcal{U}$, are appended to $\mathcal{W}$. When the wavefront $\mathcal{W}$ is exhausted, no new unvisited neighbors exist for the segmenting region. The advancing front algorithm is re-seeded at a new unvisited point until $\mathcal{U}$ is empty. Following the segmentation of $\mathcal{U}$ into multiple

visited groups $\mathcal{V}_i$, the method resolves boundary points in $\mathcal{B}$. Each boundary point b is inserted into the group $\mathcal{V}_i$ that contains the closest visited point to b.

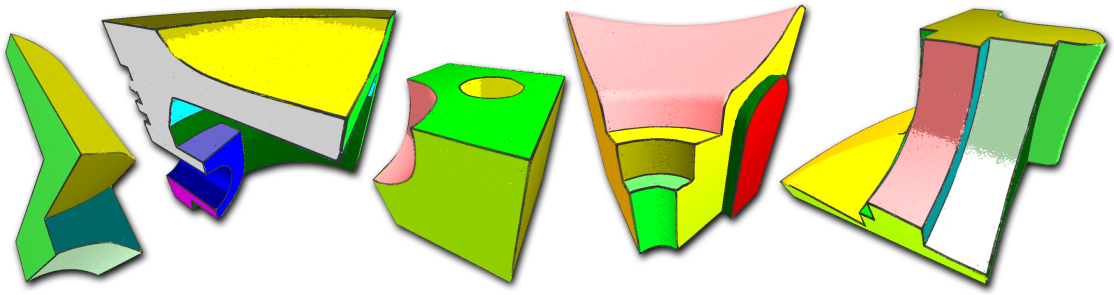
Using the precomputed feature curves, this point based algorithm is able to generate feature-aligned segmentations of the input cloud. These results are illustrated for multiple models in Figure 3.8, where the splats are oriented using MLS based normals evaluated for each segmented region $\mathcal{V}_i$ as if it were an individual point cloud. In this way, the MLS projected points and associated normals no longer smooth sharp edges, instead reconstructing the feature normals. In addition to feature-aware visualizations and MLS based projections, the surface segmentation results aid in the several applications for point based, including compression and surface reconstruction.

3.3.2 Surface Compression

Surface compression is an elementary problem in geometry processing, the goal of which is to represent 3D models compactly for the purpose of space efficient storage and fast transmission. The task of the encoder is to transform a specific surface representation into a compact bit stream, which can be decoded at the receiver side in order to reobtain the original model or an approximation of it, if the encoding is lossy. While polygonal meshes may compress point locations and connectivity data, point based models contain reduced information, being only a collection of connectionless 3D data points.

Point based compression schemes must cleverly encode the point locations, for instance, by using indicators to aid in the prediction of new 3D locations. For instance, it is possible to approximate the original model by an iteratively refined base data set; where, at each step, the position of the inserted point is guided by a local displacement normal vector that is encoded by a single scalar [51]. A second approach constructs a binary tree structure for the point set that is utilized to predict point positions at each level of the tree based on previous levels [170]. The residuals are encoded using a one-dimensional (1D) variant of the *set partitioning in hierarchical trees* (SPIHT) encoder [129].

A highly successful sheme, [117], compresses point based models by decomposing the input surface into a number of patches that are parameterized as height fields over a planar reference plane and resampled on regular grids. The resulting images with arbitrary regions of support are encoded using state-of-the-art wavelet compression of shaped images. In this method, the surface segmentation is based on a split-merge partitioning scheme: first, recursively subdividing the surface until all patches meet a

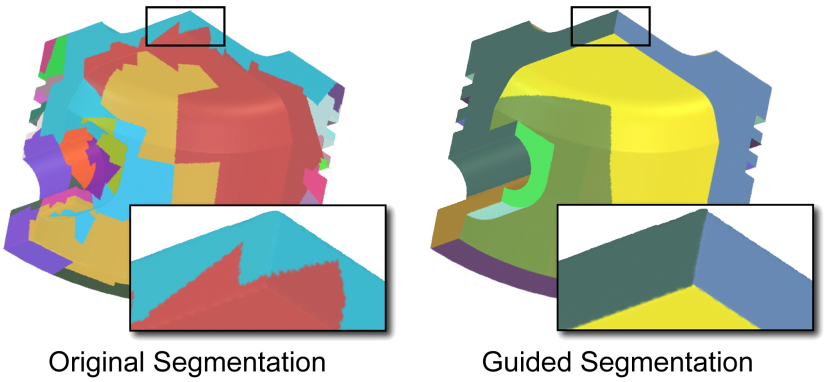


3.8: The segmentation algorithm divides the point-based model into multiple feature aligned regions. MLS projected splats upsample each region independently to reconstruct sharp feature normals on the model.

prescribed flatness constraint, then merging adjacent patches that maintain height field properties.

Although this method yields a high rate-distortion performance, the approximation quality largely depends on partition structure. As illustrated in Figure 3.9, the partitioning scheme, especially on models with sharp features, constructs segments of the model that contain features rather than identify these important structures as region boundaries. The drawback of this behavior is that sharp features within a patch corresponds to high frequency components in the corresponding wavelet signal that is hard to encode. To overcome this problem, the original partitioning procedure may be guided by the previously discussed segmentation results of Section 3.3.1. The encoder receives the input point based model with the pre-partition information that had been generated using the the feature edges. The splitting phase in [117] is performed on each individual patch of the pre-partition, while the merge operations occur in their usual setting. By guiding the partitioning scheme, the resulting segmentation is feature aligned where the individual patches show an improved structure and more regular boundaries, as illustrated in Figure 3.9.

The compression performance for the Fandisk and Quarter Piston models are illustrated in Figure 3.10 with results shown in Figure 3.11, each with $2 \cdot 10^5$ points. The original models are shown, followed by two decoded models at different bit rates. For the fandsik model heavy compression artifacts are only observed at extremely low bit rates (0.08 bits per point (bpp)), and slightly increasing the bit rate leads to significantly improved geometric reconstruction quality. The quarter piston model requires more bits (0.26 bpp) to achieve adequate reconstruction quality, due in part to the more complex



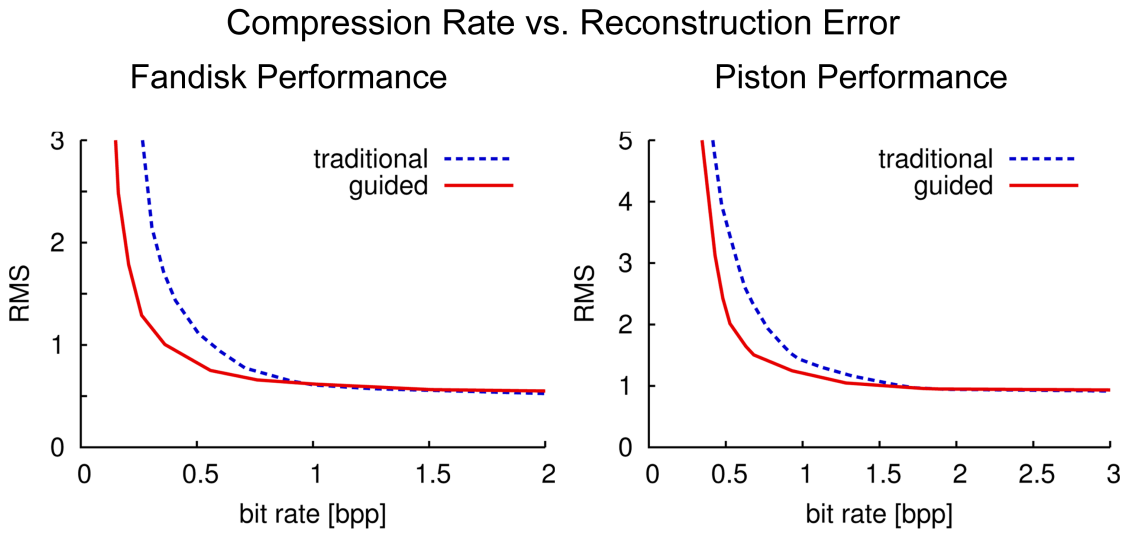
3.9: A comparison of the traditional segmentation method for an existing point based compression method and guided results using extracted feature curves for the Quarter Piston model.

geometric features of this model. An increased bit rate (0.48 bpp) yields a reconstruction that is close to the original model.

3.3.3 Surface Reconstruction

Triangle meshes are a popular representational form of surface models that are commonly used in geometry processing. The related algorithms are able to leverage the additional information inherent in mesh models as compared to point set surfaces, i.e., connectivity, to compute discrete differential operations. The robustness and efficiency of these algorithms are dependent on the quality of the input mesh, there is a demand for high quality mesh structures that motivates surface construction and remeshing algorithms.

In previous reconstruction research, [137], extending [134, 136], an advancing-front algorithm is shown to produce high quality remeshes from many different types of source models, including point-set surfaces. This approach is based on defining a *guidance field*, constructed using local curvature information that determines the size of the output triangles. Precomputing a specific guidance field allows the triangulation scheme to adapt to areas of high curvature, shrinking the allowable triangle size as the advancing fronts grow towards such regions. However, because the algorithm uses MLS for curvature computations and projections of the remesh vertices, the method adaptively shrinks the triangle sizes near sharp features and produces rounded edges. Extending the projection and guidance field computations by using RMLS enables reconstructions with sharp

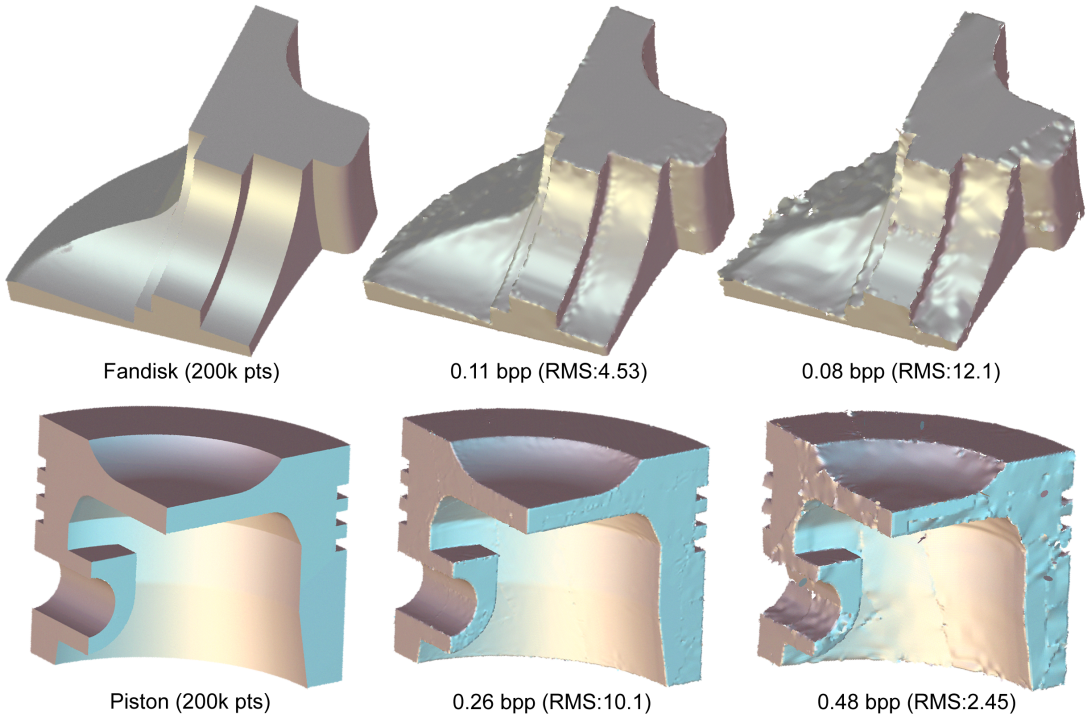


3.10: The guided partitioning improves rate-distortion performance of the compression method as compared to the traditional segmentation for the Fandisk and Piston; the error is measured as the root-mean-square (RMS).

features; however, as previously discussed, this creates jagged edges with expensive computational costs.

In response to this problem, observe that the inclusion of the feature curve information enhances the understanding of the point set surface's underlying structure to facilitate feature-aware reconstructions, illustrated in Figure 3.12. The extracted feature curves, with corresponding RMLS-defined feature normals, are used as the initial advancing fronts for the remeshing system. Without any modifications to the original algorithm, the method [137] is able to reconstruct sharp features. Furthermore, because the original projection routines and other computations remain unaffected, the computational performance is equivalent.

It is important to note that the feature-guided segmentation process is able to improve the computation of a better guidance field. Previously sharp features compute large MLS based curvature values that resulting in many small elements within these regions (Figure 3.12). In contrast, the new guidance fields may be computed over the segmented regions indepdently, so that the elements remain large near the sharp features. Consequently, the surface reconstruction results are greatly improved by leveraging the knowledge gained from the extracted feature curves, obtaining both feature-alignment and reduced element counts.

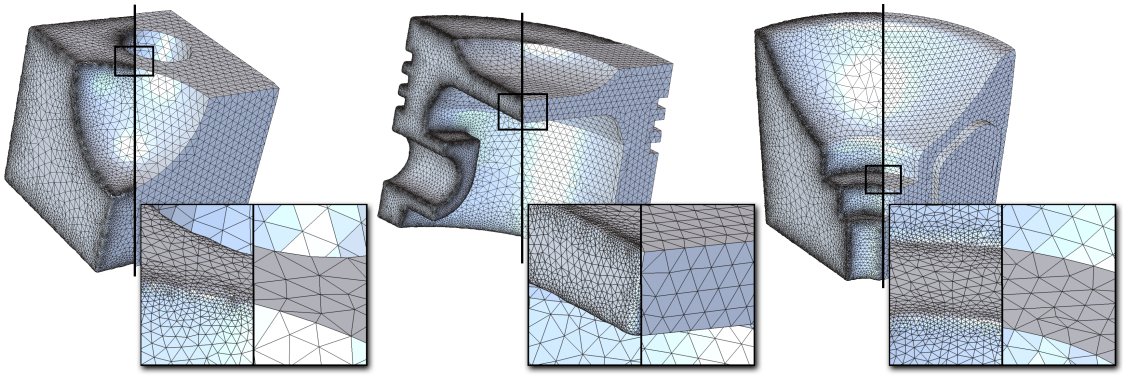


3.11: The guided partitioning compression results of the Fandisk and Piston models at several levels of compression; the error is measured as the root-mean-square (RMS).

3.4 Experimental Results and Analysis

The robustness to noise, accuracy of the extracted curves, and run time efficiency of the feature extraction method are further analyzed in this section. Figure 3.13 illustrates extracted feature lines for the piston and chess piece models at different noise levels. In this image, uniform noise is applied to a quasi-uniform point sampling of a triangle mesh, shifting each point p by a random vector $\delta(p)$ of restricted length, $\|\delta(p)\| \leq l$. The length l is the noise level, expressed in units of the bounding box diagonal length d_B of the original model. A ground truth representation of the sharp feature edges is constructed using a thresholding test to compare a user-tuned tolerance angle with the angle of separation between normals of the triangles adjacent to each mesh edge.

For low levels of uniform noise, the feature extraction procedure successfully defines smooth feature curves, as illustrated in Figure 3.13. Despite introducing a $0.02d_B$ uniform noise on the two models, the algorithm extracts a majority of feature curves within error tolerances of the *true* sharp edges. For higher levels of noise, $0.05d_B$, differentiation between the uniform noise and feature edges becomes difficult, and weak features, with

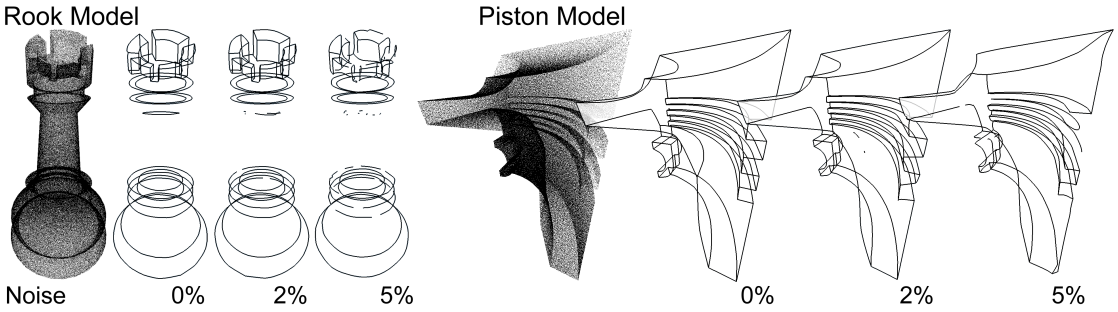


3.12: Results obtained by applying an advancing front remeshing technique to several point set surfaces. The left half of each model illustrates the original method [137], and the right half shows results guided by the extracted feature curves.

near planar defining surfaces, are not clearly extracted by the algorithm.

Further accuracy measurements are presented in Table 3.1 for the models shown throughout this chapter. Ground truth feature edges are semiautomatically constructed from mesh models used to generate the experimental point clouds. The analysis computes the mean and max distance information for the feature extraction results by projecting the extracted feature points onto the nearest ground truth feature curves. On average, the distance between the extracted features and ground truth is very small ($\leq$ 0.5% of the bounding box diagonal d_B). The percentage of extracted feature curves is also reported within Table 3.1, emphasizing the success of the feature extraction algorithm to properly annotate a majority of the sharp edges on the point set surfaces. This percentage is computed by measuring the ratio of ground truth feature curve segments with an extracted feature curve within a distance of $0.01d_B$.

The total time required to construct the polylines is determined by the number of *potential* feature points in data set. Implemented in C++ on a 2.16 GHz Intel Core 2 Duo with 2GB memory, the feature extraction run times are reported for each algorithmic stage in Table 3.1. These results process half a million points within a few minutes, where the major computational cost is attributed to the RMLS projection procedure (stage 2) that is highly parallelizable.



3.13: Feature extraction from the chess piece and piston models; features extracted from the original model, and models with progressively increased uniform noise ($\|\delta(p)\| \leq 0.02d_B$ and $\|\delta(p)\| \leq 0.05d_B$, where d_B is the length of the model bounding box diagonal).

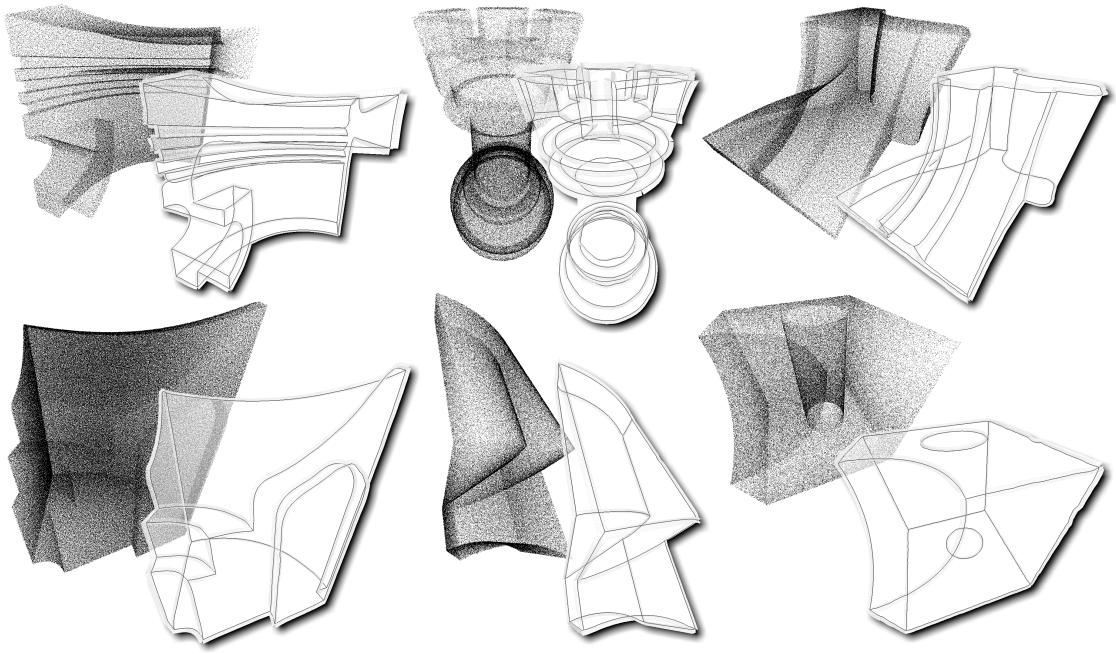
3.1: Run time performance of the feature extraction algorithm for models (some with specified uniform noise, each containing $5 \cdot 10^5$ points). The pipeline stages correspond to: point identification (1), point projection (2), curve propagation (3), curve completion (4), and spline fitting (5). The analysis includes the mean and max distance from the extracted features to a ground truth, as well as the percent of the features extracted.

Model ($\|\delta(p)\|$)	(τ, $\|\mathcal{F}\|$)	Run Time [seconds]					Analysis	
		(1)	(2)	(3)	(4)	(5)	Dist. (Mean, Max)	Extract
Fandisk	(0.013, 55.7k)	191	192	11.5	0.8	0.0	($0.0013d_B$, $0.0089d_B$)	99%
Hook	(0.006, 49.6k)	207	196	0.7	0.6	2.3	($0.0005d_B$, $0.0049d_B$)	100%
Piston	(0.006, 80.3k)	146	207	0.3	0.7	9.9	($0.012d_B$, $0.19d_B$)	99%
($\leq 0.02d_B$)	(0.017, 97.9k)	196	698	1.3	0.7	68	($0.0044d_B$, $0.25d_B$)	97%
($\leq 0.04d_B$)	(0.030, 82.2k)	169	552	18.9	1.9	29	($0.0049d_B$, $0.18d_B$)	90%
Ra	(0.012, 40.8k)	159	116	1.0	0.0	1.6	($0.0013d_B$, $0.025d_B$)	99%
Rook	(0.011, 99.6k)	216	471	5.8	3.0	8.8	($0.0022d_B$, $0.046d_B$)	99%
($\leq 0.02d_B$)	(0.030, 112k)	250	840	21.7	12	19	($0.0025d_B$, $0.055d_B$)	97%
($\leq 0.04d_B$)	(0.034, 124k)	296	1337	19.0	18	23	($0.0042d_B$, $0.057d_B$)	92%
S-Cube	(0.0175, 33.6k)	186	324	2.5	0.3	5.6	($0.0025d_B$, $0.025d_B$)	100%
Knuckle	(0.0095, 70.1k)	176	348	10.3	1.3	7.5	($0.0010d_B$, $0.026d_B$)	98%

3.5 Discussion and Limitations

This chapter presents an algorithm for extraction of feature curves from point-sampled surfaces, leveraging the robust statistical methods of RMLS to project points to all possible features. The resulting curves are accurate and do not require the point based acquiring techniques to explicitly sample the sharp features. Illustrated in Figure 3.14 are extracted feature curves and corresponding RMLS-defined surface normals for multiple point based models with varying feature types.

The robust feature extraction produces smooth results, it is limited by extremely poor



3.14: The feature extraction results for multiple point based models.

sampling quality and by time constraints. High noise levels or sparse sampling worsens the performance of the method because the distinction between sharp features and noisy regions becomes ambiguous. The RMLS projection routine is computationally expensive, restricting the utility of this feature extraction method as a preprocess.

Feature identification proves valuable, as the generated curves augment the performance of subsequent geometric processing algorithms. Most relevant to the over-arching theme of this dissertation, the feature curves facilitate feature-aligned surface reconstructions. Furthermore, the defined spline based curves constructed from the point based model address a thematic research topic that re-occurs throughout this dissertation related to the interaction between continuous and discrete geometry.

CHAPTER 4

QUADRILATERAL MESH SIMPLIFICATION

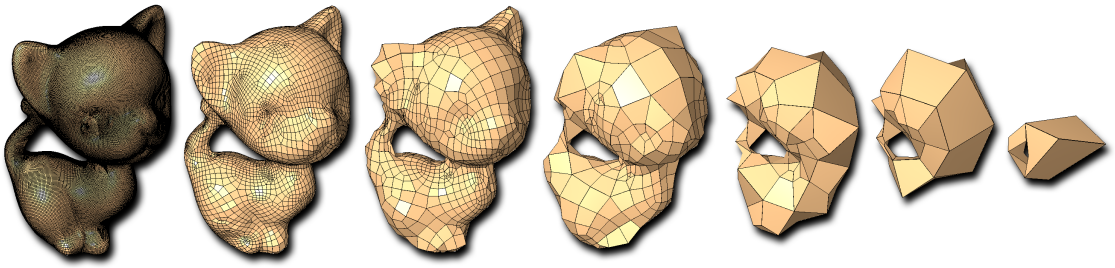
Mesh simplification is an important geometry processing operation that has been used as a building block for many higher level processing steps, including mesh compression, rendering, progressive transmission, editing operations, smoothing, parameterization, and shape reconstruction. It is for this reason that triangle mesh simplification techniques have been some of the most useful operations developed.

A major challenge associated with quadrilateral simplification, unlike triangle-based techniques, is the consideration of the structured nature of the quadrilateral elements that force global constraints on the mesh connectivity. For triangle meshes it is possible to limit the attention to local operations, that is, to collapse an edge in a triangle mesh, one only needs to consider the triangles in its one-neighborhood. In contrast, the deletion of a single quadrilateral element may require more global consideration by removing a larger collection of elements in order to maintain highly structured results [35, 152, 40].

This chapter details important observations pertaining to the quadrilateral-based simplification while addressing the challenging problems of maintaining the all-quadrilateral connectivity and mesh topology, illustrated in Figure 4.1. This discussion describes a progression of quadrilateral-based simplification techniques from deletion methods designed to operate on the dual representation to localized operators that adapt vertex sample densities in a controlled manner, comparing the advantages of the different algorithms in terms of semiregular and irregular quadrilateral mesh inputs. The simplification results are applied to remeshing polygonal models into quadrilateral-only meshes, demonstrating an effective and robust connectivity-based approach to reconstruction.

4.1 The Dual Representation

The dual representation is an important structure of a quadrilateral mesh [21] that is useful in mesh analysis and processing. The dual representation is defined to have the



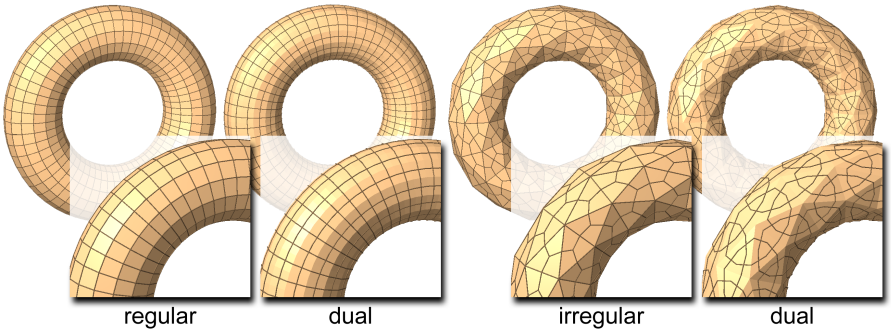
4.1: The simplification algorithm can be used to generate a pure quadrilateral level-of-detail hierarchy. The algorithm preserves topology during simplification, and attempts to optimize geometric fidelity and quadrilateral structure (vertex valences near 4) throughout the process.

following components: the dual of a quadrilateral is its *centroid*; the dual of a quadrilateral edge is the *chord* that connects the centroids of the neighboring quadrilaterals; the dual of a vertex is the *polygon* formed by connecting the centroids, in a cyclic order, of the neighboring quadrilaterals.

An important higher order structure is the *polychord*, a polyline whose adjacent segments are chords that meet at a common centroid and are dual to opposing edges in that quadrilateral. On a closed quadrilateral mesh without boundaries, the polychords always form closed loops. That is, starting at a single edge on a closed quadrilateral mesh and traversing opposite edges on adjacent quadrilaterals, the traversal will always end at the starting edge. The polychord is a generalization of the *ring* structure constructed in related quadrilateral-based research [24, 40], and is related to the *dual sheet* associated with hexahedral-based meshing [151] for which the looping property is proven.

In order to sharpen our intuition, consider the dual structure of the two different quad representations of a torus shown in Figure 4.2. A high quality quadrilateral mesh of a torus (shown on the left) with all vertices having valence four has a similar dual structure with chords that tend to meet at approximately right angles and polychordal loops that exhibit low curvature, and polygons that are dual to the vertices are generally rectangular (i.e., four-sided). The lower quality quad mesh (shown on the right) has chords intersecting with nonright angles, polychordal loops that exhibit higher curvature, and the polygons that are dual to the vertices of the mesh are typically nonrectangular (i.e., typically, are not four-sided).

The dual representation is a powerful tool, useful in assessing well-behaved quadrilateral mesh surfaces in addition to assisting geometric processing. In addition to polychord



4.2: The dual structure of a highly structured mesh is itself highly structured with low geodesic curvature (left); whereas, a nonstructured mesh contains a complex dual representation, nonquadrilateral dual faces and high curvature polychords (right).

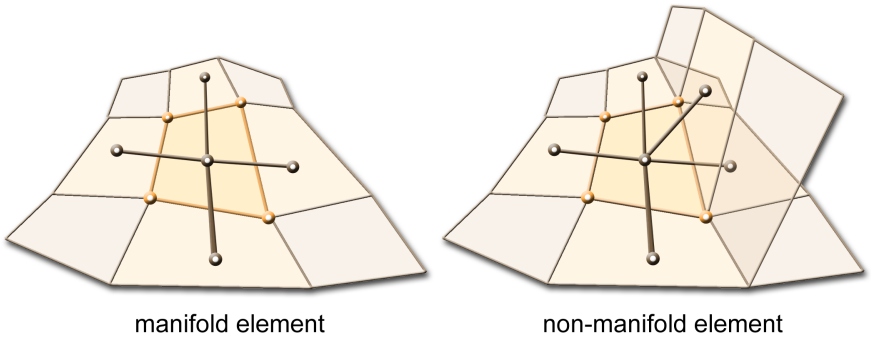
curvature observations, the dual structure can identify well-behaved surface models. For instance, as illustrated in Figure 4.3 each quadrilateral of a closed (without boundaries) two-manifold will have exactly four *unique* dual chords. This property implies that every pair of adjacent quadrilaterals will have at most one chord connecting their centroids, thus sharing at most one edge. Furthermore, this property prevents nonmanifold edges, where more than two quadrilaterals share a common edge, as these elements would contain additional chords.

4.2 Global Quadrilateral Simplification

Many simplification techniques for triangle meshes are based on edge-collapse operations. Unfortunately, if an edge is collapsed in a quad structure, triangles are introduced and structure is lost. This research adapts ideas proposed in [144] that observed, in the context of hexahedral mesh generation, that the quality of a hexahedral mesh is related to properties of its dual structure. As related to mesh simplification, the dual structure is used as a tool for designing deletion operators.

4.2.1 Deletion Operators

The following discussion outlines a set of deletion templates used to modify the dual structure of the quadrilateral mesh in order to improve its overall quality during mesh coarsening. The *polychord collapse* describes a global operator, simultaneously removing multiple quadrilaterals during each execution, while the *quadrilateral vertex merge* and *doublet removal* induce modification of the dual structure through localized deletions.



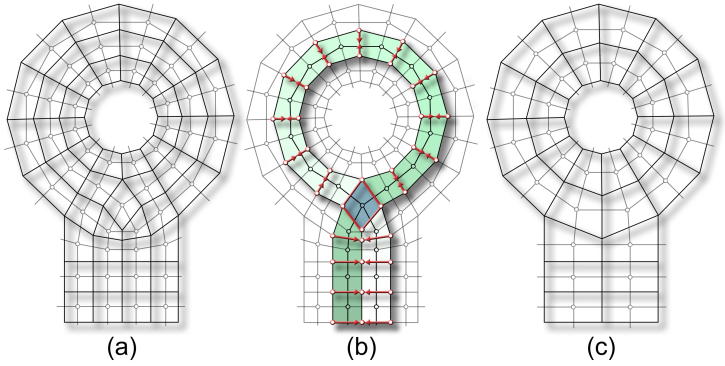
4.3: The dual structure is useful in mesh analysis: a well-behaved quadrilateral-only mesh defines four *unique* chords with each face primitive (left), while nonmanifold edges are detected by additional dual chords (right).

4.2.1.1 Polychord Collapse

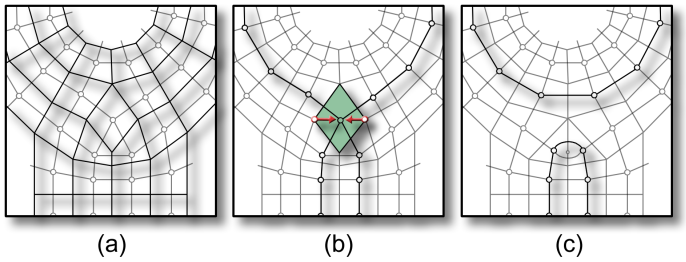
Removal of a polychord from the dual representation simultaneously deletes all quadrilaterals through which it passes by merging the vertices of each edge dual to the selected polychord, as illustrated in Figure 4.4. A polychord, as it is related to hexahedral meshing, is equivalent to the intersection of a hexahedral dual sheet [114] with the boundary surface. Because removal of hexahedral sheets has been shown to preserve the hexahedral connectivity during mesh improvement [21], it follows that the lower dimensional equivalent, polychord deletions, preserve quadrilateral-only surface elements.

4.2.1.2 Quadrilateral Vertex Merge

The quadrilateral vertex merge (*qvMerge*) is a localized deletion operator that removes a selected element by merging diagonally opposing vertices (also described as a quadrilateral close [87]), illustrated in Figure 4.5. By imagining the quadrilateral as two virtual triangle elements connected by an edge between the merging vertices, observe that the described *qvMerge* is a generalization of the triangle edge collapse. The elements modified by a *qvMerge* are limited to a local neighborhood and maintain the quadrilateral-only constraint. Note the modifications that result to the connectivity of the dual polychords, observing that the deletion operation describes a swap in the connectivity of two polychords.



4.4: The original mesh and its dual representation (a), the chosen polychord (b) and the resulting mesh after the polychord collapse (c).



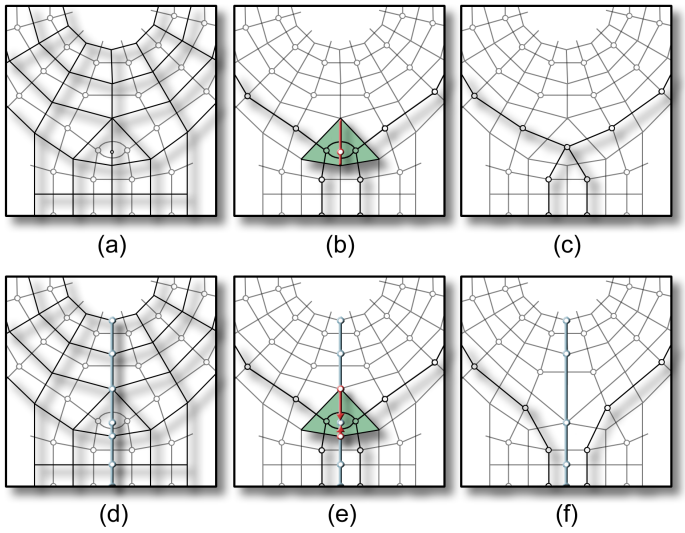
4.5: The original mesh and its dual representation (a), the chosen quadrilateral (b) and the resulting mesh after the quadrilateral-vertex merge (*qvMerge*) (c). The *qvMerge* describes a swap in the underlying dual polychords, highlighted above.

4.2.1.3 Doublet Removal

Doublets, identified as neighboring quadrilaterals that share two consecutive edges, are removed from the quadrilateral mesh following each collapse operation. The valence 2 doublet vertex describes a *degenerate extraordindary point*, associated with a degenerate dual polygon. Under normal circumstances, a doublet is removed by merging the two faces into a single quadrilateral element modifying the connectivity of the highlighted dual chords, illustrated in Figure 4.6. The removal of a doublet may generate new doublets, requiring additional deletions, and, similar to the quadrilateral collapse, modifies the structure of two polychords.

4.2.2 Prioritizing Operations

To improve the mesh connectivity and maintain geometric fidelity, it is important that the algorithm selects the elements for deletion intelligently. The prioritization of



4.6: The original mesh and its dual representation (a), the chosen doublet (b) and the resulting mesh after the doublet collapse (c). When the doublet edge is an annotated feature (d), the doublet quadrilaterals (d) are simultaneously merged to preserve the feature edge (f). Similar to the *qvMerge*, the doublet collapses describe swaps in the dual polychord structure.

the collapse operations is achieved by queuing the elements based on the impact of the deletion on the resulting mesh. A combination of factors is considered by the weighting scheme, including the quality of the final vertex valences, the geometric loss, and the area distortion associated with each collapse operator. Consequently, the prioritization scheme is able to reduce the accumulated geometric error while improving the element quality during sequential deletions.

The polychord collapse is a more complicated operator than the quadrilateral deletion so the weighting scheme discussion focuses on this operator. The error metric E assigned to a polychord p is

$$E(p) = \alpha_q(1 - e^{-E_q(p)}) + \alpha_d(1 - e^{-E_d(p)}) + \alpha_v(1 - e^{-E_v(p)})$$

where αs are user defined positive scalars that sum to 1. This function considers multiple contributors, written as $1 - e^x$ to scale the terms to the range $(0, 1]$ with a Gamma-like distribution, that weigh the impact of the deletion with a purpose similar to [150]. Deletions that evaluate with a maximum E, equal to 1, are given priority over lower values. While other types of distributions and functions have been tested, this priority metric yielded the most improved results in simplification experiments, measured as the

percentage of ideal vertices. The user defined α's allow control of the influence of each term.

The function $E_q(p)$ returns the worst case QEM error over all of the groups of merging vertices for the polychord p, representing an upper bound on the geometric impact of the collapse. QEM matrices are computed for the mesh vertices by virtually subdividing each quadrilateral into four triangular elements connected at the quadrilateral centroid. This allows for a hinging effect to accommodate for nonplanar quadrilaterals while accumulating the planar equations for the original mesh vertices using the established triangle-based QEM scheme. The function $E_d(p)$ evaluates the length of the longest edge collapse (or group of edges) due to the collapse of the polychord p. The purpose of E_d is to prioritize collapses in order to construct square elements.

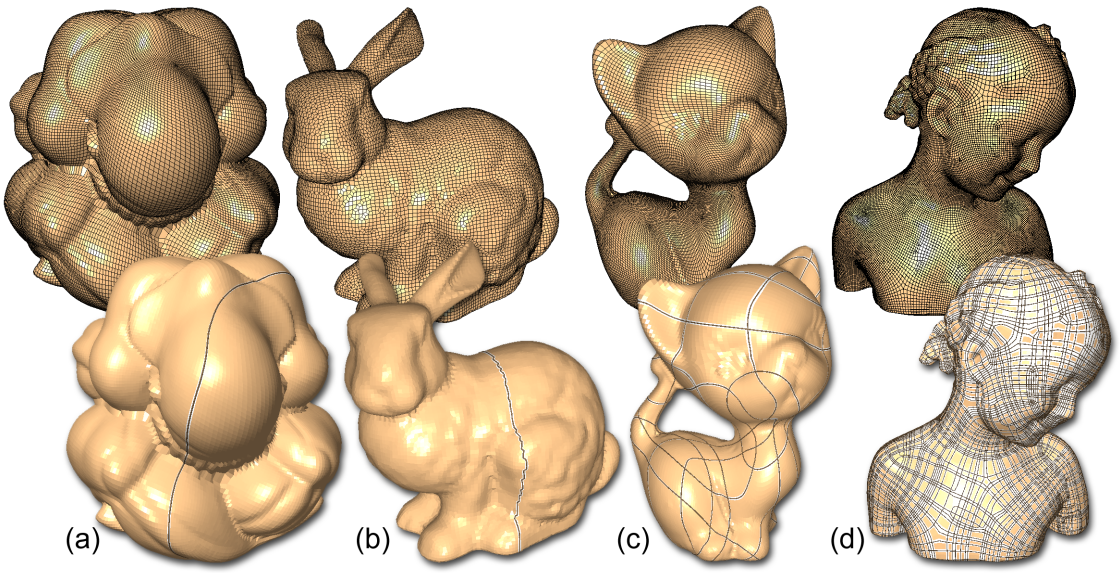
The final weighting term, $E_v(p)$, measures the change in vertex valences from the ideal 4 and penalizes polychord collapses that deteriorate the valences of neighboring vertices. The polychord p is decomposed into multiple vertex groups $\{V_i = \{v_{i,j}\}_{j=0}^{N_i}\}_{i=0}^{M}$ that merge to a single new vertex $\tilde{v}_i$, illustrated in Figure 4.4(b). The valence weighting term sums the worst created valence with the average,

$$\begin{aligned} E_v(p) = &\max_i \max_j \beta_{i,j}(|\nu(v_{i,j}) - 4| - |\nu(\tilde{v}_i) - 4|)) \\ &+ 1/M \sum_{i=0}^{M} 1/N_i \sum_{j=0}^{N_i} \beta_{i,j}(|\nu(v_{i,j}) - 4| - |\nu(\tilde{v}_i) - 4|)) \end{aligned}$$

$$\beta_{i,j} = \begin{cases} 0, \text{ if } |\nu(v_{i,j}) - 4| \leq |\nu(\tilde{v}_i) - 4| \\ 1, \text{ otherwise} \end{cases}$$

where $\nu(v)$ returns the valence of v. Thus, E_v penalizes polychord collapses that result in nonideal vertices while sorting the deletions with equivalent worst cases based on their average created valence.

The weighting metric assigned to each quadrilateral element for the quadrilateral collapse is computed similarly as $E(p)$, allowing the user to vary the influence of the geometric-, area-, and valence-based functions. For a quadrilateral q with vertices (in counter clockwise order) a, b, c, d, the QEM measurement $E_q(q)$ associated with a quadrilateral collapse is based on the loss of shape fidelity induced by merging the chosen opposing vertices, for example, merging a and c to $\tilde{v}_{ac}$. The area term $E_d(q)$ measures the distance between the two merging vertices of q.



4.7: A *single* polychord is highlighted for the above models. Some semiregular meshes (a) and irregular models (b) exhibit a polychord structure that lends itself to deletions. However, for many other semiregular (c) and irregular meshes (d), the polychords form complex knots over a significant portion of the mesh, complicating, and often invalidating, their collapse.

The final weighting term, $E_v(q)$, measures the difference between the current vertex valences and the new configuration and is biased to favor collapses that improve the local connectivity. The valence term sums the difference of the created valences from the originals,

$$\begin{aligned} E_v(q) &= \beta_a(|\nu(a)-4| - |\nu(\tilde{v}_{ac})-4|) + \delta_b(|\nu(b)-5| - |\nu(b)-4|) \\ &\quad + \beta_c(|\nu(c)-4| - |\nu(\tilde{v}_{ac})-4|) + \delta_d(|\nu(d)-5| - |\nu(d)-4|) \\ \beta_i &= \begin{cases} 0, & \text{if } |\nu(i)-4| \leq |\nu(\tilde{v})-4| \\ 1, & \text{otherwise} \end{cases} \\ \delta_i &= \begin{cases} 0, & \text{if } |\nu(i)-5| \leq |\nu(i)-4| \\ 1, & \text{otherwise} \end{cases} \end{aligned}$$

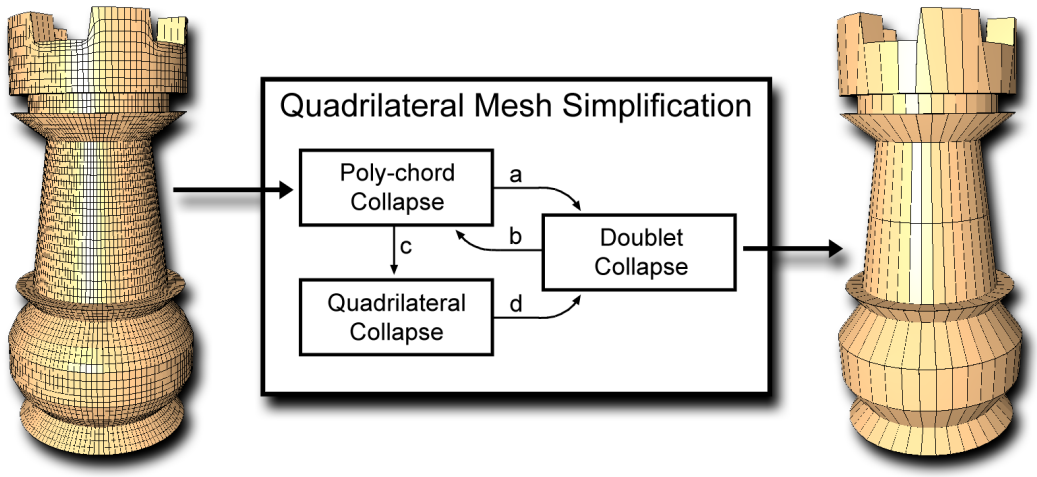
Thus, E_v penalizes quadrilateral collapses that lower the quality of the local vertex valences. Note that for a given quadrilateral two collapse configurations exist, each with distinct metrics. Finally, doublets are given the highest priority and collapsed when detected.

4.2.3 Naïve Simplification

The repeated iteration of any single collapse operator, while ideal in some circumstances, proves inadequate when applied to a broader spectrum of quadrilateral mesh connectivity. The polychord simplification scheme creates an ideal reduction of some semiregular meshes, further discussed in Section 4.2.6, that maintains high quality elements at all levels of detail. However, for irregular, and some semiregular meshes, where the polychords are not well structured, their deletions create too many high valence vertices and poorly shaped quadrilaterals. Furthermore, simplification that relies solely on polychord deletions may terminate early, unable to generate lower resolution models, because of structural complexities. As illustrated in Figure 4.7, even for well structured models, dominated by ideal vertices, the polychords may exhibit a *knotting* effect, winding over a significant portion of the model, that prohibits further deletions.

On the other hand, naïve simplification using on the *qvMerge* operator is not impeded by the dependence on the dual structure. Given irregular meshes, the weighting scheme prioritizes the collapses such that the number of ideal vertices significantly improves through simplification, further discussed in Section 4.2.6. However, this technique increases the number of nonideal valence vertices undesirably during simplification of (semi)regular quadrilateral meshes.

4.2.4 Simplification Algorithm



4.8: The QMS state diagram.

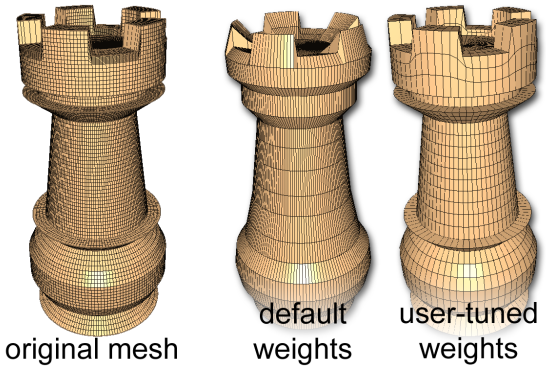
The novel quadrilateral mesh simplification algorithm, coined *QMS*, balances the global polychord and local *qvMerge* operators to achieve a flexibility that generates high quality results independent of the structure of the mesh connectivity. As shown in the state diagram, illustrated in Figure 4.8, the algorithm favors the polychord collapse, iteratively choosing the deletion that best improves the mesh's connectivity while least affecting its geometry (looping over state transitions a and b). When no further polychord collapses are available, bounded by threshold tolerances, the algorithm selects a quadrilateral element for deletion (state transition c). The *qvMerge* modifies the local neighborhood's adjacencies and alters the dual polychord structure. Consequently, the algorithm returns to the polychord collapse state (via state transitions d and b), to inspect the deletion eligibility of the newly configured polychords. Doublets are removed after each collapse (state transitions a and d), as opposed to implementing a final pass, because experiments indicated a better control over the element count and quality of the final mesh.

The state transition from the polychord collapse to the *qvMerge* enables the adaptive nature of *QMS*. This approach relies on the *qvMerge* to *unwind* complex knots that prevent polychord collapses, illustrated in Figure 4.7, to continue simplification. Furthermore, given irregular meshes, the algorithm quickly transitions to the *qvMerge* to avoid the poor quality results associated with polychord deletions.

4.2.4.1 Alpha Terms

For the simplification results shown throughout this chapter, mesh connectivity is the dominant factor of the sorting metrics ($\alpha_v = 0.9$, α_q, $\alpha_d = 0.05$). With these settings, the collapses are sorted primarily by the quality of the valences created during the deletion, greedily selecting the best possible vertex valence configuration after each iteration. The small weights given to the QEM and area metrics allow similar valence generating collapses to be sorted based on their geometric impact.

The weighting factors of the prioritizing function offer the user some control to allow tuning of the simplification performance, necessary in pathological cases. For instance, illustrated in Figure 4.9, the rook model is defined by rotating a curve about an axis, resulting in a latitude-longitude based connectivity with two polar points. For this model, the valence-based weighting scheme described earlier deletes all longitudinal polychords prior to any of the latitude oriented polychords, due to high errors associated with the



4.9: The rook model is defined by rotating a curve about an axis, resulting in two polar points. The default weights, emphasizing the valence term $\alpha_v = 0.9, \alpha_q = \alpha_d = 0.05$ (consistent with all other simplification results) generates a poorly simplified mesh. Instead, user tuning the weights for such pathological cases, $\alpha_v = 0.05, \alpha_q = \alpha_d = 0.475$, constructs a better simplification hierarchy.

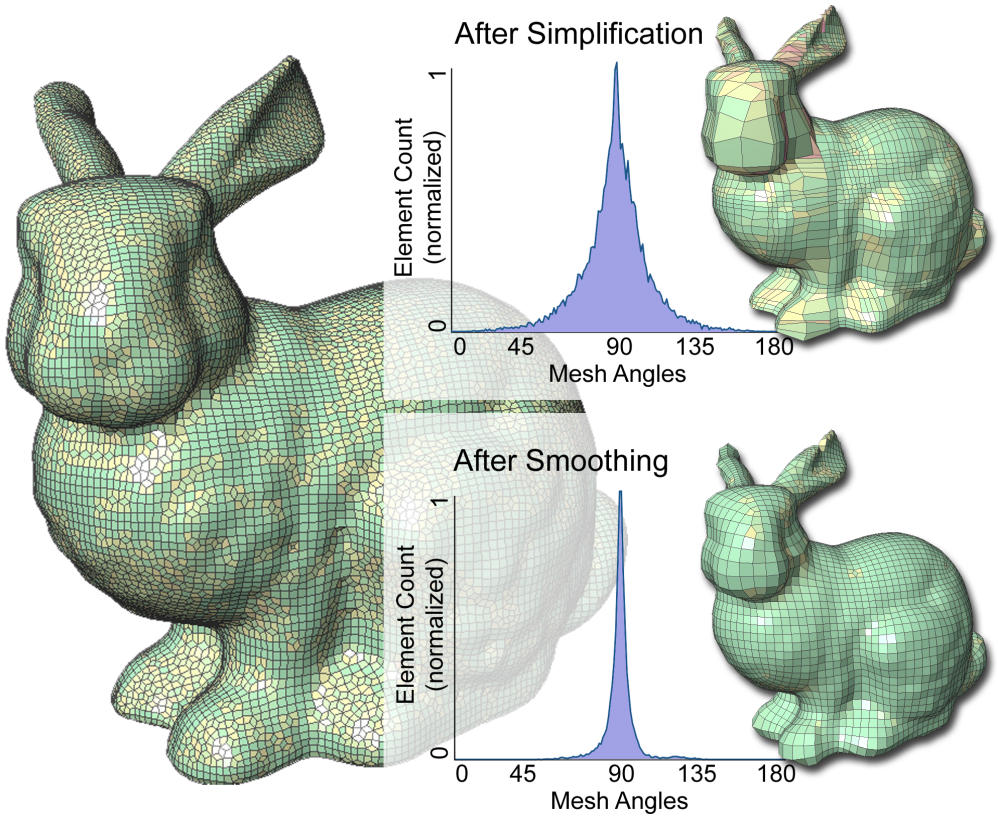
polar points. This creates many elongated rectangles, encouraging, instead, a weighting scheme that evenly splits the QEM and area terms with little or no emphasis on the valence information ($\alpha_v = 0.05$, α_q, $\alpha_d = 0.475$).

4.2.4.2 Quadric Error Metrics

To preserve geometric fidelity, *QMS* extends the well known quadric error metric (QEM) to assist in relocating simplified vertices to reduce the error incurred by each operation. The initial QEM matrices are computed for the original mesh vertices by *virtually* splitting each quad into four triangles emanating from the dual centroid. In this manner, this extension accommodates for the nonplanarity of quads while using the existing triangle-based QEM computations. The initial collapse vertices are computed by minimizing the QEM metric to reduce geometric error.

4.2.4.3 Mesh Smoothing

The QEM computations associated with each collapse are used primarily to influence the new vertex locations to better maintain the surface geometry. However, as illustrated in Figure 4.10, these vertex locations may not create high quality quadrilaterals. To improve element quality, a posteriori smoothing procedure described by [180] modifies vertex locations applied after the simplification has finished. The mesh vertices are perturbed in the normal direction to reduce mesh noise while preserving the mesh volume;



4.10: The original model, 22k quadrilaterals with 50% ideal vertices, is reduced to 5k quadrilaterals with 96% ideal vertices. The diagrams compare angle distributions (ideal 90°) before and after the smoothing process.

simultaneously, a tangential movement is included to improve the quad aspect ratios. Note that one reason to prioritize valence 4 vertices in the alpha parameters is that it augments element quality obtained through smoothing.

4.2.4.4 Topology Preservation

To preserve the topology of the manifold mesh, the quadrilateral mesh may be *virtually* defined as a simplicial mesh to use the Link condition [47]. In the neighborhood of a quadrilateral collapse, the quadrilaterals are divided into two triangles by inserting a cross diagonal edge radiating from the merging vertices, equating its link condition discussion to that of the triangular edge collapse. Because the polychord collapse consists of multiple simultaneous edge contractions, the discussion is more complicated.

Deletion of the polychord p executes multiple edge collapses simultaneously, merging

the vertex pairs V_i into the new vertices $\tilde{v}_i$ for each mesh edge dual to p. Where the polychord is adjacent to itself or self intersects, illustrated in Figure 4.4, multiple vertex pairs V_i will merge into a single vertex; however, for this discussion, each edge is inspected individually. While inspecting a vertex pair V_i, the remaining vertex pairs are temporarily replaced with their collapse vertex $V_k = \tilde{v}_k, \forall k, k \neq i$, to account for the multiple edge collapses occurring. The remaining quadrilaterals sharing the vertices in V_i are *virtually* divided into two triangles by inserting the cross diagonal edge originating from one of the vertices in V_i. The mesh topology is preserved during polychord deletions by comparing the intersection of Link conditions associated with the one-neighborhoods of the vertices in V_i to the union of those associated with the one-neighborhood of the mesh edge connecting V_i. For further details, we refer the reader to [47].

4.2.4.5 Complexity

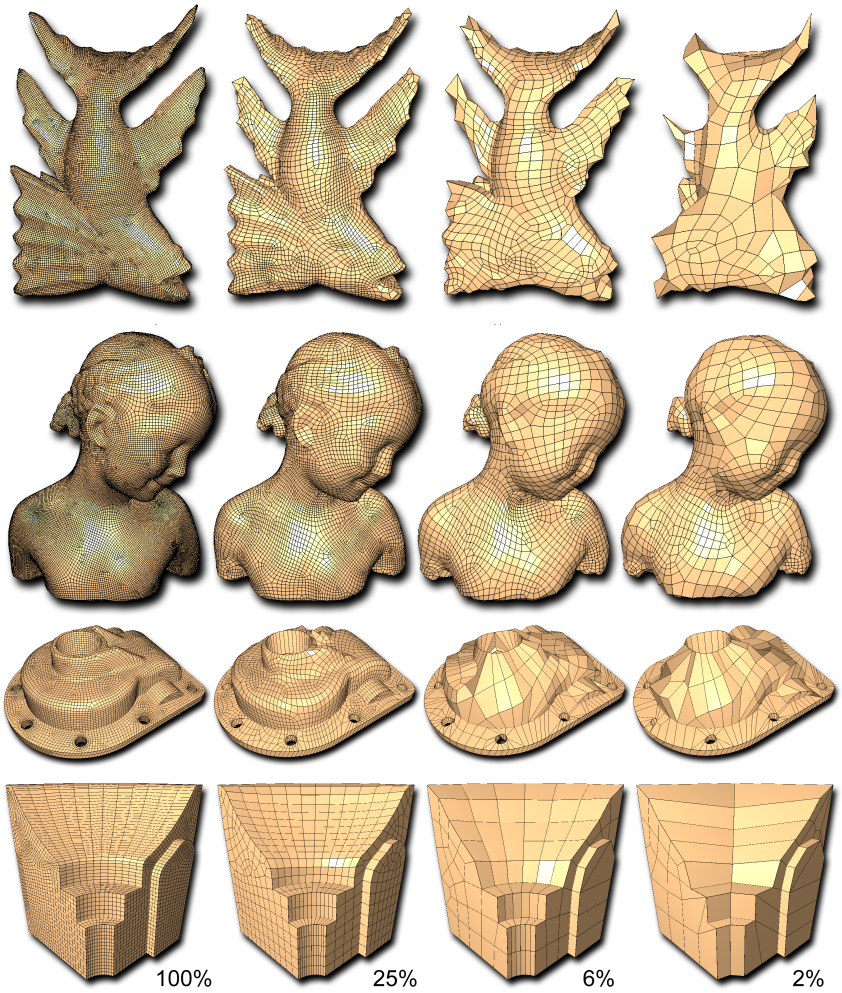
The complexity of a single call to the various operators differ. While quadrilateral and doublet collapse operators take $\Theta(1)$, a single polychord operation can take $\Theta(n)$. Still, it is not hard to use amortized analysis to show that the overall cost of each operation is still $\Theta(1)$. The argument is based on the fact that once an element is deleted, it is not touched again.

4.2.4.6 Sharp Features

In some applications, the preservation of user annotated feature edges is important. For this reason, *QMS* considers annotated edge sequences, maintaining their topology during simplification. Feature preservation is illustrated in Figure 4.11 for two manufacturable models containing user annotated features aligned to the sharp edges of these meshes.

The *QMS* implementation ensures that the simplified feature edges are homeomorphic to their original counterparts. Conceptually, triangular facets are created by connecting each feature edge to a temporary transition vertex. In this way, the feature edges belong to three polygonal faces, thus defining *degree* three edges. The remaining edges of the two-manifold quadrilateral mesh are degree-two, shared by only two quadrilateral elements. The Link condition supports the preservation of the non-two-manifold edges to ensure topological isomorphism of the annotated features.

The preservation of feature edges defines additional constraints that restrict the simplification operators and invalidate some potential collapses. While this is terminal for the



4.11: The *QMS* algorithm generates controlled level-of-detail representations for quadrilateral-only models while preserving the topology of the mesh and annotated features.

polychord and quadrilateral collapse operators, we define a special case to handle doublet deletions along feature edges. Illustrated in Figure 4.6(d-f), when the shared edges of a doublet are annotated as part of a feature edge, the two quadrilaterals are simultaneously collapsed. If the synchronized quadrilateral collapse special case maintains the topology of the mesh and features edges, the doublet is removed from the mesh.

4.2.5 Experimental Results

The QMS algorithm, as described in the previous section, is implemented in C++. This code is written with flexibility in mind, with no particular effort given to optimiza-

tions for speed at this time. The running times reported in this section were performed on a dual core AMD Opteron 2.21 GHz processor with 4GB memory. The code is robust, and the implementation has been tested on a large of quadrilateral meshes, including those shown in this paper.

In addition to the visual quality of the simplified models as shown in Figure 4.11, Table 4.1 measures several metrics to assess the quality final simplification results of a variety of quadrilateral meshes, including the connectivity of the simplified meshes as well as the quality of the quadrilateral elements. An ideal vertex, important in finite element meshing, is valence 4. Nonideal vertices are *extraordinary points* that complicate parameterization solutions and geometric computations. Included in Table 4.1 is the percentage of ideal vertices, the number of extraordinary points, and the worst case valence for each mesh.

To further investigate the quality of the quadrilateral elements, Figure 4.12 plots a histogram of the original and simplified element angles and scaled Jacobians of the models examined by the table. The scaled Jacobian is a metric equal to 1.0 for a rectangular element, 0.0 if three vertices (of the four) are colinear, and negative if the quadrilateral is not convex. Ideally, the angles of a quadrilateral mesh are near 90° and the scaled Jacobians near 1.0, corresponding to orthogonal corners in the mesh.

As shown in Table 4.1, the *QMS* algorithm generates high quality connectivity, maintaining low worst case valences and reducing the number of extraordinary points. For irregular quadrilateral models, i.e., the bumpy torus and the Pensatore[2], the simplification results significantly improve the percent of ideal valence vertices. Simplification of semiregular quadrilateral models, i.e., the Pensatore[1] (Figure 4.13) and Ra models (Figure 4.11), generates high quality results. Observe that in some cases, the number of extraordinay points increases on the semiregular models, i.e., the Bimba and casting models. This is due to a number of quadrilateral collapses executed in order to unwind complex polychords (Figure 4.7).

The improved connectivity augments the quality of the corresponding quadrilateral elements, plotted in Figure 4.12. Despite the quality of the input models, the *QMS* algorithm results in simplified meshes with good quality angles and scaled Jacobians. Some degradation is inherent, similar to the geometric error incurred during the deletions. Importantly, the element quality of the two models with poor original distributions in Figure 4.12, corresponding to the bumpy torus (used as a remeshing example in

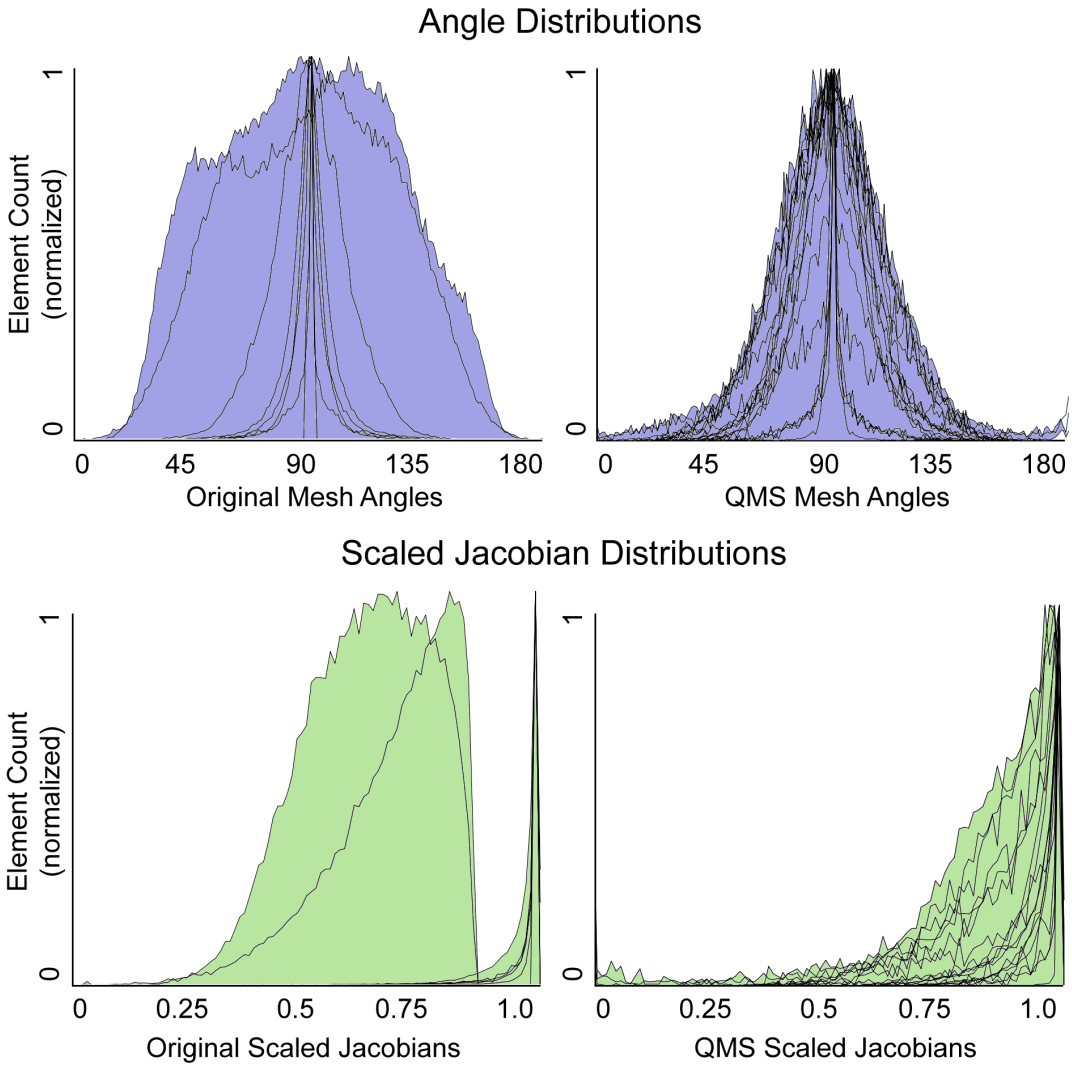
4.1: Performance and vertex valence analysis (percentage of ideal vertices, number of extraordinary vertices, and worst case valence) of models shown throughout the paper: the semiregular and irregular pensatores (^{1,2} respectively), bimba, wooden fish, casting, ra, and the bumpy torus models.

|Quads|	Time (sec)	Vertex Valences (% Ideal, |Ex|, Worst)	|Quads|	Time (sec)	Vertex Valences (% Ideal, |Ex|, Worst)
Pensatore^1 (Figure 4.13)			Pensatore^2 (Figure 4.14)		
44k	n/a	(99.9%, 8, 3)	46k	n/a	(50%, 23k, 10)
5.2k	52	(99.9%, 8, 3)	5.7k	157	(76%, 1.3k, 7)
2.5k	53	(99.9%, 8, 3)	2.9k	160	(80%, 556, 6)
Bimba (Figure 4.11)			Fish (Figure 4.11)		
62.8k	n/a	(98%, 726, 6)	32.4k	n/a	(97%, 876, 6)
15.5k	479	(94%, 898, 6)	8k	115	(91%, 678, 6)
7.8k	500	(92%, 600, 6)	4k	119	(88%, 475, 6)
3.9k	505	(89%, 398, 7)	2k	120	(84%, 314, 6)
970	506	(80%, 189, 6)	490	121	(73%, 131, 6)
Casting (Figure 4.11)			Ra Model (Figure 4.11)		
20.8k	n/a	(98%, 405, 6)	3.9k	n/a	(98%, 52, 5)
5.1k	30	(90%, 491, 6)	925	1	(94%, 52, 5)
2.6k	31	(86%, 344, 6)	450	1	(90%, 44, 5)
1.3k	32	(79%, 259, 6)	190	1	(81%, 36, 5)
630	32	(67%, 205, 6)	95	1	(69%, 28, 5)
Bumpy Torus (Figure 4.24)					
95k	n/a	(50%, 47.6k, 13)			
4k	908	(69%, 1808, 6)			
24k	n/a	(92%, 1808, 6)			

Section 4.5) and the irregular Pensatore^2 (Figure 4.14), are improved through the *QMS* processing. There are no similarly bad distributions evident in the simplified histograms. The low quality mesh angles and corresponding scaled Jacobians are attributed to the preservation of feature edges, i.e., the casting and Ra models (Figure 4.11), discussed in the following subsection.

4.2.6 Discussion and Limitations

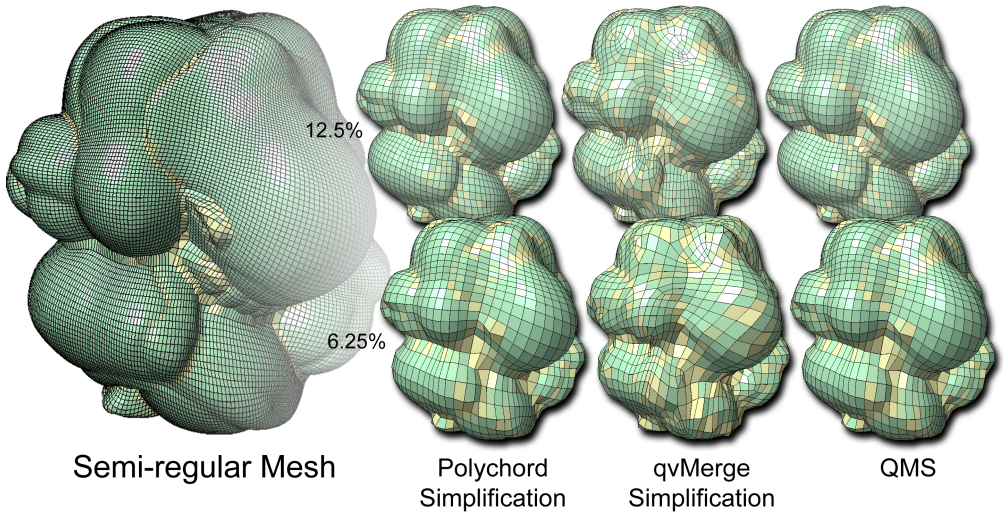
The *QMS* algorithm balances polychord and *qvMerge* operators to generate quality meshes independent of the underlying mesh structure. As illustrated in Figures 4.13 and 4.14, *QMS* is an improved approach over any single operation type. For semiregular meshes, *QMS* behaves similarly to the polychord simplification method, better maintaining mesh regularity with a minimal number of critical points and near orthogonal edges. However, where the polychord collapse performs poorly, terminating early on the irregular mesh, *QMS* instead mimics the results of the *qvMerge*-based simplification scheme. The



4.12: A comparison of the angle distributions and scaled Jacobians measured on the original and QMS simplified meshes shown throughout this paper, analyzed in Table 4.1. The *QMS* algorithm produces angles near 90° and scaled Jacobians near 1.0 despite the input distributions.

assorted deletion types enable a flexibility within the *QMS* algorithm, generating quality LOD meshes for arbitrary mesh structures.

Quadric error metrics are used to reduce the loss of geometric fidelity experienced during the simplification process. Consequently, even when the weighting metrics used to sort the simplification operators heavily emphasize connectivity improvement, the reduced meshes maintain high shape fidelity. Illustrated in Figure 4.11 are several organic and machined models simplified using the QMS algorithm, weighing the deletions solely based

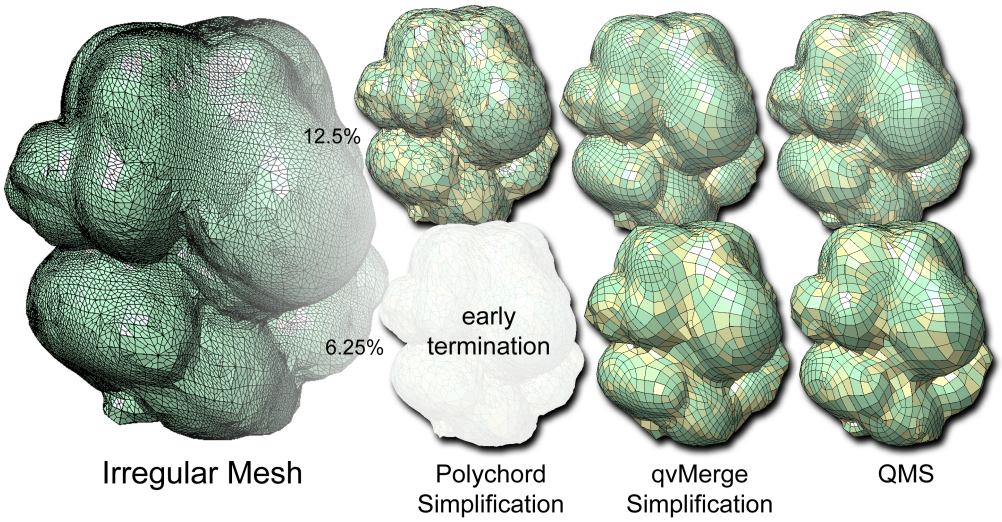


4.13: Simplification results for a *semiregular* mesh using the polychord-, *qvMerge*-based and QMS algorithms. The QMS algorithm behaves similarly to the polychord collapses, maintaining the high quality mesh structure in comparison to the *qvMerge*-based simplification.

on connectivity.

The *QMS* algorithm is unable to discern when too many elements have been deleted from constrained regions surrounding the user annotated feature edges. In practice, simplification to coarse representations generates low quality edge angles, evident in Figure 4.12, sometimes resulting in negative scaled Jacobians, indicating inverted elements. These poorly shaped elements typically occur in regions around the constrained feature edges. While the mesh topology is preserved, the inverted elements may inhibit computations and visualization of the mesh.

When compared to triangle simplification (Figure 4.15), it is harder to maintain small detailed features like the scales of the dragon because of the global nature of the polychord collapses. Furthermore, a motivating factor for the use of quadrilateral elements is based on their natural ability to align to important geometric attributes, i.e., principal curvature directions; yet, the *QMS* prioritization metrics disregard related information during simplification. The remainder of this chapter investigates a localized simplification method aimed at improving adaptive sampling and mesh alignment based on surface geometry and associated user specified attribute data while developing simplification hierarchies with high quality mesh structures.



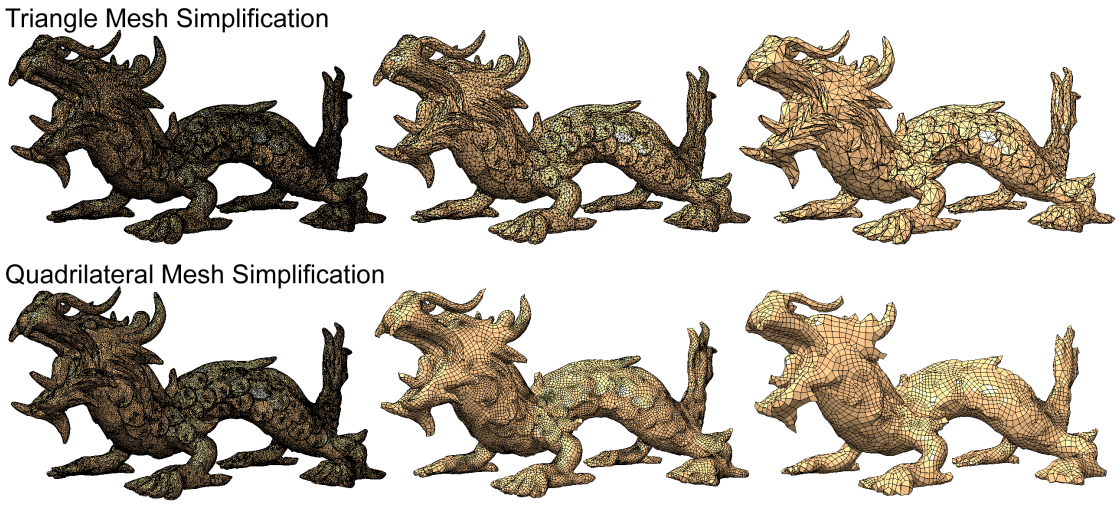
4.14: Simplification results for an *irregular* mesh using polychord-, *qvMerge*-based and QMS algorithms. The QMS algorithm mimics the results of the *qvMerge*-based simplification, improving the mesh structure and element quality of irregular meshes. The polychord simplification is unable to complete due to the complex polychord structures.

4.3 Local Quadrilateral Simplification

It has been well articulated that the polychord collapse is instrumental in developing and maintaining high quality quad meshes. As such, operations related to this dual structure have been useful in quadrilateral-based geometric processing [35, 24, 152]. Unfortunately, when the polychord spans a significant portion of the mesh, its deletion, as described in the previous section, has a global effect.

Globalized coarsening operations have adverse effects on element sizing and preservation of surface attributes, illustrated in Figure 4.15. An important challenge of mesh simplification is the controlled degradation of elements to achieve an adaptive sampling that respects important surface features. In the conventional sense, this translates to the preservation of important surface geometry [56], but also extends to appearance attributes [57, 70], i.e., color and material properties. In contrast to triangle mesh approaches, quad mesh algorithms may also have the added complexity of achieving element alignment to orthogonal vector fields defined over the surface, i.e., principal curvature directions. However, to my knowledge, no quadrilateral-based simplification scheme considers these properties.

The remainder of this section introduces an algorithm that simultaneously addresses the challenging problem of developing attribute aware meshes while producing high



4.15: The *QMS* algorithm generates controlled level-of-detail representations. The figure above provides a visual comparison between QEM-based triangle mesh simplification and my method for a dragon model with 150k elements (left) at 37.5k (middle) and 9.4k elements (right).

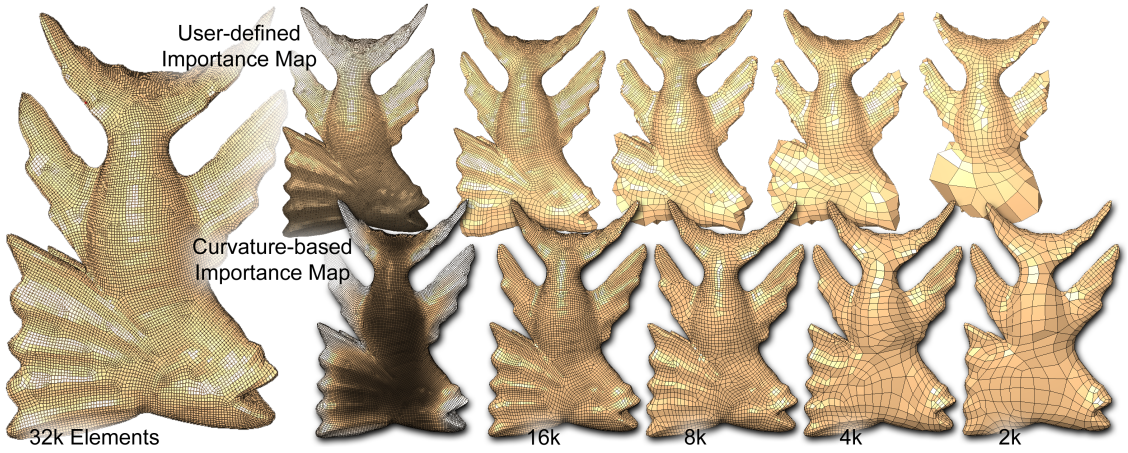
quality quadrilateral-only connectivity at all levels-of-detail, illustrated in Figure 4.16. In this image, the wooden fish model is coarsened based on the associated importance map attributes, where darker elements correspond to lower weights. The mesh hierarchy contains well shaped elements, with few extraordinary vertices, and vertex sample density and mesh edge alignment based on the input attribute data.

4.3.1 Deletion Operations

In addition to the previously described *qvMerge* (Section 4.2.1.2) and the doublet removal (Section 4.2.1.3), the localized quadrilateral mesh simplification method introduces a novel deletion template. The purpose of this deletion operator is to locally reproduce the effects of the polychord collapse. Discussed in more detail in Section 4.4, this set of localized deletion operators better enables attribute aware mesh simplification hierarchies, with quality results on both irregular and (semi)regular quadrilateral meshes.

4.3.1.1 Quadrilateral Edge Merge

The quadrilateral edge merge operator (*qeMerge*) merges two adjacent quads to their shared edge then remeshes neighboring elements to maintain quadrilateral-only connectivity. Merging two quads to a shared edge generates an even number of triangular elements (either 0, 2 or 4). The remeshing phase, illustrated in Figure 4.17, is able to remove the



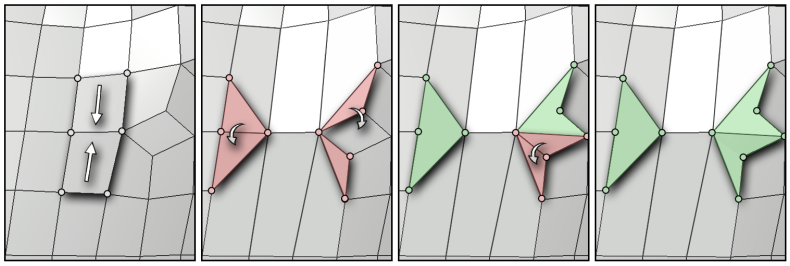
4.16: The *qCoarsen* algorithm is a localized quadrilateral mesh simplification that generates quadrilateral-only mesh hierarchies, sensitive to attribute data, i.e., quality elements scaled to user defined (top) or automatically computed curvature-based (bottom) attributes.

triangles by examining the edges that emanate from the endpoints of the merged edge. Similar to the wandering edge swaps that occur in Delaunay triangulations [172], the process can be described as rotating the edge shared by a triangle and quad until two triangles are adjacent and combine to form a single element. The remeshing is limited to the one-ring neighborhood of the merged edge and is executed once for each pair of triangles, so that the *qeMerge* deletes 4 elements with each execution.

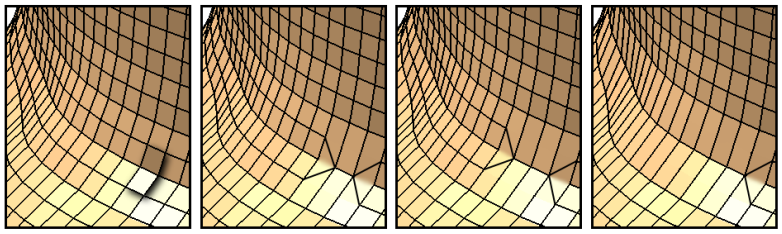
This localized deletion method can reproduce the results of the polychord deletions (Figure 4.18). The *qeMerge* initiates a *zipper-like* effect that locally reproduces the deletion results of two parallel polychords. In this way, the method may be used on structured models to maintain high quality connectivity. But, unlike the polychord deletion, each iteration is localized to avoid the propagation of the deletion by terminating the *zippering* effects as determined by the surface attributes, i.e., important geometry detected as high curvature.

4.3.2 Algorithmic Inputs

The localized coarsening algorithm, coined *qCoarsen*, requires an input quad mesh and (optionally) annotated feature edges, importance map, and alignment vector data. In the absence of user defined importance and vector attributes, the principal curvature magnitudes and directions are approximated over the discrete model [7] by virtually



4.17: The *qeMerge* operator generates an even (0, 2 or 4) triangles, that are resolved by a subsequent remeshing phase.



4.18: The *qeMerge* followed by multiple *qvMerge*rs executes a *zippering* effect that iteratively reproduces the polychord collapse.

subdividing each quad into four triangles. In practice the best results are observed when the scalar importance values are normalized and scaled over the range [1.0,3.0]. The simplification process is guided by the attribute data, preserving the topology of feature edges while generating simplified models that scale element sizes based on the importance map and improve element alignment to the given vector field.

4.3.3 Prioritizing Operations

To improve and maintain high quality mesh connectivity while obtaining attribute awareness, it is important to intelligently select the elements for deletion. Simplifying the state diagram in comparison to the previously described *QMS* algorithm, *qCoarsen* implements a single priority queue to sort the *qeMerge* and *qvMerge* operations based on the impact of each deletion on the resulting mesh. The weighting function, E, influenced by [150], balances the quality of the vertex valences with sensitivity to importance attributes,

$$E(\cdot) = (\frac{V_{before}(\cdot)}{V_{after}(\cdot)} + \alpha)^2 (D(\cdot) + \alpha)^2 (\frac{A_{before}(\cdot)}{A_{after}(\cdot)} + \alpha)^2,$$

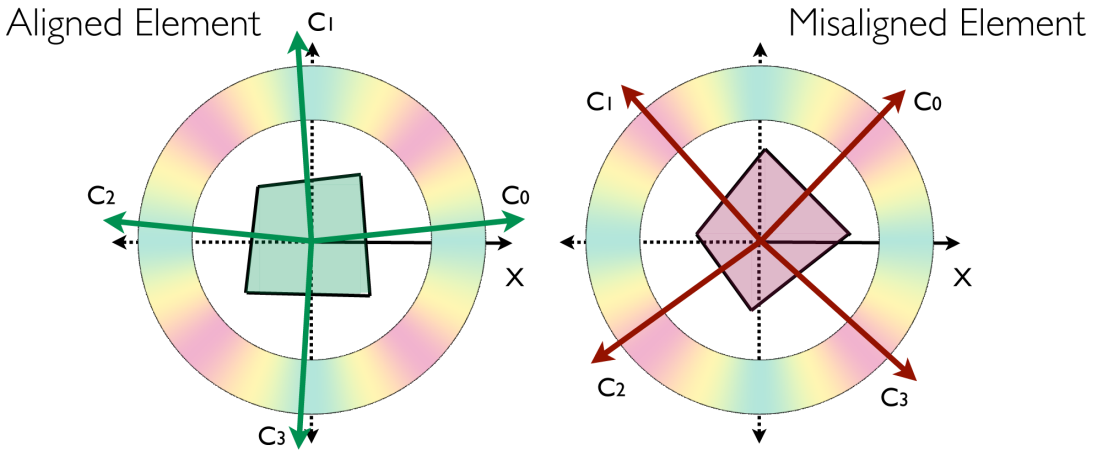
where V measures the ratio of ideal vertices to total vertices, before (V_{before}) and after the deletion and doublet removals (V_{after}); D evaluates the average distance between the merging vertices weighted by the importance scalars; and, A measures the quad alignment to the attribute vector fields, before (A_{before}) and after (A_{after}) the deletion. The constant α (in practice 0.01) is added to each term to ensure that the error metric is greater than 0, critical in the absence of importance or vector alignment data.

The ratio of before versus after ideal valences improves the mesh connectivity. For a quad mesh, the ideal valence is 4 and nonideal vertices are *extraordinary* as they complicate parameterization solutions and subdivision schemes. This term contributes lower error values for collapses that improve the mesh structure while penalizing those that lead to the deterioration in quality of the connectivity.

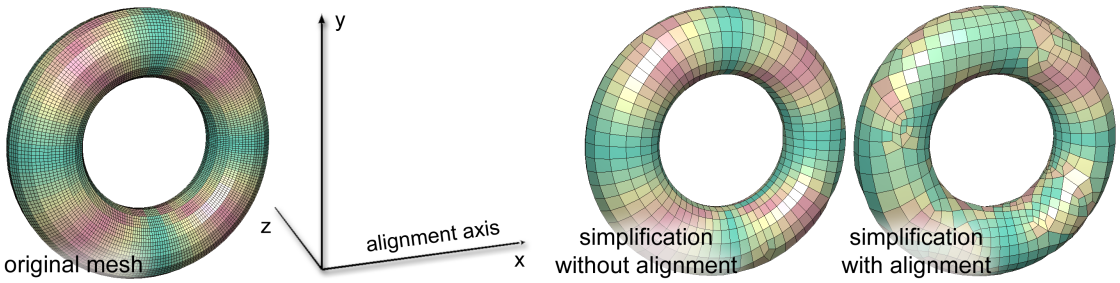
The distance importance term D causes the element gradation illustrated in Figure 4.16. For the *qeMerge* operation, D is the average of the distances between the two pairs of merging vertex groups with the midpoint of their respective group measured as a percentage of the bounding box diagonal. D is weighted by the importance attribute scalar assigned the quadrilateral. In contrast, for the *qvMerge* operation, D measures the importance weighted distance percentage between the two merging vertices. In this way, D attributes low errors for elements with small area and elements with low importance attributes.

The alignment term A generates low error values for collapses that improve the element alignment to associated attribute vector fields. A_{before} measures the average angle of separation prior to the collapse, modulo 90°, between the images of dual chords of the deleting quadrilaterals and their associated attribute vectors projected on the normal planes evaluated at the quadrilaterals' centroids, illustrated in Figure 4.19. A is evaluated as $\sum_{i=0}^{3}(2\langle x, c_i\rangle^2 - 1)^2$, where x is the attribute vector assigned the element, and c_i is the image of the dual chord across the i^{th} edge on the tangent plane defined by the quadrilateral's normal evaluated at its centroid. Well aligned elements are assigned a metric near 1, while misaligned elements are near 0. For the *qeMerge* this consists of the two quads being removed, while the *qvMerge* considers a single element. A_{after} computes the similar metric based on the angle of separation between the edges to which the quadrilaterals collapse with the attribute vectors.

The effects of attribute vector fields are illustrated and emphasized in Figure 4.20 for simplification of the torus model. The elements are colored based on their alignment,



4.19: *qCoarsen* measures element alignment based on the angle of separation between the assigned attribute vectors and the dual chords. A well aligned element is oriented where its sides reflect the associated vector frame, while a misaligned element is rotated by 45°.



4.20: *qCoarsen* weights attribute vector field alignment. This proof of concept example exaggerates the effects by forcing alignment to an unnatural vector field (the x-axis projected onto the surface).

computed as previously described, to a constant vector field in the x-direction. Such a vector field is not typical, but is useful in illustrating the effects of the alignment on the simplification results. In Figure 4.20, *red* designates misaligned quads with angle of separation between the dual chords and assigned attribute vectors near 45°; and, *green* represents well aligned elements, corresponding to angles measuring 0° or 90°. Simplification without vector field alignment is achieved by assigning null attribute vectors to the mesh.

To prevent the creation of many extraordinary vertices with high worst case valences,

a conditional term is added to E. If the worst case valence count after a collapse is greater than the worst case before, $V^w_{after} > V^w_{before}$, then

$$E(\cdot) = E(\cdot) + \beta \cdot V^w_{after}(\cdot),$$

where β (in practice 100) is a large constant. In this way, large error terms are awarded to collapse scenarios that greatly degrade the quality of the mesh structure. In practice, this conditional worst case valence term effectively reduces the creation of new extraordinary vertices.

4.3.3.1 Mesh Smoothing

To improve the element quality and attribute sensitivity during simplification, a centroidal-based smoothing procedure is executed on the vertices belonging to the quads in the one-ring neighborhood of each collapse. For a vertex v with neighboring quads $\{q_i\}$ corresponding to dual centroids $\{c_i\}$, surface normals $\{n_i\}$ and importance attributes $\{a_i\}$, the new point $\tilde{v}$ is evaluated as the weighted average of the centroids, $\tilde{v} = (\frac{\sum a_i c_i}{\sum a_i})$. This point is projected to the tangent plane, $\sum n_i$, defined at v to better preserve the surface geometry.

Consistent with the *QMS* algorithm, QEMs are used to determine the vertex locations following a deletion to maintain mesh fidelity. While the smoothing operation will modify vertex locations, the associated QEMs remain unaffected. Future deletions initialize the new neighborhood with the QEM-based vertex locations as if previous smoothing did not occur. In practice, this results in quality representations that respect the original mesh geometry.

4.3.3.2 Topology Preservation

During simplification, *qCoarsen* maintains well-behaved surfaces by disallowing collapse operators that generate nonmanifold models. Similar to the *QMS* algorithm, each deletion operation can be validated using Link condition [47]. However, for ease of implementation each collapse is allowed within *qCoarsen*, building an intermediate neighborhood. An analysis obtained by exploring the dual representation of the intermediate connectivity verifies that each quadrilateral contains exactly four unique chords. In such cases, the deletion results in a well-behaved mesh and the intermediate neighborhood is updated within the current mesh.

4.3.4 Experimental Results

The *qCoarsen* algorithm was implemented in C++ and running times reported in Table 4.2 were performed on a 2.16 GHz Intel Core 2 Duo with 4GB memory. In these results, the vertex smoothing operation is performed once at the end of the simplification routine, as opposed to following each deletion step, typically resulting in a 4x speedup. The algorithmic cost of the *qCoarsen* implementation is $O(mk \log n - m)$ for m deletions, k updates per deletion where queue updates are $O(\log n)$ for n quads. Because each iteration of the deletion triggers a two-ring neighborhood update, a pathological mesh may be organized so that this region includes the entire model, thus $k = n$. However, in practice the update neighborhood is much smaller, $k < 40$, where the cost is $O(m \log n)$.

The code has been rigorously tested by constructing mesh hierarchies in the order of a few minutes for various models, illustrated in Figure 4.21 and analyzed in Table 4.2. The table quantifies the quality of the meshes, measuring statistics of the quads' scaled Jacobians; the structure, documenting the percentage of ideal vertices, the number of extraordinary vertices and the worst case valence information; and lastly, the approximation error measuring the Hausdorff distance between the simplified mesh and its original in terms of the bounding box diagonal d_B. The *qCoarsen* algorithm generates well-behaved, homeomorphic surfaces despite the quality of the input mesh; most notably, significantly improving the mesh structure of irregular models while maintaining close approximations of the original surface.

4.4 Simplification Comparison

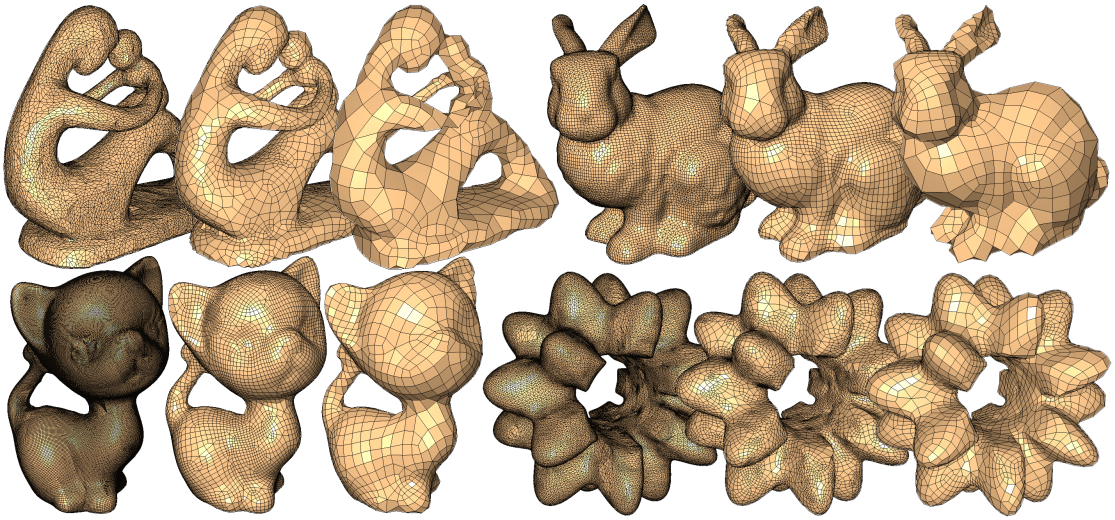
Comparison of the *QMS* results to those of the *qCoarsen* algorithm illustrates the advantages of each approach. The use of a global deletion operator with a valence heavy weighting metric allows the *QMS* algorithm to exploit highly structured dual representations. A subset of semiregular quad meshes, including polycube-based remeshes [169], dual contour surface [24], swept and rotated geometries, some spline-based models, and some quadrilateral meshes generated from morse-based techniques [42], exhibit the necessary structure because the dual polychords exhibit low geodesic curvature while defining concise loops over the model. For these meshes, the *QMS* algorithm generates highly structured models through simplification. At the expense of additional extraordinary vertices, *qCoarsen* generates more accurate simplifications with adaptive sampling characteristics. Furthermore, due to the highly structured nature of the dual

4.2: Simplification results (computation time, mesh quality (Scaled Jacobian statistics), mesh structure (percent of ideal vertices, number of extraordinary vertices, and worst case valence), and error) for various meshes.

	Time	S.Jacobian		Vertex Valence			Error
|Quads|	(sec.)	Median	Worst	Ideal	|Ex|	Worst	(10^{-3})
Wooden Fish (Figure 4.16)							
33k	n/a	0.99	0.44	97%	876	6	n/a
16k	20	0.98	0.3	96%	673	6	$13.9d_B$
8k	30	0.95	0.14	94%	509	6	$11.9d_B$
4k	35	0.92	−0.17	91%	363	6	$20.5d_B$
2k	37	0.91	0.18	89%	220	6	$23.0d_B$
Egea (Figure 4.23)							
27k	n/a	0.63	0.02	51%	13.1k	11	n/a
3k	21	0.91	0.42	80%	592	6	$17.0d_B$
12k	n/a	0.97	0.52	95%	592	6	$10.9d_B$
Stanford Bunny (Figure 4.21)							
22k	n/a	0.92	0.0	56%	9.6k	6	n/a
5k	22	0.96	0.52	93%	337	6	$11d_B$
1k	27	0.93	0.44	88%	120	6	$31d_B$
Fertility Model (Figure 4.21)							
23k	n/a	0.65	0.09	51%	11k	12	n/a
5k	17	0.87	0.22	72%	1396	7	$8d_B$
2k	21	0.84	0.11	81%	366	6	$24.7d_B$

representation, *QMS* is extremely efficient, removing many quads with each pass of the polychord deletion, exemplified by the simplification performance of the pensatore model in Figure 4.22 and Table 4.3. However, as illustrated by the timings for the other simplification examples, this behavior is atypical.

For the remaining quad models, a majority of irregular and many semiregular meshes, where the dual representation does not demonstrate the necessary structure that lends itself to polychord deletions, the advantages of *qCoarsen* are more pronounced. For example, the bimba model, illustrated in Figure 4.22, is generated by the *periodic global parameterization* algorithm [126] and is dominanted by ideal vertices (99%); yet, the extraordinary vertices do not describe a coarse cube-like decomposition of the model so that the polychords describe complex knots, see Figure 4.7. Under such circumstances, *qCoarsen* outperforms *QMS* in every way, documented in Table 4.3, with faster computations, fewer extraordinary vertices, and lower approximation errors (measured in terms of the bounding box diagonal d_B) with adaptive sampling to better describe complex geometric details.

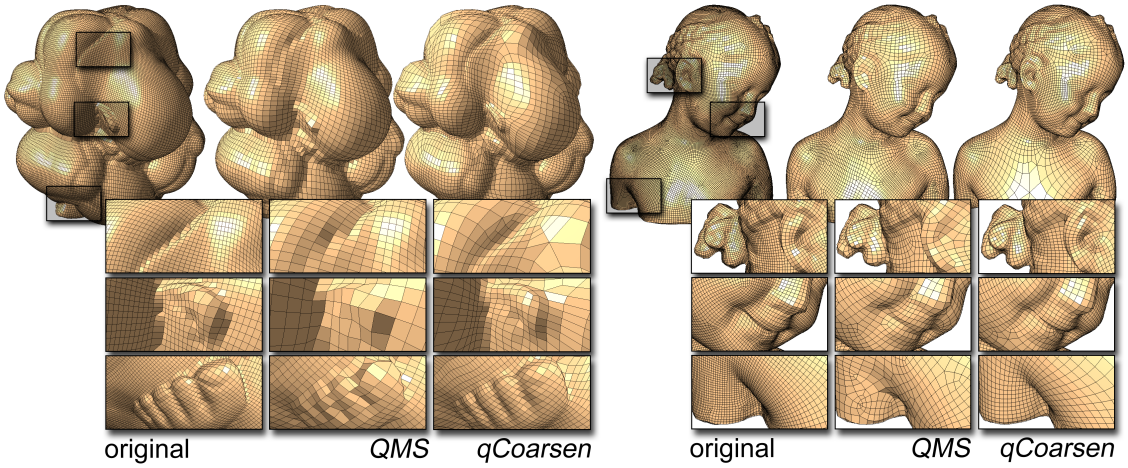


4.21: The simplification hierarchies for multiple models generated by *qCoarsen*.

4.5 Simplification-based Remeshing

The introduced techniques for simplification can be used to robustly guide a quadrilateral-only reconstruction of arbitrary topological and polygonal surface meshes. The remeshing pipeline, illustrated in Figure 4.23, constructs a quadrilateral-only mesh by splitting the polygonal elements based on the rules of Catmull-Clark subdivision. The split model is simplified using *QMS* or *qCoarsen* to a quarter of the desired element count. In Figure 4.23, automatically computed curvature-based attributes are associated with the model to guide the *qCoarsen* simplification method. Finally, the model is subdivided and vertices projected to the original surface model using a bounding volume hierarchy to improve nearest neighbor searches. This technique is based on the observation that a single iteration of the Catmull-Clark subdivision yields a quadrilateral-only representation independent of the polygonal types present in the original mesh. Furthermore, while the splitting results in an explosion of elements and generates many extraordinary vertices, the simplification algorithms are able to reduce the element count while significantly improving the mesh structure (Tables 4.1 and 4.2).

Geometric error thresholds during simplification are used to constrain the output remesh within a bounded distance of the original surface. Additionally, the simplification validations maintain mesh topology and user annotated feature edges, thus recreating important, i.e., sharp, curves from the original model. Because this remeshing procedure is entirely connectivity-based, relying on the deletion templates, the technique is both



4.22: Comparison between the two quad simplification algorithms (*QMS*) and *qCoarsen*, for a highly structured (left) and less structured (right) meshes. While *QMS* strives to maintain mesh structure and element quality, *qCoarsen* introduces additional extraordinary vertices to reduce approximation errors and respect surface attributes.

robust and simple to implement, with multiple results of varying genus are illustrated in Figure 4.24, addressing the challenging problem of quadrilateral-only mesh generation.

4.6 Summary

This chapter shows how it is possible to use the dual structure to develop a quadrilateral mesh simplification, through the (*QMS*) technique. To the best of my knowledge, *QMS* is the first fully automatic technique for simplification of quadrilateral meshes using quality metrics. Robustness is one of the key strengths of this approach; the *QMS* algorithm is successfully demonstrated on a large number of models. Another is simplicity; it is relatively easy to implement the algorithm and reproduce the associated results.

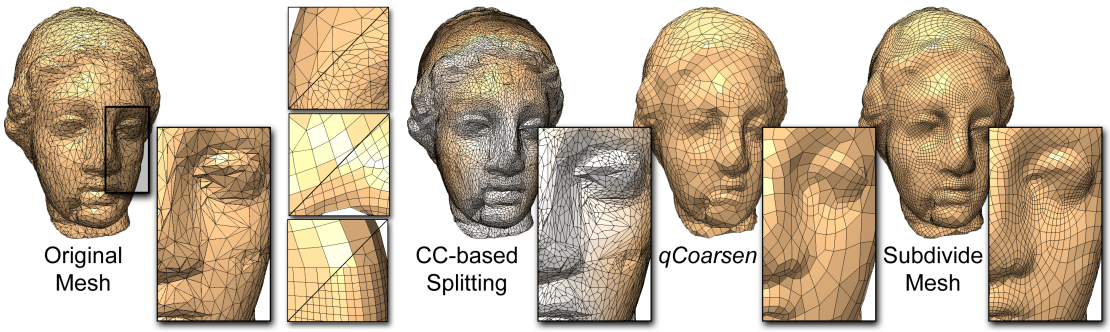
The localized variant of quadrilateral mesh simplification, the *qCoarsen* algorithm, describes a fully automated technique for quadrilateral-based simplification that simultaneously generates well structured quadrilateral-only meshes, dominated by ideal, valence four vertices, while exhibiting augmented attribute awareness via controlled element sizing and alignment to geometric features. User interaction is optional and straightforward, allowing an importance map and a vector field defined over the surface, to enable tailored algorithmic behavior. The localized deletion operators, while improving the attribute awareness of the simplification results, further motivate the robustness and simplicity of

4.3: Quantitative comparison of *QMS* and *qCoarsen*.

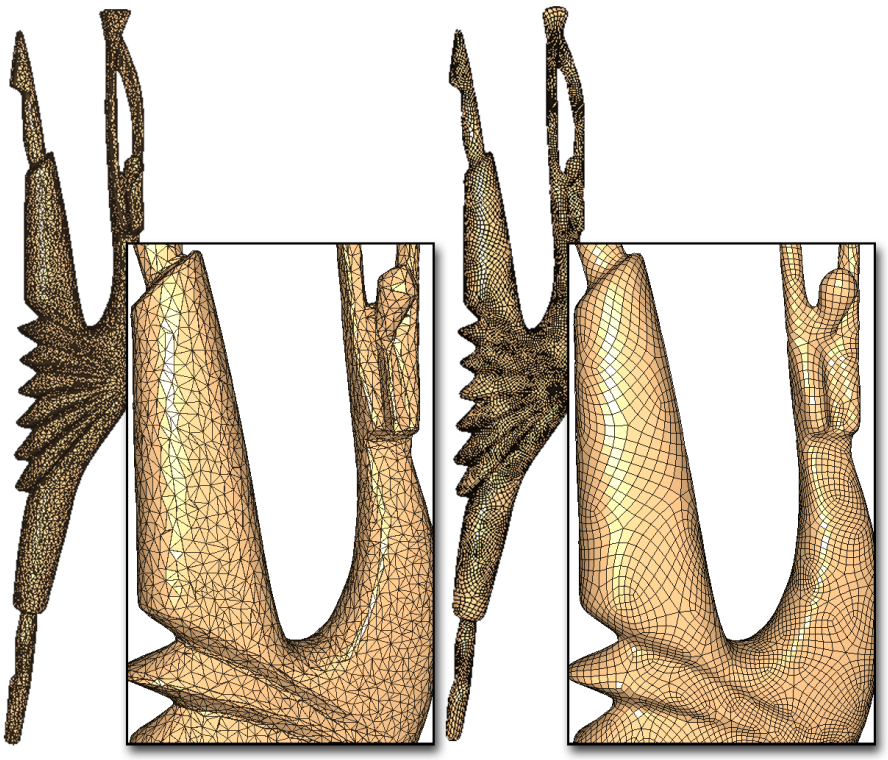
|Quads|	Time (sec.)	Vertex Valence (Ideal,|Ex|,Worst)			Error (10^{-3})
Pensatore (Figure 4.22)					
43650	n/a	99%	8	3	n/a
11k (*QMS*)	46	99%	8	3	$8d_B$
11k (*qCoarsen*)	38	99%	123	6	$1.3d_B$
Bimba (Figure 4.22)					
62842	n/a	99%	726	6	n/a
15.5k (*QMS*)	479	94%	898	6	$4.5d_B$
15.5k (*qCoarsen*)	56	98%	432	6	$3.8d_B$
Wooden Fish (Figures 4.11 and 4.16)					
32.4k	n/a	97%	876	6	n/a
2k (*QMS*)	120	84%	314	6	$56.0d_B$
2k (*qCoarsen*)	37	89%	220	6	$23.0d_B$
Bumpy Torus (Figure 4.21)					
95256	n/a	50%	47.6k	13	n/a
23.5k (*QMS*)	878	66%	8006	10	$14.8d_B$
23.5k (*qCoarsen*)	69	66%	7888	9	$9.7d_B$

connectivity-based approach to quadrilateral-only mesh processing.

I touch only on the potential applications of this framework with the description of a simple quadrilateral remeshing approach. Based on the usefulness of triangle-based simplification in a wide variety of geometric processing algorithms, I believe that the discussion in this chapter can be used as a building block in many other quadrilateral-based processing techniques. The following chapter further motivates this research by improving techniques for simplification guided remeshing of semiregular, quadrilateral-only surfaces.



4.23: The elements of a triangular mesh are split using a Catmull-Clark based scheme to generate a quadrilateral-only mesh. A single iteration of the Catmull-Clark iteration is shown to generate quadrilateral-only models for arbitrary polygonal types, i.e., triangles, quad-dominant and t-junction models. The quadrilateral-only model is simplified to a fraction of the desired element count, then subdivided and reprojected to the original surface.



4.24: Simplification-based remeshing can *robustly* generate quad-only meshes from polygonal meshes with arbitrary genus.

CHAPTER 5

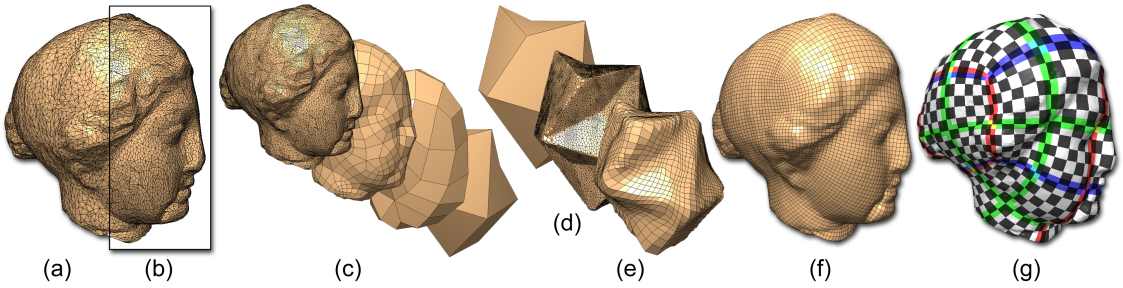
REMESHING FROM SIMPLIFIED BASE DOMAINS

Simplification-based remeshing is a robust approach to quadrilateral-only mesh generation, relying on connectivity-based operators that are simple to implement and understand. Refinement methods insert ideal vertices so that no new extraordinary vertices are generated with each iteration of subdivision, to define semiregular mesh structures. However, the method proposed in Section 4.5 is limited in its application by requiring that the simplified base domain is within a reasonable distance of the original model to properly compute the backward projections.

The natural progression of simplification-based remeshing is to address the coarseness of the base domain model. This chapter introduces an algorithm to remesh input polygonal-based surfaces of arbitrary genus using simplification and refinement methods, while creating a mapping function of the original model onto the simplified mesh at every level-of-detail, illustrated in Figure 5.1. This chapter describes the implementation of an hierarchical mapping technique as well as an adaptive resampling of the base domain. These methods describe a semiregular, quadrilateral-only remeshing algorithm that can be generalized to arbitrary polygonal remeshing.

This work uses quadrilateral-based simplification to build the base domains. It is well articulated in Chapter 4 that the quadrilateral element enforces structural constraints on the mesh, where the deletion of a single quadrilateral may require the removal of a larger collection of elements to preserve an all-quadrilateral mesh or to maintain the mesh structure. While this work is inspired by the *multiresolution adaptive parameterization of surfaces* (MAPS) [94], and its variant [85], for triangle-based remeshing, a key novelty is the development of a mapping technique that is not dependent on the particular coarsening operations as required by quadrilateral-based simplification.

While semiregular, quadrilateral-only meshes demonstrate structural advantages useful in subsequent applications, their constructions are complicated by parameterization-



5.1: The algorithm splits an input mesh of arbitrary polygonal type (a) into a quadrilateral-only mesh (b), simplifies the model while maintaining critical levels-of-detail (c) to guide the map of the original geometry to the base domain (d). The base domain is refined, the vertices relaxed to accommodate for area distortions in the map (e), then the vertices are reprojected to the original surface (f). A surface parameterization is a byproduct of the method to facilitate geometry processing, i.e., texture mapping (g).

based challenges. The base domain coarseness is constrained by the genus and geometric complexities of the model, where too few base elements typically lead to distorted elements and poor surface approximation, further discussed in Section 5.1.5. To address these problems, an adaptive sampling of the base domain uses an approximation to the surface area as well as element quality to reduce the parametric distortion and improve remeshing errors.

5.1 Semiregular Remeshing

This algorithm relies on the observation from Section 4.5 that a single execution of Catmull-Clark subdivision [28] generates a quadrilateral-only representation of an input mesh, independent of the original polygonal elements. The splitting inserts ideal vertices at the midpoints of the input mesh edges, and vertices with valence equal to the number of polygon sides at each face centroid. The remainder of this section describes the simplification, mapping and refinement processes for remeshing.

5.1.1 Deletion Operators

Quadrilateral-based simplification constructs the base domain mesh, while maintaining a mapping from the original model to the coarsened mesh at each levels-of-detail. In contrast to triangle-based techniques [94], mapping quadrilateral-based simplifications have the challenge of supporting a large cast of deletion templates, including global operators. In particular, as discussed in Chapter 4, the *dual polychord* collapse operator

is a critical deletion operator for quadrilateral meshes.

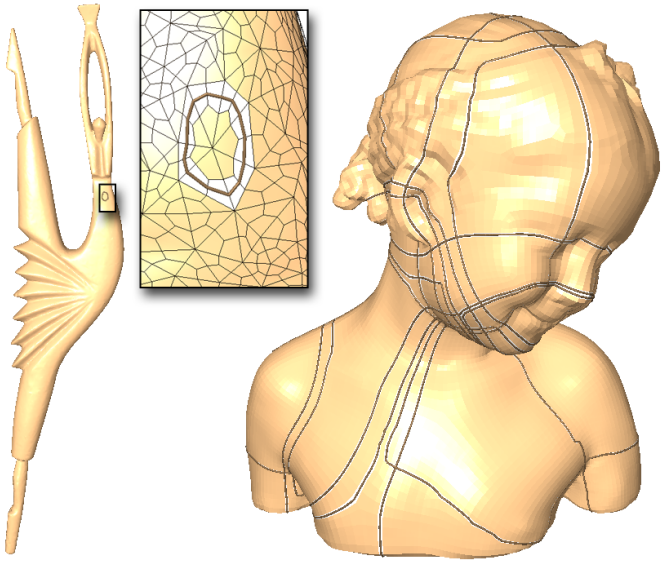
While mapping of some polychord neighborhoods to the plane may be straightforward, the complex knots and global nature of these structures can quickly complicate the parameterization method, as illustrated in Figure 5.2. In this work, the *QMS* and *qCoarsen* algorithms, described in Chapter 4, are used to drive the simplification process. Together, these techniques describe various deletion operators, as well as their own weighting functions for the automated prioritization of the element deletions. A novel hierarchical mapping scheme operates independent of the underlying simplification method, supporting both *QMS* and *qCoarsen* without special case handling.

5.1.2 Keyframe Meshes

The function ϕ defines a bijective mapping of the vertices V of an input quad mesh M to the base domain mesh M^0, $\phi : M \rightarrow M^0$. To support arbitrary deletion operators without special case handling, ϕ is constructed by storing a set of *keyframe* meshes $K^{\{M,\ldots,0\}}$ during the simplification process, illustrated in Figure 5.3. The term keyframe is intended to evoke a popular animation technique, where important locations and poses are defined through which a character deforms. Analogously, the keyframe meshes dictate the path progression for points as they map from M to M^0.

The original model M is pushed onto the stack of keyframe meshes, denoted as K^M. During simplification a new keyframe mesh K^{m-1} is committed to stack as necessitated by an inspection routine executed after each deletion iteration. The current simplified mesh M^c is committed to the keyframe stack if the Hausdorff distance between M^c and K^m, the previously committed keyframe mesh, is greater than a specified distance d; or if the projection of K^m onto M^c has flipped elements. Lastly, the base domain mesh M^0 is committed as the final keyframe mesh K^0.

To improve performance, this inspection process is localized to a subset of quads Q^m of K^m. Consider that a deletion operator processes a set quads Q^c of M^c, including the element(s) intended for deletion and their one-neighborhood, returning a new set of quads Q^{new}, where $|Q^{new}| < |Q^c|$. Only the subset of quads Q^m of K^m within the distance d of the original quad group Q^c are considered. The vertices of Q^m are projected onto Q^{new} using a closest point projection, testing the distance threshold and for flipped elements. Localizing the inspection improves performance.



5.2: A *single* polychord is highlighted on the two models above. While mapping some polychord neighborhood boundaries to the plane is straightforward (left), the global nature of these structures may require more complicated parametric domains (right).

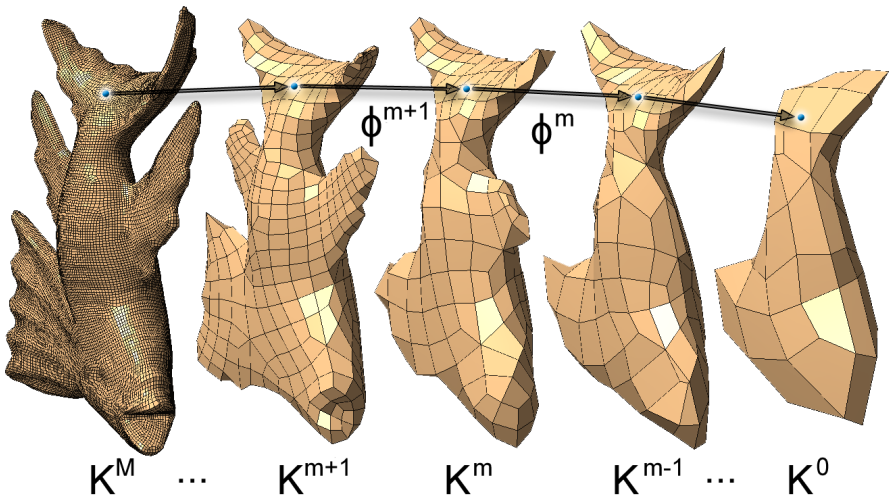
5.1.3 Hierarchical Keyframe Mapping

The development of $\phi : M \rightarrow M^0$ is guided by the keyframe meshes, where individual functions are independently developed to map each keyframe mesh to the next coarsest representation, $\phi^m : K^m \rightarrow K^{m-1}$. As illustrated in Figure 5.4, the function ϕ^m is obtained through iterative ray casting and relaxation of the vertices of K^m over K^{m-1} until inverted elements are resolved. Figure 5.4 illustrates a $2D$ diagram of the projection and relaxation phase results, as well as an example image of the fold over evident within a projected mesh.

A new mesh $\tilde{K}^m$ that is the projection of K^m onto K^{m-1}, produces an initial ϕ^m. To ensure that ϕ^m is a bijective mapping, flipped elements in $\tilde{K}^m$ are resolved via a relaxation phase. A movement vector $\tilde{m}$ corresponding to a vertex $\tilde{v}$ of $\tilde{K}^m$ is computed towards the weighted average of the centroids of the neighboring quads $\tilde{q}_i$ of $\tilde{v}$:

$$\begin{aligned} \tilde{m} &= \frac{\sum_i C(\tilde{q}_i)\theta(\tilde{N}^m(C(\tilde{q}_i)),N^{m-1}(\tilde{v}))}{\sum_i \theta(\tilde{N}^m(C(\tilde{q}_i)),N^{m-1}(\tilde{v}))} - \tilde{v}, \\ \theta(n_1,n_2) &= \begin{cases} 10, \langle n_1, n_2 \rangle \geq 0.0 \\ 0, \text{otherwise} \end{cases} \end{aligned}$$

where $C(q)$ computes the centroid of quad q, $\tilde{N}^m(v)$ returns the normal of $\tilde{K}^m$ evaluated at v, $N^{m-1}(v)$ returns the normal of K^{m-1} evaluated at the projection of v, and $\langle n_1, n_2 \rangle$



5.3: Keyframe meshes K^m are discrete samplings of the simplification hierarchy, used to guide the mapping of a point from K^M to K^0.

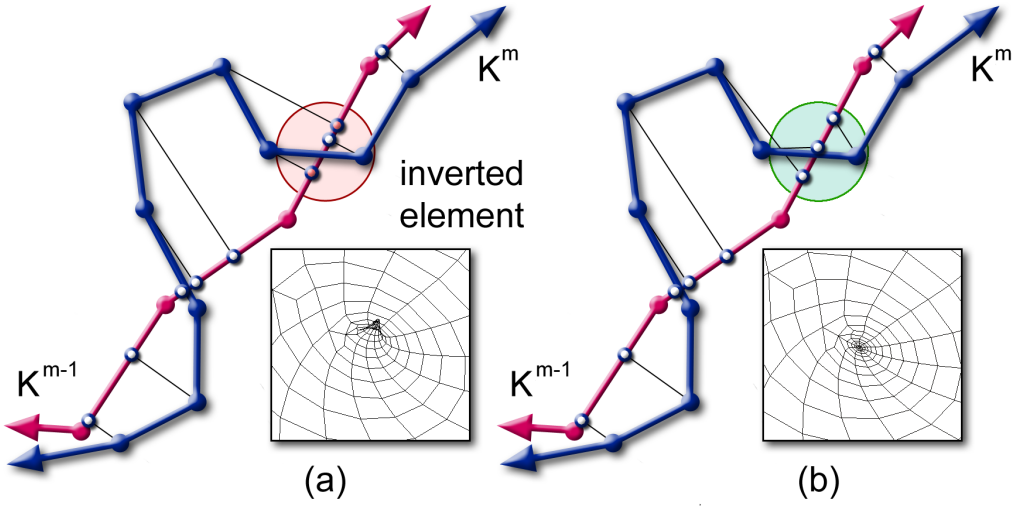
is the inner product of the two vectors. The relaxation process resolves flipped elements of $\tilde{K}^m$ by assigning larger weights, θ, to nonflipped quads. The weighting differential results in a pulling effect that spreads points away from flipped regions. The movement vector $\tilde{m}$ is projected and scaled, $\tilde{v} = \tilde{v} + \alpha(\tilde{m} - \langle N^{m-1}(\tilde{v}), \tilde{m}\rangle N^{m-1}(\tilde{v}))$, in practice $\alpha = 0.5$, and $\tilde{v}$ is reprojected onto K^{m-1}.

Developing the mapping functions ϕ^m between keyframe meshes allows the procedure to be parallelized with the simplification process and each other. This hierarchical mapping technique improves computational performance required in the resolution of flipped elements, because each keyframe mesh stores a reduced number of vertices, especially in comparison to M. The projection of a point on M to M^0 through the keyframe mapping functions ϕ^m necessitates the use of barycentric coordinates, illustrated in Figure 5.3. In this work, the barycentric coordinates are computed by virtually dividing each quadrilateral into four triangles, radiating about the centroid.

5.1.4 Downward Projection

A point p on K^m is assigned the barycentric coordinates (α, β, γ) for the subtriangle t of the quadrilateral q. The vertices of q are indexed $q.v_i$, $i = (0, 1, 2, 3)$, the centroid is $q.c$, and the subtriangle t is described by vertices $(q.v_t, q.v_{(t+1)\%4}, q.c)$. The projection of p onto K^{m-1} is computed based on ϕ^m, illustrated in Figure 5.5.

When the vertices of q map to the same subtriangle t' of quadrilateral q' on K^{m-1},



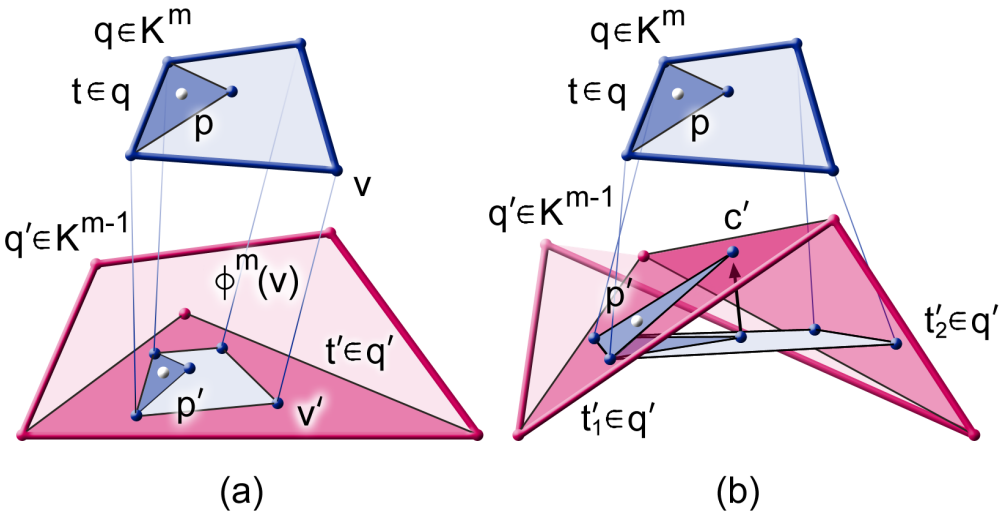
5.4: The function ϕ^m maps the vertices and connectivity of the keyframe mesh K^m onto the next keyframe mesh K^{m-1}, developed as a two phase process: ray cast projection (a) and relaxation to resolve inverted elements in the projection of K^m (b).

mapping p onto K^{m-1} is straightforward (Figure 5.5a). Barycentric coordinates are computed for each vertex $q.v_i$ on t' as $(\alpha_i, \beta_i, \gamma_i)$, and those assigned the centroid are an average of the four vertices, $(\alpha_c, \beta_c, \gamma_c) = \frac{1}{4}\sum_{i=0}^{4}(\alpha_i, \beta_i, \gamma_i)$. The new barycentric coordinates for p within t' are computed as a weighted combination,

$$\begin{aligned}(&\alpha'_t \cdot \alpha + \alpha'_{(t+1)\%4} \cdot \beta + \alpha'_c \cdot \gamma,\\ &\beta'_t \cdot \alpha + \beta'_{(t+1)\%4} \cdot \beta + \beta'_c \cdot \gamma,\\ &\gamma'_t \cdot \alpha + \gamma'_{(t+1)\%4} \cdot \beta + \gamma'_c \cdot \gamma).\end{aligned}$$

The more challenging problem is when the vertices of q map to multiple subtriangles on K^{m-1}. When the vertices of q map to two adjacent subtriangles, t'_1 and t'_2, on K^{m-1}, the triangles may be locally flattened. On this plane, new barycentric coordinates are computed for p after projecting the vertices $q.v_i$ and $q.c$. However, when additional subtriangles are involved, more intricate flattening strategies are required.

Instead, to compute new barycentric coordinates of p, we decided to use a ray casting approach (Figure 5.5b). The vertices $q.v_i$ correspond to $q.v'_i$ of $\tilde{K}^m$, $q.v'_i = \phi^m(q.v_i)$. Because the vertices of q map to multiple subtriangles of K^{m-1}, the simple projection case (Figure 5.5a) does not apply. Instead, the projected centroid $q.c'$ is the average of the mapped vertices $q.c' = \frac{1}{4}(\sum_i q.v'_i)$. This point $q.c'$ is projected in the normal direction $\tilde{N}^m(q.c')$ onto K^{m-1}.



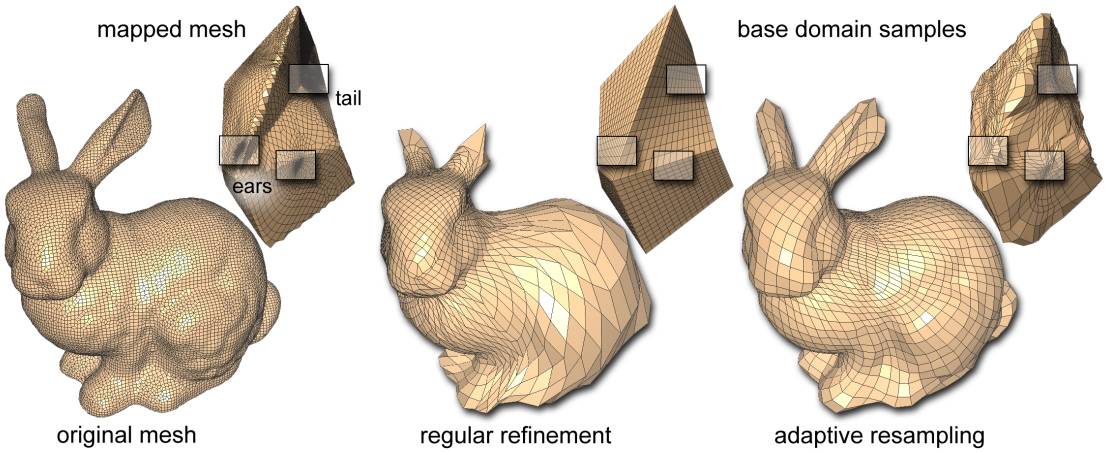
5.5: The barycentric coordinates of the point p within subtriangle t of $q \in K^m$ are known. If all vertices of q map, $\phi^m(v)$, to the same subtriangle (a) $t' \in K^{m-1}$, then new barycentric coordinates assigned to p are computed at p' within t'. If the vertices of q map to multiple subtriangles (b), the mapped centroid is projected in a normal direction to K^{m-1}, c', and p' is computed. If p' is not on a subtriangle of K^{m-1}, then it is projected in a normal direction.

If the vertices $q.v'_t$, $q.v'_{(t+1)\%4}$, and $q.c'$, map to the same subtriangle of K^{m-1}, then new barycentric coordinates for p may be computed as a weighted combination, previously discussed. However, when these vertices map to multiple subtriangles, p' is computed on the triangle formed by these vertices, $p' = \alpha q.v'_t + \beta q.v'_{(t+1)\%4} + \gamma q.c'$. This point is projected in the normal direction $\tilde{N}^m(p')$ onto K^{m-1}, computing new barycentric coordinates for p at the intersection.

The ray casting based downward projections yield similar results to the previously described unhinging technique, without special cases for the various neighborhood scenarios. For improved performance, the vertices and centroids of each keyframe mesh are projected once and stored. Future projections require only normal projections of p' for a subset of the subtriangles in K^m.

5.1.5 Adaptive Resampling

Following the computation of a map, typically a semiregular remesh is computed through regular refinement of K^0 and backward projection of the vertices onto K^M. However, this approach is unable to accommodate for nonequiareal mappings that results in poor surface approximations. We allow the base domain remesh R^0 to adaptively



5.6: Regular refinement of the base domain may poorly approximate the surface, due to area-based distortions in the mapping. Our refinement and adaptive resampling better accommodates these regions, highlighted on the tail and ears.

resample K^0 (Figure 5.6), guided by surface area, approximation error and the element quality of the final remesh R^M.

The vertices of R^0 lie on the base domain determined by regular refinement of M^0. The area of the original model M associated with each vertex $v \in R^0$ can be computed by integrating the area of M that maps onto M^0 nearest to v. Relaxation of v occurs by moving towards the area weighted centroid of its neighboring remesh vertices. Iterative execution of the relaxation improves the distribution of the remesh vertices, more evenly sampling the original model.

Because our keyframe mapping approach does not describe a conformal mapping, the angles formed by edges of R^0 do not translate to similar angles on R^M. A second relaxation phase is integrated within the resampling to reduce parametric distortion on R^M by adapting a *ballooning scheme* that improves element quality and approximation [140]. Each vertex in R^M moves in the direction of the vertex normal, scaled by the accumulated error value measured as the signed distance between each neighboring quadrilateral centroid and M. The vertices are simultaneously relaxed toward the the average of their connected neighbors, and projected onto K^M. The vertices of R^0, v_i^0 corresponding to v_i^M on R^M, are updated to reflect these relaxations, $v_i^0 = \phi(v_i^M)$.

This process leverages both representations of the remesh on the base domain R^0 and the original model R^M. The relaxations improve element quality while allowing

the remesh to cope with parametric distortions in the map, illustrated in Figure 5.6. Furthermore, hierarchical resampling, achieved by interleaving the refinement and two relaxation phases produces faster convergence of the remesh vertices, a method used for the remeshes illustrated throughout this paper.

5.1.6 Area Approximation

Wavefront propagation used to compute the surface area associated with each remesh vertex, as described above, is time consuming and costly. Instead, to quickly approximate the area of the original model as it maps to the base domain based on the keyframe maps, we construct two point clouds, using a kd-tree to facilitate nearest neighbor searches: P^M is a near equiareal, random sampling of K^M, and P^0 is its mapping onto K^0 through the keyframe maps, ϕ^m.

Given P^0, the approximation of the surface area associated with a remesh vertex v of R^0 is computed by summing the number of points in P^0 within a specified distance of v. The search radius is evaluated as one half the average distance between v and each of its neighbor vertices in R^0. Because the sampling of P^M is near equiareal, the neighborhood count serves as a sufficient scalar to approximate an area-based weight that can be assigned to each point in R^0.

5.1.7 Point Projection

It is possible to project all of the original vertices through ϕ onto the base domain, illustrated in Figure 5.6, for precise backward projection computations. However, for improved computational performance, we leverage the correspondence between the points in P^M and the projected points P^0. An approximation of the backward projection for a point p on K^0 with neighboring points $n_i \in P^0$ that correspond to $n_i' \in P^M$, is computed as

$$p' = (\sum_i \frac{n_i'}{\|p - n_i\|})/(\sum_i \frac{1}{\|p - n_i\|}).$$

With a dense sampling of P^M (in practice $2k$ points), the technique is fast and adequate for our purposes, avoiding the mapping of potentially many vertices in M to K^0. The subsequent ballooning, relaxation and projection will ensure that p' is placed on M. Furthermore, the relationship between P^M and P^0 can be further exploited during the downward projections while updating the point locations of R^0 to reflect relaxations that occur on R^M.

5.1.8 Feature Preservation

The simplification operators maintain the topology of feature edges, important structures annotated on the original mesh [35]. The keyframe-based remeshing method reconstructs feature edges with special case handling during the adaptive resampling. When refining the feature annotated edges of the base domain K^0, the new vertices are mapped to the corresponding feature on K^M and moved to evenly resample the original feature. These vertices are later fixed during the subsequent smoothing iterations, relying on movement of normal vertices to resolve inverted elements.

5.2 Results

The quadrilateral mapping described in Section 5.1 was implemented in C++. The remeshes were performed on a 2.16 GHz Intel Core 2 Duo with 2GB memory, taking in the order of a few minutes to compute, further detailed in Table 5.1 for remeshes shown throughout the paper. The timings measure four phases, monitoring the simplification of the model (I), the additional time needed to complete the keyframe mappings (II), and the adaptive resampling and backward projections (III). This code had been built to emphasize its ability to operate independent of the simplification technique, supporting any variety of deletion operations without special case handling.

The implementation is tested on a range of models with varying genus, geometric complexities, and input polygonal types, illustrated in Figure 5.7. These remeshes test multiple quadrilateral-based simplification algorithms that support locally- and globally-based operations, while developing quadrilateral-only reconstructions of triangle-, quad-dominant and irregular quadrilateral-only meshes. All of the remeshes shown throughout this paper emphasize the advantages of our simplification-based algorithm by supporting very coarse base domains.

Table 5.1 quantitatively analyzes the quality of the remeshes, documenting approximation errors, number of extraordinary vertices (nonideal valence four), worst case valence, and the orthogonality of the resulting parameterization. The remesh error measures the Hausdorff distance of the remesh and the original model, relative to the bounding box diagonal d_B. The number of extraordinary vertices is related to the number of coarse quad regions that segment the model for parameterization or semiregular remeshing. The scaled Jacobian statistics indicate the orthogonality of the mesh elements, average and worst case, where 1 corresponds to a rectangle, 0 to a quad with 3 colinear vertices, and

5.1: Analysis of the original quad models (following the initialization subdivision when necessary), and the remesh results, including remesh times (simplification (I), keyframe mapping (II), adaptive remesh (III) and total (T)), vertex information (total, extraordinary, and worst case valence), element quality (median and worst Scaled Jacobian, median and standard deviation mesh angles), and approximation errors of the models shown throughout this chapter.

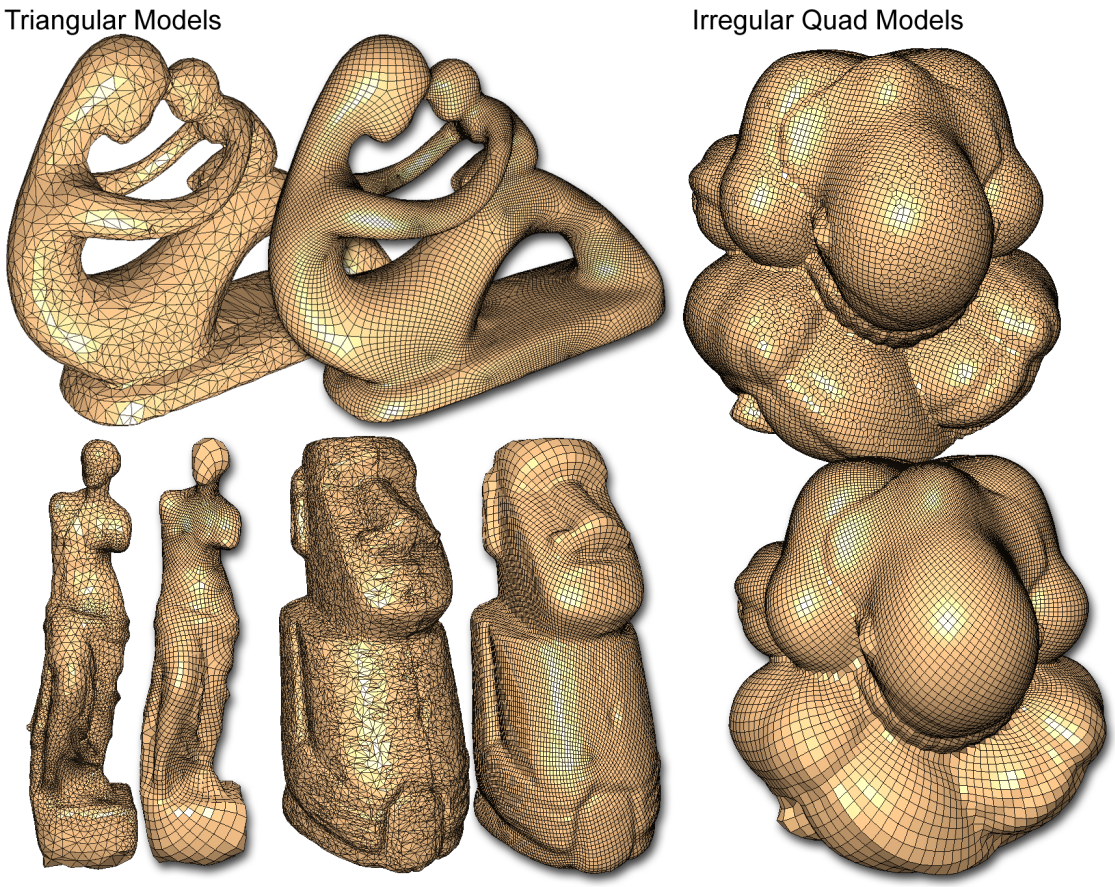
Time (seconds) (I,II,III) total	Vertices (|V|, |Ex|, wc)	S.Jacobians (median, worst)	Mesh Angles (mean, σ)	Error (10^{-2})
Egea (Figure 5.1)	(27k, 13.1k, 11)	(0.63, 0.02)	(90°, 36.6°)	n/a
(26, 30, 28) 84	(10.2k, 8, 5)	(0.98, 0.66)	(89.9°, 9.6°)	$0.71d_B$
Bunny (Figure 5.6)	(21.7k, 9.6k, 6)	(0.92, 0.0)	(89.9°, 18.1°)	n/a
(34, 1348, 33) 1415	(2.3k, 8, 5)	(0.88, −0.25)	(89.3°, 32.5°)	$4.3d_B$
Fertility (Figure 5.7)	(22.5k, 11.1k, 11)	(0.65, 0.09)	(90°, 34.3°)	n/a
(18, 9, 42) 69	(31.9k, 110, 6)	(0.98, 0.06)	(89.9°, 9.7°)	$1.0d_B$
Venus (Figure 5.7)	(28.1k, 14k, 11)	(0.83, −0.96)	(89.9°, 23.3°)	n/a
(139, 15, 39) 193	(25.6k, 26, 6)	(0.98, 0.45)	(89.8°, 10.7°)	$1.4d_B$
Moai (Figure 5.7)	(63.5k, 31k, 11)	(0.72, −1.0)	(89.9°, 28.8°)	n/a
(293, 7, 24) 324	(12.8k, 12, 5)	(0.98, 0.45)	(89.7°, 10.7°)	$0.083d_B$
Pensatore (Figure 5.7)	(31.1k, 16.4k, 6)	(0.88, 0.05)	(89.9°, 20.1°)	n/a
(66, 31, 33) 130	(19.5k, 8, 5)	(0.99, 0.41)	(89.8°, 8.6°)	$0.16d_B$

inverted elements are less than 0.

Further analysis compares the results of the adaptive remeshing versus traditional regular refinement, illustrated in Figure 5.6. Regular refinement, especially when simplifying to coarse base domains, yields higher approximation error ($E = 0.16d_B$) than our adaptive technique ($E = 0.043d_B$). Without incorporating the relaxation scheme, regular refinement does not handle parametric distortions, generating a median scaled Jacobian of 0.81 with a worst case -0.94. Our adaptive resampling relaxes vertex locations based on element quality to improve these metrics, with a median scaled Jacobian equal to 0.88 and a worst case -0.25. The coarseness of the base domain (10 faces) can result in the negative scaled Jacobians, despite our adaptive resampling technique. In these cases, the small number of user desired extraordinary vertices over constrains the structure and optimization (Table 5.1). Allowing more extraordinary verices and concomitantly more faces in the base domain gives the flexibility needed to improve the remesh quality.

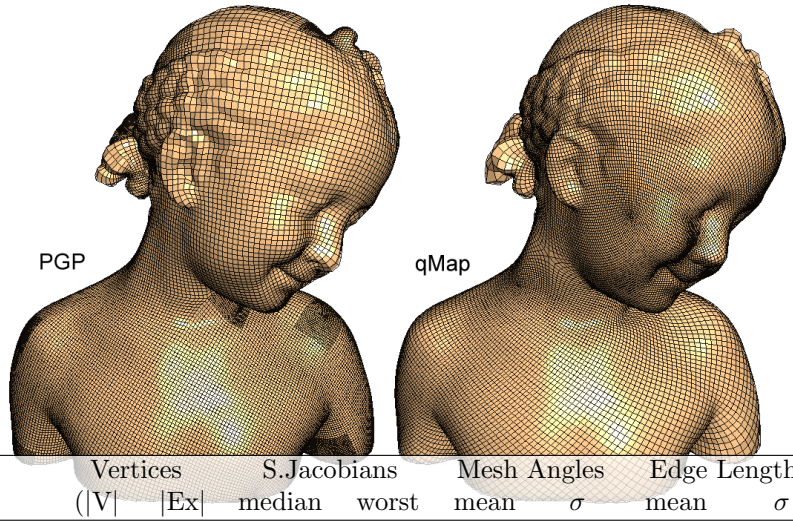
5.2.1 Remesh Comparison

This study measures the quality of the remesh elements, a statistical analysis of the scaled Jacobians and the mesh angles, as well as a comparison of parametric stretching related to the mesh edge lengths. Our bimba remesh is compared to a model acquired on-



5.7: Semiregular, quadrilateral-only remeshes, supporting both local- and global-based simplification algorithms, for input triangle and quadrilateral-only models.

line, remeshed using *periodic global parameterization* (PGP) [126] in Figure 5.8. The PGP remesh describes a coarse quad segmentation, generating a semiregular, quadrilateral-only mesh. The 915 T-junctions on the model were *not* included in the count of extraordinary vertices. In comparison to PGP, our model, that was specifically constructed to have a similar number of extraordinary vertices, exhibits a similar statistical analysis (mean and standard deviation) of the mesh angles and edge lengths, while improving the worst case scaled Jacobian. This comparison, and similar quality metrics evaluated on our remeshes (Table 5.1), illustrates that it is possible to create remeshes of similar quality with our method in comparison to existing remeshing methods, while improving the ability to control the number of base patches and extraordinary vertices of the final remesh.



	Vertices		S.Jacobians		Mesh Angles		Edge Lengths	
	(|V|	|Ex|	median	worst	mean	σ	mean	σ
PGP	47.3k	26	0.99	−0.75	89.9°	10.8°	0.0063	0.0027
qMap	31.7k	24	0.98	0.53	89.9°	11.5°	0.0059	0.0026

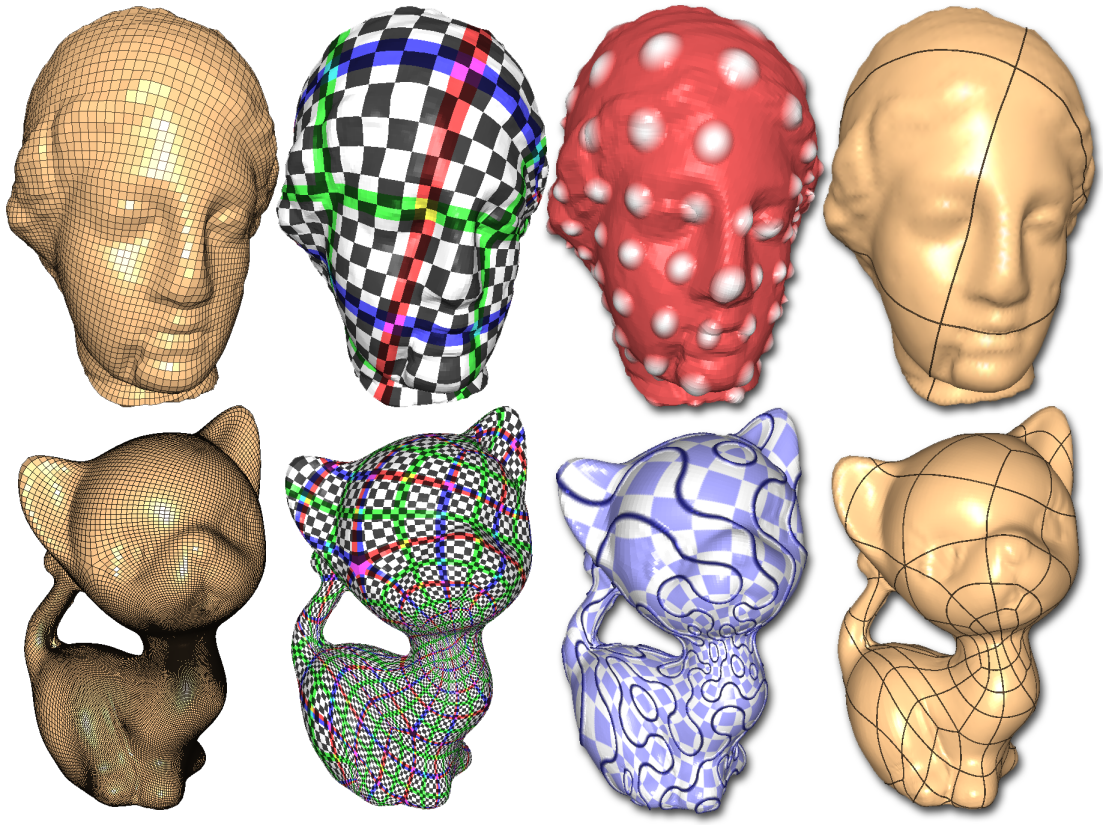
5.8: Remesh results of *periodic global parameterization* (PGP) [126] and our remeshing algorithm (qMap). A quantitative analysis of the illustrated meshes compares the vertex information (original count, remesh count, and extraordinary count), and statistics related to the mesh angles and edge lengths.

5.2.2 Applications

Geometric processing algorithms are able to take advantage of the neighborhood structure offered by semiregular polygonal meshes. Particularly, simplification-based remeshing schemes facilitate mesh improvement [24], consistent remeshing [135] and deformation [93] applications. As illustrated in Figure 5.1, the semiregular, quadrilateral-only remeshes are coupled with a surface parameterization as a byproduct of their construction. Texturing and displacement mapping applications, as well as spline-based modeling is straightforward, illustrated using several remeshes in Figure 5.9. Out of scope of this dissertation, yet applicable to the spline-based modeling, are related research projects [32] and [31] in which I address intersurface continuity between adjacent B-spline patches fit over input geometry.

5.2.3 Limitations

The hierarchical mapping technique can generalize to other surface representations, supports arbitrary deletion methods, and describes a hierarchical approach to the map development. However, the relaxation that resolves inverted projections can require many



5.9: A surface parameterization is a byproduct of the remeshing algorithm, facilitating texture and displacement mapping, as well as spline-based modeling.

iterations, especially in resolving large fold over regions that may occur while reducing to a very coarse base domain. Most remeshes are obtained within a few minutes (Table 5.1); however, the Stanford bunny (Figure 5.6) required 23 minutes because of the base domain coarseness (i.e. only 8 faces). This mapping technique is unable to handle cases where the simplification generates self intersections.

Future research will address improving the placement of the remesh vertices, in particular, extraordinary vertices. An advantage of our approach is that it will facilitate processing, by computing vertex shifting, element refinement, and other methods on the coarse base domains. An important and challenging aspect of quad meshes is to address the placement of base domain extraordinary vertices and the integration of attribute alignment [92].

5.3 Summary

This chapter introduces a simplification-based technique for the semiregular, quad-only remeshing of arbitrary topological polygonal meshes that operates independent of the deletion operations by leveraging *keyframe* meshes to guide a hierarchical mapping algorithm. It is shown that this method can produce models with similar quality elements as an existing quad remeshing scheme (PGP), while providing tools for more direct control over the number of extraordinary vertices to produce *very* coarse quad regions. The remesh vertices are sampled in a way that reduces parametric distortions and approximation errors. The remeshing algorithm is able to significantly simplify the input geometry by implementing an adaptive resampling scheme of the base domain to accommodate for area distortions in the mapping functions. The modified resampling supports coarser segmentations than other simplification-based remeshing [35] and mapping-based methods [94, 85, 1], by which this work in inspired.

CHAPTER 6

QUADRILATERAL MESH IMPROVEMENT

To promote themes evident throughout the dissertation, this chapter motivates the use of connectivity-based operators while emphasizing structure in the underlying dual representation. The following discussion applies polychord-based methods within an interactive mesh improvement framework. The preliminary results of this research further demonstrates the potential of polychord-based methods for quadrilateral mesh processing, while presenting methods that are straightforward to implement.

6.1 Interactive Mesh Improvement

The goal of an mesh editting system is to provide the user with a set of tools to guide a set of semiautomated methods that augment characteristics and features of the model. Analogously, photo editors provide an artist many painting, cutting, and other templates to manipulate the input. While there are many modifications a user may wish to control on the mesh, including multiresolution editting [17, 93], optimization [71] and smoothing [161, 39, 180], this work focuses on a novel approach to augment the structure of a quadrilateral mesh, reducing and isolating extraordinary vertices to specific regions on the model.

In this research, the user specifies the alignment of an *ideal* polychord. In practice, I have chosen ideal polychord alignment to be guided by surface curvature, forming concise loops around cylindrical components of a model. Following the creation of a target polychord, the method alternates between simplification methods to remove existing elements from the model, and ideal insertion techniques to refine the element count back towards the desired number. The user is able to control the area of influence around the ideal polychord, localizing the effects of the improvement scheme. The following subsections further documents the refinement operators, user interface, and details of the outlined method for mesh improvement.

6.1.1 Polychord Pillowing

The polychord pillowing operator is designed to generate a polychord of desired structure or shape on the model. This operator inserts a set of quadrilaterals in such a way that it forms a polychord that runs parallel to a selected looping set of mesh edges. As illustrated in Figure 6.1, given a linked list of mesh edges (a), the method separates the mesh along this curve creating a void with two boundary curves with an identical number of vertices (b). The pillowing method fills the void by connecting the duplicated vertices to generate the new polychord (c). The polychord pillowing method is able to generate desired structures within the dual representation independent of the original polychords.

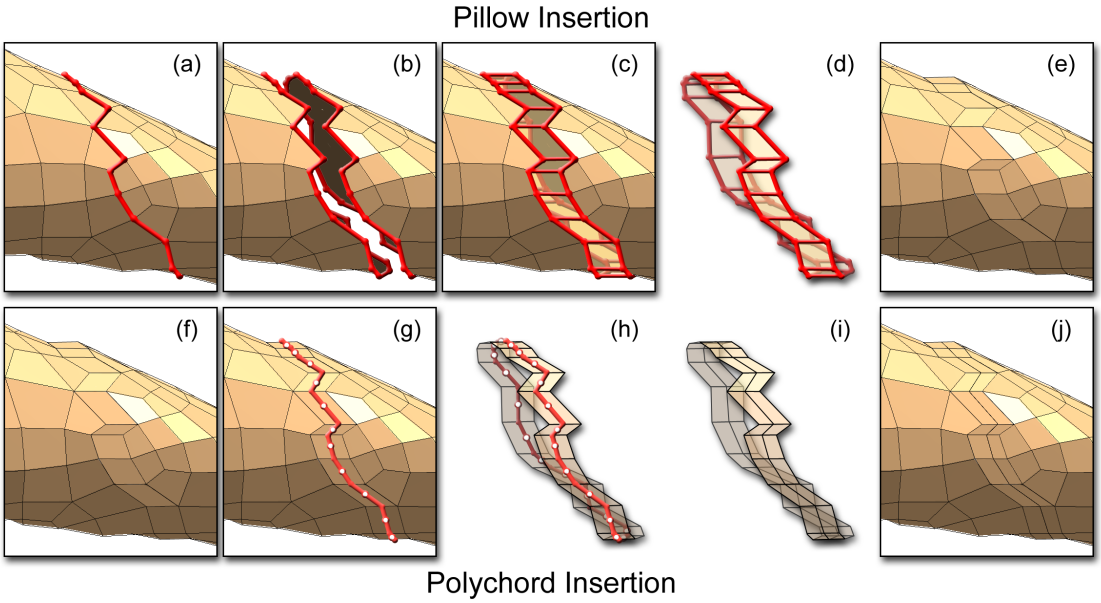
6.1.2 Polychord Insertion

Polychord insertion, more aptly described as duplication, results in the inclusion of an additional polychord within the dual representation. In contrast to the polychord collapse, the insertion method simultaneously creates multiple quadrilaterals in the mesh. As illustrated in Figure 6.1, the method splits the mesh elements by sequentially connecting the midpoints of edges to which the polychord is dual. Because of the looping property of the polychord, the polychord insertion will not generate any T-junctions, maintaining a quadrilateral-only model. Note that the polychord insertion is in fact a constrained version of the polychord pillowing, where the mesh edge loop parallels a polychord. However, to ease this discussion, the operators are considered distinct to better distinguish between them.

6.1.3 User Interface

To initialize the algorithm, as illustrated in Figure 6.2, the user identifies information pertaining to the desired polychord alignment and region of improvement. First, the user identifies the looping mesh edges that will seed the polychord pillowing and insertion methods. Second, the user defines the region within which the edge loop will be duplicated over the model.

Two methods exist to define the linked list of mesh edges. For wandering (or obscure) paths, the user may iteratively walk the mesh from a seed vertex to grow the edge sequence. However, in practice the selected edge loops often contain a simple structure, circumnavigating cylindrical regions of the model. A semiautomated method facilitates the definition of edge loops of this type, requiring the user to identify a seed vertex and a



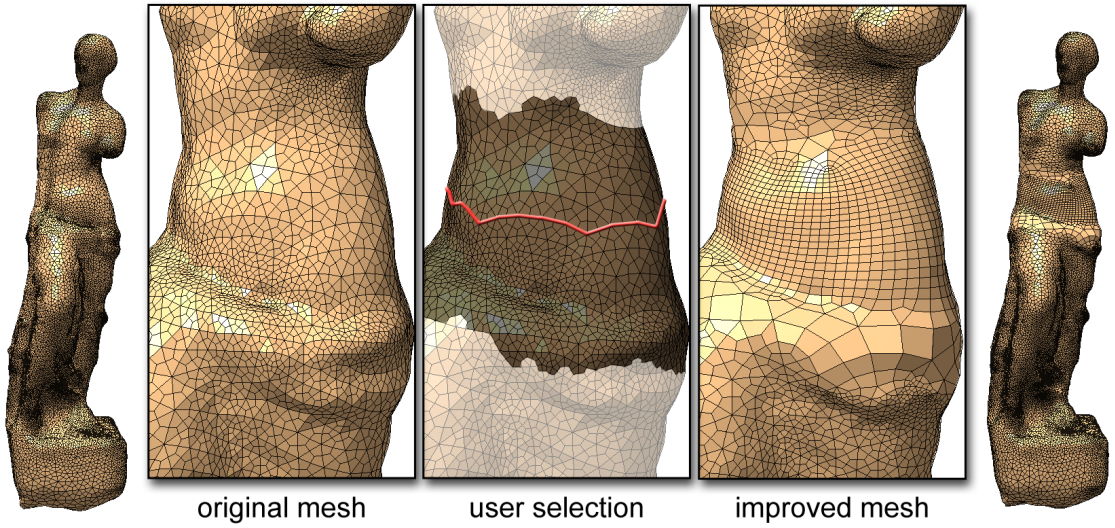
6.1: The *pillow insertion* (top), for a selected looping set of mesh edges (a), separates the mesh along the edge loop (b), then inserts elements to fill the void (c) generating a polychord that aligns with the edge loop (d). The *polychord insertion* (bottom) splits the polychord's quadrilaterals (g,h), by connecting midpoints of the edges dual to the polychord (i).

grow direction of the path. Given these inputs, a variant of the Djikstra's shortest path algorithm that disregards edges whose direction has a negative inner product with the defined growth vector, computes a looping path around the model.

The mesh improvement is executed over the region surrounding the looping mesh edge sequence defined by the user through a *painting* interface. As illustrated in Figure 6.2, the boundaries of this region are set by allowing the user to select the deletable elements on the mesh. An adaptable brush size facilitates the *painting* process as the user applies strokes via their mouse to the mesh.

6.1.4 Improvement Algorithm

The semiautomatic algorithm is initialized with the user highlighted mesh region and specified edge loop to guide the mesh simplification and polychord-based refinement. First, the method produces the desired (or *ideal*) polychord by pillowing the specified mesh edge loop (Figure 6.1). Then a variant of the localized *qCoarsen* simplification algorithm, described in Chapter 4, is used to coarsen the elements within the *painted* deletable element region. Because the improvement algorithm uses the orientation of



6.2: Given an input quadrilateral mesh, the modeler selects a looping set of mesh edges and the region over which a polychord aligned to these edges should be propogated. Given these inputs, the improvement algorithm pillow inserts a polychord along the identified edges, then iterates between simplification and polychord insertion to obtain an improved remesh of the region.

the looping edge sequence to align elements during insertion, the simplification priority metric focuses on vertex valence and element size to prioritize the deletions.

As the original elements are removed from the model, new elements are created through polychord insertion, and vertex relaxation propagates the elements over the coarsened area. The success of this improvement scheme stems from the observation that the polychord insertion does not generate any extraordinary vertices. The method alternates between the simplification process and refinement, fluctuating within a window of the desired element count; in practice, reducing elements to 90% the desired count before refining to the original element count. The remeshing algorithm terminates when a fraction, in practice $< 25\%$, of the original elements that had been marked for deletion, remain on the model. Experiments indicate that deleting all of the *painted* elements during the remeshing scheme leads to a poor transistion in element density between the original mesh and the remeshed quadrilaterals.

Localization of the improvement method and operating within a small window of the desired element count, in practice, keeps the remesh surface within a tight envelope of the original surface. As such, following the simplification and refinement methods, a post process smoothing and projection routine is executed. This better distributes the inserted

elements, improves quadrilateral quality and reduces the remesh approximation error.

6.1.5 Preliminary Studies

The refinement operators, polychord pillowing and insertion, introduce ideal vertices aligned to the user-specified mesh edges. As such, the improvement scheme operates independently of the underlying geometry, allowing the user to create the appropriate polychord structure. Application of the improvement method to multiple regions defined over the model will allow a global improvement of the entire mesh structure. In this vision, the user individually paints regions of the model, identifying cylindrical regions where the looping polychord is straightforward. The improvement algorithm run separately for each region isolates extraordinary vertices within the unpainted regions of the model.

While this improvement scheme is guided by user input, the final remeshes exhibit characteristics of path tracing remeshing schemes that are guided by curvature tensor fields [7, 107, 126, 20]. In practice, the selected mesh edges typically align to low frequency curvature vectors, looping principal curvature directions of cylindrical elements. Furthermore, the regions that remain unpainted during the editting process typically correspond the area surrounding curvature vector singularities. This results in an isolation of extraordinary vertices to similar regions as the path tracing methods, while producing large expanses of structured elements.

The goal of the improvement algorithm is to be applied globally over the model; however, two important aspects limit the current approach. First, the simplification procedure removes elements based on their impact on mesh connectivity and area of collapse, unconcerned with the distance to the inserting polychord wavefronts. Applying the described method globally over the model, may result in significant degradation of the model far removed from the inserting wavefronts. Second, the currently described system is unable to process multiple seed edge loops that initiate the insertion process. While support of the additional looping polychords may be straightforward, some consideration of their inter-relationship is critical. For instance, it may become important to provide a means to describe a correspondence between neighboring seed edge loops to align the elements and prevent twisting artifacts. Furthermore, based on the parallels to the curvature-based remeshing, discussed earlier, it may be possible to completely automate this improvement algorithm, leveraging both path tracing methods to identify ideal polychords, a correspondance between them, and regions ideal for extraordinary vertices,

while connectivity-based operators can robustly improve the mesh structure.

6.2 Summary

The work outlined in this chapter further motivates the connectivity-based theme using simplification and refinement methods, as well as the study of the polychord structures. The results illustrate the effectiveness of polychord deletions and insertions for interactive mesh improvement. The work demonstrated throughout this chapter represents preliminary results and raises interesting questions and challenges. In particular, the methods do not consider the preservation and recreation of important feature curves over the model, discussed throughout this dissertation as sharp features. The generalization of these techniques to hexahedral remeshing presents a feasible and attractive direction for future research.

CHAPTER 7

CONCLUSIONS

The contributions of this research span multiple tasks affiliated with surface reconstruction, developing techniques that bridge the gap between discrete and continuous geometry. At face value, these topics address continuous feature curves defined over point-based geometry and surface remeshing and parameterization through simplification of quadrilateral models. A meaningful contribution achieved through these studies is the understanding of the dual representation for quadrilateral meshes and the advantages of structure imposed on this form.

Beyond the continuation of the preliminary research explored in Chapter 6, because quadrilateral meshing is increasing in popularity, there exist many interesting avenues of pursuit within the area. While this work stresses the robustness of connectivity-based approaches, it has yet to significantly consider the surface geometry in its operations. Drawing from lessons of anisotropic-based path tracing [7, 107, 20] and variational shape approximation methods [155, 108], the intelligent placement of extraordinary vertices, located at regions of singularities in curvature tensor fields and in regions of high curvature, improves mesh descriptions. Integration of connectivity-based algorithms and numerical-based path tracing schemes may prove robust while augmenting attribute awareness in the final meshes.

7.1 Constraining the Structure

It has been observed in this research, particularly through the global-based simplification, that the structure of the dual representation is important to defining and creating high quality mesh elements. While semiregular remeshes define a local structure that is advantageous for many geometry processing algorithms, there is also the consideration of the global structure related to the dual representation. As illustrated in Figure 4.7, many semiregular models define dual polychords with knots that complicate quadrilateral-based simplification and refinement applications where maintaining mesh structure and quality

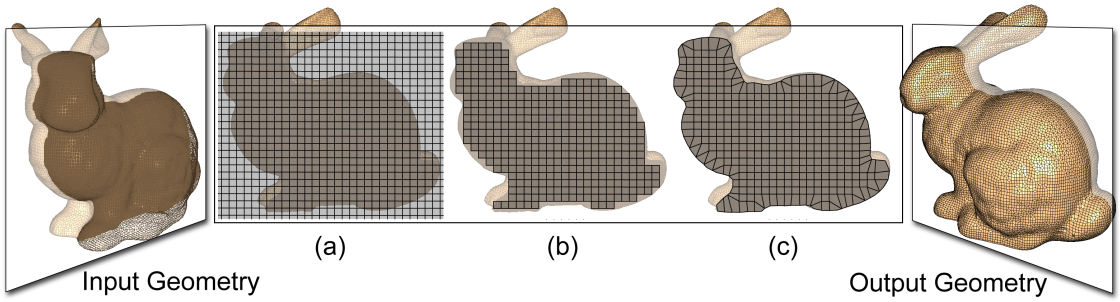
is important.

Polycube maps [160] produce seamless textures for arbitrary objects by extending the ideas of cube mapping, used in environment mapping and texturing cube-like surfaces. The polycube map describes a set of axis aligned cubes stacked together in a way that resembles the original model, capturing large geometric features. The original polycube mapping requires user interaction to model the input geometry. More recently it has been shown that the Reeb graph is useful in designing a decomposition of the model that can guide the polycube map construction [101]. Because the two shapes are relatively close in general shape, mapping the original model onto the polycube domain describes a parameterization with low distortion. By virtue of the quadrilateral base domain, polycube maps can guide quadrilateral-based remeshing and parameterization [169].

The global structure of the polycube map is unique. A polychord of the axis aligned cube configuration concisely loops over the polycube model along the boundary of a hexahedral dual sheet [21]. In contrast to a generic semiregular mesh, semiregular meshes with constrained polychords that exhibit a polycube-like structure may facilitate hexahedral mesh generation techniques that are based on input quadrilateral meshes [153], as well as simplification and refinement processing of surface models [24].

Expansion of the ideas introduced throughout this dissertation, in concert with techniques based on *surface grafting* [151] or *dual contouring* [79, 133], may lead to the robust, connectivity-based development of polycube remeshes. Surface grafting generates structured quadrilateral meshes of arbitrary orientable manifolds, requiring only a projection operator. Illustrated in Figure 7.1, the method embeds the surface within a uniform hexahedral grid (a), culls away exterior (completely or partially) voxels (b), then projects the remaining exterior nodes to the surface (c).

The polychords of a grafted surface describe a concise loops around the model, demonstrated in a related voxel-based remeshing scheme [24]. The polychords correspond to the intersection of hexahedral dual sheets that correspond to the rows, columns and layers of the underlying hexahedral grid. The variant of surface grafting implements polychord deletions to remove many extraordinary vertices from grafted quadrilateral models. While this work focuses on a single minded task of reducing extraordinary vertices, further improvement to the approach may consider the allowance of select extraordinary vertices, using persistence-based weighting methods to intelligently guide deletions and insertions.



7.1: The surface grafting algorithm remeshes an input geometry by embedding it within a uniform grid (a), culling external voxels (b), and projecting the boundary nodes to the surface (c).

7.2 Future Research

Surface models play a critical role in many facets of computer graphics where the ability to interact with geometry is vital. For instance, entertainment driven animations, isosurface extraction for medical imaging, manufacturing design, planning, and reverse engineering, physically-based simulations, (non)photorealistic rendering and scientific visualization, among many other applications, leverage computations on virtual representations of real world objects. To offer the different functionality required by the variety of driving applications, a plethora of surface representational forms have been created.

The trade offs assorted with the different surface model types are associated with the structure of the underlying connectivity. For instance, discrete connectionless point samples are often utilized for the reconstruction and visualization of virtual models generated from range scanners; however, exact mathematical computations of shape are impossible. In contrast, spline-based models, requiring highly structured quadrilateral control meshes, provide continuous surface definitions and exact differential operations that are instrumental in manufacturing and computer aided design applications. Polygonal meshes balance the trade offs between computational power and discrete approximation, supporting fast computations and error bounded representations.

The research I present outlines a fundamental tool in the pursuit of a *representational independent modeling* (RIM) *system*. Such a system would enable the creation of an attribute aware surface model of any desired structural format, generated from a virtual model that may be potentially composed of multiple heterogenously represented components. Support of error bounded conversions between the assorted modeling formats greatly increases the number of available algorithms with which a designer may interact

with a virtual model. As opposed to redesigning geometry processing methods, originally created for a specific structural format, for all model types, a task that is neither straightforward nor necessarily possible, the RIM system would generate the appropriate model structure for the desired processing task.

A semiregular quadrilateral mesh, by virtue of its relationship to spline-based modeling, presents an attractive representational format to operate as the hub of the described conversion network. However, before such a system is recognized, further research is required. Development of the algorithms that form the error bounded conversion network is paramount, and means to track modifications made to the model by external geometry processes is important. The improved alignment and adaptive sampling of semiregular meshes and support of direct and intuitive controls for user manipulation of the semiregular, quadrilateral-only models is a critical task. The generalization of these techniques for hexahedral models is both daunting and important. The solution to these challenges, developed within a coherent software package, will revolutionize the way in which we operate on virtual models.

REFERENCES

[1] Ahn, M., Guskov, I., and Lee, S. Out-of-core remeshing of large polygonal meshes. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1221–1228.

[2] Alexa, M. Merging polyhedral shapes with scattered features. In *IEEE Shape Modeling and Applications* (1999), pp. 202–210.

[3] Alexa, M., and Adamson, A. On normals and projection operators for surfaces defined by point sets. In *Symposium on Point-Based Graphics* (2004), pp. 149–155.

[4] Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., and Silva, C. T. Point set surfaces. In *IEEE Visualization* (2001), pp. 21–28.

[5] Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., and Silva, C. T. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics 9*, 1 (2003), 3–15.

[6] Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., and Silva, C. T. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics 9*, 1 (2003), 3–15.

[7] Alliez, P., Cohen-Steiner, D., Devillers, O., Lévy, B., and Desbrun, M. Anisotropic polygonal remeshing. In *ACM SIGGRAPH* (2003), pp. 485–493.

[8] Alliez, P., Cohen-Steiner, D., Tong, Y., and Desbrun, M. Voronoi-based variational reconstruction of unoriented point sets. In *ACM Symposium on Geometry Processing* (2007), pp. 39–48.

[9] Amenta, N., and Bern, M. Surface reconstruction by voronoi filtering. *Discrete and Computational Geometry 22*, 4 (December 1999), 481–504.

[10] Amenta, N., Bern, M., and Kamvysselis, M. A new voronoi-based surface reconstruction algorithm. In *Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), ACM Press, pp. 415–421.

[11] Amenta, N., Choi, S., and Kolluri, R. The power crust, union of balls and the medial axis transform. *Computational Geometry 19*, 2 (July 2001), 127–153.

[12] Amenta, N., and Kil, Y. J. Defining point-set surfaces. *ACM Transactions on Graphics 23*, 3 (2004), 264–270.

[13] Attene, M., Falcidieno, B., Rossignac, J., and Spagnuolo, M. Sharpen-bend: Recovering curved sharp edges in triangle meshes produced by feature-insensitive sampling. *IEEE Transactions on Visualization and Computer Graphics 11*, 2 (2005), 181–192.

[14] Attene, M., Falcidieno, B., Spagnuolo, M., and Wyvill, G. A mapping-independent primitive for the triangulation of parametric surfaces. *Graphical Models 65*, 5 (2003), 260–273.

[15] Balmelli, L., Morris, C., Taubin, G., and Bernardini, F. Volume warping for adaptive isosurface extraction. In *IEEE Visualization* (2002), pp. 467–474.

[16] Bernardini, F., Mittleman, J., Rushmeier, H., and nad Gabriel Taubin, C. S. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualizations and Computer Graphics 5*, 4 (October–December 1999), 349–359.

[17] Biermann, H., Martin, I., Bernardini, F., and Zorin, D. Cut-and-paste editing of multiresolution surfaces. *ACM Transactions on Graphics 21*, 3 (2002), 312–321.

[18] Blacker, T., and Stephenson, M. Paving: A new approach to automated quadrilateral mesh generation. *Numerical Methods in Engineering* (May 1991).

[19] Boissonant, J.-D. Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics 3*, 4 (October 1984).

[20] Bommes, D., Zimmer, H., and Kobbelt, L. Mixed-integer quadrangulation. In *ACM SIGGRAPH* (2009).

[21] Borden, M., Benzley, S., and Shepherd, J. Hexahedral sheet extraction. In *International Meshing Roundtable* (September 2002), pp. 147–152.

[22] Botsch, M., and Kobbelt, L. A remeshing approach to multiresolution modeling. *ACM Symposium on Geometry Processing* (2004), 189–196.

[23] Bow, S. *Pattern Recognition and Image Preprocessing*. Marcel Dekker Inc., New York, NY, 1992.

[24] Bremer, P. T., Porumbescu, S. D., Hamann, B., and Joy, K. I. *Automatic Semi-Regular Mesh Construction from Adaptive Distance Fields*. 2002.

[25] Canny, J. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 8*, 6 (1986), 679–698.

[26] Carr, J. C., Beatson, R. K., Cherrie, J. B., Mitchell, T. J., Fright, W. R., McCallum, B. C., and Evans, T. R. Reconstruction and representation of 3d objects with radial basis functions. In *ACM SIGGRAPH* (2001), pp. 67–76.

[27] Cass, R., Benzley, S., Meyers, R., and Blacker, T. Generalized 3d paving: An automated quadrilateral surface mesh generation algorithm. *Numerical Methods in Engineering 39* (1998), 1475–1498.

[28] Catmull, E., and Clark, J. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer Aided Geometric Design 10*, 6 (1978), 350–355.

[29] Chew, P. Guaranteed quality mesh generation for curved surfaces. In *In Proceedings of the 9th Annual Symposium on Computational Geometry* (1993), pp. 274–280.

[30] Cohen-Steiner, D., and Da, F. A greedy delaunay-based surface reconstruction algorithm. *The Visual Computer 20*, 1 (April 2004), 4–16.

[31] Daniels, J., and Cohen, E. Surface creation and curve deformation between two closed complex spatial spline curves. *Geometric Modeling and Processing* (2006).

[32] Daniels, J., Cohen, E., and Johnson, D. Converting molecular meshes into smooth interpolatory spline solid models. *ASME: Computers and Information in Engineering* (2005).

[33] Daniels, J., Ha, L., Ochotta, T., and Silva, C. Robust smooth feature extraction from point clouds. *Shape and Modeling International* (2007).

[34] Daniels, J., Ochotta, T., Ha, L., and Silva, C. Spline-based feature curves from point-sampled geometry. *The Visual Computer* (2008).

[35] Daniels, J., Silva, C., Shepherd, J., and Cohen, E. Quadrilateral mesh simplification. *ACM SIGGRAPH Asia* (2008).

[36] Demarsin, K., Vanderstraeten, D., Volodine, T., and Roose, D. Detection of closed sharp feature lines in point clouds for reverse engineering applications. Tech. Rep. TW 458, Department of Computer Science, K.U.Leuven, Belgium, 2006.

[37] Desbrun, M., Meyer, M., and Alliez, P. Intrinsic parameterizations of surface meshes. *Computer Graphics Forum 21* (2002), 209–218.

[38] Desbrun, M., Meyer, M., Schröder, P., and Barr, A. H. Implicit fairing of irregular meshes using diffusion and curvature flow. *Computer Graphics 33* (1999), 317–324.

[39] Desbrun, M., Meyer, M., Schroder, P., and Barr, A. H. Implicit fairing of irregular meshes using diffusion and curvature flow. In *ACM SIGGRAPH* (1999), pp. 317–324.

[40] Dewey, M. Automated quadrilateral coarsening by ring collapse. Master's thesis, Bringham Young University, 2008.

[41] Dey, T. K., and Goswami, S. Tight cocone: a water-tight surface reconstructor. In *ACM Solid Modeling and Applications* (2003), pp. 127–134.

[42] Dong, S., Bremer, P.-T., Garland, M., Pascucci, V., and Hart, J. C. Spectral surface quadrangulation. In *ACM SIGGRAPH* (2006), pp. 1057–1066.

[43] Dong, S., Kircher, S., and Garland, M. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Computer Aided Geometric Design 22*, 5 (2005), 392–423.

[44] Du, Q., Faber, V., and Gunzburger, M. Centroidal voronoi tesselations: Applications and algorithms. *SIAM Review 41*, 4 (1999), 637–676.

[45] Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., and Stuetzle, W. Multiresolution analysis of arbitrary meshes. In *ACM SIGGRAPH* (1995), pp. 115–123.

[46] Eck, M., and Hoppe, H. Automatic reconstruction of b-spline surfaces of arbitrary topological type. In *ACM SIGGRAPH* (1996), vol. 30, pp. 325–334.

[47] Edelsbrunner, H. *Geometry and Topology for Mesh Generation*. Cambridge University Press, New York, NY, USA, 2006.

[48] Edelsbrunner, H., and Guoy, D. Sink-insertion for mesh improvement. In *In Proceedings of the 17th Annual Symposium on Computational Geometry* (2001), pp. 115–123.

[49] Eldar, Y., Lindenbaum, M., Porat, M., and Zeevi, Y. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing 6*, 9 (1997), 1305–1315.

[50] Field, D. Laplacian smoothing and delaunay triangulations. *Communications in Applied Numerical Methods 4*, 6 (June 2005), 709–712.

[51] Fleishman, S., Cohen-Or, D., Alexa, M., and Silva, C. T. Progressive point set surfaces. *ACM Transactions on Graphics 22*, 4 (2003).

[52] Fleishman, S., Cohen-Or, D., and Silva, C. T. Robust moving least-squares fitting with sharp features. *ACM Transactions on Graphics 24*, 3 (2005), 544–552.

[53] Floater, M. S. Parameterization and smooth approximation of surface triangulations. *Computer Aided Geometric Design 14*, 3 (1997), 231–250.

[54] Floater, M. S. Mean value coordinates. *Computer Aided Geometric Design 20*, 1 (2003), 19–27.

[55] Floater, M. S., and Hormann, K. Surface parameterization: a tutorial and survey. In *Advances in multiresolution for geometric modelling*, N. A. Dodgson, M. S. Floater, and M. A. Sabin, Eds. Springer Verlag, 2005, pp. 157–186.

[56] Garland, M., and Heckbert, P. Surface simplification using quadric error metrics. In *ACM SIGGRAPH* (1997), pp. 209–216.

[57] Garland, M., and Heckbert, P. Simplifying surfaces with color and texture using quadric error metrics. In *IEEE Visualization* (1998), pp. 263–270.

[58] Garland, M., and Heckbert, P. S. Surface simplification using quadric error metrics. In *ACM SIGGRAPH* (1997), pp. 209–216.

[59] Garland, M., Willmott, A., and Heckbert, P. Hierarchical face clustering on polygonal meshes. In *Symposium on Interactive 3D Graphics* (New York, NY, USA, 2001), ACM Press, pp. 49–58.

[60] Glimm, J., Grove, J. W., Li, X. L., ming Shyue, K., Zeng, Y., and Zhang, Q. Three-dimensional front tracking. *SIAM Journal on Scientific Computing 19*, 3 (1998), 703–727.

[61] Gopi, M., Krishnan, S., and Silva, C. Surface reconstruction based on lower dimensional localized delaunay triangulation. *Computer Graphics Forum 19*, 3 (September 2000), 467–478.

[62] Gotsman, C., Gu, X., and Sheffer, A. Fundamentals of spherical parameterization for 3d meshes. *ACM Transactions on Graphics 22*, 3 (2003), 358–363.

[63] Gu, X., Gortler, S., and Hoppe, H. Geometry images. *ACM SIGGRAPH* (2002), 355–361.

[64] Gu, X., He, Y., Jin, M., Luo, F., Qin, H., and Yau, S.-T. Manifold splines with single extraordinary point. In *ACM Solid and Physical Modeling* (2007), pp. 61–72.

[65] Gu, X., He, Y., and Qin, H. Manifold splines. In *ACM Solid and Physical Modeling* (2005), pp. 27–38.

[66] Gumhold, S., Wang, X., and McLeod, R. Feature extraction from point clouds. In *International Meshing Roundtable* (2001).

[67] Guskov, I., Vidimce, K., Sweldens, W., and Schroeder, P. Normal meshes. In *ACM SIGGRAPH* (2000), pp. 95–102.

[68] He, Y., Wang, K., Wang, H., Gu, X., and Qin, H. Manifold t-spline. *Geometric Modeling and Processing 4077* (2006).

[69] Hildebrandt, K., Polthier, K., and Wardetzky, M. Smooth feature lines on surface meshes. In *ACM Symposium on Geometry Processing* (2005), pp. 85–90.

[70] Hoppe, H. New quadric metric for simplifiying meshes with appearance attributes. In *IEEE Visualization* (1999), pp. 59–66.

[71] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. Mesh optimization. *Computer Graphics* (August 1993), 19–26.

[72] Hoppner, F., Klawonn, F., Kruse, R., and Runkler, T. *Fuzzy Clustering Analysis: Methods for Classification, Data Analysis and Image Recognition.* John Wiley and Sons, New York, NY, USA, 1999.

[73] Hormann, K., and Greiner, G. Mips: An efficient global parameterization method. *Curve and Surface Design* (2000), 153–162.

[74] Huang, J., Zhang, M., Ma, J., Liu, X., Kobbelt, L., and Bao, H. Spectral quadrangulation with orientation and alignment control. *ACM SIGGRAPH Asia* (2008).

[75] Hubeli, A., and Gross, M. Multiresolution feature extraction for unstructured meshes. In *In Proceedings of IEEE Visualization* (2001), pp. 287–294.

[76] Jenke, P., Wand, M., Bokeloh, M., Schilling, A., and Strasser, W. Bayesian point cloud reconstruction. *Computer Graphics Forum 25*, 3 (2006), 379–388.

[77] Jeong, W. K., and Kim, C. H. Direct reconstruction of displaced subdivision surface from unorganized points. *Graphical Models 64*, 2 (2002), 78–93.

[78] Jones, T., Durand, F., and Desbrun, M. Non-iterative, feature-preserving mesh smoothing. *ACM Transactions on Graphics* (2003), 943–949.

[79] Ju, T., Losasso, F., Schaefer, S., and Warren, J. Dual contouring of hermite data. In *ACM SIGGRAPH* (2002), pp. 339–346.

[80] Kalaiah, A., and Varshney, A. Statistical point geometry. In *ACM Symposium on Geometry processing* (2003), pp. 107–115.

[81] Kalberer, F., Nieser, M., and Polthier, K. Quadcover: Surface parameterization using branched coverings. *Computer Graphics Forum 26*, 3 (September 2007), 375–384.

[82] Kass, M., Witkin, A., and Terzopoulos, D. Snakes: Active contour models. *International Journal of Computer Vision 1* (1988), 321–331.

[83] Katz, S., and Tal, A. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transaction on Graphics 22*, 3 (2003), 954–961.

[84] Kharevych, L., Springborn, B., and Schroder, P. Discrete conformal mappings via circle patterns. *ACM Transactions on Graphics 25*, 2 (2006).

[85] Khodakovsky, A., Litke, N., and Schroder, P. Globally smooth parameterizations with low distortion. *ACM Transactions on Graphics 22*, 3 (2003), 350–357.

[86] Khodakovsky, A., Schroder, P., and Sweldens, W. Progressive geometry compression. In *ACM SIGGRAPH* (2000), pp. 271–278.

[87] Kinney, P. Cleanup: Improving quadrilateral finite element meshes. In *International Meshing Roundtable* (1997), pp. 437–447.

[88] Kobbelt, L. 3-subdivision. In *ACM SIGGRAPH* (2000), pp. 103–112.

[89] Kobbelt, L., Vorsatz, J., Labsik, U., and Seidel, H. A shrink wrapping approach to remeshing polygonal surface. In *Computer Graphics Forum* (1999), vol. 18, pp. 119–130.

[90] Kobbelt, L., Vorsatz, J., Labsik, U., and Seidel, H.-P. A shrink wrapping approach to remeshing polygonal surfaces. *Computer Graphics Forum 18*, 3 (1999), 119–130.

[91] Krishnamurthy, V., and Levoy, M. Fitting smooth surface to dense polygon meshes. In *ACM SIGGRAPH* (1996), pp. 313–324.

[92] Lai, Y.-K., Kobbelt, L., and Hu, S.-M. An incremental approach to feature aligned quad dominant remeshing. In *ACM Solid and Physical Modeling* (2008).

[93] Lee, A., Dobkin, D., Sweldens, W., and Schroeder, P. Multiresolution mesh morphing. In *ACM SIGGRAPH* (1999), pp. 343–350.

[94] Lee, A. W. F., Sweldens, W., Schröder, P., Cowsar, L., and Dobkin, D. Maps: multiresolution adaptive parameterization of surfaces. In *ACM SIGGRAPH* (New York, NY, USA, 1998), ACM, pp. 95–104.

[95] Lee, I.-K. Curve reconstruction from unorganized points. *Computer Aided Geometric Design 17*, 2 (2000), 161–177.

[96] Lee, Y., Kim, H. S., and Lee, S. Mesh parameterization with a virtual boundary. *Computers and Graphics 26*, 5 (2002), 677–686.

[97] Levin, D. The approximation power of moving least-squares. *Mathematics of Computation 67*, 224 (1998), 1517–1531.

[98] Levin, D. *Geometric Modeling for Scientific Visualization*. Springer-Verlag, 2003, ch. Mesh-independent surface interpolation, pp. 37–49.

[99] Levy, B., Petitjean, S., Ray, N., and Maillot, J. Least squares conformal maps for automatic texture atlas generation. *ACM SIGGRAPH* (2002), 362–371.

[100] Li, W. C., Ray, N., and Levy, B. Automatic and interactive mesh to t-spline conversion. In *ACM Symposium on Geometry Processing* (2006).

[101] Lin, J., Jin, X., Fan, Z., and Wang, C. Automatic polycube maps. *Geometric Modeling and Processing* (2008), 3–16.

[102] Lin, J. L., Chuang, J. H., Lin, C. C., and Chen, C. C. Consistent parameterization by quinary subdivision for remeshing and mesh metamorphosis. In *In Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Austalasia and South East Asia* (2003), pp. 151–158.

[103] Lloyd, S. Least square quantization in pcm. *IEEE Transactions on Information Theory 28* (1982), 129–137.

[104] Loop, C., and Schaefer, S. Approximating catmull-clark subdivision surfaces with bicubic patches. *ACM Transactions on Graphics 27* (March 2008).

[105] Lorensen, W., and Cline, H. Marching cubes: A high resolution 3d surface construction algorithm. In *Computer Graphics and Interactive Techniques* (New York, NY, USA, 1987), ACM Press, pp. 163–169.

[106] Lucchese, L., and Mitra, S. Color image segmentation: A state-of-the-art survey. In *In Proceedings of the Indian National Science Academy of Image Processing, Vision, and Pattern Recognition* (2001), vol. 67, pp. 207–221.

[107] Marinov, M., and Kobbelt, L. Direct anisotropic quad-dominant remeshing. In *In Proceedings of Pacific Conference on Computer Graphics and Applications* (October 2004), pp. 207–216.

[108] Marinov, M., and Kobbelt, L. A robust two-step procedure for quad-dominant remeshing. *Computer Graphics Forum 25*, 3 (September 2006), 537–546.

[109] McInerney, T., and Terzopolous, D. T-snakes: Topology adaptive snakes. *Medial Image Analysis 4*, 2 (June 2000), 73–91.

[110] Mederos, B., Amenta, N., Velho, L., and de Figueiredo, L. H. Surface reconstruction from noisy point clouds. *ACM Symposium on Geometry Processing* (2005).

[111] Meyer, M., Kirby, R. M., and Whitaker, R. Topology, accuracy, and quality of isosurface meshes using dynamic particles. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (September-October 2007).

[112] Meyer, M. D., Georgel, P., and Whitaker, R. T. Robust particle systems for curvature dependent sampling of implicit surfaces. In *Shape and Modeling International* (2005).

[113] Miller, G. A time efficient delaunay refinement algorithm. In *In Proceedings of the 15th Annual Symposium on Discrete Algorithms* (2004), pp. 400–409.

[114] Murdoch, P., Benzley, S., Blacker, T., and Mitchell, S. The spatial twist continuum: A connectivity based method for representing all-hexahedral finite element meshes. *Finite Element in Analysis and Design 28*, 2 (December 1997), 137–149.

[115] Ni, X., Garland, M., and Hart, J. C. Fair morse functions for extracting the topological structure of a surface mesh. In *ACM SIGGRAPH* (2004), pp. 613–622.

[116] Nielson, G., and Hamann, B. The asymptotic decider: Resolving the ambiguity in marching cubes. In *IEEE Visualization* (October 1991), pp. 83–91,413.

[117] Ochotta, T., and Saupe, D. Compression of point-based 3D models by shape-adaptive wavelet coding of multi-height fields. In *ACM Point-Based Graphics* (2004), pp. 103–112.

[118] Owen, S., Staten, M., Canann, S., and Saigal, S. Q-morph: An indirect approach to advancing front quad meshing. *Numerical Methods in Engineering 44*, 9 (March 1999), 1317–1340.

[119] Pal, N., and Pal, S. A review on image segmentation techniques. *Pattern Recognition 26*, 9 (1993), 1277–1294.

[120] Pauly, M., Gross, M., and Kobbelt, L. P. Efficient simplification of point-sampled surfaces. In *IEEE Visualization* (2002), pp. 163–170.

[121] Pauly, M., Keiser, R., and Gross, M. Multi-scale feature extraction on point-sampled surfaces. *Computer Graphics Forum 22*, 3 (2003), 281–290.

[122] Peng, J., Strela, V., and Zorin, D. A simple algorithm for surface denoising. In *IEEE Visualization* (2001), pp. 107–112.

[123] Peyre, G., and Cohen, L. D. Geodesic remeshing using front propagation. *International Journal of Computer Vision 69* (2006).

[124] Praun, E., and Hoppe, H. Spherical parameterization and remeshing. In *ACM SIGGRAPH* (2003), pp. 340–349.

[125] Praun, E., Sweldens, W., and Schroder, P. Consistent mesh parameterization. In *ACM SIGGRAPH* (2001), pp. 179–184.

[126] Ray, N., Li, W. C., Levy, B., Sheffer, A., and Alliez, P. Periodic global parameterization. *ACM Transactions on Graphics 25*, 4 (2006), 1460–1485.

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

[128] Sahoo, P. K., Soltani, S., Wong, A. K., and Chen, Y. C. A survey of thresholding techniques. *Computer Vision, Graphics and Image Processing 41*, 2 (1988), 233–260.

[129] Said, A., and Pearlman, W. A. An image multiresolution representation for lossless and lossy image compression. *IEEE Transactions on Image Processing 5*, 9 (1996), 1303–1310.

[130] Sander, P., Snyder, J., Gortler, S., and Hoppe, H. Texture mapping progressive meshes. pp. 409–416.

[131] Sander, P., Wood, Z., Gortler, S., Snyder, J., and Hoppe, H. Multi-chart geometry images. In *ACM Symposium on Geometry Processing* (2003), pp. 246–255.

[132] Sapiro, G. Color snakes. *Computer Vision and Image Understanding 68*, 2 (1997), 247–253.

[133] Schaefer, S., Ju, T., and Warren, J. Manifold dual contouring. *IEEE Transactions on Visualization and Computer Graphics 13*, 3 (2007), 610–619.

[134] Scheidegger, C., Fleishman, S., and Silva, C. Triangulating point set surfaces with bounded error. In *ACM Symposium of Geometry Processing* (2005).

[135] Schreiner, J., Asirvatham, A., Praun, E., and Hoppe, H. Inter-surface mapping. In *ACM SIGGRAPH* (2004).

[136] Schreiner, J., Scheidegger, C., and Silva, C. Direct (re)meshing for efficient surface processing. *Computer Graphics Forum*, 3 (2006), 527–536.

[137] Schreiner, J., Scheidegger, C., and Silva, C. High-quality extraction of isosurfaces from regular and irregular grids. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (September–October 2006), 1205–1212.

[138] Sederberg, T. W., Cardon, D. L., Finnigan, G. T., North, N. S., Zheng, J., and Lyche, T. T-spline simplification and local refinement. In *ACM SIGGRAPH* (New York, NY, USA, 2004), ACM, pp. 276–283.

[139] Shafarenko, L., Petrou, M., and Kittler, J. Histogram-based segmentation in perceptually uniform color space. In *IEEE Transactions on Image Processing* (1998), vol. IP-7, pp. 1354–1358.

[140] Sharf, A., Lewiner, T., Shamir, A., Kobbelt, L., and Cohen-Or, D. Competing fronts for coarse-to-fine surface reconstruction. *Computer Graphics Forum 25*, 3 (2006), 389–398.

[141] Sheffer, A., Levy, B., Mogilnitsky, M., and Bogomyakov, A. Abf++: Fast and robust angle based flattening. *ACM Transactions on Graphics 24*, 2 (2005), 311–330.

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

[143] Shekhar, R., Fayyad, E., Yagel, R., and Cornhill, F. Octree-based decimation of marching cubes surfaces. In *IEEE Visualization* (1996).

[144] Shepherd, J. *Topologic and Geometric Constraint-Based Hexahedral Mesh Generation*. PhD thesis, School of Computing, University of Utah, May 2007.

[145] Shimada, K. *Physically-based mesh generation: automated triangulation of surfaces and volumes via bubble packing*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1993.

[146] Shimada, K. Quadrilateral meshing with directionality control via close packing of square cells. *SIAM Conference on Geometric Modeling* (1999).

[147] Shimada, K., and Gossard, D. C. Bubble mesh: automated triangular meshing of non-manifold geometry by sphere packing. In *ACM Solid Modeling and Applications* (1995), pp. 409–419.

[148] Shimada, K., Liao, J.-H., and Itoh, T. Quadrilateral meshing with directionality control through the packing of square cells. In *International Meshing Roundtable* (1998), pp. 61–76.

[149] Sifri, O., Sheffer, A., and Gotsman, C. Geodesic-based surface remeshing. In *International Meshing Roundtable* (2003), pp. 189–199.

[150] Smith, J., and Boier-Martin, I. Combining metrics for mesh simplification and parameterization. In *ACM SIGGRAPH Sketches* (2005), p. 135.

[151] S.R. Jankovich, S.E. Benzley, J. S., and Mitchell, S. The graft tool: An all-hexahedral transition algorithm for creating a multi-directional swept volume mesh. In *International Meshing Roundtable* (1999), pp. 387–392.

[152] Staten, M., Benzley, S., and Scott, M. A methodology for quadrilateral finite element mesh coarsening. *Engineering with Computers* (2008), 241–251.

[153] Staten, M., Owen, S., and Blacker, T. Unconstrained paving and plastering: A new idea for all hexahedral mesh generation. In *14th International Meshing Roundtable* (2005), pp. 399–416.

[154] Staten, M. L., and Canann, S. A. Post refinement element shape improvement for quadrilateral meshes. *ASME AMD: Trends in Unstructured Mesh Generation* (1997), 9–16.

[155] Steiner, D. C., Alliez, P., and Desbrun, M. Variational shape approximation. In *ACM SIGGRAPH* (2004), pp. 905–914.

[156] Surazhsky, V., Alliez, P., and Gotsman, C. Isotropic remeshing of surfaces: a local parameterization approach. In *International Meshing Roundtable* (2003).

[157] Surazhsky, V., and Gotsman, C. Explicit surface remeshing. In *ACM Symposium on Geometry Processing* (2003), pp. 20–30.

[158] Szeliski, R., and Tonnesen, D. Surface modeling with oriented particle systems. In *International Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1992), ACM Press, pp. 185–194.

[159] Takeuchi, S., Suzuki, H., Kimura, F., Kanai, T., and Shimada, K. Subdivision surface fitting with qem-based mesh simplification and reconstruction of approximated b-spline surfaces. In *In Proceedings of the 8th Pacific Conference on Computer Graphics and Applications* (2000).

[160] Tarini, M., Hormann, K., Cignoni, P., and Montani, C. Polycube-maps. In *ACM SIGGRAPH* (New York, NY, USA, 2004), ACM, pp. 853–860.

[161] Taubin, G. Curve and surface smoothing without shrinkage. *IEEE International Conference on Computer Vision 0* (1995), 852.

[162] Tong, Y., Alliez, P., Cohen-Steiner, D., and Desbrun, M. Designing quadrangulations with discrete harmonic forms. In *ACM Symposium on Geometry Processing* (2006), pp. 201–210.

[163] Tristano, J., Owen, S., and Canann, S. Advancing front surface mesh generation in parametric space using a riemannian surface definition. In *International Meshing Roundtable* (1998).

[164] Turk, G. Re-tiling polygonal surfaces. In *ACM SIGGRAPH* (1992), pp. 55–64.

[165] Tutte, W. T. How to draw a graph. *London Mathematical Society 13* (1963), 743–768.

[166] Uccheddu, F., Corsini, M., and Barni, M. Wavelet-based blind watermarking of 3d models. In *International Multimedia Conference* (2004), pp. 143–154.

[167] Viswanath, N., Shimada, K., and Itoh, T. Quadrilateral meshing with anisotropy and directionality control via close packing of rectangular cells. In *International Meshing Roundtable* (2000), pp. 227–238.

[168] Vorsatz, J., Rossl, C., and Seidel, H.-P. Dynamic remeshing and applications. In *In Proceedings of ACM Symposium on Solid Modeling and Applications* (New York, NY, USA, 2003), ACM, pp. 167–175.

[169] Wang, H., He, Y., Li, X., Gu, X., and Qin, H. Polycube splines. In *ACM Solid and Physical Modeling* (New York, NY, USA, 2007), ACM, pp. 241–251.

[170] Waschbusch, M., Gross, M., Eberhard, F., Lamboray, E., and Wurmlin, S. Progressive compression of point-sampled models. In *Symposium on Point-Based Graphics* (2004), pp. 95–102.

[171] Watanabe, K., and Belyaev, A. G. Detection of salient curvature features on polygonal surfaces. *Computer Graphics Forum 20*, 3 (2001), 385–392.

[172] Watson, D. Computing the n-dimensional delaunay tessellation with application to voronoi polytopes. *The Computer Journal 24* (1981), 167–172.

[173] White, D., and Kinney, P. Redesign of the paving algorithm: Robustness enhancements through element by element meshing. In *International Meshing Roundtable* (1997), pp. 323–335.

[174] Witkin, A., and Heckbert, P. Using particles to sample and control implicit surfaces. In *Computer Graphics and Interactive Techniques* (New York, NY, USA, 2005), ACM Press.

[175] Woo, H., Kang, E., Wang, S., and Lee, K. H. A new segmentation method for point cloud data. *International Journal of Machine Tools and Manufacture 42* (January 2002), 167–178.

[176] Xu, R., and II, D. W. Survey of clustering algorithms. *IEEE Transactions on Neural Networks 16*, 3 (May 2005), 645–678.

[177] Yagou, H., Ohtake, Y., and Belyaev, A. Mesh smoothing via mean and median filtering applied to face normals. In *Geometric Modeling and Processing* (2002), p. 124.

[178] Yang, M., and Lee, E. Segmentation of measured point data using a parameteric quadric surface approximation. *Computer Aided Geometric Design 31*, 7 (June 1999), 449–457.

[179] Zayer, R., Rossl, C., and Seidel, H. Setting the boundary free: A composite appraoch to surface parameterization. *ACM Symposium on Geometry Processing* (2005), 91–100.

[180] Zhang, Y., Bajaj, C., and Guoliang, X. Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow. In *14th International Meshing Roundtable* (2005), pp. 449–468.

[181] Zhang, Y., Bajaj, C., and Guoliang, X. Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow. *Numerical Methods in Engineering* (November 2007).