Connectivity Control for Quad-Dominant Meshes

by

Chi-Han Peng

A Dissertation Presented in Partial Fulfillment
of the Requirement for the Degree
Doctor of Philosophy

Approved October 2014 by the
Graduate Supervisory Committee:

Peter Wonka, Co-Chair
Ross Maciejewski, Co-Chair
Gerald Farin
Anshuman Razdan

ARIZONA STATE UNIVERSITY

December 2014

ABSTRACT

Quad-dominant (QD) meshes, i.e., three-dimensional, 2-manifold polygonal meshes comprising mostly four-sided faces (i.e., quads), are a popular choice for many applications such as polygonal shape modeling, computer animation, base meshes for spline and subdivision surface, simulation, and architectural design. This thesis investigates the topic of *connectivity control*, i.e., exploring different choices of mesh connectivity to represent the same 3D shape or surface. One key concept of QD mesh connectivity is the distinction between *regular* and *irregular* elements: a vertex with valence 4 is regular; otherwise, it is irregular. In a similar sense, a face with four sides is regular; otherwise, it is irregular. For QD meshes, the placement of irregular elements is especially important since it largely determines the achievable geometric quality of the final mesh.

Traditionally, the research on QD meshes focuses on the automatic generation of pure quadrilateral or QD meshes from a given surface. Explicit control of the placement of irregular elements can only be achieved indirectly. To fill this gap, in this thesis, we make the following contributions. First, we formulate the theoretical background about the fundamental combinatorial properties of irregular elements in QD meshes. Second, we develop algorithms for the explicit control of irregular elements and the exhaustive enumeration of QD mesh connectivities. Finally, we demonstrate the importance of connectivity control for QD meshes in a wide range of applications.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my adviser Prof. Peter Wonka. His insights, passions, work ethics, and the sheer amounts of intelligence deeply inspires me, and is indeed the main driving force behind the fruition of this thesis. In addition, I would like to thank him for the numerous life-changing opportunities he has given to me, including the research scholarship at ASU, the visits to KAUST, TU Vienna, and UCL, and the collaborations with world-renowned experts in many disciplines.

Besides my adviser, I would like to thank the rest of my thesis committee: Prof. Ross Maciejewski, Prof. Anshuman Razdan, Prof. Gerald Farin, and Prof. Yalin Wang, for their insightful comments and helpful assistances.

My sincere thanks go to Prof. Eugene Zhang. We collaborated on a wonderful and fruitful research project, and during the process I learned a lot from his rigorous theoretical background. I cherish the collaboration with Prof. Yoshihiro Kobayashi, whose interdisciplinary expertises and perspectives are invaluable to me.

I would like to thank Prof. Helmut Pottmann, Prof. Niloy Mitra, Prof. Michael Wimmer, and Prof. Leonidas Guibas, for their invaluable advises and helps to my research. I would also like to express my gratitude to the co-authors of the papers stemmed from this thesis: Michael Barton, Caigui Jiang, and Yong-liang Yang. I appreciate my time at Adobe ATL and the mentoring of Dr. Nathan Carr and Dr. Radomir Mech.

I would like to thank the helps from Fan Bao, Dongming Yan, Xiang Sun, Chengcheng Tang, Jun Wang, Christopher Grasso, Virginia Unkefer, Qi-Xing Huang, Prof. Alla Sheffer, David Bommes, Alexander Schiftner, Prof. Yu-Kun Lai, Prof. Leif Kobbelt, Martin Marinov, Prof. Hao Zhang, Tom Kelly, Cheng Pan, Yifan Zhang, Feng Wang, Yafeng Lu, and many others. I gratefully acknowledge supports by the National Science Foundation, Adobe, and KAUST visiting sponsorship.

Finally, my fullest gratitudes are reserved for my wonderful family - my parents, my brother, my amazing wife Lydia Lee, and my amusing son Leo Peng, and Jesus Christ my shepherd.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

Polygonal meshes are important representations of geometric surfaces. Such representations are based on the idea of *cell decomposition*: a continuous, geometric surface is represented with an assembly of (possibly many) simple polygonal cells. In particular, quad-dominant (QD) meshes, which comprise mostly 4-sided faces (i.e., quads), are extensively studied. They are a popular choice for many applications such as polygonal shape modeling (Southern (2011)), computer animation, base meshes for spline and subdivision surface [Panozzo and Puppo (2010); Stam and Loop (2003); Myles *et al.* (2008); Sederberg *et al.* (2003, 2004)], simulation [D'Azevedo (2000); Shepherd and Johnson (2008)], and architectural design [Liu *et al.* (2006); Pottmann *et al.* (2008); Eigensatz *et al.* (2010); Zadravec *et al.* (2010); Yang *et al.* (2011)].

This thesis investigates the topic of *connectivity control*, i.e., exploring different choices of mesh connectivity to represent the same 3D shape or surface. For QD meshes, this is an especially important issue since the achievable geometric quality of the final mesh is largely determined by the mesh connectivity chosen in the first place. Examples are shown in Figure 1.1 and 1.2. The reasons are given next.

To begin with, a major reason behind QD meshes' popularity is due to the fact that there exists a natural matching between QD meshes and cross fields, i.e., an assignment of a pair of dominant directions to each surface point. In a QD mesh, a vertex is typically adjacent to four edges, or two pairs of opposing edges, that can be naturally aligned to the pair of directions in the cross field. The same can be said about the sides of a face. To generate high quality QD meshes, the mesh lines should align to the cross field of the underlying principal curvature directions.

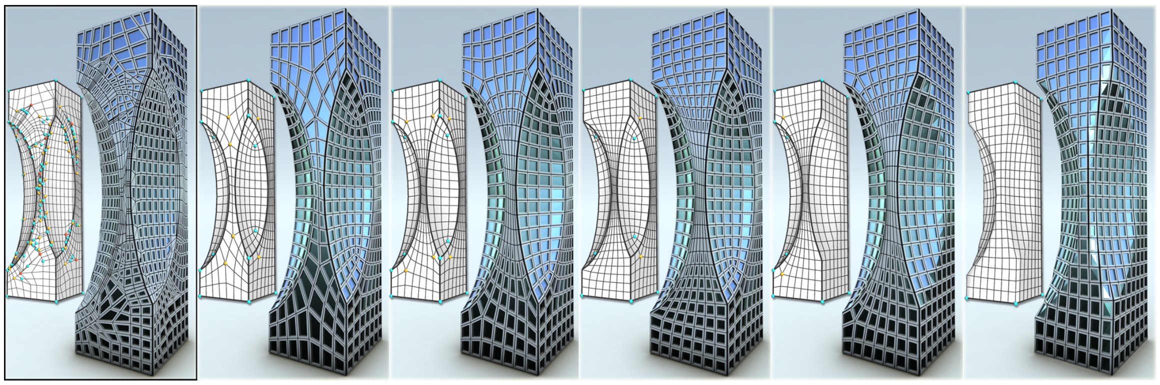


Figure 1.1: Comparing Different Ways to Present the Same Architectural Tower Model as Pure Quadrilateral Meshes.

Left: the input mesh with a large number of irregular vertices and small faces. Second left to right: five requadrangulations of the input mesh. The first and second both consider the hyperboloid-shaped facade on the front as an 8-sided patch, while the former favors uniform quad size and the latter favors alignment of irregular vertices. The third alternatively considers the front facade as a 4-sided patch, and four valence-3 irregular vertices are removed. The fourth no longer preserves the sharp feature on the right, and the two adjacent patches are merged. It has fewer irregular vertices at the cost of less sharp feature fidelity. The fifth discards all sharp features on the front in exchange for even fewer irregular vertices.

It follows that one important concept when dealing with QD meshes is the distinction between *regular* and *irregular* elements (vertices and faces): a vertex with valence 4 is regular; otherwise, it is irregular. In a similar sense, a face is regular if it has 4 sides; otherwise, it is irregular. An irregular element can no longer align with a cross field. Rather, it corresponds to a field singularity of the matching index. For example, given a cross field of principal curvature directions, elements with degree lower than 4 are best positioned at areas with positive Gaussian curvatures (i.e., elliptical regions), and elements with degree higher than 4 are best positioned at areas with negative Gaussian curvatures (i.e., hyperbolic regions).

Irregular vertices are also important for other reasons. It has been shown that irregular vertices are necessary at transition regions between the denser and sparser parts of the mesh Panozzo and Puppo (2010); Myles *et al.* (2010). Irregular vertices

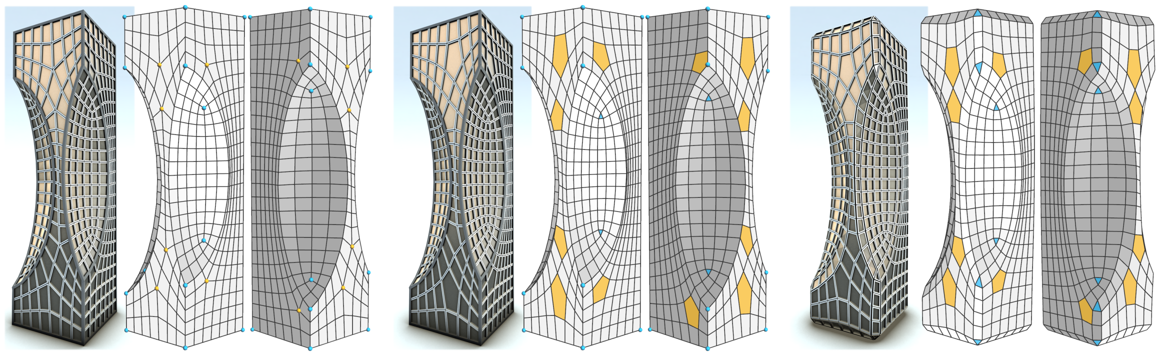


Figure 1.2: Comparing Irregular Vertices and Irregular Faces in the Context of Design and Construction.

In general, irregular vertices are necessary to maintain sharp features (corners and edges), but they create higher angle deviations in mesh lines in smooth regions (left). Irregular faces lead to smoother mesh lines, but they cannot maintain sharp features (right). The ability to model with a mixture of irregular vertices and faces would give more flexibility to the user, e.g., creating a design with sharp features and smooth mesh lines (middle).

also determine the layout of *separatrices* in a QD mesh. That is, edge strips connecting one irregular vertex to another (possibly the same) irregular vertex or mesh boundary. Separatrices partition the mesh into several regular 2D arrays of quads that can be mapped into the 2D plane to obtain a parameterization. To minimize the number of regular patches, irregular vertices should be positioned in the way that the separatrices are aligned as much as possible Bommes *et al.* (2011); Tarini *et al.* (2011); Campen *et al.* (2012).

Traditionally, the research about QD meshes focuses on the automatic generation of QD [Alliez *et al.* (2003); Marinov and Kobbelt (2006); Ray *et al.* (2006)] or pure quadrilateral [Kälberer *et al.* (2007); Bommes *et al.* (2009); Zhang *et al.* (2010)] meshes from a given surface, i.e., QD or quad remeshing. This is largely a parameterization problem and is usually tackled by optimization approaches driven by curvatures. For a broader background on this topic we suggest the survey by Bommes et al. [Bommes *et al.* (2012)]. For automatic generation methods, control of the placement of irregular vertices is usually achieved indirectly by tweaking the optimization parameters [Bommes *et al.* (2009)] or editing the field singularities [Zhang *et al.* (2006); Tong *et al.* (2006); Palacios and Zhang (2007); Ray *et al.* (2006, 2009)]. The process is often slow and unintuitive.

On the other hand, research on the connectivity aspects of QD meshes has been relatively sparse. Examples include connectivity editing operators for quad strips [Daniels *et al.* (2008); Bommes *et al.* (2011)] and certain types of irregular elements [Maza *et al.* (1999); Tarini *et al.* (2010)], and combinatorial methods to quadrangulate an empty region with specific goals [Schaefer *et al.* (2004); Nasri and Yasseen (2009); Bessmeltsev *et al.* (2012); Takayama *et al.* (2013)]. However, none of these offer a systematic analysis and editing framework for the connectivity control of QD meshes.

To fill this gap, the aim of this thesis is to develop theories and algorithms for the understanding and direct control of QD mesh connectivity, with the following contributions:

- We present a novel study about the fundamental combinatorial properties and the explicit control of irregular vertices in pure quadrilateral meshes (Chapter 3).

- We extend the study to general QD meshes, of which irregular elements include both vertices and faces (Chapter 4).

- We present a method for the efficient enumeration of all possible quadrangulations (i.e., pure quadrilateral connectivities) with an upper bound of the number of irregular vertices (Chapter 5).

The research results have led to the following technical papers (myself as the first author):

- Connectivity Editing for Quadrilateral Meshes. Chi-Han Peng, Eugene Zhang, Yoshihiro Kobayashi, and Peter Wonka. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH ASIA 2011). Peng *et al.* (2011); Peng and Wonka (2013).

- Connectivity Editing for Quad-Dominant Meshes. Chi-Han Peng and Peter Wonka. Eurographics Symposium on Geometry Processing (SGP) 2013. Peng and Wonka (2013).

- Exploring Quadrangulations. Chi-Han Peng, Michael Barton, Caigui Jiang, and Peter Wonka. ACM Transactions on Graphics (presented at SIGGRAPH 2014). Peng *et al.* (2014a).

- Computing Layouts with Deformable Templates. Chi-Han Peng, Yong-Liang Yang, and Peter Wonka. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2014). Peng *et al.* (2014b).

A detailed discussion about related work is given in the next chapter.

Chapter 2

RELATED WORK

Polygonal mesh processing: The majority of research about polygonal mesh processing is for triangle meshes and usually focuses on preserving geometric details in the underlying surface or achieving optimal aspect ratio and size of triangles in the mesh under a reasonable face count budget [Hoppe (1996); Desbrun *et al.* (1999); Alliez *et al.* (2002)]. For a complete reference of past work in mesh processing we refer the readers to [Botsch *et al.* (2010)].

QD **Mesh Generation:** There has been much recent work in the automatic generation of a pure quadrilateral or QD mesh given an input triangular mesh. Typical approaches include tracing evenly spaced hyperstreamlines [Alliez *et al.* (2003); Marinov and Kobbelt (2004); Dong *et al.* (2005); Bauer *et al.* (2010)], constructing a global parameterization [Ray *et al.* (2006); Kälberer *et al.* (2007); Bommes *et al.* (2009); Zhang *et al.* (2010); Bommes *et al.* (2013); Panozzo *et al.* (2014)], and generating a patch layout on the surface that facilitates quadrilateral remeshing [Dong *et al.* (2006); Tong *et al.* (2006); Nieser *et al.* (2010b); Myles *et al.* (2010); Bessmeltsev *et al.* (2012); Takayama *et al.* (2013)]. In general, the generation of pure quadrilateral meshes is more challenging than generating general QD meshes because discrete constraints are involved to ensure the resulting mesh is free of non-quad faces (see [Bommes *et al.* (2009)] for a discussion). For approaches depending on hyperstreamline tracing and global parameterizations, a guiding 4-way rotational symmetry (4-RoSy) field is needed. The quality of the remeshes depends on the quality of the fields, and this has led to work on generating geometry-aware fields either manually [Zhang *et al.* (2006, 2007); Palacios and Zhang (2007); Ray *et al.* (2008)] or automatically [Ray *et al.* (2009); Nieser *et al.* (2010a)]. Control of the placement of irregular vertices is usually achieved indirectly by tweaking the optimization parameters [Bommes *et al.* (2009)] or editing the field singularities [Zhang *et al.* (2006); Tong *et al.* (2006); Palacios and Zhang (2007); Ray *et al.* (2006, 2009)]. The process is often slow and unintuitive. Moreover, controlling singularities and irregular elements have different degrees of freedom, see [Palacios and Zhang (2007)].

Contrary to the aforementioned field-guided methods, there has also been some work in formulating combinatoric algorithms to quadrangulate an empty region with specific goals such as having fewest irregular vertices possible [Schaefer *et al.* (2004); Nasri and Yasseen (2009); Bessmeltsev *et al.* (2012); Takayama *et al.* (2013)]. Our quadrangulation enumeration method ([Peng *et al.* (2014a)]) shares the same motivation.

Alternative *QD* mesh generation methods include approaches work by vertex alignments [Tchon and Camarero (2006); Lai *et al.* (2008)], space partitioning (octree [Marchal (2009)] or quadtree [Frey and Marchal (1998)]), or use segmentation as an important ingredient [Marinov and Kobbelt (2006)]. The generated meshes can also satisfy additional requirements, e.g., remeshing with planar quads [Liu *et al.* (2006); Zadravec *et al.* (2010). Advancing fronts (i.e., paving) methods [Blacker and Stephenson (1991); White and Kinney (1997); Park *et al.* (2007)] incrementally grow quad faces from the patch boundaries and they could be used for finding and enumerating pure quadrilateral topologies. However, one problem is that the algorithm does not terminate without a geometric heuristic and can grow fronts indefinitely.

QD **Mesh Connectivity:** There has been some work that stressed the importance of irregular element control, as the number, location and type of irregular elements are often intrinsically linked to the geometric features on the surface [Ray *et al.* (2006)]. See [Akleman and Chen (2006)] for a discussion on this topic. Irregular elements are also important in many other aspects. The distribution of irregular elements can impact the quality of inverse subdivision [Taubin (2002)]. As mentioned previously, irregular elements are necessary at transition regions between the denser and sparser parts of the mesh [Panozzo and Puppo (2010); Myles *et al.* (2010)]. For pure quadrilateral meshes, irregular vertices also determine the layout of *separatrices*, i.e., edge strips connecting one irregular vertex to another (possibly the same) irregular vertex or mesh boundary. Separatrices partition the mesh into several regular 2D arrays of quads that can be mapped into the 2D plane to obtain a parameterization. To minimize the number of regular patches, irregular vertices should be positioned in the way that the separatrices are aligned as much as possible [Bommes *et al.* (2011); Tarini *et al.* (2011); Campen *et al.* (2012)].

An early example of operators for the control of irregular elements in QD meshes can be found in [Maza *et al.* (1999)], where a pair of triangles and a pair of valence-3 and valence-5 irregular vertices at diagonal positions can be moved together. Note that these are a subset of all possible pair-wise irregular element movement operators proposed in our studies [Peng *et al.* (2011); Peng and Wonka (2013)]. Local operators have been proposed for pure quadrilateral mesh simplification [Tarini *et al.* (2010)]. Surprisingly, however, local operators such as quad collapse, edge split, vertex rotation, and edge flip all increase the number of irregular vertices when applied to a single irregular vertex. Therefore, it is important to investigate operators on larger regions that do not increase the number of irregular vertices. Examples include operators on quad strips [Daniels *et al.* (2008)], and so called GP operators [Bommes *et al.* (2011)], which operate on non-aligned quad strips. We can find similar concepts in related fields. In subdivision and spline surfaces, operators for editing T-junctions in T-spline control meshes has been proposed [Sederberg *et al.* (2004)]. In crystallography, Burgers vectors have been used to displace irregular vertices [Hull and Bacon

(2001)].

Architectural Geometry: An important venue of applications for QD meshes, as well as inspirations for many parts of this research, lies in the field of architectural geometry. It is because QD meshes, especially the special class of planar quad (PQ) meshes of which most faces are quad and close to planar, possess several favorable properties for architecture (see the discussion in Glymph *et al.* (2004)). A strong emphasis have been put on the generation of PQ meshes and variants for free-form architectures [Liu *et al.* (2006); Zadravec *et al.* (2010); Pottmann *et al.* (2007b); Schiftner and Balzer (2010); Liu *et al.* (2011); Jiang *et al.* (2014); Tang *et al.* (2014)]. For a detailed discussion of architectural geometry we refer the readers to [Pottmann *et al.* (2007a)].

Chapter 3

CONNECTIVITY EDITING FOR QUADRILATERAL MESHES

Quadrilateral and hexahedral meshes are popular choices in simulation and shape modeling due to the natural tensor product property that they possess. Quadrilateral meshes can also facilitate architectural modeling as well as texture and geometry synthesis. Important aspects of a quadrilateral mesh include the location, orientation, type, and number of irregular vertices. While there has been some work in quad mesh connectivity editing Daniels *et al.* (2008); Bommes *et al.* (2011), achieving irregular vertex control is challenging and many questions about what editing operations are possible and impossible still need to be answered.

In this chapter, we propose three operations that move an irregular vertex pair (two valence 3, two valence 5, or one valence 3 and one valence 5) over the mesh. To show that these are fundamental operations for quad mesh editing, we will establish the following properties:

- These editing operations impact the smallest possible region on the mesh and are therefore as local as possible (in a convex region).
- A region containing only one irregular vertex cannot be edited.
- A region containing two irregular vertices can be edited by changing the location of the irregular vertices within the region. However, they cannot be canceled. Some irregular vertex pairs can be merged while others cannot, depending on their graph distance in the initial configuration.
- A region with three irregular vertices can be edited by canceling or merging the irregular vertices.

- Our three movement operations can perform all possible edits within a (convex) region that contains two irregular vertices.

While the three vertex pair movement operations are at the core of this chapter, we also introduce several supplementary operations to control the type and number of irregular vertices: splitting, merging, cancellation, and alignment. All of these operations can be realized through three graph-level editing operations: quad collapses, edge flips, and edge splits.

We do not extensively deal with the geometric consequence of mesh editing. We consider feature edges in our implementation and analysis, but otherwise treat editing operations that result in isomorphic graphs as identical. Our analysis is based on recent work in triangular mesh editing Li *et al.* (2010). However, we make the contribution of providing the enumeration (and parameterization) of all valid requadrangulations given a regular convex region. Furthermore, while Li et al. Li *et al.* (2010) demonstrate that it is possible to move an irregular vertex pair, we establish the condition under which this is possible with an explicit algorithm to requadrangulate a region.

3.1 Chapter Overview

The input to our system is a quadrilateral mesh M that represents a closed manifold surface. The *valence* of a vertex v in M, which we denote as $l(v)$, is the number of edges in the mesh incident to v. A vertex with a valence of n is denoted as vn, e.g., $v3$ and $v5$. A $v4$ vertex is considered as *regular*, and vertices of other valences are referred to as *irregular*. A pair of valence m and n vertices is denoted as $m-n$ pair.

To simplify the discussion, we only consider irregular vertices with a valence of 3 or 5. Other irregular vertices can be transformed to multiple $v3$ or $v5$ vertices through our atomic type change operations (Section 3.6).

The focus of this chapter is to introduce and analyze the following three operations: $3-5$, $3-3$, and $5-5$ pair movement operations. To realize these operations, we develop a *three-level hierarchy* of editing operations: basic operations, atomic semantic operations, and the three pair-wise movement operations.

1. Basic operations (Section 3.2) include quad collapse, edge split, and edge flip. All semantic operations are achieved by a combination of these three types of operations.

2. Atomic semantic operations (Section 3.3) are single basic operations with semantic interpretations. Users can generate and remove irregular vertices as well as move, split, and merge adjacent irregular vertices.

3. Pair-wise movement operations (Section 3.4) include the aforementioned $3-5$, $3-3$, and $5-5$ pair movement operations. These operations are accomplished by multiple atomic semantic operations.

In Section 3.5 we provide theoretical analysis of the proposed operations. We first prove that it is impossible to requadrangulate a convex region containing a single irregular vertex without introducing additional irregular vertices. Consequently, possible local operations require at least two irregular vertices. By analyzing all possible connectivity edits for a pair of irregular vertices within a convex region, we conclude that the three proposed irregular vertex pair movement operations are sufficient to generate all possible local edits within a convex region containing two irregular vertices.

In Section 3.6 we present several useful operations built upon the three pair-wise movement operations. These additional operations can be used to reduce the number of irregular vertices, align $3-3$ and $5-5$ pairs, and merge one $3-3$ or $5-5$ pair into one $v2$ or $v6$ vertex. The aforementioned editing operations are *topological* by design and may lead to a loss of geometric details in the underlying surface and quadrilaterals with poor aspect ratios. We provide several geometric operations to remedy such problems. In Section 3.7 we compare our editing operations to the irregular vertex editing operations for triangular meshes proposed in Li *et al.* (2010). In Section 3.8 several applications of our editing operations are shown.

3.2 Basic Operations

Our set of basic operations has the following desirable properties: First, the support for these operations is local. Second, their implementations are relatively easy with a low computational cost. Third, it is straightforward to combine multiple basic operations, incurring no limitations and special cases (Figure 3.1).

Quad Collapse: Quad collapse can be intuitively understood as merging a pair of diagonally opposing vertices (v_1, v_2) sharing the same face. The face they share will be deleted, and the valence of the merged vertex will be $l(v_1) + l(v_2) - 2$, while the valence of the other pair of diagonally opposing vertices will be decreased by one per vertex.

Edge Split: Edge split can be intuitively understood as bloating a pair of connected edges (e_1, e_2) into one face and the central vertex v between them is split into two. After an edge split, a new face and a new vertex will be created. The pair of edges (e_1, e_2) separates the remaining edges incident to v_1 into two groups containing d_1 and d_2 edges, respectively. After the edge split, the valences of the two vertices are $d_1 + 2$ and $d_2 + 2$, respectively. The valence of the other two vertices involved in the operation will be increased by one.

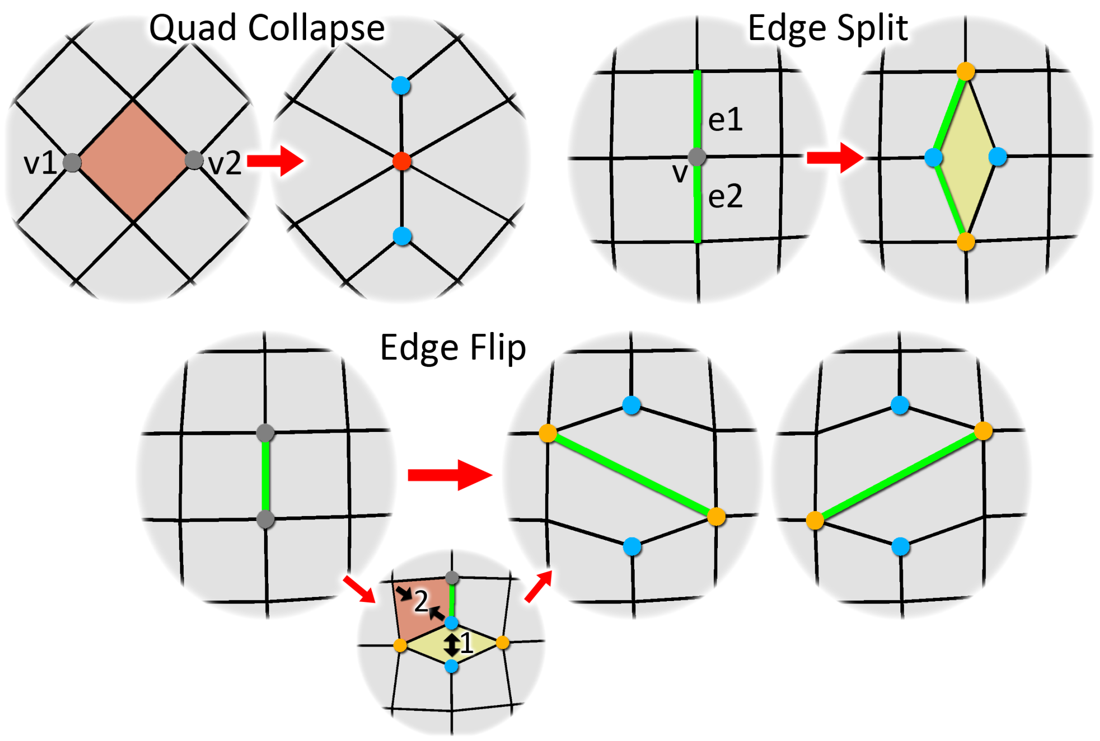


Figure 3.1: Basic Operations.
Quad collapse: a pair of diagonally opposing regular vertices (gray) is merged. The face to be deleted is shown in red. The $v3$ vertices are shown in blue and the $v6$ vertex in red. Edge split: a pair of connected edges (green) is bloated into a face (yellow). The $v5$ vertices are shown in orange. Edge flip: an edge (green) is flipped in a counter-clockwise and clockwise direction. One way to realize an edge flip is by one edge split followed by one quad collapse (shown below).

Edge Flip: Edge flip can be intuitively understood as rotating an edge in either the counter-clockwise or clockwise direction. The valence of the two vertices on the edge will be decreased by one, while the valence of the other two involved vertices will be increased by one.

Quad collapse and edge split are inverse to each other; they are both atomic in the sense that each cannot be realized by any combination of the other two basic operations. On the other hand, the inverse of an edge flip is an edge flip in the other direction; it is not atomic because it can be realized by one edge split and one quad collapse.

3.3 Atomic Semantic Operations

The influence of the aforementioned basic operations on the valence is non-trivial and lacks semantic meaning, which makes it difficult to use them directly to control irregular vertices. By providing semantic interpretations for basic operations we define a collection of atomic semantic operations.

Adjacent $3-5$ Pair Movement: A pair of adjacent $v3$ and $v5$ vertices can be moved in the direction of its six adjacent vertices (Figure 3.2).

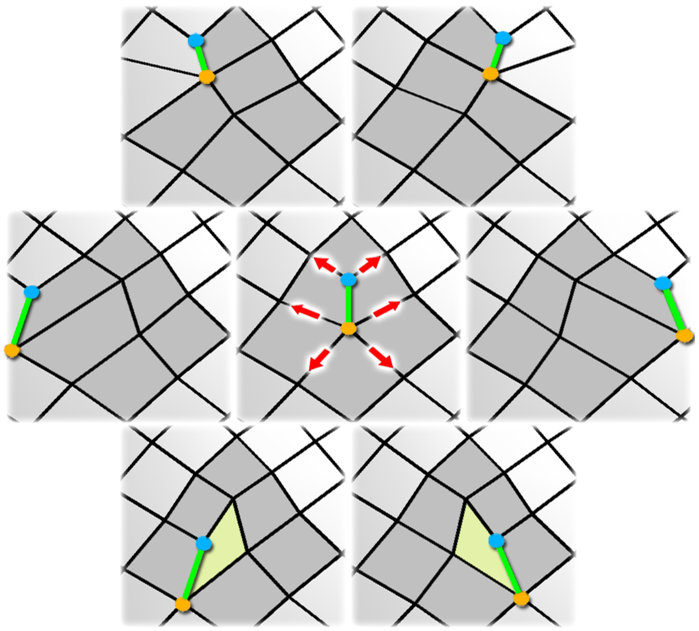


Figure 3.2: Possible Directions of an Adjacent $3-5$ Pair Movement Operation. Moving to the upper-left and upper-right corners are achieved by one quad collapse. Moving to the lower-left and lower-right corners is achieved by one edge split. Moving to the left and right is achieved by one edge flip. Faces adjacent to the $3-5$ pair are shown in gray to assist comparisons. In the lower-left and lower-right cases faces created by edge split are shown in yellow.

$v3(v5)$ Movement and $3-5$ Pair Generation: A $v3$ vertex can be moved to one of its adjacent locations, and one adjacent $3-5$ pair is created. Each direction can be achieved by two kinds of quad collapses and two kinds of edge flips. Similarly, a $v5$ vertex can be moved to one of its five adjacent locations, and one adjacent $3-5$

pair is created. Each direction can be achieved by two kinds of edge splits and two kinds of edge flips. Figure 3.3 shows one possible moving direction for a $v3$ and a $v5$ vertex.

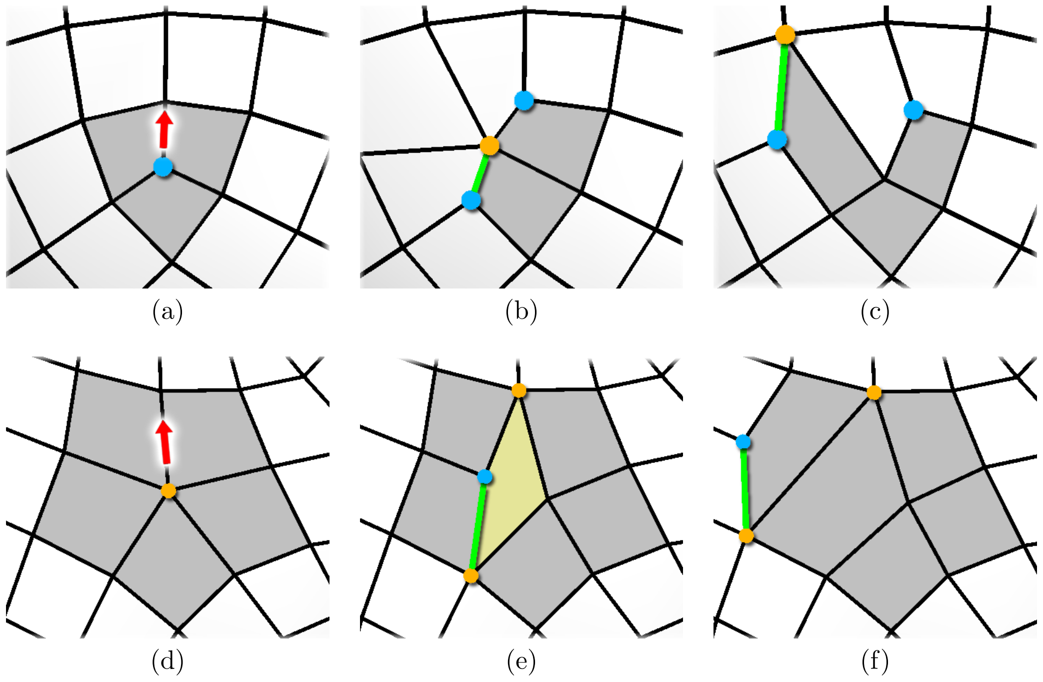


Figure 3.3: Examples of Moving an Irregular Vertex Upward from its Current Location.

(a) to (c): the $v3$ vertex is moved upward by either (b) collapsing the face on the left or (c) flipping the left edge clockwise. It can also be achieved by collapsing the face on the right or flipping the right edge counter-clockwise. (d) to (f): the $v5$ vertex is moved upward by either (e) splitting the edge pair on the left or (f) flipping the left edge counter-clockwise. It can also be achieved by splitting the edge pair on the right or flipping the right edge clockwise. Note that in all these scenarios a $3-5$ irregular vertex pair is created as a result of the movement operation.

$v3(v5)$ **Movement and** $3-5$ **Pair Removal:** This operation is the inverse of the operation described above. It can be understood by reading Figure 3.3 in the reverse direction.

$3-3-5-5$ **generation/removal:** By one edge flip two $v3$ and two $v5$ vertices can be generated. Inversely two $v3$ and two $v5$ vertices can be removed by one edge flip in the other direction.

Type Change Operations: Type change operations increase or decrease vertex valences. As mentioned before, all irregular vertices other than $v3$ or $v5$ can be transformed into a set of $v3$ or $v5$ irregular vertices by using type changing operations.

A vertex's valence can be increased by applying a quad collapse to one of its adjacent faces, at the cost of creating adjacent irregular vertices. For example, applying a quad collapse to a $v2$ vertex with a regular diagonally opposing vertex can transform the $v2$ vertex into a regular vertex, while decreasing valence of the other pair of diagonally opposing vertices by one (Figure 3.4a). Similarly, a vertex's valence can be decreased by applying an edge split to one of the adjacent edge pairs between it, at the cost of creating adjacent irregular vertices. For example, splitting a $v6$ vertex along an edge pair that evenly separates its six adjacent edges will create two regular new vertices, while increasing the other two vertices' valence by one (Figure 3.4b).

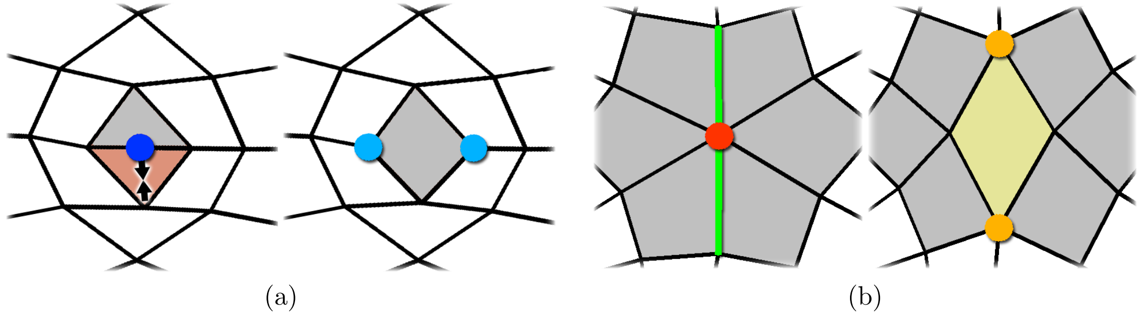


Figure 3.4: Applying Type Change Operations to Convert $v2$ and $v6$ Vertices into $v3$ and $v5$ Vertices.

(a) A $v2$ vertex is converted to two $v3$ vertices by one quad collapse. (b) A $v6$ vertex is converted to two $v5$ vertices by one edge split.

3.4 Pair-wise Movement Operations

While the aforementioned atomic operations only have local influences, we can move non-adjacent pairs of irregular vertices ($3-5$, $3-3$, and $5-5$) by a combination of multiple atomic operations. We also show how the pair can be moved even when the shortest path between the vertices intersects some feature edge that has to be preserved. These operations result in quads being inserted into and deleted from the mesh. Therefore, we also need to interleave smoothing operations with these operations to avoid degrading the mesh quality.

$3-5$ **Pair Movement:** A non-adjacent $3-5$ pair can be moved together in the same direction in the following three-step pipeline:

1. Move the $v3$ vertex to a user-specified adjacent location by applying a $v3$ movement and $3-5$ pair generation operation.

2. Apply multiple adjacent $3-5$ pair movement operations to transport the generated $3-5$ pair toward the $v5$ vertex until they become adjacent.

3. Apply a $v5$ and $3-5$ pair removal operation to remove the $3-5$ pair and shift the $v5$ vertex. The relative position of the two vertices remains the same.

Alternatively the pipeline can be executed reversely by moving the $v5$ vertex first and then colliding the generated $3-5$ pair toward the $v3$ vertex. Each movement has four moving directions, which can be understood as moving on a 2D Cartesian grid with a nearby regular vertex as the origin. After one movement the relative distance between the $3-5$ pair, defined by the number of edges of their two connecting separatrices (Definition 3.5.7) will be preserved. In Figure 3.5 the four moving directions for $3-5$ pairs in different configurations are analyzed.

$3-5$ **Pair Movement with Sharp Features:** The transportation of the generated $3-5$ pair in the second step does not need to follow the shortest path between the irregular vertices. Figure 3.6 illustrates this with an example. If the generated $3-5$ pair is transported through the shortest path, the sharp feature will be modified. Alternatively, the generated $3-5$ pair can be transported through a longer path to avoid modifying the sharp feature.

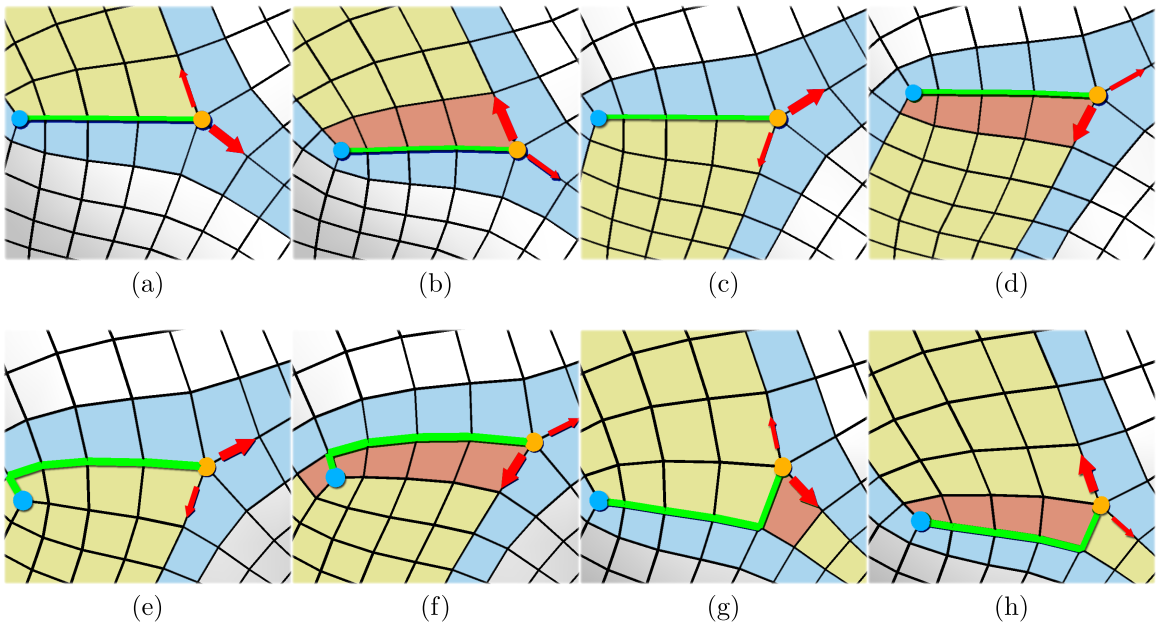


Figure 3.5: Moving a $3-5$ Pair.
The four moving directions (red arrows) for $3-5$ pairs with aligned (top row) and misaligned (bottom row) separatrices. Each green line denotes a shortest path between the pair. The blue faces denote the nearest unchanged quad strips that enclose the affected region. The yellow and red faces are generated or deleted depending on which direction the $3-5$ pair moves, while red faces denote the ones that are immediately to be created or deleted. The length of the connecting path determines the number of faces created or deleted in one movement.

$3-3$ **Pair Movement:** The mechanics of moving a non-adjacent $3-3$ irregular vertex pair is identical to moving a $3-5$ pair. There are also four directions of movement. The major difference is that the graph distance between the two irregular vertices changes. We define the number of edges on each of the two connecting separatrices as d_1 and d_2 (See Figure 3.7). If the two vertices are directly connected by a separatrix one of these two values is equal to zero. One step of movement can be labeled by the changes to (d_1, d_2) in one of the four possible ways: $(+1,+1)$, $(+1,-1)$, $(-1,+1)$, and $(-1,-1)$.

$5-5$ **Pair Movement:** This operation is similar to moving a $3-3$ pair in that there are four directions of movement and the graph distance between the vertices

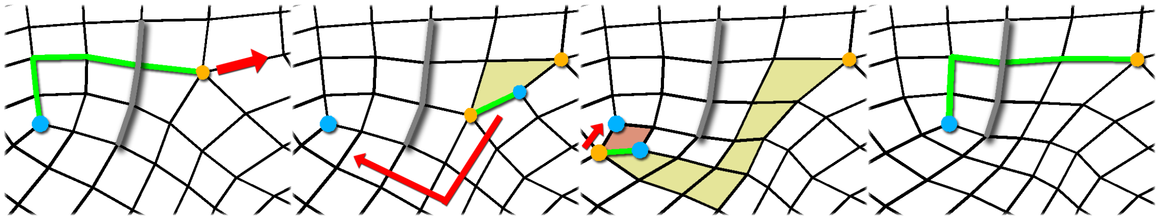
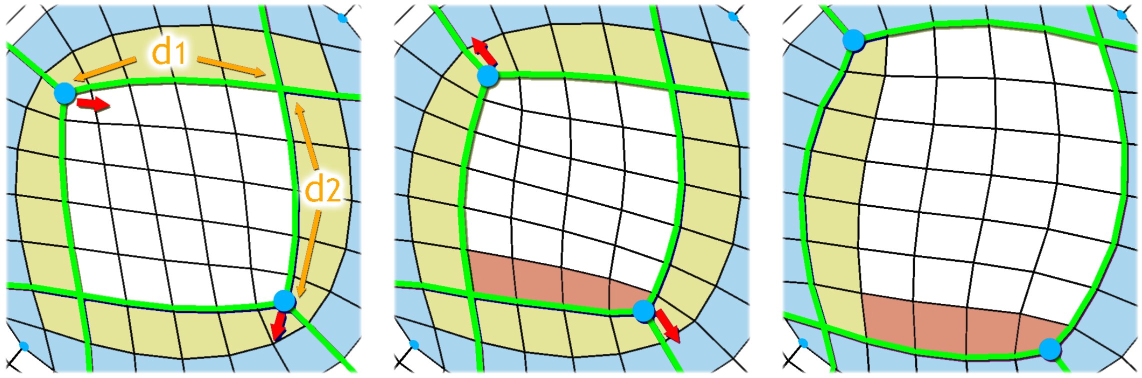

(a) (b) (c) (d)

Figure 3.6: Moving a $3-5$ Pair Whose Shortest Path Intersects Sharp Features.
Faces generated in the process are shown in yellow and faces deleted are shown in red. (a) The connecting path, shown in green, intersects the sharp feature shown in grey. (b) to (d) The generated $3-5$ pair is transported toward the $v3$ vertex through an alternate path that avoids the sharp feature.



(a) (b) (c)

Figure 3.7: Examples of $3-3$ Pair Movement.
(a) to (b) The $3-3$ pair is moved in the $(-1,+1)$ direction (shown as red arrow). (b) to (c) The $3-3$ pair is moved again in the $(+1,+1)$ direction. Generated faces are shown in red.

changes in the same fashion.

UI Implementation: Figure 3.8 shows our UI implementation of the three pair-wise movement operations. The four moving directions for $3-5$, $3-3$, and $5-5$ pair movements are shown as colored arrow pairs to help the user predict the effect of a movement before actually executing it.

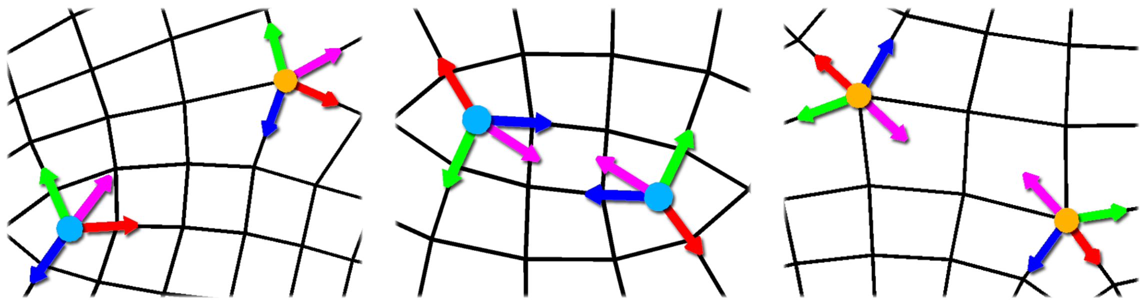


Figure 3.8: UI Implementation of the Three Pair-wise Movement Operations. Each of the four possible moving directions is shown as one pair of arrows with the same color.

3.5 Topological Analysis

In this section we provide theoretical analysis on why the three pair-wise movement operations are fundamental. Unlike vector and tensor field editing in which it is possible to move a singularity and cancel a singularity pair with opposite singularity indexes, it is impossible to move an irregular vertex (Theorem 3.5.1), and topological changes between a pair of irregular vertices have to follow specific constraints. For example, an irregular vertex pair whose discrete Gaussian curvature sum to zero cannot be canceled. The following theoretical discussion explains all possible edits within a convex region containing one or two irregular vertices ($v3$ or $v5$). This discussion explains why our suggested operations are the simplest possible operations that do not increase the number of irregular vertices. Note that some terminologies used in Theorem 3.5.1, 3.5.2, 3.5.3, and 3.5.4 are defined in Definitions 3.5.5, 3.5.6, and 3.5.7.

Theorem 3.5.1 *Consider a convex region R that contains exactly one irregular vertex v_0 in its interior. When $l(v_0)$, the valence of v_0, is not a multiple of 4, it is impossible to remesh the interior quadrilaterals of R to have a different configuration that still contains only one irregular vertex.*

The proof of Theorem 3.5.1 is adapted from Li *et al.* (2010) and is omitted here. The following three theorems deal with two irregular vertices in a convex region. A convex N-sided polygonal region is *regular* if all N-sides have equal length measured in graph distance. In this case we call the length of each side to be the *side length* of the region.

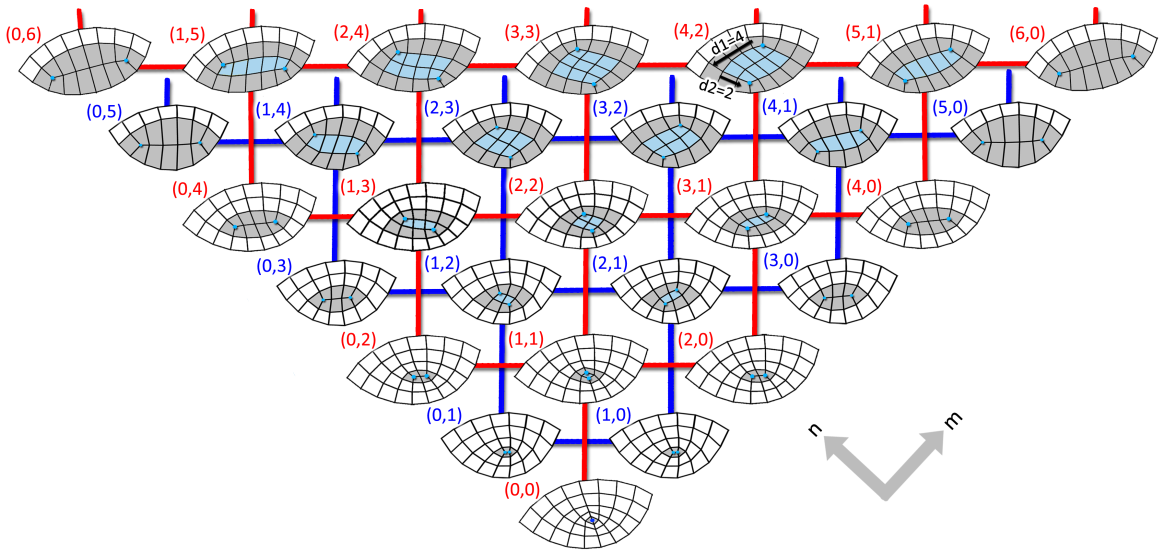


Figure 3.9: All Possible Configurations of a Region that Contains a $3-3$ Pair. Note that this figure extends to infinity. All possible requadrangulations of a specific region would be a sub-grid (either blue or red) that is cut along a line $n+m < c$, where c is a constant. R_α, the smallest region between the pair, is shown in blue, and the smallest regular digons containing R_α are shown in grey. Each configuration is labeled by its parameterization (m, n) and represented by a node positioned at (m, n) in a 2D coordinate system rotated by $\frac{\pi}{4}$. Every node has at most four neighbors, indicating four possibilities of movement given a configuration. Note that each pair of nodes $(x, 0)$ and $(0, x)$ on the boundaries (not including the $v2$ case) represent the same degenerate case parameterized differently, thus each boundary configuration can also have at most four neighbors. Note that the grids of configurations with even (red) and odd (blue) L are disconnected, i.e., it is impossible to requadrangulate a configuration with even L to any configuration with odd L and vice versa. Note that the grids of configurations with odd and even L are dual to each other, i.e., each face corresponds to a vertex and each pair of adjacent faces corresponds to an edge.

Theorem 3.5.2 *Let R be a regular convex digon with a side length L. If R encloses a $3-3$ irregular vertex pair, there are $\tau(L)$ mutually distinctive requadrangulations of R*

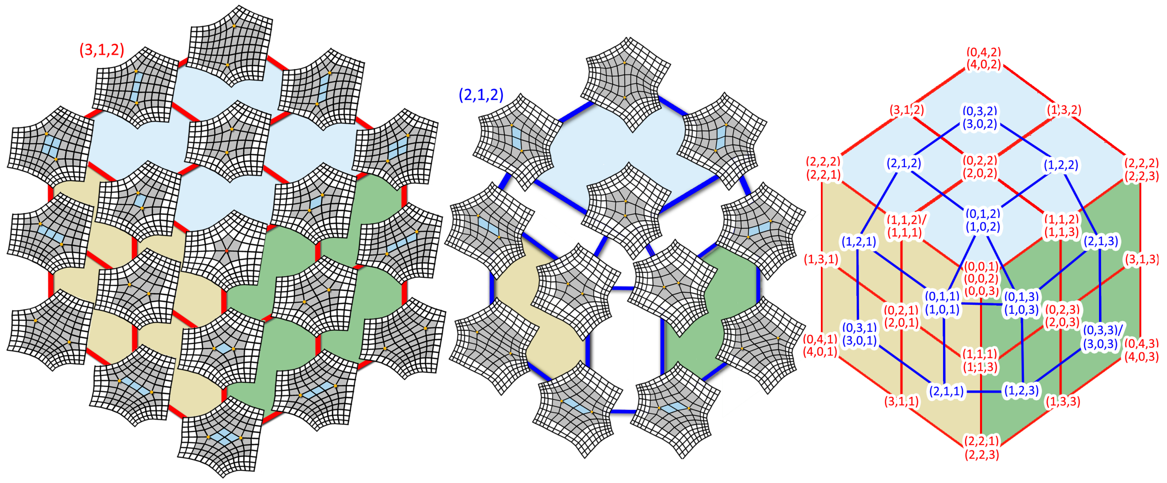


Figure 3.10: All Possible Configurations within a Regular Hexagon of Side Length Six (Left) and Five (Middle) that Contains a $5-5$ Pair.

Each configuration is labeled by its parameterization (m, n, p). For brevity we show all parameterizations in the right. Note that the grids of configurations with odd and even L are dual to each other, i.e., each face corresponds to a vertex and each pair of adjacent faces corresponds to an edge.

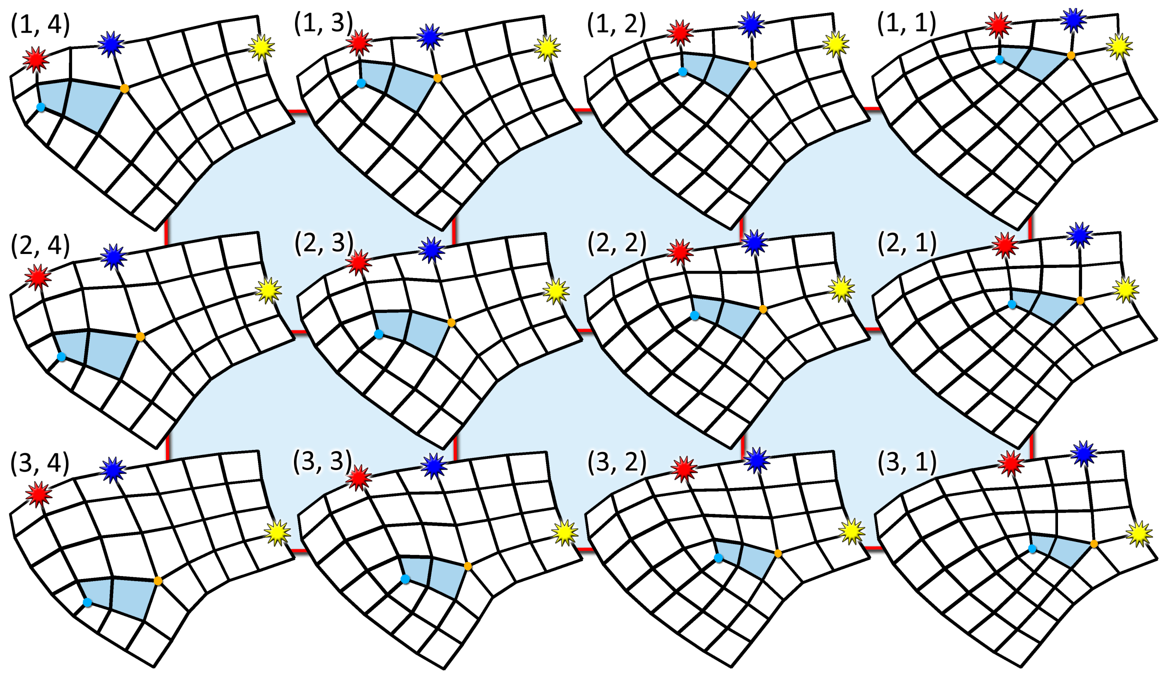
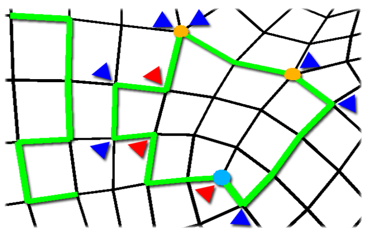


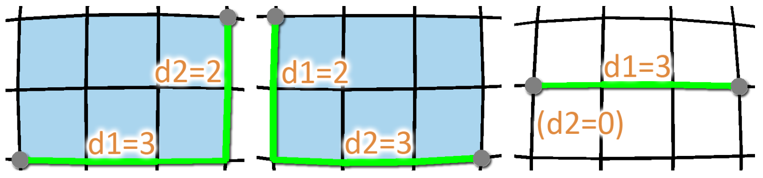
Figure 3.11: All Possible Configurations in a Region that Contains a $3-5$ Pair.
Each configuration is labeled by its parameterization (m, n). Notice they correspond to translations over a regular 2D grid parameterized by the locations of the red and yellow stars. Every node has at most four neighbors, indicating four possible moving directions given a configuration.

where $\tau(L) = N(N-1)$ *when* L *is even and* N^2 *when* L *is odd. Here* $N = \lfloor \frac{L}{2} \rfloor$*. These configurations can be parameterized by the set* $\{(m,n) \mid m \geq 0, n \geq 0, 0 < m+n \leq L-2\}$ *(Figure 3.9). Any one-step requadrangulation from the case* (m_0, n_0) *must be one of the following possible scenarios: (1)* (m_0+1, n_0-1)*, (2)* (m_0-1, n_0+1)*, (3)* (m_0+1, n_0+1)*, and (4)* (m_0-1, n_0-1)*. The operations can correspond to distance-preserving mutual spinning as well as distance-varying movement. Furthermore, when* R *is not regular, the number of valid configurations is at most* $\tau(L)$ *where* L *is the side length of the smallest regular digon that contains* R*.*

Theorem 3.5.3 *Let* R *be a regular convex hexagon with a side length* L*. If* R *encloses a* $5-5$ *irregular vertex pair, there are* $\tau(L)$ *mutually distinctive requadrangulations of* R *where* $\tau(L) = 3N(N-1)$ *when* L *is even and* $3N^2$ *when* L *is odd. Here* $N = \lfloor \frac{L}{2} \rfloor$*. These configurations can be parameterized by the set* $\{(m,n,p) \mid m \geq 0, n \geq 0, 0 < m+n \leq L-2, 1 \leq p \leq 3\}$ *(Figure 3.10). Any one-step requadrangulation from the case* (m_0, n_0, p_0) *must be one of the following possible scenarios: (1)* (m_0+1, n_0-1, p_0')*, (2)* (m_0-1, n_0+1, p_0')*, (3)* (m_0+1, n_0+1, p_0')*, and (4)* (m_0-1, n_0-1, p_0')*. Here* p_0' *can be* p_0*,* p_0-1*, or* p_0+1*, depending on the situation. The operations can correspond to distance-preserving mutual spinning as well as distance-varying movement. Furthermore, when* R *is not regular, the number of valid configurations is at most* $\tau(L)$ *where* L *is the side length of the smallest regular hexagon that contains* R*.*

Theorem 3.5.4 *Given a convex region* R *that encloses a* $3-5$ *pair connected by separatrices of lengths* d_1 *and* d_2*, there are exactly* $K = MN$ *mutually distinctive requadrangulations of* R *that still contain a* $3-5$ *pair. Here* M *and* N *are constants derived from the side lengths* E_i*'s of* R *and* d_1 *and* d_2*. The set of the requadrangulations is parameterized by the following set:* $\{(m,n) \mid 1 \leq m \leq M, 1 \leq n \leq N\}$ *(Figure 3.11). Any one-step requadrangulation from the case* (m_0, n_0) *must be one of*


(a)



(b)

Figure 3.12: An Open Path and a Loop.
An open path (left) and a loop (right). Blue triangles indicate normal convex turn points and red triangles indicate concave turn points. (b) All possible cases of the smallest convex region containing a vertex pair connected by separatrices with length d_1 and d_2 in the counter-clockwise order. (left) and (middle) Two non-degenerate cases. (right) Degenerate case where d_2 is 0.

the following: (1) (m_0+1, n_0)*, (2)* (m_0-1, n_0)*, (3)* (m_0, n_0+1)*, and (4)* (m_0, n_0-1)*. Each of these moves corresponds to the irregular vertex pair moving toward one of the four sides of R without mutual spinning or change in distance between the irregular vertex pair.*

To prove these theorems we need the following definitions adapted from Li *et al.* (2010).

Definition 3.5.5 *A* path γ *(Figure 3.12a) on the mesh M consists of a sequence of edges* $e_i = (v_i, v_{i+1})$ *for* $0 \leq i < N$*. N is the* length *of* γ*. A path is a* loop *(Figure 3.12a right) if* $v_0 = v_N$*. Otherwise,* γ *is an* open path *(left). A loop* γ *is* degenerate *if there exists a vertex in* γ *that is incident to at least three edges in* γ*.*

A *degenerate open path* can be defined in a similar fashion. An open path γ consisting of only regular vertices is *regular*. For a regular open path γ, the edges in γ divide the 1-ring neighborhood of any interior vertex v on γ into two subsets of quadrilaterals. v is a *turn point* if there are three quadrilaterals on one side and one quadrilateral on the other side. Otherwise, it is a *non-turn point*. For non-regular open paths turn points and non-turn points are undefined. For any vertex v on a loop γ, v is a *non-turn point* if there are two quadrilaterals in the exterior of the loop, otherwise it is a *turn point.* A path γ is *straight* if every interior vertex in γ is regular and non-turning.

A *region* R is a subset of the quadrilaterals in M whose dual graph is connected. That is, for any two quadrilaterals s and t in R, there is a sequence of quadrilaterals such that $t_0 = s$, $t_N = t$, and t_i and t_{i+1} share an edge in R for all $0 \leq i < N$. The boundary of R, denoted by ∂R, is a loop. R is *degenerate* if ∂R is a degenerate loop, otherwise it is non-degenerate. In this chapter we assume a region is non-degenerate and has exactly one boundary loop unless otherwise specified. A turn point on ∂R is *convex* if it has more than two adjacent quadrilaterals in $\bar{R}$, the exterior of R. Otherwise the turn point is *concave* (Figure 3.12a). The *angle* of a turn v, denoted by $k(v)$, is $(m(v) - 2)\frac{\pi}{2}$ where $m(v)$ is the number of incident quadrilaterals of v in $\bar{R}$. Because we only consider boundary vertices with a valence up to six, a convex turn point can have a turning angle up to π $(m(v) = 4)$. We refer to such points as a *sharp* convex turn point. A sharp convex turn point is considered as two *normal* convex turn points $(m(v) = 3)$ connected by an edge of length zero. Finally, a region R is *convex* if there are no concave turn points on ∂R. Otherwise, it is *concave.*

Definition 3.5.6 *Two regions are* compatible *if their boundaries are indistinguishable from the exterior. That is, there exists a way to walk each boundary and produce the same sequence of turning angles of the vertices.* Remeshing *a region is defined as finding a different but compatible region.*

A *side* of a region R is a sequence of edges of ∂R between two normal convex turn points. Alternatively we can say two regions are compatible if there exists a way to walk each boundary and produce the same sequence of length of sides.

Definition 3.5.7 *An* open separatrix *is an open and straight path whose interior vertices are regular and at least one of the end vertices is irregular. A* closed separatrix *is a loop that contains exactly one irregular vertex. In the remainder of the chapter we are only interested in open separatrices. There are $l(v_0)$ separatrices emanating from v_0.*

The discrete Gauss-Bonnet theorem relates the total turning angle $\sum_{v\in\partial R} k(v)$ along the boundary ∂R to the total discrete Gauss curvature of the vertices of R as follows:

$$\sum_{v\in\partial R}(m(v)-2)+\sum_{v\in intR}(4-l(v))=4\chi(M) \tag{3.1}$$

where $\chi(M)$ is the *Euler characteristic* of M.

To prove Theorems 3.5.2, 3.5.3 and 3.5.4, first we need the following lemmas.

Lemma 3.5.8 *Under the assumption of Theorems 3.5.2, 3.5.3 and 3.5.4, the smallest region between v_1 and v_2, R_α, is unique and must be a rectangular grid of size $d_1 \times d_2$ in the counter-clockwise order. If d_1 or d_2 is zero we have the degenerate case of the two irregular vertices connected by a single separatrix.*

The proof of Lemma 3.5.8 is based on Li *et al.* (2010) and is omitted here. All possible cases of the smallest convex region are shown in Figure 3.12b.

We note that the smallest *enclosing* convex region for an irregular vertex pair is the 1-ring neighborhood of R_α.

Lemma 3.5.9 *Given a convex region R that contains no irregular vertices and no sharp convex turn points on its boundary ∂R, the number of sides (equal to the number of normal convex turn points) in ∂R is given by $4 - G(R)$ where $G(R) =$ is total discrete Gaussian curvature of R.*

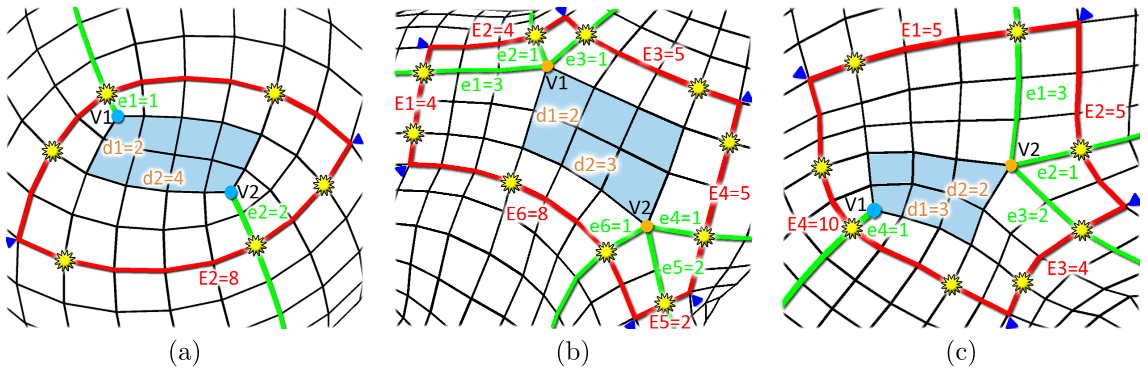


Figure 3.13: Examples of Convex Regions.
Examples of convex regions R enclosing exactly one (a) $3-3$, (b) $5-5$ and (c) $3-5$ pair v_1 and v_2 connected by separatrices of length d_1 and d_2. The smallest region between the pair R_α is shown in blue. ∂R is shown in red. Blue triangles indicate normal convex turn points. Separatrices of v_1 and v_2 that go outward from R_α are shown in green. E_1 to E_6 denote the length of sides of R. e_1 to e_6 denote the number of quad strips extended from R_α along the outward separatrices. The intersection points between separatrices and the region boundary are marked with yellow stars.

Proof From Equation 3.1 we have $\sum_{v\in\partial R}(m(v)-2) = 4\chi(R) - \sum_{v\in intR}(4-l(v))$. Since R has only convex turn points, for any normal convex turn point on ∂R we have $m(v) = 3$ and non-turn point $m(v) = 2$. Consequently, the left side of this equation, i.e., $\sum_{v\in\partial R}(m(v)-2)$, is equal to the number of normal convex turn points in ∂R. On the right side of this equation, $\chi(R) = 1$ since R is simply-connected. Furthermore, $G(R) = \sum_{v\in intR}(4-l(v))$ is the total discrete Gaussian curvature of R. ∎

A consequence of this lemma states that the number of sides of a convex region R contains a $3-3$, $5-5$, and $3-5$ irregular vertex pair has 2, 6, and 4 sides, respectively (see Figure 3.13 for examples).

Lemma 3.5.10 *The atomic step to expand a convex region R while keeping R convex without introducing irregular vertices is to add a strip of quadrilaterals adjacent to one side of R.*

Proof To expand a region at least one quadrilateral belonging to $\bar{R}$ adjacent to an edge in ∂R has to be added. Consequently, for the two ending vertices of the edge, denoted as v_a and v_b, both gain an adjacent quadrilateral and $m(v_a)$ and $m(v_b)$ decrease by 1. Thus v_a and v_b will become concave turn points if it is not a convex turn point ($m(v_a), m(v_b) <= 2$). To avoid creating concave turn points adding more quadrilaterals adjacent to edges in ∂R right next the first edge is necessary until both ending vertices are convex turn points. ∎

Theorem 3.5.11 *Given a convex region R enclosing a $3-3$, $5-5$, or $3-5$ pair, the configuration is* uniquely *determined by the following vector of couples (S_{ij}, u_{ij}) where i is the index of the irregular vertex, j is the index of a separatrix emanating from v_i, S_{ij} is the side on ∂R that intersects with this separatrix, and u_{ij} is the graph distance of this intersection point from the starting point of this side (counterclockwise). In Figure 3.13 each of these locations is highlighted with a yellow star-shaped symbol.*

The proof for Theorem 3.5.11 is similar to that of Theorem 3.5.1. Basically the location of the intersection points on the boundary and the side lengths of the region boundary are related to the distance of the irregular vertex pair to each edge of ∂R. This relation can be characterized by a linear system that has a unique solution. Consequently, should the solution have only positive entries (all distances are positive), there is a unique configuration corresponding to this set of parameters. ∎

The following lemmas show the underlying structure of R containing exactly a $3-3$, $5-5$ or $3-5$ pair:

Lemma 3.5.12 *Under the assumption of Theorem 3.5.2, arbitrary convex regions R can be constructed as follows: from the smallest convex region between the $3-3$ pair R_α, there are e_1 strips of quadrilaterals extending from one $v3$ vertex parallel to its outward separatrix and e_2 strips of quadrilaterals extending from another $v3$ vertex parallel to its outward separatrix. An example is shown in Figure 3.13a.*

Lemma 3.5.13 *Under the assumption of Theorem 3.5.3, arbitrary convex regions R can be constructed as follows: from the smallest convex region between the $5-5$ pair R_α, there are e_1, e_2 and e_3 strips of quadrilaterals extended from one $v5$ vertex parallel to its three outward separatrices, and e_4, e_5 and e_6 strips of quadrilaterals extended from another $v5$ vertex parallel to its three outward separatrices. An example is shown in Figure 3.13b.*

Lemma 3.5.14 *Under the assumption of Theorem 3.5.4, arbitrary convex regions R can be constructed as follows: from the smallest convex region between the $3-5$ pair R_α, there are e_1, e_2 and e_3 strips of quadrilaterals extended from the $v5$ vertex parallel to its three outward separatrices, and e_4 strips of quadrilaterals extended from the $v3$ vertex parallel to its outward separatrix. An example is shown in Figure 3.13c.*

Proof For R_α, which is unique according to Lemma 3.5.8, Lemma 3.5.12 holds with e_1 and e_2 equals 0, Lemma 3.5.13 holds with e_1 to e_6 equals 0, and Lemma 3.5.14 holds with e_1 to e_4 equals 0. Consider R_α as the unique first step in constructing R. By Lemma 3.5.9 we know that R has exactly 2, 6, and 4 sides when containing exactly a $3-3$, $5-5$, and $3-5$ pair. By Lemma 3.5.10 we know that the atomic step to expand a convex region is to add one strip of quadrilaterals adjacent to one side. It can be easily verified that each side is perpendicular to one outward separatrix of one $v3$ or $v5$ vertex. Since the strip of quadrilaterals added is parallel to one side, it is also perpendicular to one outward separatrix of one $v3$ or $v5$ vertex, thus Lemmas 3.5.12, 3.5.13 and 3.5.14 hold after the expansion.

Since these lemmas hold for the unique first step of constructing R and hold after all possible atomic steps to expand R, by mathematical induction we know that they hold for arbitrary convex regions R, which must be constructed from the unique R_α by several atomic expansion steps. ∎

We now prove Theorem 3.5.2.

Proof By Lemma 3.5.12 we denote the two sides of R adjacent to the e_1 and e_2 strips of quadrilateral as E_1 and E_2, as shown in Figure 3.13a. It can be easily verified that the following equations hold: $E_1 = d_1 + d_2 + 2e_2$ and $E_2 = d_1 + d_2 + 2e_1$.

Consequently, e_1 and e_2 are uniquely determined given E_1, E_2, d_1, and d_2. We now seek to enumerate (and parameterize) all possible such requadrangulations when $E_1 = E_2$.

After remeshing we denote the changes of each variable as Δd_1, Δd_2, Δe_1 and Δe_2. Since $E_1 = E_2$ are constant we have the following set of equations: $\Delta d_1 + \Delta d_2 + 2\Delta e_2 = 0$ and $\Delta d_1 + \Delta d_2 + 2\Delta e_1 = 0$.

From these equations we know that $\Delta d_1 + \Delta d_2$ must be even otherwise Δe_2 and Δe_1 will not be an integer. In other words Δd_1 and Δd_2 have to be either both even or both odd. Furthermore, if $\Delta d_1 = \Delta d_2 = 0$ then $\Delta e_2 = 0$ and $\Delta e_1 = 0$. All variables do not change thus R remains identical.

In the case of a regular digon containing a $3-3$ pair that is not connected by a single separatrix, each edge of the digon will intersect one separatrix from one vertex, say v_1, and two separatrices from the other, say v_2. Suppose that the first edge in the digon intersects separatrix 1 of v_1 and separatrices 0 and 2 of v_2, i.e., $S_{11} = S_{20} = S_{22} = 1$. Recall that u_{11}, u_{20}, and u_{22} denotes the location of the intersection points. Consequently, we have $u_{20} < u_{11} < u_{22}$. Notice that $d_1 = u_{22} - u_{11}$ and $d_2 = u_{11} - u_{20}$. When the irregular vertex pair is connected by a separatrix, each edge in the digon will only intersect one separatrix from each vertex. In this case, $d_1 = u_{22} - u_{11}$ and $d_2 = 0$.

When $d_1 + d_2 = E_1$, the irregular vertex pair must appear on the boundary of R. Since we discuss only convex regions free of irregular vertices on its boundary, this is not allowed. Furthermore, once d_1 and d_2 are given, the set of intersection points between any separatrix from v_1 or v_2 with any edge will be determined. According to Theorem 3.5.11, there is at most one valid configuration (requadrangulation) that satisfies the requirements on positions of intersection points. Consequently, all possible requadrangulations of R with a $3-3$ pair can be parameterized by m and n as follows: $\{(m,n) \mid m \geq 0, n \geq 0, 0 < m+n < E_1 - 2\}$. Here $m = d_1$ and $n = d_2$. For degenerate cases, i.e., $d_2 = 0$, both $(d_1, 0)$ and $(0, d_1)$ are allowed. This gives rise to a pyramid-shaped grid such as the one shown in Figure 3.9. When E_1 is even, there are $N(N-1)$ mutually distinguishable configurations. When E_1 is odd, there are N^2 mutually distinguishable configurations. Here $N = \lfloor \frac{L}{2} \rfloor$ and $L = E_1 = E_2$. Every grid point corresponds to a valid requadrangulation, and $(0,0)$ corresponds to the case in which the irregular vertex pair is merged into a single $v2$ vertex.

Given a configuration, it is possible to move the irregular vertex pair in at most four directions, corresponding to the four neighboring grid points. Each of these four atomic changes can be realized using a pair-wise movement operation. Given this, we now have an explicit algorithm of realizing any valid configuration inside a regular digon. For configurations where $m+n$ is even, we start with the unique configuration corresponding to the $(0,0)$ case. For configurations where $m+n$ is odd, we start with the unique configuration corresponding to the $(0,1)$ case. The uniqueness is a result of Theorem 3.5.1. Next, we find a shortest path in the grid between the starting case and the target configuration and perform necessary steps to move from the former towards the latter through appropriate graph-level operations. This demonstrates that each valid configuration inside a regular digon can be realized, i.e. there is one and only one remeshing corresponding to every grid point. ∎

The proof for Theorem 3.5.3 is similar, except that the relationships governing these variables are given by:

$$
\begin{aligned}
E_1 &= d_1 + e_2 + e_6 \\
E_2 &= e_1 + e_3 \\
E_3 &= d_2 + e_2 + e_4 \\
E_4 &= d_1 + e_3 + e_5 \\
E_5 &= e_4 + e_6 \\
E_6 &= d_2 + e_1 + e_5
\end{aligned}
\tag{3.2}
$$

When R is a regular hexagon containing a $5-5$ pair, i.e., $E_1 = E_2 = E_3 = E_4 = E_5 = E_6$, there is again a two-fold rotational symmetry which maps two different configurations to the same remeshing. These two cases corresponding to the positions of the irregular vertex pair are swapped. In the remainder of the discussion we will treat the two configurations as one, which is equivalent to modulating out the two-fold symmetry. When the irregular vertices are not connected by a single separatrix, it is straightforward to verify that there are exactly four edges in R that intersect two separatrices, one from v_1 and v_2 each. Due to the two-fold symmetry in the hexagon. These four edges form two pairs of opposing edges. Consequently, it is sufficient to consider only two consecutive edges from the four.

After the initial modulation, there is still a three-fold rotational symmetry, which corresponds to three pairs of consecutive sides. For example, in Figure 3.13b the sides with length E_3 and E_4 are one such pair. Every valid configuration for one pair can be used to generate a solution for the other two pairs through an appropriate rotation, and vice versa. Consequently, it is sufficient to consider only the case where the pair of sides each intersect with one separatrix from v_1 and v_2. Notice that the distance between the two intersection points on the first side is equal to d_1, and the distance between the intersection points on the second side is d_2.

For the special case when there is a connecting separatrix between v_1 and v_2, we consider that d_2=0. Consequently, the set of solutions can be parameterized by the same set of m and n as for the $3-3$ pair. However, due to the aforementioned three-fold symmetry, we add the third index p which ranges from 1 to 3 to distinguish them. This leads to three squares being glued together to form the surface of a half cube (Figure 3.10). Like the $3-3$ case, in a regular hexagon there is one and only one valid configuration corresponding to each point in the grid. Any valid configuration can be explicitly realized. ▮

We now prove Theorem 3.5.4.

Proof By Lemma 3.5.14 we denote the four sides of R adjacent to the e_1 and e_4 strips of quadrilateral as E_1 and E_4, as shown in Figure 3.13c. It can be easily verified that the following equations hold:

$$\begin{aligned} E_1 &= d_1 + e_2 + e_4 \\ E_2 &= e_1 + e_3 \\ E_3 &= d_2 + e_2 + e_4 \\ E_4 &= d_1 + d_2 + e_1 + e_3 \end{aligned} \tag{3.3}$$

After remeshing we have $\Delta E_i = 0$ for $1 \leq i \leq 4$. This implies that $\Delta d_1 = \Delta d_2 = 0$, i.e., any remeshing of R cannot change the relative position of the two irregular vertices. However, these configurations can differ in terms of e_1 and e_2. Given the values of E_1, E_2, e_1, and e_2 such that $1 \leq e_1 < E_2$ and $1 \leq e_2 < E_1 - d_1$, there exists a unique legal requadrangulation of R. Consequently, the set of possible requadrangulations can be parameterized by e_1 and e_2, and the range of $m = e_1$ and $n = e_2$ are given by $[1, E_2 - 1]$ and $[1, E_1 - d_1 - 1]$, respectively. This is illustrated in Figure 3.11. Any atomic requadrangulation will translate R_α towards one of the sides of ∂R, corresponding to the four neighboring grid points. ∎

3.6 Complementary Operations

Irregular Vertex Cancellation: We can move a $v3$ vertex to collide with a $v5$ vertex, or vice versa, by applying multiple pair-wise movement operations. When one $v3$ and one $v5$ vertex collide they cancel each other and both become regular. At least one other irregular vertex needs to be involved in this cancellation. In this fashion we develop a $3-5$ **pair cancellation** operation. It is possible that the last step of a $3-5$ pair cancellation is equivalent to one $3-3-5-5$ removal operation and two pairs of irregular vertices are canceled at once. Examples can be found in Figures 3.15 and 3.16.

Irregular Vertex Merging: A $3-3$ pair can be merged to a $v2$ vertex and a $5-5$ pair can be merged to a $v6$ vertex when their graph distance is even. Theorems 3.5.2 and 3.5.3 provide the theoretical analysis that is related to such a merge.

Irregular Vertex Alignment: Under the assumptions of Theorems 3.5.2 and 3.5.3, arbitrary $3-3$ and $5-5$ pairs can be aligned by applying multiple movement operations until $d_1 = 0$ or $d_2 = 0$.

Smoothing: We use iterative Laplacian mesh smoothing to improve the geometry if the connectivity edits degrade the shape of the mesh above a user-defined tolerance. The user can select uniform weights or cord-length weights, and elect to preserve sharp features by constraining the positions of vertices on sharp edges. The smoothing scheme can improve the aspect ratios of modified faces. After each iteration all vertices are projected back onto the original mesh. We have also experimented with a scheme in which newly generated vertices are pulled towards vertices in the original mesh if the distance between the new and original vertices is above a threshold. The projection and pulling scheme can narrow the difference to the original mesh.

3.7 Comparisons to Triangle Mesh Editing

We show the analogy between the editing operations for triangular meshes proposed in Li *et al.* (2010) and our operations for quadrilateral meshes. First, it is impossible to generate, move, or delete a single irregular vertex within a convex region for both triangular and quadrilateral cases. Thus the simplest possible operations that do not increase the number of irregular vertices must involve an irregular vertex pair. For the triangular case they are $5-7$, $5-5$, and $7-7$ pair movements, and for the quadrilateral case they are $3-5$. $3-3$, and $5-5$ pair movements.

The $5-7$ pair (triangular case) and $3-5$ pair (quadrilateral case) movements both will not change the relative distance between the pair, measured in the length of their connecting separatrices.

Similarly, the $5-5$ / $7-7$ pair (triangular case) and $3-3$ / $5-5$ pair (quadrilateral case) movements both will change the relative distance under specific constraints. In general in both cases the pair will move in a symmetric, rotating fashion.

We note that despite the similarity in the suites of editing operations for both triangular and quadrilateral meshes, i.e., both comprise a hierarchy of basic, atomic semantic, and composite operations, our chapter is the first to be able to enumerate all possible configurations given a regular convex region. Such analysis was not present in Li *et al.* (2010). Consequently, while they demonstrated that it is possible to move a $5-5$, $7-7$, or $5-7$ pair in practice, no guarantee was provided, neither was an explicit algorithm given to transform from one valid configuration to a different valid configuration within the regular convex region.

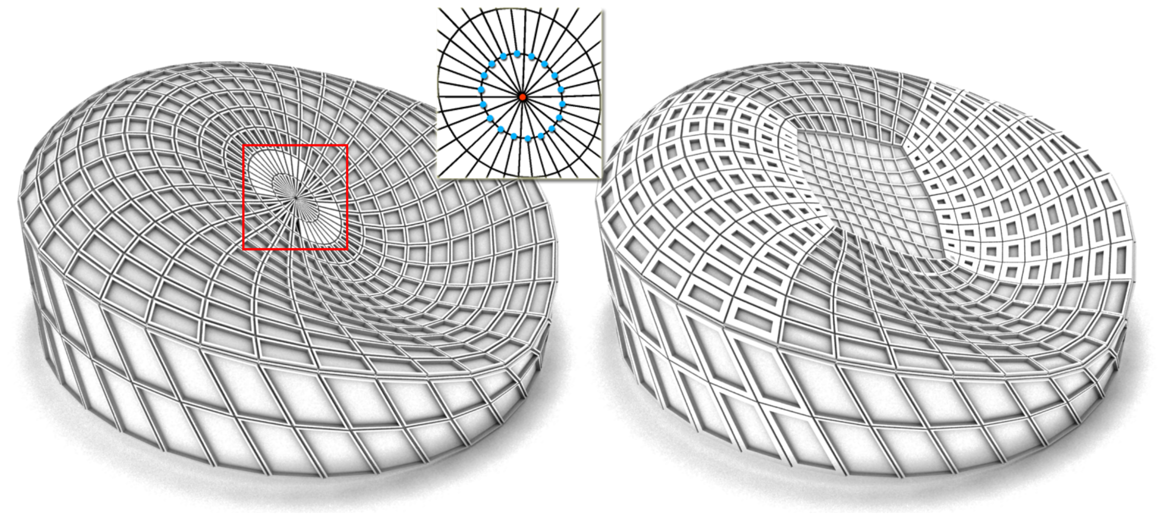


Figure 3.14: Converting a Disc Shaped Mesh with Higher Order (36) Irregular Vertices to a Semi-regular Mesh.

The faces and edges of the mesh define the panels of the designed structure.

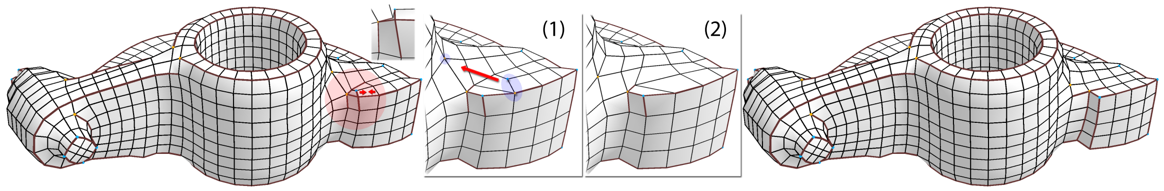


Figure 3.15: Using Our Editing Framework to Improve a Remeshed Rockarm Model.

(left) Original mesh with an ill-shaped corner part. The diagonal $3-5$ pair leads to a highly non-planar face in between and distorts the nearby faces on the upper-left side. (1) The misplaced $v3$ vertex is made regular by a quad collapse. (2) The generated $3-5$ pair is moved upper-left to cancel with the $v5$ vertex by a $3-5$ pair cancellation. Movement of vertices on sharp features are constrained. (right) Mesh improved by our editing framework. Now the corner part has a nice structure.

3.8 Applications

Irregular Vertex Cancellation and Alignment: In Figure 3.19 we improve a highly irregular structure designed by an architect that consists of quadrilateral glass panels. We reduce the numbers of irregular vertices from 168 (92 $v3$, 62 $v5$, and 14 $v6$) to 24 (12 $v3$ and 12 $v5$) by multiple $3-5$ pair cancellation operations. Then we align the irregular vertices to improve the flow of the panel structure. The editing

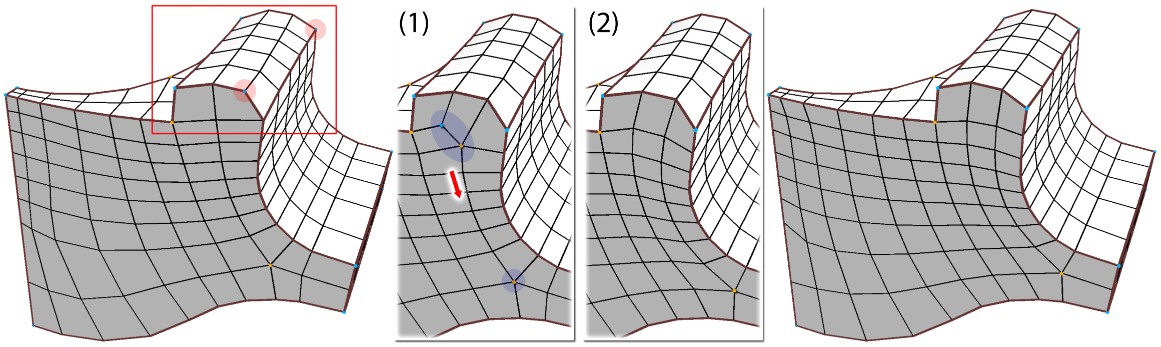


Figure 3.16: Using Our Editing Framework to Improve a Remeshed Fandisk Model. (left) Original mesh with an ill-shaped upper part caused by a mis-aligned $3-3$ pair. (1) The misplaced $v3$ vertex is moved to proper location by an edge flip. (2) The created $3-5$ pair is moved downward to cancel with the $v5$ vertex by a $3-5$ pair cancellation. (right) Mesh improved by our editing framework. Now the upper part has a nice structure and the face strips flow consistently through the front surface.

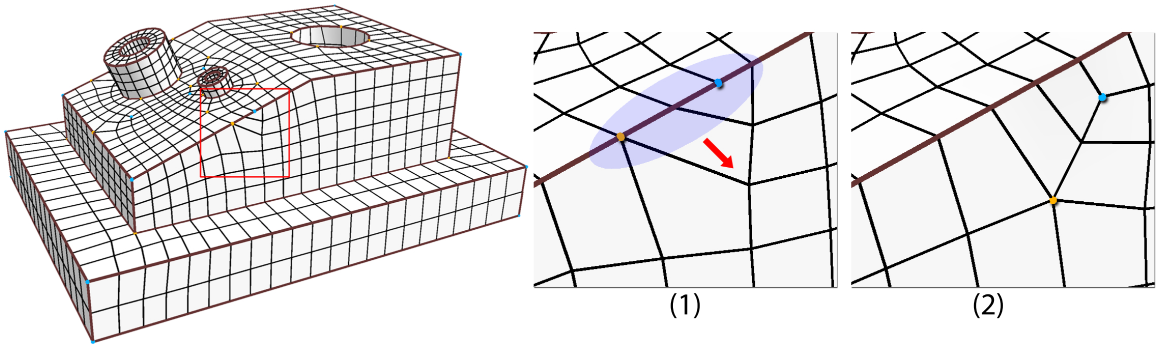


Figure 3.17: Using Our Editing Framework to Fix a Defect on a Remeshed Accessory Model.

(left) Original mesh with a $v3$ irregular vertex positioned on the feature edge. The $v3$ vertex's deficiency of valence leads to a degenerated, triangular shaped quad face on its lower side. (1) to (2) The $v3$ vertex is pulled off the feature edge by a $3-5$ pair movement. The shape of nearby faces are improved because of the better mesh topology.

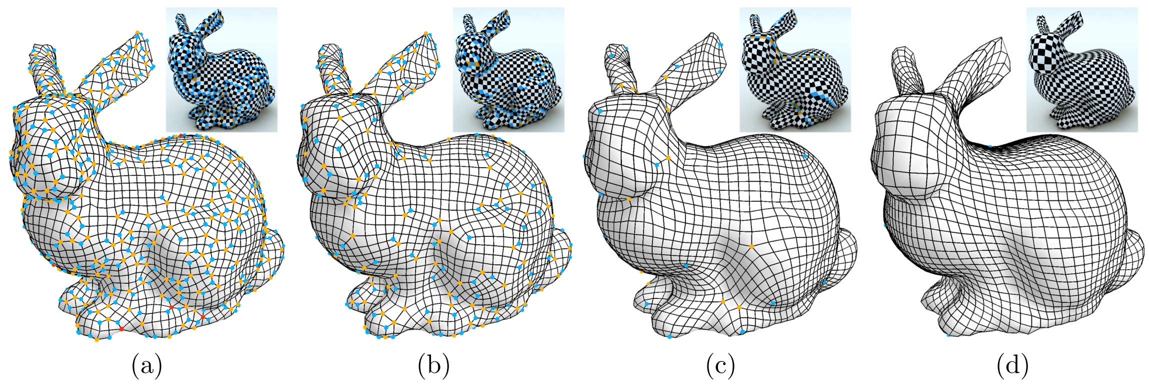


Figure 3.18: Progressive Irregular Vertex Cancellation.
(a) Bunny model produced by a mesh simplification algorithm, in which 956 of the total 3006 vertices are irregular. (b) and (c) Intermediate results with 400 and 48 irregular vertices. (d) Maximally reduced form with only eight $v3$ irregular vertices. The smaller figures show the same models with a checkerboard pattern by greedy coloring (so that adjecent quad faces have different colors).

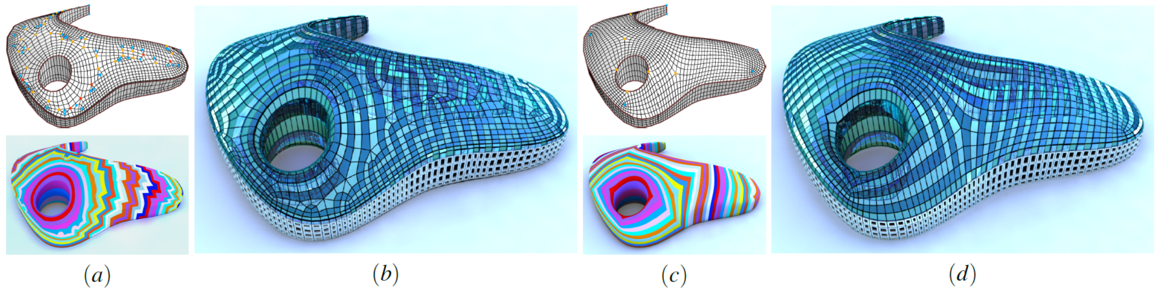


Figure 3.19: Irregular Vertex Redunctions.
We introduce connectivity editing operations to control irregular vertices in quadrilateral meshes. This can lead to improved results in the design of a glass structure: (a) top: the original mesh with irregular vertices as colored dots, (a) bottom: a stripe pattern applied to the mesh, (b) a rendering of the design as glass construction. In (c) and (d) we show the edited mesh. The glass panels on the roof are generated from the edges in the meshes.

process takes approximately 12 minutes (7 minutes for irregular vertices cancellation and 5 minutes for alignment).

In Figure 3.14 we transform a mesh with higher order irregular vertices (36 $v3$, 2 $v36$) to a semi-regular (4 aligned $v3$ vertices) mesh by several valence reduction operations and several irregular vertex alignment operations.

Connectivity Improvement: Figure 3.15, 3.16, and 3.17 show our editing framework is capable of repairing various kinds of defects found in results of state-of-art quadrangulation approaches. Even though these existing quad remeshing algorithms produce excellent results, they are essentially heuristics to tackle an NP-complete problem resulting from the fact that the selection and positioning of irregular vertices is a discrete optimization problem. Therefore, we believe that it is unlikely that a general solution can be found. In our analysis a user can identify several meaningful edits that can improve a mesh for many automatically generated results. We therefore argue that manual editing tools are an essential component of a complete mesh processing pipeline.

Chapter 4

CONNECTIVITY EDITING FOR QUAD-DOMINANT MESHES

We propose an editing framework for the connectivity of arbitrary two-manifold quad-dominant (QD) meshes with the ability to explicitly control the location, type, and number of the irregular vertices (with more or fewer than four neighbors) and faces (non-quads) in the mesh. In the primal domain, the large number of combinations of different irregular elements makes connectivity analysis difficult. Therefore, we propose to edit QD meshes in an alternative pure quad mesh domain building on existing work in quad mesh connectivity editing Tarini *et al.* (2010); Bommes *et al.* (2011); Peng and Wonka (2013). We are able to answer the following fundamental questions about the connectivity of QD meshes: what is the discrete Gauss-Bonnet theorem for QD meshes? In particular, can irregular vertices and faces be counted in the same way? What editing operations are fundamentally possible and impossible for QD meshes? Can we create, delete, or move a single irregular element? How can we interchange irregular faces and vertices? What can we do with T-junctions? In what circumstances is being quad-dominant, i.e. having non-quad faces, preferable to being pure quad for remeshing a surface? We then identify the simplest possible editing operations for QD meshes in the sense that they affect the smallest possible regions and involve the fewest irregular elements, which are moving irregular elements in pairs.

To evidence that the ability to edit irregular elements in a QD mesh is valuable, we compare different ways to QD remesh a given surface, e.g. having only irregular vertices, only irregular faces, or both, in terms of geometric properties. We demonstrate that the users can use our editing framework to alter the connectivity of existing QD

meshes with ease.

4.1 Chapter Overview

We organize this chapter as follows. In Section 4.2.1, we review the $\sqrt{CC}$ domain of QD meshes and argue why it is a suitable platform for analyzing the connectivity of QD meshes. In Section 4.2.2, we formulate the generalized version of the discrete Gauss-Bonnet theorem for QD meshes, which can be used to predict the number of irregular elements in an arbitrary QD mesh given the boundaries and the Euler characteristic. In Section 4.2.3, we analyze what kinds of editing operations are impossible and possible for QD meshes. Backed by the aforementioned theoretical analysis, in Section 4.3, we propose a connectivity editing framework for QD meshes. In Section 4.4, we compare various ways of QD mesh design for a given surface in terms of geometric properties, and show examples of QD mesh connectivity edits.

4.1.1 Basic Definitions

We limit our discussion to two-manifold QD meshes. The valence of a vertex v, which we denote as $l(v)$, is the number of edges in the mesh incident to v. A vertex with valence n is denoted as vn, e.g. a $v3$ and a $v5$. A vertex with valence 4 is considered as *regular*, otherwise it is *irregular*. The degree of a face f, which we denote as $d(f)$, is f's number of vertices. A face with degree d is denoted as fd, e.g. a triangle ($f3$), a quad ($f4$), and a pentagon ($f5$). A face with degree four is considered as *regular*, otherwise it is *irregular*. We introduce an *index* function to measure irregularity. We define the index as $4 - n$ for a vn and $4 - d$ for an fd.

To simplify our discussion, we consider irregular vertices with valences lower than 3 or higher than 5 as multiple $v3$ or $v5$ collocated together and therefore count them as multiple irregular vertices. Similarly, irregular faces with degree lower than 3 or higher than 5 are considered as multiple $f3$ or $f5$ collocated together and counted as multiple irregular faces. We also use the following definitions:

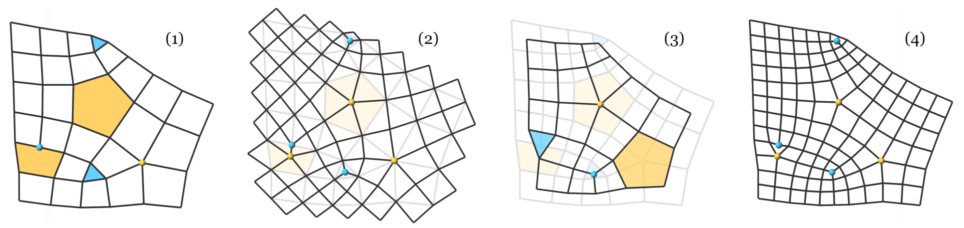
Definition 4.1.1 *A* path γ *on a* QD *mesh* M *consists of a sequence of edges* $e_i = (v_i, v_{i+1})$ *for* $0 \leq i < N$, *where* N *is the* length *of* γ. *A path is a* loop *if* $v_0 = v_N$. *We assume that* γ *is non-degenerate, i.e. there is no vertex in* γ *that is incident to at least three edges in* γ.

Definition 4.1.2 *A* region R *on a* QD *mesh* M *is a connected subset of the faces in* M *(Figure 4.1b). We assume that* R *has a single boundary unless otherwise specified. The boundary of* R, *denoted by* ∂R, *is a loop. The* index *of a boundary vertex* v *is* $1 - e(v)$ *where* $e(v)$ *denotes the* internal valence *of* v, *i.e. the number of* v*'s adjacent edges connecting to internal vertices. Intuitively, we denote a boundary vertex as a* non-corner *if its index is zero, a* convex corner *if its index is* 1, *and a* concave corner *if its index is negative. A region is* convex *if it has no concave corners, otherwise it is* concave. *A* side *of a region* R *is a sequence of edges of* ∂R *between two corners.*

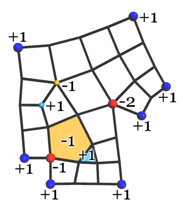
In our context, a region's boundary configuration is often specified as a sequence of vertices together with their internal valences. Note that the indices of irregular vertices, irregular faces, and boundary vertices are related to the Gaussian curvature (for irregular elements) and geodesic curvature (for boundary vertices) of their suitably remeshed neighborhoods.

4.2 Theoretical Analysis

We describe the major insights for connectivity editing of QD meshes in the following.



(a)



(b)

Figure 4.1: Different Domains of a QD Mesh.
(a) A QD mesh with a single boundary loop in its (1) primal domain, (2) $\sqrt{CC}$ domain, (3) dual domain, and (4) CC domain. $v3$ and $f3$ are drawn in blue. $v5$ and $f5$ are drawn in yellow. Note that the mapping from (1) to (2) and from (2) to (4) can be done by a $\sqrt{CC}$ subdivision, the mapping from (2) to (1) can be done by an inverse $\sqrt{CC}$ subdivision, and the mapping from (2) to (3) can be done by an alternative inverse $\sqrt{CC}$ subdivision with the p-vertices and d-vertices switched (dangling vertices are omitted). Note that the T-junction in the bottom left corner is mapped to an adjacent $v3$-$v5$ pair in the $\sqrt{CC}$ domain. (b) A region in a QD mesh. Convex corners are shown in blue and concave corners are shown in red. We denote the indices of the boundary vertices and irregular elements. Note that the discrete Gauss-Bonnet theorem (Equation 4.1) is satisfied.

4.2.1 $\sqrt{CC}$ Domain

Analyzing connectivity editing for QD meshes in their primal domain would be too complicated because all possible combinations of irregular faces and vertices need to be considered. For example, the triple combinations of $v3$, $v5$, $f3$, and $f5$ amount to 20 possibilities, as compared to 4 in a quad mesh with just $v3$ and $v5$. An alternative domain where both irregular vertices and faces are mapped to elements of the same type is thus desired.

A key insight of our work is the selection of a suitable domain for QD mesh connectivity editing. We prefer domains where meshes are pure quad so that existing quad mesh editing work can be directly applied. Therefore, the dual, Doo-Sabin Doo and Sabin (1978), and $\sqrt{3}$-subdivision Kobbelt (2000) are infeasible. One reasonable choice may be the CC domain in which subdivided meshes are pure quad and both irregular vertices and faces in the primal domain are mapped to irregular vertices of the same degree. However, only a subset of quad meshes is inversely CC subdivisible, e.g. irregular vertices must have graph distances of at least two, which means that it is possible to edit a CC subdivided mesh to make it no longer inversely subdivisible. This fact again makes analysis difficult. It turns out that the $\sqrt{CC}$ domain of a QD mesh Kobbelt (1996) is a suitable platform for our analysis. We review key properties of the $\sqrt{CC}$ domain as follows.

A $\sqrt{CC}$ subdivision can be understood as a half step of achieving the connectivity of a full CC subdivision (Figure 4.1a). Every primal vertex and face is mapped to a $\sqrt{CC}$ vertex. The former is denoted as a *p-vertex* and the latter is denoted as a *d-vertex* (corresponding to dual vertices of the primal mesh). We position the p-vertices at the same locations of their corresponding primal vertices and the d-vertices at the centers of the corresponding primal faces for visualization purposes. Every p-vertex is connected to the d-vertices of its corresponding primal vertex's adjacent faces and every d-vertex is connected to the p-vertices of its corresponding primal face's adjacent vertices. For meshes with boundaries we add an imaginary adjacent boundary face to each primal boundary edge. In this manner every primal edge, non-boundary or boundary, would have two adjacent faces and would be mapped to a $\sqrt{CC}$ face. Any $\sqrt{CC}$ domain mesh is thus pure quad since every $\sqrt{CC}$ face has four adjacent vertices (two p-vertices and two d-vertices). Furthermore, the $\sqrt{CC}$ vertices have the same degree as their corresponding primal vertices or faces. It is straightforward to see that every QD mesh has exactly one corresponding $\sqrt{CC}$ mesh. Another advantage of a $\sqrt{CC}$ subdivision is that it increases the number of faces by a factor of about two (converting every primal edge to a $\sqrt{CC}$ face), as compared with a factor of about four by a CC subdivision.

Inverse $\sqrt{CC}$ Subdivision: As noted by Taubin Taubin (2002), an inverse $\sqrt{CC}$ subdivision to recover the primal mesh can be done by inserting a diagonal connecting the two p-vertices for each $\sqrt{CC}$ face (and recover the dual by inserting diagonals connecting two d-vertices) for quad meshes that are 2-colored (as p-vertices and d-vertices). Note that for a mesh with boundaries it is possible that the recovered dual mesh has valence 1, i.e. dangling, vertices corresponding to the valence-2 boundary $\sqrt{CC}$ vertices. The following proposition describes the feasibility of inverse $\sqrt{CC}$ subdivision of quad meshes:

Proposition 4.2.1 *A region in a quad mesh with a sphere-like or disk-like topology (with at most one boundary and no handles) can be inversely $\sqrt{CC}$ subdivided in exactly two ways.*

Proof It is known Harary (1969) that a graph can be 2-colored, i.e. is bipartite, if and only if it has no odd graph cycles. Any graph cycle in such a region is the boundary of a quadrangulation with a disk-like topology thus has an even length (a property of pure quad meshes). It is straightforward to see that the 2-coloring for a 2-colorable graph is unique (up to switching of all colors). By considering the vertices with the first color as either p-vertices or d-vertices we have two ways to do inverse $\sqrt{CC}$ subdivision. ∎

Conversely, a region with more than one boundary or with handles may be inversely $\sqrt{CC}$ subdivisible or not. Proposition 4.2.1 implies that we can freely edit a $\sqrt{CC}$ mesh and the edited mesh is still inversely $\sqrt{CC}$ subdivisible as long as the changes are contained in a disk-like region. Furthermore, the retrieved primal domain meshes are consistent if we fix the coloring of a fixed vertex in the $\sqrt{CC}$ domain by the following proposition:

Proposition 4.2.2 *An inverse $\sqrt{CC}$ subdivision applied to a quad mesh region with at most one boundary converts two $\sqrt{CC}$ vertices to be of the same type (primal vertex or face) if and only if their graph distance is even.*

The proof is straightforward by inspecting the 2-coloring of a shortest path between the two $\sqrt{CC}$ vertices and is omitted here.

4.2.2 Discrete Gauss-Bonnet Theorem for Quad-Dominant Meshes

We formulate the discrete Gauss-Bonnet theorem for an arbitrary QD mesh M as follows:

$$\sum_{v \in \partial M} (1 - e(v)) + \sum_{v \in intM} (4 - l(v)) + \sum_{f \in M} (4 - d(f)) = 4\chi(M), \tag{4.1}$$

where $\chi(r)$ denotes the Euler characteristic of M. Recall that $l(v)$ denotes vertex v's valence, $e(v)$ denotes the number of v's adjacent edges connecting to internal vertices, and $d(f)$ denotes face f's number of vertices.

Equation 4.1 implies that for an arbitrary QD mesh (including a region) the sum of the indices of the boundary vertices plus the sum of the indices of the irregular vertices and faces equals a constant solely determined by its Euler characteristic. It is useful for determining the minimal number of irregular elements in a QD mesh with known boundaries and Euler characteristic and vice versa. For example, any connectivity edits would not change the sum of the indices of the irregular elements within a QD region with a fixed boundary.

Equation 4.1 can be proved in many ways, e.g. by analyzing the CC domain of M or summing the angle defects of the irregular elements and boundary vertices. Here we provide a proof that demonstrates that Equation 4.1 is a pure combinatorial fact directly derived from the Euler characteristic.

Proof The Euler characteristic of M states that $V - E + F = \chi(M)$, where V, E, and F respectively denote the number of vertices, edges, and faces in M. V equals $V_b + V_i$ where V_b denotes the number of boundary vertices and V_i denotes the number of internal vertices. E equals $E_{bb} + E_{bi} + E_{ii}$ where E_{bb} denotes the number of boundary edges that are incident to two boundary vertices, E_{bi} denotes the number of boundary-internal edges that are incident to one boundary vertex and one internal vertex, and E_{ii} denotes the number of internal edges that are incident to two internal vertices. The Euler characteristic of M can then be rewritten as $4V_b + 4V_i - 4E_{bb} - 4E_{bi} - 4E_{ii} + 4F = 4\chi M$, which can be reformulated as:

$$\begin{gathered}(V_b - E_{bi})+\\ (3V_b - 3E_{bb})+\\ (4V_i - 2E_{ii} - E_{bi})+\\ (4F - 2E_{ii} - 2E_{bi} - E_{bb}) = 4\chi M.\end{gathered}$$

The first part $(V_b - E_{bi})$ equals $\sum_{v \in \partial R}(1 - e(v))$. The second part $(3V_b - 3E_{bb})$ equals zero since the number of boundary vertices and the number of boundary edges are the same. The third part $(4V_i - 2E_{ii} - E_{bi})$ equals $\sum_{v \in intM}(4 - l(v))$ since, by summing the valences of all internal vertices, we count each internal edge twice and each boundary-internal edge once. The fourth part $(4F - 2E_{ii} - 2E_{bi} - E_{bb})$ equals $\sum_{f \in M}(4 - d(f))$ since by summing the degrees of all faces, we count each internal and boundary-internal edge twice and each boundary edge once. ∎

4.2.3 Fundamental Editing Operations

In this section, we describe what editing operations are fundamentally impossible and possible for irregular elements in a QD mesh. In general, our findings stem from analyzing the $\sqrt{CC}$ domain of QD meshes to which theorems about quad mesh editing can be applied.

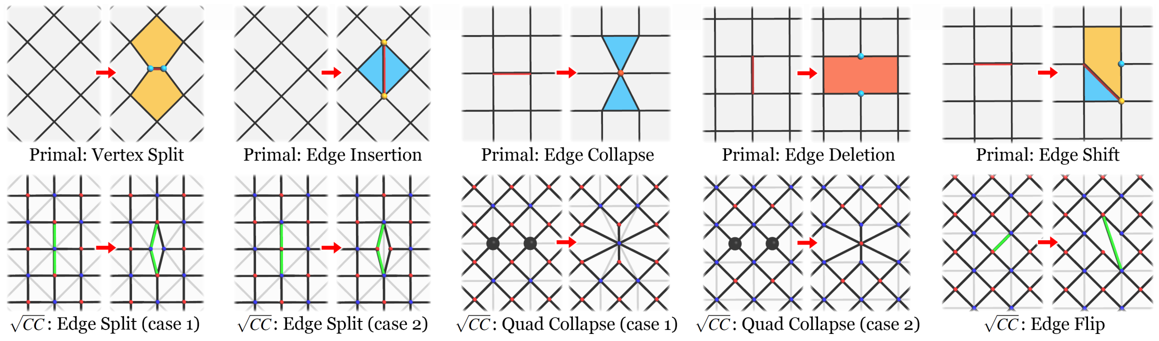

Primal: Vertex Split
Primal: Edge Insertion
Primal: Edge Collapse
Primal: Edge Deletion
Primal: Edge Shift
$\sqrt{CC}$: Edge Split (case 1)
$\sqrt{CC}$: Edge Split (case 2)
$\sqrt{CC}$: Quad Collapse (case 1)
$\sqrt{CC}$: Quad Collapse (case 2)
$\sqrt{CC}$: Edge Flip

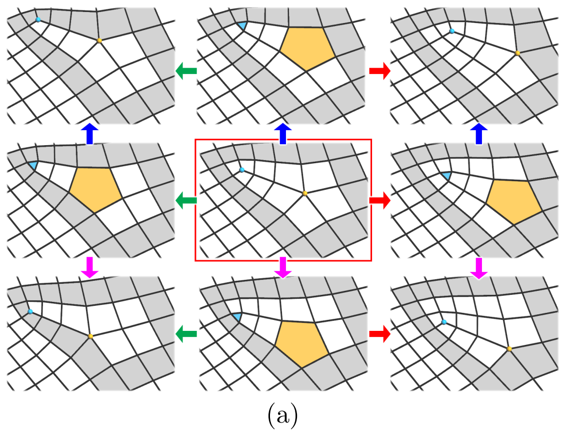
Figure 4.2: The Five Basic Editing Operations for QD Meshes and Their Corresponding Operations in the $\sqrt{CC}$ Domain.

Both a vertex split and an edge insertion add a primal edge. They are equivalent to an edge split of a $\sqrt{CC}$ edge pair with a p-vertex (blue) and a d-vertex (red) in between, respectively. Both an edge collapse and an edge deletion delete a primal edge. They are equivalent to a quad collapse of a d-vertices pair and a p-vertices pair, respectively. An edge shift shifts a primal edge toward one of its adjacent faces. It is equivalent to an edge flip in the $\sqrt{CC}$ domain. The edited meshes are not smoothed.

To begin with, we note that by Proposition 4.2.2, moving a single irregular element without type-changing implies that the corresponding $\sqrt{CC}$ irregular vertex's graph distance to another vertex with fixed labeling is changed by an even number, while type-changing an irregular element implies that the graph distance is changed by an odd number.

Proposition 4.2.3 *It is impossible to create, delete, move, or type-change a single irregular element in an otherwise regular QD region R.*

Proof Being able to do so implies that we can produce another QD region R' with the same boundary but with a single created, deleted, moved, or type-changed irregular element. R and R''s $\sqrt{CC}$ domains are two quad meshes with the same boundary but with a created, deleted, or moved single irregular vertex. This contradicts Theorem 7.1 of Peng *et al.* (2011) (with an imaginary convex quad mesh region extended from R''s $\sqrt{CC}$ domain mesh). ∎



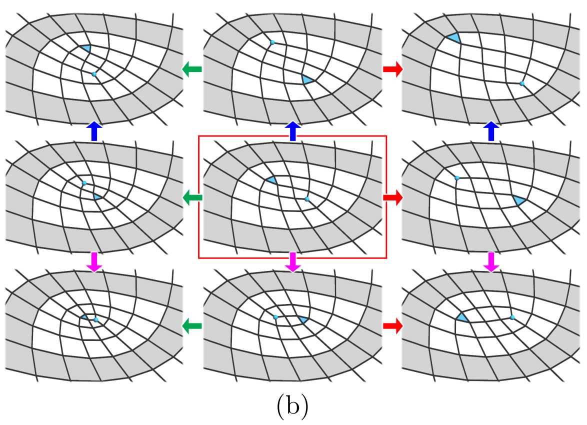


Figure 4.3: All Possible Moving Directions of a $v3$-$v5$ Pair.
(a) A $v3$-$v5$ pair can be moved in the left (green arrows), right (red arrows), up (blue arrows), and down (purple arrows) direction. (b) A $v3$-$f3$ pair can be moved closer (green arrows), farther apart (red arrows), rotating clockwise (blue arrows), and rotating counter-clockwise (purple arrows). Note that a single step would switch their types, thus a type-preserving movement can be realized by two consecutive steps. The gray faces are marked for ease of inspection.

Because editing a single irregular element is impossible, we are interested in editing irregular elements in pairs.

Proposition 4.2.4 *Two irregular elements can be moved together in an otherwise regular QD region. Irregular elements of the opposite indices, e.g. an $f3$-$v5$ pair, translate in the same direction. Irregular elements of the same index, e.g. an $f5$-$f5$ pair, rotate and scale around a fixed point in the middle between the two irregular elements. Furthermore, the irregular elements change types at each step. See Figure 4.3a and 4.3b for illustrations.* ■

Proof A sketch of the proof is as follows. Two irregular elements can move in the same way as two irregular vertices in the corresponding $\sqrt{CC}$ mesh as described in Theorem 7.2, 7.3, and 7.4 of Peng *et al.* (2011). Regarding the type-changing behavior, a single movement in the $\sqrt{CC}$ mesh would change both the irregular vertices' graph distances to another vertex with a fixed labeling by one, thus changing the types of their corresponding primal elements.

Proposition 4.2.5 *An irregular element pair can be merged into a single irregular element with a higher absolute index, e.g. two* $v3$ *can be merged into a single* $v2$ *and two* $f5$ *can be merged into a single* $v6$*, if and only if they have the same type and same index.*

Proof An irregular element pair of the opposite indices translates in the same direction thus cannot be merged. For a pair with the same index and different types, e.g. a $v3$-$f3$ pair, the graph distance between their corresponding $\sqrt{CC}$ irregular vertices is odd by Proposition 4.2.2, thus cannot be merged into a single irregular vertex by Theorem 7.2 and 7.3 of Peng *et al.* (2011). Otherwise it can be merged. ∎

4.2.4 T-Junction Editing

T-junctions, i.e. adjacent $v3$-$f5$ pairs, are often of special interest in QD remeshing. We describe how a single T-junction can be moved and how a pair of T-junctions can be canceled as follows.

Proposition 4.2.6 *A T-junction can be moved in exactly four directions denoted as up, down, left, and right.*

Proof A T-junction can be moved exactly in the same way as the corresponding adjacent $v3$-$v5$ pair (the $v3$ is a p-vertex and the $v5$ is a d-vertex) in the $\sqrt{CC}$ domain. The left and right directions are both realized by applying an edge split followed by a quad collapse. The up direction is realized by two consecutive quad collapses. The down direction is realized by two consecutive edge splits. There are no other combinations. Note that applying a single step would switch the T-junction to be an $f3$-$v5$ pair. See Figure 4.4 for an illustration. ∎

Proposition 4.2.7 *A pair of T-junctions in an otherwise regular convex region* R *can be completely canceled if and only if* R *is a parallelogram, i.e. a 4-sided region* *with both pairs of opposite sides of the same graph length. Otherwise it can be reduced to be a single irregular element pair of opposite indices and of the same type, e.g. a* $v3$-$v5$ *or an* $f3$-$f5$ *pair.*

Proof We first note that R must have 4 sides since it is convex and has a sum of indices of irregular elements of 0. If R is a parallelogram, it is straightforward to see that it can be remeshed as a regular grid, thus canceling the pair of T-junctions. Otherwise we perform a triple cancellation of irregular vertices in the $\sqrt{CC}$ domain to cancel one $v3$-$v5$ pair against the $v3$ or $v5$ of the second pair. In either case the result is a single $v3$-$v5$ pair, of which the graph distance is two, thus having the same type. ■

Based on Proposition 4.2.7, we can identify four possible configurations (in terms of relative orientations) of a T-junction pair. These configurations including their cancellations are shown in Figure 4.5.

4.3 Editing Framework

To make the user interface intuitive, we design our editing framework such that edits can be done directly in the primal domain. However (as stated in Section 4.2.1), implementing editing operations in a QD mesh directly would be very complicated since the operations need to handle the large number of combinations of both kinds of irregular elements. Our proposed solution is as follows. First, user inputs in the primal domain are mapped to the $\sqrt{CC}$ domain. Actual edits are then carried out in the $\sqrt{CC}$ domain by building on existing work in quad mesh connectivity editing. Finally, the edited QD mesh is obtained by an inverse $\sqrt{CC}$ subdivision. In our prototype we also allow editing in the $\sqrt{CC}$ domain to enable easier analysis of the underlying concepts. The user can edit using the operations described in the following.

Basic Operations: For analysis purposes and full flexibility, we identify five basic operations for QD meshes that operate on a per-edge level (Figure 4.2): a *vertex split* and an *edge insertion* both add a single edge, an *edge collapse* and an *edge deletion* both delete a single edge, and an *edge shift* shifts an edge toward one of its adjacent faces. They all can be realized by applying three basic operations for quad meshes (edge split, quad collapse, and edge flip) in the $\sqrt{CC}$ domain, thus eliminating the need for implementing each of them explicitly. Edge splits and quad collapses can each map to two different operations in the primal domain. The difference arises because the vertices in the $\sqrt{CC}$ domain have two different labels (p-vertices or d-vertices), resulting in two different outcomes of inverse $\sqrt{CC}$ subdivisions. We point out that an edge collapse and an edge shift is respectively equivalent to a collapse and a shift step of the GP operators proposed by Bommes et al. Bommes *et al.* (2011).

Pair-wise Movement: The user can move two irregular elements of arbitrary indices and types, e.g. a $v3$-$v3$, a $v3$-$f5$, and an $f5$-$f5$, in four possible directions. Each of the four possible movement directions is visualized by a pair of arrows with the same color in both the primal and $\sqrt{CC}$ domains (Figure 4.6). The user can select one moving direction for one irregular element and the movement of the other irregular element is constrained.

With this pair-wise movement operation, it is possible to do triple cancellations (e.g. an $f3$-$f5$-$v5$ to a single $v5$) by selecting a pair of irregular elements and colliding one element with a third element of an opposite index. Four irregular elements can be canceled at once when both elements of an irregular element pair collide with other elements of the opposite indices simultaneously. Cancellation can be done automatically, by computing the best movement path using a shortest path algorithm. The shortest path algorithm should include topological as well as geometric cost terms (e.g. curvature) and is therefore only a heuristic. Therefore, the manual selection of the movement path needs to remain as an important option.

Type-Change: As mentioned in Section 4.2.3, an odd number of pair-wise movements in the $\sqrt{CC}$ domain would not only move but also change the types of both irregular elements, e.g. a $v3$-$f5$ to an $f3$-$v5$ and a $v3$-$v3$ to an $f3$-$f3$. This is useful for users to change the types of irregular elements without introducing additional ones.

Single Conversion: The user can also change the type of a single irregular element at the cost of introducing an adjacent irregular element pair of opposite indices, e.g. an $f5$ to a $v5$ plus an $f3$-$v5$ pair or an $f3$ to a $v3$ plus an $v3$-$f5$ pair (Figure 4.7). Alternatively, it can be viewed as changing the sign of a single irregular element's index and introducing an adjacent irregular element pair of the same index. It is realized by applying a $v3/v5$ movement and $v3$-$v5$ pair generation operation in the $\sqrt{CC}$ domain.

T-junction Movement and Cancellation: As described in Section 4.2.4, the user can move a T-junction in the up, down, left and right directions. With this T-junction movement operation, it is possible to completely cancel or reduce a pair of T-junctions to be a single irregular element pair by colliding their adjacent $v3$-$v5$ pairs in the $\sqrt{CC}$ domain, depending on the conditions described in Proposition 4.2.7. Like the triple cancellation operation, the movement path can be found automatically with topological and geometric heuristics or manually by the user.

Smoothing: The user can smooth the mesh using Laplacian smoothing with back projection similar to the methods used in Tarini *et al.* (2010). T-junctions can be treated as a special case to ensure that the two edge segments are collinear as described in Lai *et al.* (2008). The visualizations of the edited meshes shown in this chapter are generated in this fashion unless otherwise specified.

4.4 Applications and Results

We implemented our mesh editing framework in C++ using CGAL cga (2014) such that all edits could be performed interactively.

Comparing Various QD Mesh Design Strategies: We evaluated different QD mesh editing strategies in the context of mesh design and mesh optimization. In our first example, we modeled three versions of a wing of the Yas-Island architectural model and optimized the three meshes for planarity and angle deviation (approximated by the fairness term) Liu *et al.* (2006). In Figure 4.9, we visualize the optimization results. Our analysis shows that pure quad meshes are best for ensuring planarity, although meshes with mixed types of faces can achieve better angle deviation (and better smoothness of mesh lines).

In Figure 4.8, we show three versions of a tower model. The first version is a pure quad mesh and the third version is the dual of the first mesh and has therefore no irregular vertices. The second version has both irregular vertices and faces. This example was chosen to illustrate that the flexibility of quad-dominant meshes generally allows the designer to achieve smoother mesh lines using irregular faces in smooth or flat regions while preserving sharp features using irregular vertices.

Mesh Connectivity Improvement: In Figure 4.10, we illustrate the T-junction editing capabilities of our framework. T-junctions can be merged or moved to alter a mesh design.

Finally, in Figure 4.11, we show connectivity editing of a highly irregular mesh model (generated by the remeshing algorithm of Lai et al. Lai *et al.* (2008)). Even though there are highly irregular regions (including a 9-sided polygon), our editing framework can easily facilitate local mesh improvement. While we do not aim to replace automatic remeshing algorithms, local mesh editing has many unique advantages and provides a nice complement.

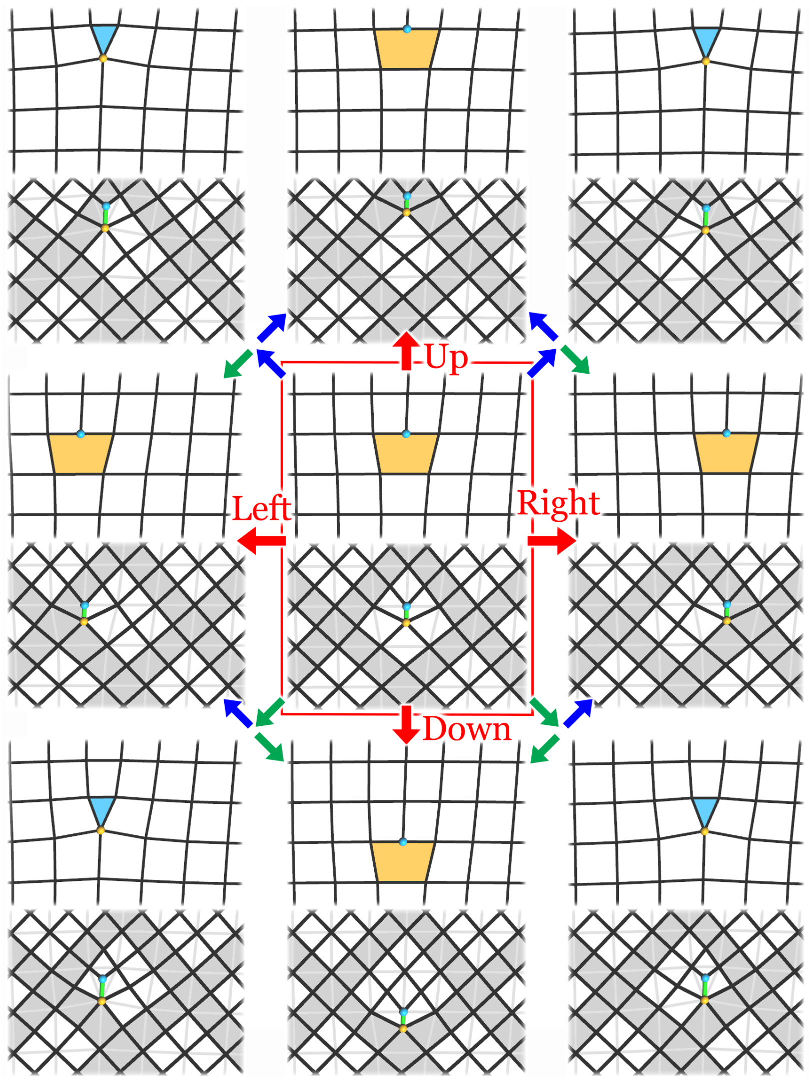


Figure 4.4: The Four Possible Movement Directions for a T-junction.
Each direction (a red arrow) is realized by applying two consecutive atomic basic operations (quad collapses, blue arrows, and edge splits, green arrows) to the corresponding $v3$-$v5$ pair in the $\sqrt{CC}$ domain. The gray faces are marked for ease of inspection.

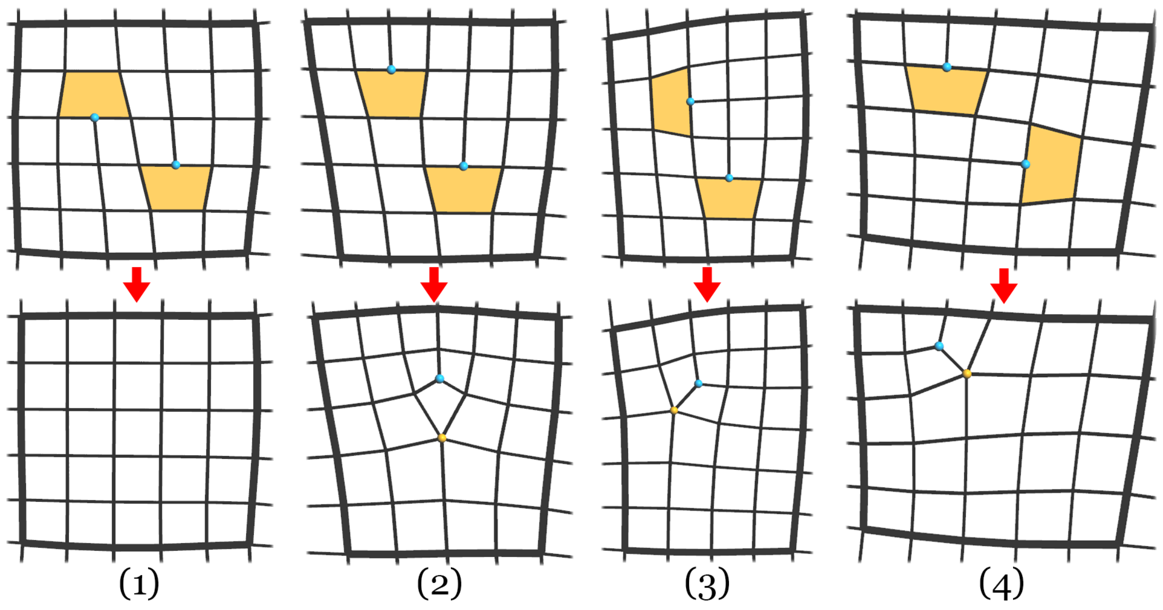


Figure 4.5: Cancellation of a T-junction Pair.
How a T-junction pair can be canceled in the four possible relative orientations (up to rotational symmetry) within an otherwise regular (4-sided) region R. Two T-junctions can be completely canceled if and only if they are facing the opposite sides of R (1). Otherwise they are either facing the same side (2) or two adjacent sides (3,4) and can be reduced to an irregular element pair of the same type.

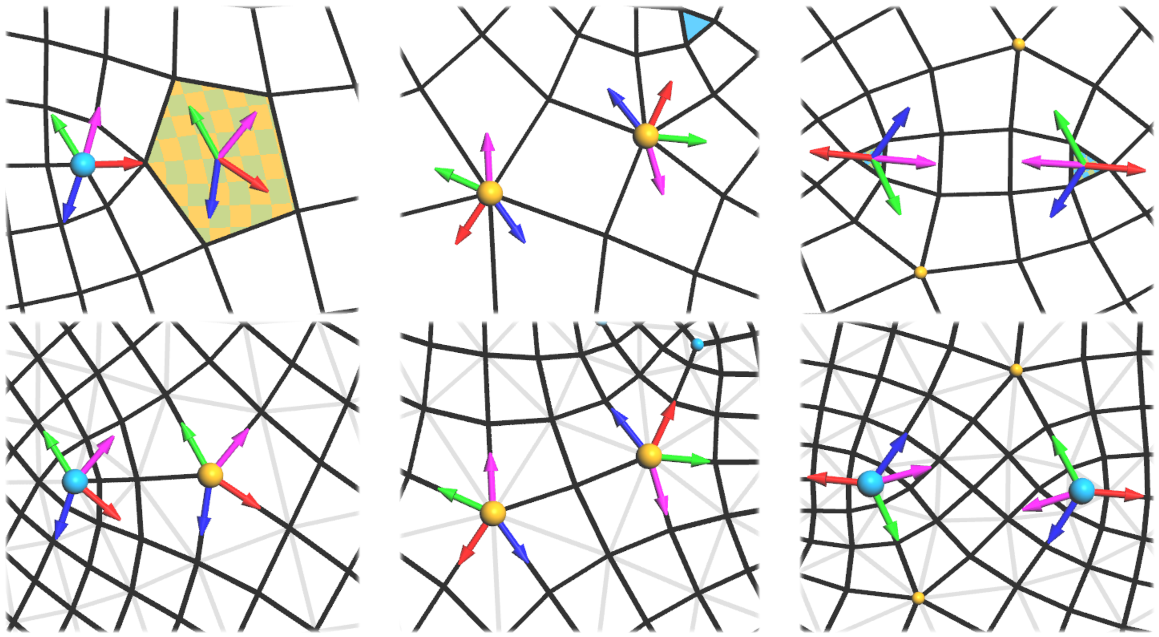


Figure 4.6: Visualization of the Four Possible Movement Directions of a Pair of Irregular Elements.

Left: a $v3$-$f5$ pair. Middle: a $v5$-$v5$ pair. Right: an $f3$-$f3$ pair) in both the primal (top) and $\sqrt{CC}$ (bottom) domains.

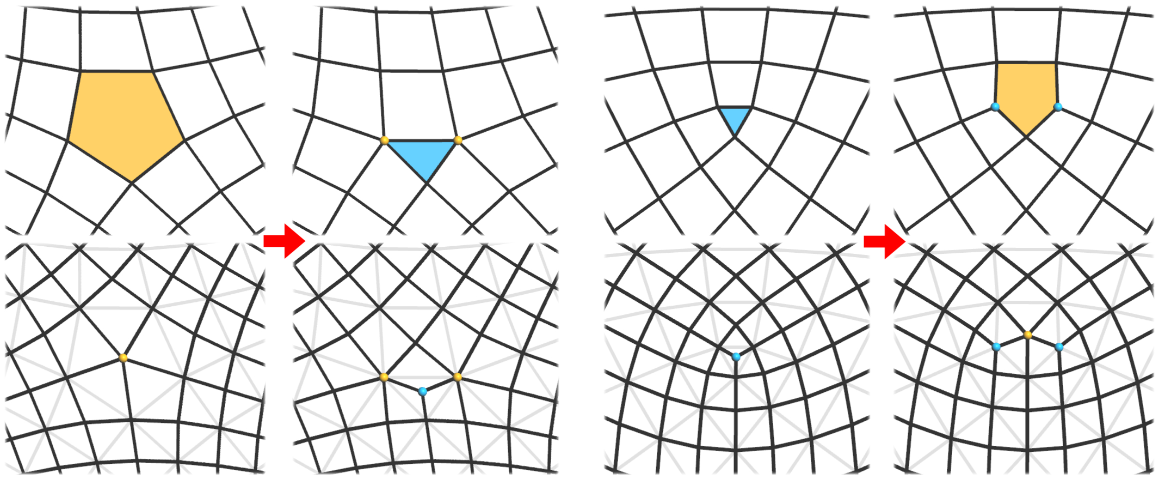


Figure 4.7: Converting the Type of a Single Irregular Element.
Left: an $f5$ is converted to an $f3$-$v5$-$v5$ triple. It can be viewed as type-changing the $f5$ to a $v5$ or changing the sign of its index (an $f5$ to an $f3$). An adjacent irregular element pair is introduced. Right: a similar operation where an $f3$ is converted to a $v3$-$v3$-$f5$ triple. The corresponding $v3/v5$ movement and $v3$-$v5$ pair generation operations in the $\sqrt{CC}$ domain are shown below.

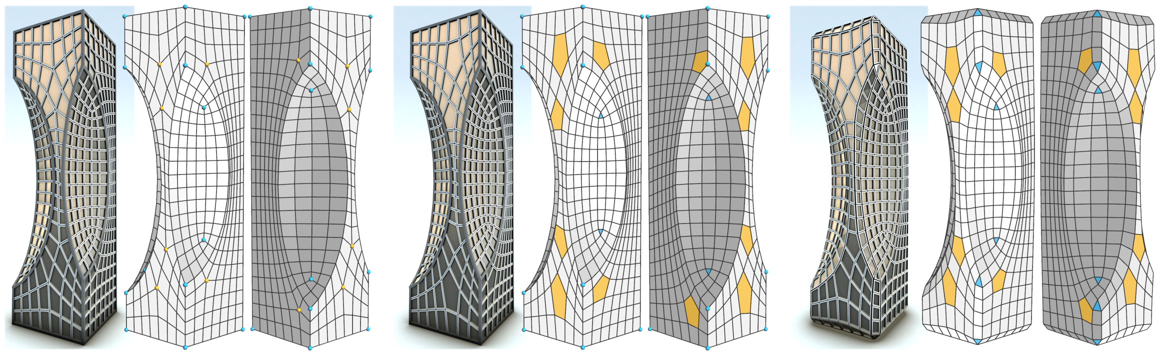


Figure 4.8: A Comparison of Different Kinds of QD Mesh Irregularity.

In our framework, the user can edit irregular vertices (with more or fewer than four neighbors) and irregular faces (non-quads) of a quad-dominant mesh. Each type of irregularity has different advantages and disadvantages in design and construction. Irregular vertices are necessary to maintain sharp features (corners and edges), but they create higher angle deviations in mesh lines in smooth regions (left). Irregular faces lead to smoother mesh lines, but they cannot maintain sharp features (right). The ability to model with a mixture of irregular vertices and faces gives more flexibility to the user, e.g. creating a design with sharp features and smooth mesh lines (middle). We render each model in a style that highlights the sharp features.

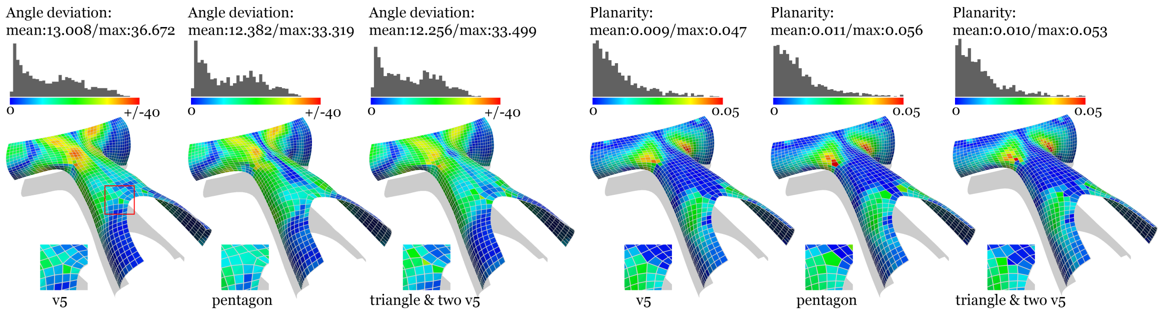


Figure 4.9: Geometric Optimizations of Three QD Mesh Designs of an Architectural Panel Structure.

Geometric optimizations of three QD mesh designs of an architectural panel structure: a pure quad mesh with four $v5$, a QD mesh with four pentagons, and a QD mesh with four triangles plus eight adjacent $v5$. Left three: the three meshes are optimized toward equiangular faces, which is approximated by the fairness term Liu *et al.* (2006). Right three: the three meshes are optimized toward planar faces. Their mean and max errors are listed and visualized (explained in Figure 4.10). In general, by allowing a mixture of face types the meshes can be optimized with better angles. On the contrary, pure quad meshes work slightly better with planarity optimizations.

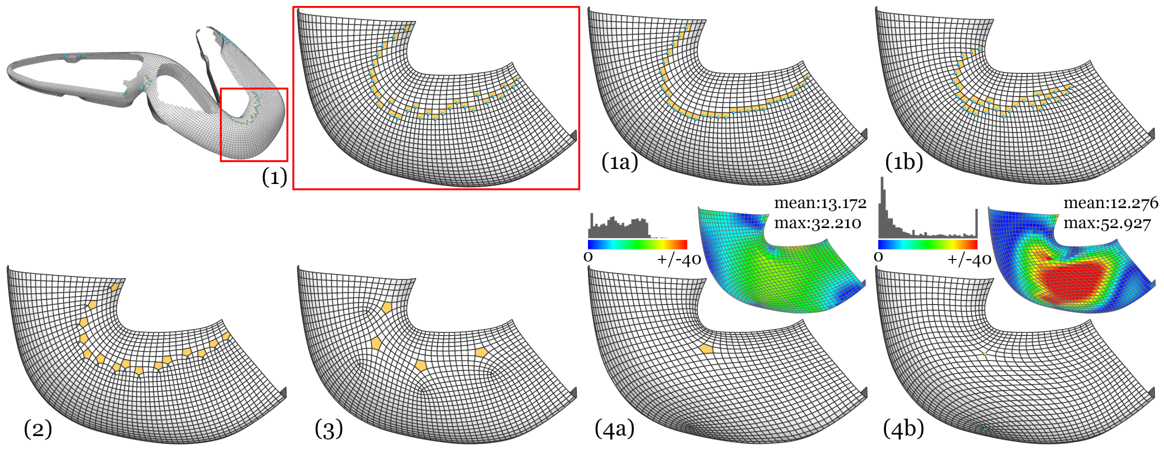


Figure 4.10: Finding Alternative QD Meshes for a Semi-Regular Quadrangulation with T-junctions of an Architectural Structure.

(1) We focus on a part of the whole structure that has an excessive amount of T-junctions in order to align with the underlying curvature. We first explore alternative T-junction patterns by moving them vertically (1a) and horizontally (1b). (2) to (4a) We progressively merge nearby T-junctions until we are left with a pair of irregular elements with opposite indices (an $f3$-$f5$ pair), at the cost of being less aligned with the curvature. The user can choose a version that compromises mesh regularity and curvature alignment. (4b) Alternatively, we change the $f3$-$f5$ pair to a $v3$-$v5$ pair. Interestingly, the QD mesh with irregular faces (4a) can be smoothed to a greater degree than can the QD mesh with irregular vertices (4b), resulting in smoother mesh-lines and less extreme angle deviations (on top we show the histograms and color visualizations of the angle deviations. Each face is visualized by the averaged deviation from the mean value of corner angles).

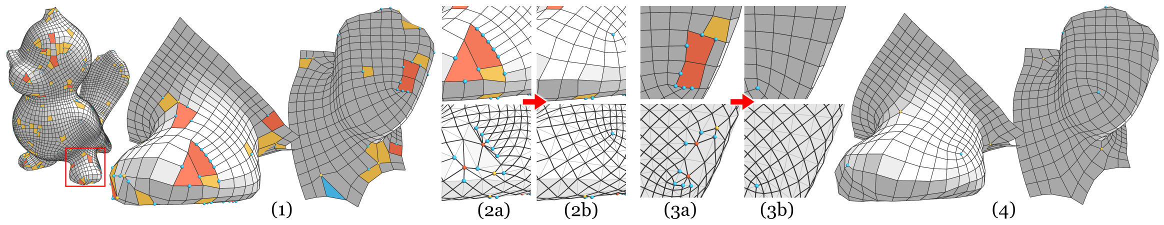


Figure 4.11: Connectivity Improvement of a QD Remeshing from Lai *et al.* (2008). (1). (2a) to (2b) We improve a region with faces with very large degrees (due to multiple incoming T-junctions) by regularizing the corresponding $\sqrt{CC}$ domain (shown below). (3a) to (3b) Another improved region. (4) The final improved mesh. We first remove excessive and nearby irregular elements. Next we move the remaining irregular elements to regions with corresponding curvatures. For demonstration purposes, we further change all irregular faces to be irregular vertices to make the mesh pure quad.

Chapter 5

EXPLORING QUADRANGULATIONS

Existing quadrangulation algorithms tackle surface remeshing problems using optimization frameworks. While users may obtain slightly different results by adjusting the optimization parameters, a systematic exploration of alternative quadrangulations is not feasible. By contrast, our goal is to help users to explore all possible topologically unique quadrangulations of an input mesh in an efficient and organized way.

The first challenge is the enumeration of topologically unique quadrangulations. Without constraints, the possibilities are innumerable. Thus, we take a two-stages approach: in the first stage, the input mesh is segmented into surface patches, typically along sharp features. Since patch boundaries need to be matched, i.e., no T-junctions are allowed, finding the boundary configurations that make all patches jointly quadrangulatable may be challenging. We show that the problem can be formulated as a linear integer program. In the second stage, topologies are enumerated for each patch with the guarantee that the patch boundaries will match.

Exhaustively enumerating all possible quadrilateral topologies for a patch within a reasonable time is made possible by the following fact: assuming that the number of irregular vertices should be minimized, any complex patch of a quadrangular mesh can be subdivided into a collection of certain patches that we call *simple patches*. Fueled by this idea, our enumeration algorithm subdivides each patch into a collection of *simple patches* that are quadrangulated by a closed-form solution. The task of enumerating topological variations thus become much more manageable because we only need to enumerate different ways to perform the aforementioned subdivisions.

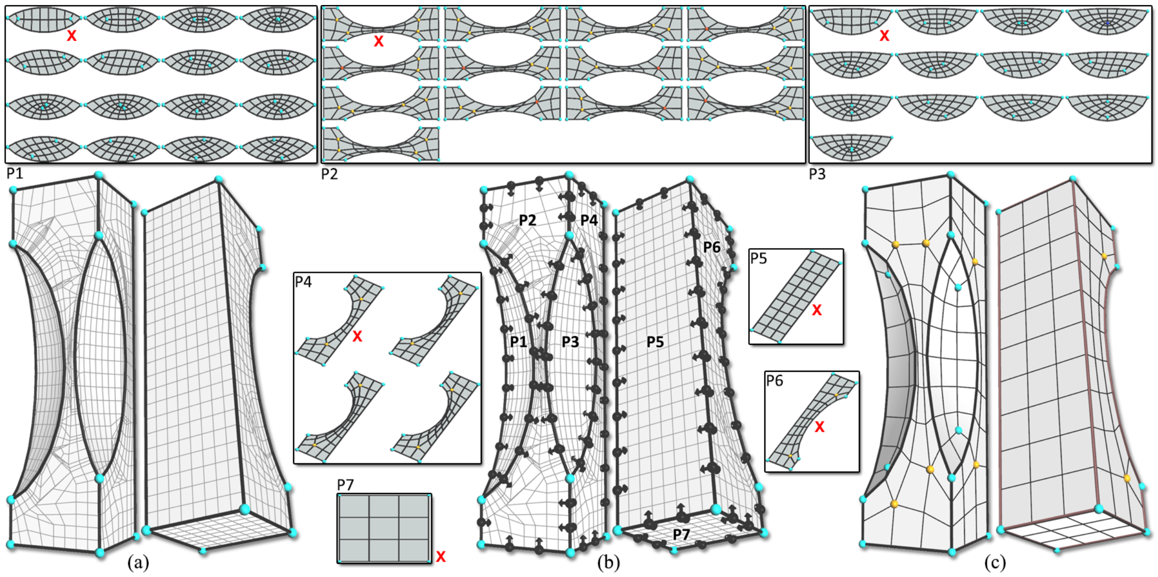


Figure 5.1: Overview of Our Quadrangulation Framework.
(a) A control graph is generated by segmenting the underlying input mesh into a collection of surface patches. (b) The numbers of vertices on each edge can be computed by integer programming such that all patches are quadrangulatable by the minimum number of irregular vertices. Exhaustive enumerations of all topologies with the minimal number of irregular vertices for every patch are shown. (c) By picking one desired topology for each patch (marked), a full requadrangulation of the mesh can be generated.

Inundating users with hundreds, even thousands, of possible variations in an arbitrary order is a job only half done. The second challenge is to help users efficiently navigate the solution space. We propose three approaches to tackle this challenge. First, the enumeration can be guided by a sampling method, so that a snapshot of the whole solution space can be quickly retrieved. Second, variations that are topologically similar, i.e., isomorphic graphs under a rotational symmetry, can be clustered to reduce visual clutter. Third, variations can be sorted by both the topological and the geometric characteristics of the quadrangulation.

Finally, we demonstrate why topological exploration is important by showing several examples for which alternative quad mesh layouts can be useful.

5.1 Chapter Overview

5.1.1 Basic Definitions

The *valence* of a vertex, v, which we denote as $l(v)$, is the number of edges in the mesh incident to v. A vertex with valence n is denoted as vn, e.g., $v3$ and $v5$. A $v4$ vertex is considered as *regular*, and vertices of other valences are referred to as *irregular*. We consider irregular vertices with valences lower than 3 or higher than 5 as multiple $v3$ or $v5$ collocated together (and therefore count them as multiple irregular vertices).

Definition 5.1.1 *A* path γ *is a sequence of edges* $e_i = (v_i, v_{i+1})$ *for* $0 \leq i < R$. R *is the* length *of* γ. *A path is a* loop *if* $v_0 = v_R$. *Otherwise,* γ *is an* open path.

Definition 5.1.2 *A (quadrilateral)* patch P *is a connected subset of the quadrilaterals in a mesh* M *without handles and it is enclosed by one or multiple loops, which we denote as* P*'s* boundaries. *A patch is* regular *if there are no irregular vertices in its interior. Each vertex along the boundary can be a* non-corner, convex corner, *or* concave corner. *Convex and concave corners divide the boundary into several subpaths, which we denote as* sides. *A patch is* convex *if it does not contain any concave corners; otherwise, it is* concave.

Definition 5.1.3 *Each boundary loop of a patch can be encoded by the length of sides and the type (convex or concave) of corners encountered during a closed, counterclockwise walk, beginning at an arbitrary vertex. During the walk, we keep a conceptual facing direction as a signed integer, beginning at zero. It is incremented by one when a convex corner is encountered, and decremented by one when a concave corner is encountered. We assign each side a direction given the current facing direction when* *it is encountered. Sides with the same direction are grouped together to form an* effective side*. A patch with N effective sides is called an N-sided polygon or N-gon for short.*

Definition 5.1.4 *For a patch with all irregular vertices in its interior, Int, the total valence deficit, TVD, is $\sum_{i \in Int} 4 - l(v_i)$.*

The extension to patches with irregular vertices on the boundary is straightforward but cumbersome to describe so we omit it here. Interestingly, the TVD of a patch can be derived from its boundary loops, a direct result of the discrete Gauss-Bonnet theorem for surface with boundaries. For a patch with one boundary loop, its TVD can be derived as $4 - n$, where n is the number of convex corners minus the number of concave corners on the boundary. The TVD of patches with multiple boundary loops is described in Section 5.3.1.

5.1.2 Framework Overview

An overview of our framework is shown in Figure 5.1. The input to our system is a polygon mesh of arbitrary type, e.g., triangular, quadrilateral, or hybrid, representing a two-manifold surface. The input mesh serves as guidance for generating the *control graph*, which is a cage-like structure that encodes a patch segmentation of the input mesh (Section 5.2). The patches of the control graph serve as the inputs to our quadrangulation algorithm that has the ability to exhaustively enumerate all possible quadrilateral topologies, i.e., remeshing, within each patch (Section 5.3). To help users navigate the potentially huge space of possibilities, we propose a sampling strategy such that a snapshot of the whole solution space can be quickly retrieved (Section 5.3.4). In Section 5.4, we show that the task of finding boundary constraints that make all patches jointly quadrangulatable can be formulated as a linear integer programming problem.

5.2 Control Graph Modeling

A control graph is a cage-like structure that encodes a segmentation of the input mesh. In essence, it is a coarse two-manifold mesh comprised of (curved) edges on the input mesh and faces that we call *surface patches.* The surface patch is required to be pathwise-connected, without handles, with at least one boundary loop, but not necessarily simply connected, e.g., a topological disc with zero or more holes. In our system, a control graph is generated either automatically by detecting sharp features using angle thresholds, or interactively by clicking on specific edges, or imported from another existing algorithm.

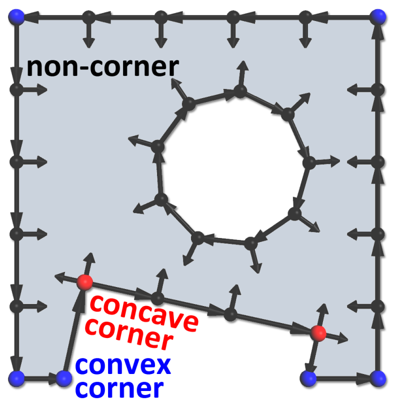
The control graph also encodes topological constraints for the subsequent patch quadrangulations that can be obtained by linear integer programming to satisfy additional requirements, e.g., all patches need to be quadrangulatable. For each edge of the control graph, we determine the number of vertices on the edge such that the number of boundary vertices for all patches is given. Further, we define the number of *inward* edges for each vertex on the boundary by classifying the vertices as convex, concave, or non-corner emanating none, two, or one inward edges, respectively. See Figure 5.2a for an illustration. In the following, we present an algorithm and interface to enumerate and explore the different topologies for one patch at a time with fixed boundary constraints. In a typical usage scenario, the user iterates between exploring the topologies of different patches and editing topological boundary constraints.

Parameterization: We build a 2D parameterization for each patch to initialize the vertices with 2D positions during the topological enumeration. We typically use LSCM Lévy *et al.* (2002) in favor of its conformal and open boundary traits. We emphasize that a pure topological enumeration can work even without parameterization. The parameterization is used simply for visualization and sampling/ranking purposes. Inputs to the parameterization are the faces and vertices of the input mesh enclosed by the patch's boundary. Since sharp features are typically captured in the control graph, we assume that the parameterization will be of reasonable quality. In practice, LSCM may generate non-bijective parameterizations if the patch's shape is highly concave or has inner boundaries. We currently work around the problem by subdividing patches. A better solution to this problem would require implementation of more advanced parametrization algorithms.

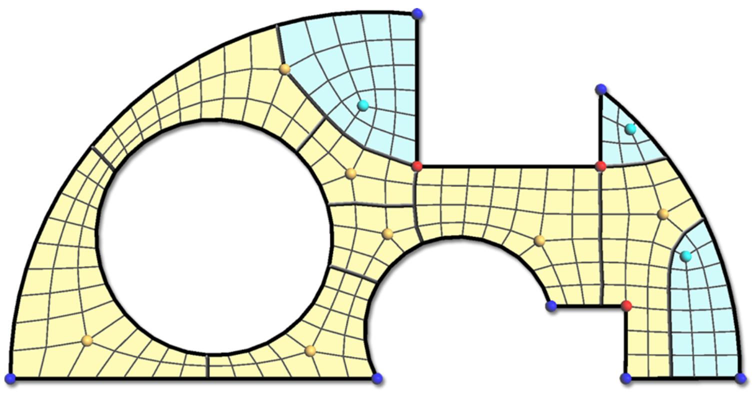
5.3 Enumerating Quadrangulations for Patches

The input for this stage is a patch including its boundary configuration, i.e., the vertices at the boundary and their prescribed number of inward edges. A quadrangulation has to form connections between all inward edges. At a glance, the task can be intimidating. There are $O(x!)$ possible ways to connect x inward edges such that an exhaustive enumeration is simply infeasible. Besides, inner irregular vertices, of which the types, numbers, and positions are unknown, can divert the connections and create additional complexity. The main idea of our approach is to use the observation that every quad mesh can be partitioned into certain *simple* patches that contain one or zero irregular vertices (See Figure 5.2b). The enumeration problem is thus greatly reduced to enumerating subdivisions into simple patches.

Another important idea of our approach is to devise an exploration strategy that can enumerate a reasonable subset of all possible quad meshes. Since every non-trivial boundary corresponds to infinitely many pure quad meshes, we need to restrict the enumeration somehow. Our assumption is that irregular vertices are typically considered as undesirable in our primary target applications, because they break the pattern on the surface. Our exploration algorithm is therefore geared towards



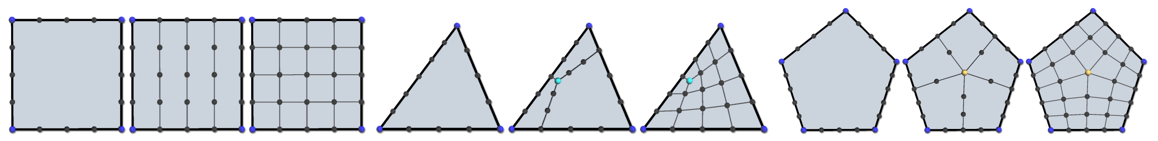
(a)



(b)

Figure 5.2: Definition of a Path and the Decomposition of a Complex Quadrangulation.

(a) A boundary configuration of a patch with two boundary loops, each shown as a strip of arrows in counter-clockwise (outer) and clockwise order (inner). Prescribed inward edges are shown as arrows pointing inwards. Note that convex corners (shown in blue) emanate no inward edge while concave corners (shown in red) emanate two inward edges. (b) Decomposing a complex quadrangulation into a collection of simple triangle (blue) and pentagon (yellow) patches.



(a) (b) (c)

Figure 5.3: Subdivision Steps to Quadrangulate a Simple Convex Patch.
Subdivision steps to quadrangulate a (a) parallelogram, (b) triangle, and (c) pentagon. Inner $v3$s are shown in cyan and inner $v5$s are shown in orange. For a triangle and a pentagon, the topological position of the irregular vertex is uniquely obtained by solving a linear system. For visualization purposes, we locate the inner $v3$ or $v5$ at the least squares solution of the location that is perpendicular to the nearest boundary vertex on each side in the parameterization domain.

enumerating solutions with the minimal number of (inner) irregular vertices ($v3$ and $v5$) k, $k \geq |TVD|$. We can therefore also bound the number of quads to a reasonable number as a secondary criterion.

In this section, we first present the overall subdivision algorithm in Section 5.3.1. The algorithm relies on the efficient enumeration of subdivisions, which is described in Section 5.3.2. Strategies to filter redundantly generated subdivisions are described in Section 5.3.3. For larger examples, the enumeration might be time consuming. We therefore propose a fast sampling strategy to generate interesting topology variations early in Section 5.3.4. Finally, we propose tools to preview and arrange the topological variations in Section 5.3.5.

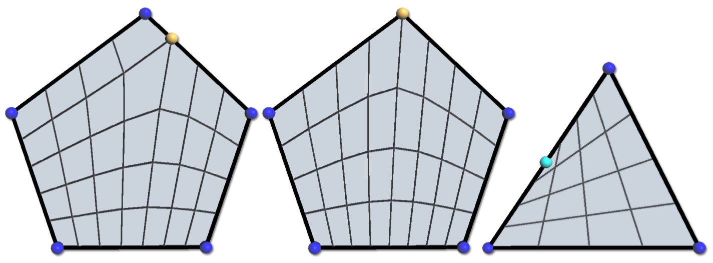
5.3.1 Subdivision-based Quadrangulation of Patches

Our quadrangulation algorithm hierarchically subdivides a patch into smaller sub-patches until every sub-patch has become a quad. The process can be described by a *subdivision tree*: the root node is the input patch, the internal nodes are sub-patches that need further subdivision, and the leaf nodes are quads. We explore this tree depth first, and for each interior node we first try to classify the patch and then subdivide it. In order of complexity, we distinguish five categories: 1) simple convex patches, 2) simple concave patches, 3) patches with $|TVD| \leq 1$, 4) general patches with a single boundary loop, and 5) patches with multiple boundary loops. We describe how to classify and quadrangulate these five categories of patches in the following. The categories form a nested hierarchy, so that each lower category is a subset of all higher ones. The strategy of the quadrangulation algorithm is then to split higher-category patches into lower-category patches.

Simple Convex Patches

A simple convex patch is either a parallelogram, simple triangle, or simple pentagon, defined as follows.

Definition 5.3.1 *A parallelogram is a convex* 4*-gon with two pairs of opposite sides of the same length. A simple triangle is a convex* 3*-gon that can enclose exactly one* $v3$*. A simple pentagon is a convex* 5*-gon that can enclose exactly one* $v5$ *(Figure 5.3).*



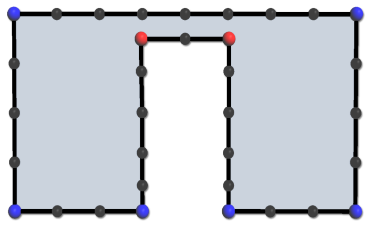


Figure 5.4: Examples of Boundary-case and Non-embeddable Patches.
(a) Left and middle: Two examples of boundary-case pentagons. Right: A boundary-case triangle. (b) A patch with a boundary that is incompatible with its potential embedding in a 4-by-8 parallelogram.

In the following discussions, we refer to simple triangles and pentagons as triangles and pentagons. Recall from Peng *et al.* (2011) that the topological position, i.e., the nearest boundary vertex on each side and the graph distances in between, of a single $v3$ or $v5$ within a convex 3-gon or 5-gon can be uniquely derived by solving a linear system formed by the side lengths. An inner irregular vertex is feasible if and only if all distances to the sides are positive. If some distances are zero, the irregular vertex actually lies on the corresponding boundaries, which we denote as *boundary cases*. Interestingly the feasibility criteria are analogous to their Euclidean space counterparts: For a convex 3-gon to be a triangle, the length of the longest side has to be smaller than the sum of the other two. For a convex 5-gon to be a pentagon, the sum of the longest consecutive two sides has to be smaller than the sum of the other three.

Quadrangulation of Simple Convex Patches: A parallelogram is recursively subdivided into smaller parallelograms along the longer pair of opposing sides (Figure 5.3a). For a triangle or a pentagon, we first create the inner $v3$ or $v5$ and connect it to each side according to the solution of the linear system mentioned previously. The connections subdivide the patch into three or five parallelograms, which are subsequently subdivided (Figure 5.3b, 5.3c). Special care is taken for boundary cases: a triangle with a boundary $v3$ is equivalent to a parallelogram and a pentagon with a boundary $v5$ is equivalent to a combination of multiple parallelograms (Figure 5.4). In practice, we pre-calculate the quadrangulations of simple convex patches of various side lengths to accelerate the algorithm.

Simple Concave Patches

A simple concave patch is a single-boundary loop concave patch that can be embedded in a simple convex one with the same TVD, denoted as the patch's *extended patch*. This is achieved by the *cave-filling algorithm* (Appendix A). Figure 5.5 shows simple concave patches embedded inside a parallelogram, triangle, or pentagon.

To determine if patch P is simple concave, we check the following in order. 1) The TVD of P has to match a parallelogram (0), triangle (1), or pentagon (-1). 2) P has a valid extended patch, $\hat{P}$, according to the cave-filling algorithm. 3) The side lengths of $\hat{P}$, which is a convex polygon with three to five sides, have to be compatible with a triangle, parallelogram, or pentagon. 4) We check if the embedding is valid, i.e., the boundary of P does not penetrate the boundary of $\hat{P}$ (computed by a boundary walk). A counter-example is shown in Figure 5.4b.

Quadrangulation of Simple Concave Patches: Conceptually, we first quadrangulate the extended patch and then remove the quads that were added by the cave-filling algorithm. For an accelerated implementation we can directly compute the splits for the concave patch in closed form (Figure 5.6).

Patches with $|TVD| \leq 1$

This category covers arbitrary (possibly concave) single-boundary loop patches, P, with $|TVD| \leq 1$ that do not fall in the previous two categories. There are the following possibilities: 1) P does not have a valid extended patch $\hat{P}$ according to the

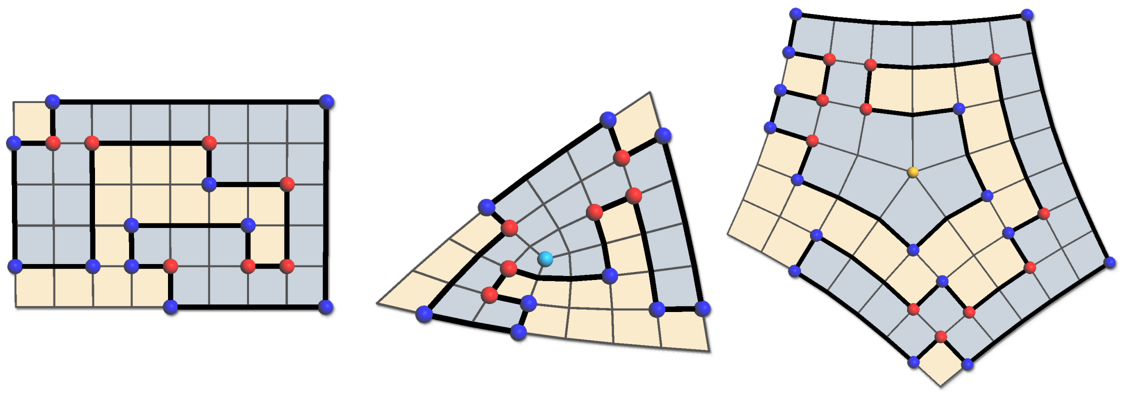

(a) (b) (c)

Figure 5.5: Embeddings of Simple Concave Patches.
Simple concave patches embedded inside a (a) parallelogram, (b) triangle, and (c) pentagon. P is shown in gray and the embedding into $\hat{P}$ is shown in yellow.

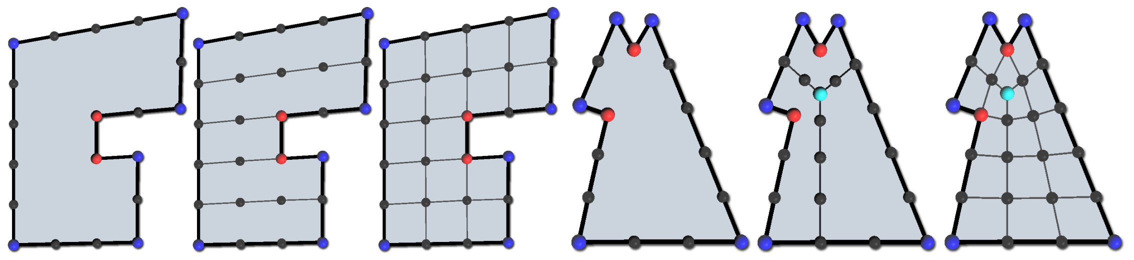

(a) (b)

Figure 5.6: Closed-form Solutions for Quadrangulating Simple Concave Patches Embeddable Inside a Parallelogram (a) and a Triangle (b).

cave-filling algorithm. 2) The side lengths of $\hat{P}$ are incompatible with a parallelogram, triangle, or pentagon. 3) P's embedding is invalid, i.e., the boundary of P penetrates the boundary of $\hat{P}$. Our strategy is to subdivide such a patch recursively until it is decomposed into a collection of simple concave or convex patches, explained in the following.

For the second kind of patch, additional inner $v3$-$v5$ pairs are necessary for the patch to be quadrangulatable. Note that $v3$-$v5$ pairs do not affect the patch's TVD. We distinguish three cases:

- $\hat{P}$ is a convex 4-gon but not a parallelogram. The quadrangulation requires one or more $v3$-$v5$ pairs. We subdivide P into a triangle, which can be quadrangu-

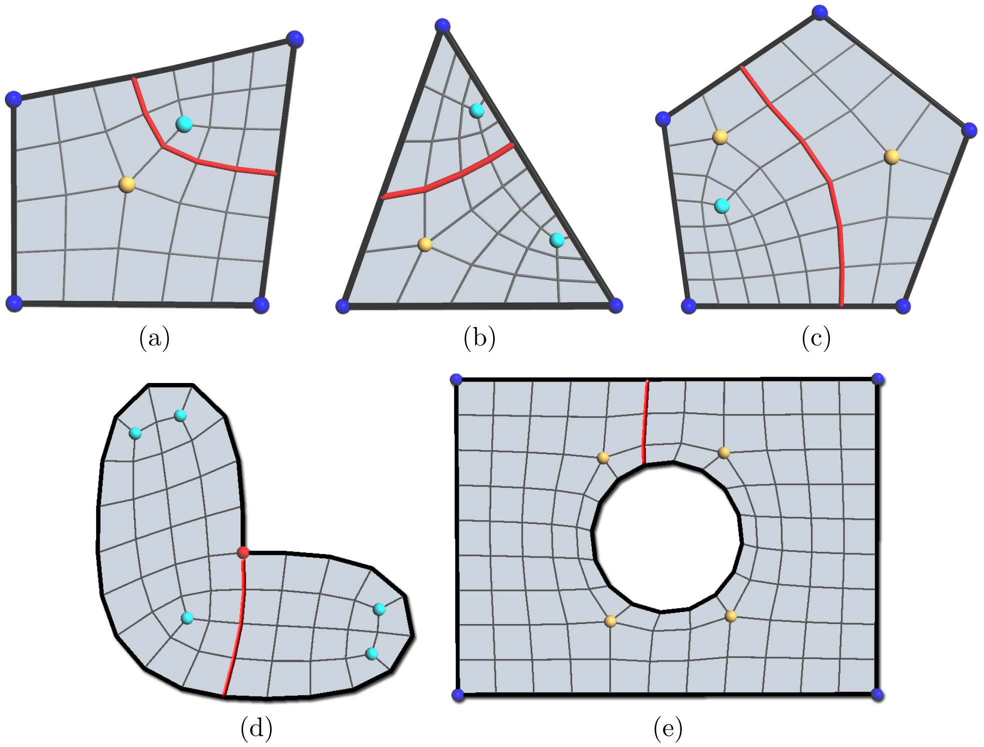


Figure 5.7: Subdivisions of Non-simple Patches.
(a) A non-simple patch with $TVD = 0$ is subdivided into a triangle and a pentagon. (b) A non-simple patch with $TVD = 1$ is subdivided into a triangle and a general 4-gon. (c) A non-simple patch with $TVD = -1$ is subdivided into a pentagon and a general 4-gon. (d) A patch with $TVD = 5$ is subdivided into two sub-patches of $TVD = 3$ (left) and 2 (right). (e) A patch's two boundary loops are combined into one by making a cut connecting the outer and inner boundary loops. The subdivisions/cuts are shown in red.

lated in closed form, and a general 5-gon, which is subsequently quadrangulated (Figure 5.7a).

- $\hat{P}$ is a convex 3-gon but not a triangle. The quadrangulation requires a $v3$ plus one or more $v3$-$v5$ pairs. We subdivide P into a triangle and a general 4-gon (Figure 5.7b).

- $\hat{P}$ is a convex 5-gon but not a pentagon. The quadrangulation requires a $v5$ plus one or more $v3$-$v5$ pairs. We subdivide P into a pentagon and a general 4-gon (Figure 5.7c).

For the first and third kinds of patch, we simply subdivide it heuristically by the scoring function described in Section 5.3.4.

General Single-Boundary Loop Patches

This category covers arbitrary single-boundary loop patches (not classifiable in the previous categories). Since these patches have to be subdivided, our aim is to minimize the number of subdivisions by keeping the TVD of the two sub-patches as equal as possible. Empirically, this leads to fewer splits and approximately logarithmic time complexity. Thus, at every recursion, a patch of $TVD = t$ is subdivided into two sub-patches of $TVD = \lceil |t|/2 \rceil$ and $\lfloor |t|/2 \rfloor$. An example is shown in Figure 5.7d where a patch with $TVD = 5$ is subdivided into two sub-patches of $TVD = 3$ and 2.

Multi-Boundary Loop Patches

This category covers patches with $g + 1$ boundary loops ($g > 0$). Since the patch is connected, the boundary loop with the largest bounding box in the 2D parameter domain is distinguished as the outermost contour, while all other boundary loops are considered contours surrounding inner holes. To quadrangulate P, edge strips connecting the outermost contour and each inner contour are required. A connection bridges the inner contour to the outermost contour with four additional convex corners at the two joints. Eventually, all boundary loops are merged and the patch has a unified boundary loop. The TVD of the patch thus can be calculated as the number of convex corners minus the number of concave corners on all boundary loops, plus the additional four convex corners per connection.

To quadrangulate a multi-boundary loop patch, we first transform it into single-boundary loop by making g aforementioned connections suggested by the scoring function described in Section 5.3.4. An example is shown in Figure 5.7e.

5.3.2 Enumerating Subdivisions

Recall that our quadrangulation algorithm recursively subdivides a patch until it is decomposed into a collection of simple convex and simple concave patches, which are then uniquely quadrangulated. At each subdivision, we face the choices of 1) which pairs of inward edges to connect and 2) how many inner vertices to generate on the connecting path, i.e., its (topological) length. Enumerating these choices at every recursion is equivalent to enumerating the topologies of a patch. The guidelines to constrain the enumeration are given below.

The choices of which pair of inward edges to connect are well constrained: when subdividing patches of the third and fourth categories, only connections that would lead to sub-patches with desired TVD are acceptable. When subdividing patches of the fifth category, only connections of inward edges of different boundary loops are acceptable. Constraints for the length of the connection path are described as follows.

1. The lengths of boundaries of the resulting sub-patches need to be even, otherwise no quadrangulation exists.

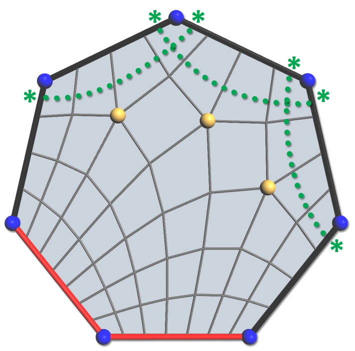
2. Except for parallelograms, which can be quadrangulated without inner irregular vertices, the length of every effective side needs to be ≥ 2 since such a sub-patch must accommodate at least a triangle (for $v3$) or a pentagon (for $v5$), of which every effective side length is at least 2.

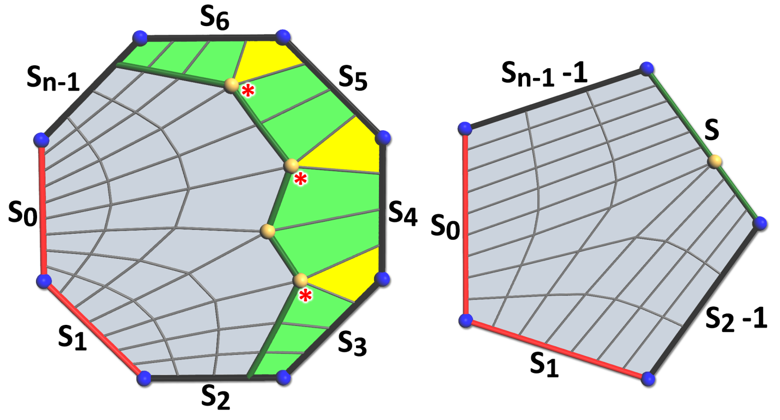
3. Requirements for sub-patches to be simple (parallelogram, triangle, and pentagon), which occur at subdividing patches of the third category, constrain the length of the connection to be a single value (parallelogram) or within a range (triangle and pentagon).

4. Requirements for sub-patches to have more than five effective sides ($TVD <$

-1), which occur at subdividing patches of the fourth category, constrain the length of the connection under the assumption of Theorem 5.3.2.

Note that the third and fourth constraints are applicable only under the assumption that redundant $v3$-$v5$ pairs are to be avoided; otherwise, connections of arbitrarily long lengths can be accommodated by an arbitrary amount of $v3$-$v5$ pairs.


(a)



(b)

Figure 5.8: Supporting Figures for the Proof of Theorem 5.3.2.
(a) A quadrangulation of a 7-sided convex patch with the inequality in Theorem 5.3.2 being exactly satisfied, i.e., the longest consecutive pair of sides (red) cannot be any longer if the lengths of other sides are fixed. The $2|TVD|$ edges on the other sides and their emanating polychords that can never reach the longest pair are marked in green. (b) Left: quadrangulation of an 8-sided convex patch with the inequality in Theorem 5.3.2 being exactly satisfied. Right: the corresponding base pentagon quadrangulated with a boundary-case solution. Side S of the pentagon is shown in green, which is expanded to be the patch's sides S_3 to S_6 with a strip of quads (green and yellow) and the three inner $v5$s (marked) inserted accordingly. Note that the lengths of S_2 and S_{n-1} are also increased by one.

Theorem 5.3.2 *Under the assumption that a patch is convex, $TVD < -1$, and the sum of the lengths of boundaries is even. Then, the patch is quadrangulatable without inner $v3$-$v5$ pairs $\Longleftrightarrow$ the sum of the lengths of the longest consecutive pair of sides $\leq$ the sum of the lengths of all other sides minus $2|TVD|$.*

Proof $\Rightarrow$: Every polychord (Daniels *et al.* (2008)) emanating from an edge of the longest consecutive pair of sides must end at the other sides; otherwise, it would

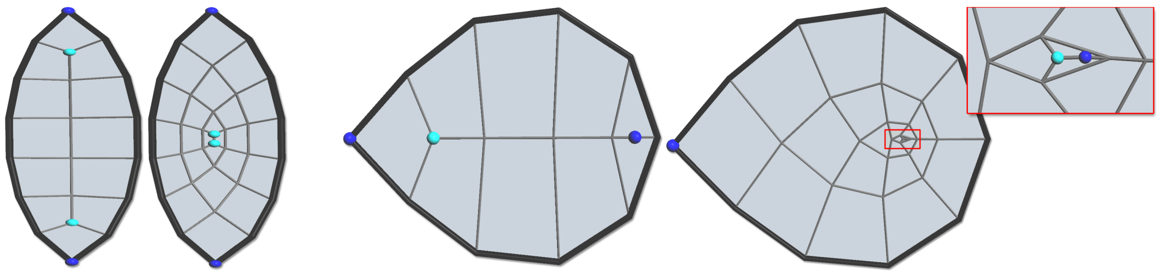


Figure 5.9: The Quadrangulations with the Minimal and Maximal Numbers of Quads for a 2-sided Patch.

The quadrangulations with the minimal and maximal numbers of quads for a 2-sided patch with side lengths $(7, 7)$ (left) and a 1-sided patch with side length 10 (right). Note that there are two $v3$ collocated as a $v2$ in the quadrangulations of the 1-sided patch.

subtract a 2-sided polygon (ending at the same side) or a triangle (ending at the adjacent side) out of the patch. In both cases, an inner $v3$ is implied, a contradiction to our assumption that patch has $TVD < -1$ (thus having $v5$) and no $v3$-$v5$ pairs. Furthermore, there are $2|TVD|$ edges on the other sides that can never emanate a polychord to the consecutive pair of sides (Figure 5.8a).

$\Leftarrow$: We denote the lengths of the longest consecutive pair of sides as s_0 and s_1 and the lengths of the other sides as s_2 to s_{n-1} in counter-clockwise order. Our idea is to show that there exists a corresponding *base* pentagon, see Figure 5.8b, that can be quadrangulated with a boundary-case solution, and that there always exists a way to extend the quadrangulation of the pentagon to be a quadrangulation of the patch. The lengths of the base pentagon's sides are: s_0, s_1, $s_2 - 1$, S, and $s_{n-1} - 1$ in counter-clockwise order, where $S = \sum_{i=3}^{n-2} s_i - 2(|TVD| - 1)$. Note that $s_i \geq 2$, $0 \leq i < n$ by the aforementioned second constraint. It is straightforward to see that the pentagon can be quadrangulated with a solution in which the $v5$ is lying in the interior of side S. A quadrangulation of the patch can be then generated by inserting a strip of quads and $|TVD| - 1$ $v3$-$v5$ pairs (the $v3$s are on the patch's boundary, serving as corners, and the $v5$s are internal) at corresponding locations right next to side S. See Figure 5.8b for an example. ∎

On the other hand, for sub-patches with $TVD > 1$, the length of an effective side can be arbitrarily long, since there exists a polychord that begins and ends at the same effective side. Since the choice of the length of the connection is unbounded, we resort to the constraints for the maximal number of quads described next.

Theorem 5.3.3 *When quadrangulated with two v3s and no v5, the maximal number of quads in a 2-sided patch ($TVD = 2$) with side lengths b_0 and b_1 is $\lfloor b_0/2 \rfloor \lceil b_1/2 \rceil + \lceil b_0/2 \rceil \lfloor b_1/2 \rfloor$.*

Theorem 5.3.4 *When quadrangulated with three v3s and no v5, the maximal number of quads in a 1-sided patch ($TVD = 3$) with side length b_0 is $(b_0 b_0)/4 - 1$.*

Proofs of Theorem 5.3.3 and 5.3.4 are based on analyzing the decomposition of the patches into triangles and are omitted here. A 2-sided and a 1-sided patch quadrangulated with the maximal and minimal numbers of quads are shown in Figure 5.9. Finally, for sub-patches with $TVD \geq 4$, the number of quads can be arbitrarily large even with fixed side lengths, since such a patch may contain an infinite amount of inner polychord cycles. For such cases, we constrain the length of the connection by geometric heuristics.

Thresholding the Number of Irregular Vertices: $|TVD|$ determines the lower bound of (inner) irregular vertices ($v3$ and $v5$) required to quadrangulate a patch. The lower bound can be achieved only if irregular vertices of one type are solely used ($v3$ or $v5$). However, such solutions may not be feasible, e.g., for skewed patches, additional $v3$-$v5$ pairs will be needed. Restricting the number of irregular vertices is thus equivalent to setting an upper limit on the number of $v3$-$v5$ pairs. To retrieve solutions with the minimally possible numbers of irregular vertices, our strategy is to enumerate quadrangulations with an increasing upper limit of $v3$-$v5$ pairs, beginning at zero. Empirically, we found that the space of enumerations grows exponentially with the number of allowed $v3$-$v5$ pairs. The computational overhead of the unsuccessful trials with insufficient upper limits for $v3$-$v5$ pairs can thus be neglected.

5.3.3 Filtering Redundant Enumerated Topologies

A challenge of our enumeration framework is that multiple enumerated subdivision trees can result in equivalent topologies, corresponding to the multiple ways to decompose a quadrangulation into simple patches. We reduce exploration of redundant subdivision trees by filtering *parallel* connections defined as follows.

Definition 5.3.5 *Two connections, $V1 - V2$ connected by $L1$ edges and $V3 - V4$ connected by $L2$ edges, are* parallel *if: 1) $V1$, $V3$ are on the same side and $V2$, $V4$ are on the same side, 2) $L1 = L2$, and 3) the lengths of the boundaries between $V1$, $V2$ and $V3$, $V4$ are the same.*

From a set of parallel connections, we pick the highest ranked one according to the heuristics described in Section 5.3.4 and discard the rest. Examples of parallel connections are shown in Figure 5.10a. It is straightforward to see that the resulting topologies of the parallel connections are different only by a parallelogram in between (assuming that $v3$-$v5$ pairs are to be avoided) and thus are equivalent. Note that this filtering strategy is not yet exhaustive, and there may be redundant connections being explored. An analysis of the filtering performance is provided in Section 5.5.

Finally, we filter enumerated topologies that are equivalent, i.e., graph isomorphic. Recognizing graph isomorphism can be done in linear time in our case since the topologies share the same patch boundaries. An algorithm is described in the

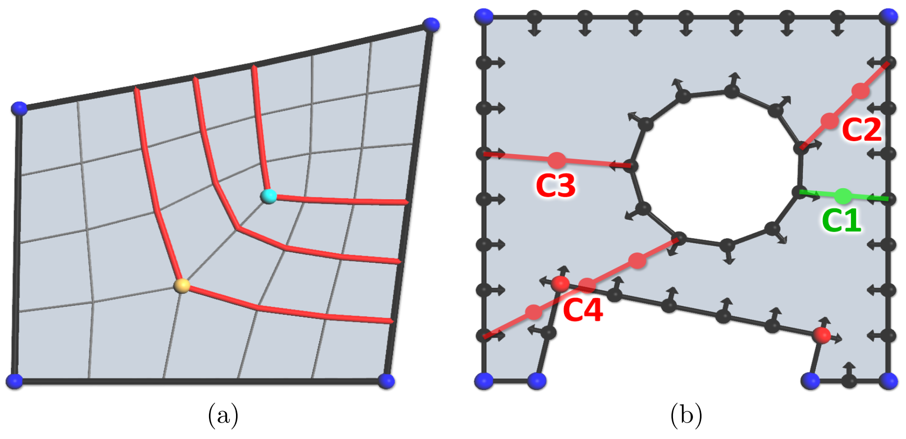
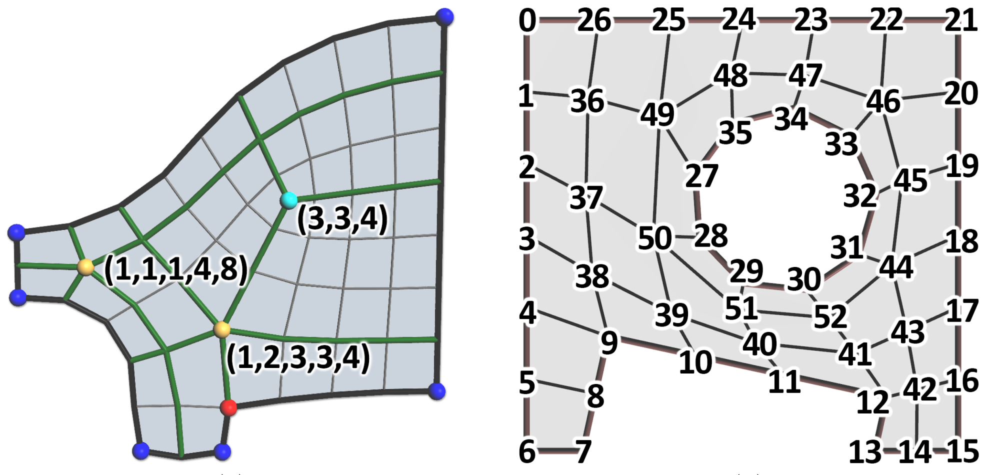


Figure 5.10: Parallel Connections and the Ranking of Connections.
(a) Three parallel connections are shown in red. All of them subdivide the patch into a triangle and a pentagon and result in equivalent topologies. Note that the top one leads to a boundary-case triangle and the bottom one leads to a boundary-case pentagon. (b) Comparing the ranking of connections. Connection $C1$ is top-ranked. $C2$ is ranked lower because it is less perpendicular to the boundary edges at its two vertices. $C3$ is also ranked lower because the geometric length of its subdivided edges deviates from the average length of the patch's boundary edges. $C4$ is ranked lowest because it goes outside the 2D parameter domain.

following.

Recognizing Graph Isomorphism: Vertices of a certain topology are ordered in increasing order according to their topological distance from the patch boundary. The boundary vertices, of which the distances are 0, are sorted in a counter-clockwise fashion starting at a fixed boundary vertex. Inner vertices with distance d ($d > 0$) are sorted as follows. We define the *parent vertex* of an inner vertex as the one that is sorted foremost among its adjacent vertices of distance $d - 1$. For two inner vertices $V1$ and $V2$, $V1$ is sorted prior to $V2$ if $V1$'s parent vertex is sorted prior to $V2$'s and vice versa. If their parent vertices are the same, $V1$ is sorted prior to $V2$ if it is prior among the parent vertex's neighbor list and vice versa. Such orderings can be found by a simple flooding strategy in linear time. It is straightforward to see that such an ordering is unique for a given topology. This algorithm is thus guaranteed to detect graph isomorphism among equivalent topologies. See Figure 5.11b for an example.



(a) (b)

Figure 5.11: Topology Profile and Ordering of Vertices.
(a) A quadrangulation of which the topology profile is $\{(3,3,4),(1,1,1,4,8),(1,2,3,3,4)\}$. (b) Ordering of vertices in a patch for the graph isomorphism test.

5.3.4 Ranking and Sampling Topological Variations

We can retrieve a quick snapshot of the whole space of enumerations by sampling only a subset of connections at each recursion. First, we rank each connection by a simple scoring heuristic considering the following geometric properties. Highly ranked connections have a better chance to generate quadrangulations with better geometric qualities and vice versa. Assuming that the connection is a straight line in the 2D parameter domain connecting $V1$ and $V2$ uniformly subdivided into $L1$ edges, we first measure how perpendicular the connection is to the boundary edges at $V1$ and $V2$. A more perpendicular connection means that we have a better chance of forming right angles at $V1$ and $V2$. Second, we measure the difference of the geometric length of the $L1$ subdivided edges to the average geometric length of the patch's boundary edges. A smaller difference implies a more appropriate choice of $L1$. Third, we favor the geometrically shorter among connections that are equally ranked by the first two properties. Finally, we penalize connections that go outside the 2D parameter domain, which may happen for patches with geometrically concave parts. An illustration of the ranking is shown in Figure 5.10b.

After the connections are sorted by the heuristic, we can perform sampling in an uniform or greedy way. In the uniform way, we sample one connection among similarly ranked connections at each recursion. A snapshot that covers a variety of quadrangulations can thus be retrieved. In the greedy way, we simply sample the top-ranked connections, such that a subset of quadrangulations with better geometric qualities can be retrieved. In both ways, the exploration is done in a depth first manner but with the number of child nodes at each branch reduced.

5.3.5 Navigating Topological Variations

We develop a real-time browsing interface to help users visually navigate topological variations. Note that only unique topologies are presented after the aforementioned filtering process. Since enumerated topologies are updated in real time, the aforementioned ranking heuristics play an important role by determining which topologies are presented first. A reasonable 2D visualization for each topology is generated by applying Laplacian smoothing with the boundary vertices fixed.

Users can choose to sort the topological variations by three types of statistics: 1) density (number of quads), 2) geometric quality criteria described in Yang *et al.* (2011), and 3) the topology profile described next.

Assume that there are n (inner) irregular vertices in a topology, v_i, $0 \leq i < n$, sorted in ascending order of valences: $l(v_i) \leq l(v_{i+1})$, $0 \leq i < n-1$. For each irregular vertex, v_i, the sequence, $s_{i,j}$ for $0 \leq j < l(v_i)$, denotes the lengths of emanated separatrices stopped by hitting the boundary or other interior irregular vertex, which are sorted in ascending order. Sequences of each irregular vertex are further sorted in ascending order by considering each sequence as a decimal number. For example, a sequence of $3-4-5$ from a $v3$ is placed before a sequence of $0-1-2-3-4$ from a $v5$. The sequences of all irregular vertices congregated in the sorted order serve as a topological profile summarizing the relative positions of the inner irregular vertices, see Figure 5.10.

Furthermore, topological variations that are rotationally equivalent, i.e., corresponding to one topology being rotated, have the same topological profile. Such variations can be identified and clustered to reduce visual clutter.

5.4 Integer Programming

Assigning numbers of vertices lying on each boundary edge of a control graph path, i.e., their (topological) length, can be a non-trivial task when there exist additional requirements for the patches. We formulate the task as a linear, pure integer programming problem in which the variables are the lengths of edges and each patch imposes its requirements as linear constraints. We denote the length of the i-th edge as a positive integer variable, $L_i, 0 \leq i < n$, where n is the number of edges. A minimal requirement for all patches to be quadrangulatable is that the sum of side lengths is even for every patch, written as $\sum_{L_i \in P_j} L_i - 2S_j = 0$ for every patch P_j (S_j are positive integer slack variables). It is straightforward to see that feasible solutions exist, e.g., if every side length is even. To further require that certain patches can be quadrangulated without $v3$-$v5$ pairs, the following constraints are added:

- For a patch with non-zero TVD, lengths of every side ≥ 2. This corresponds to the second constraint of connection lengths in Section 5.3.2.

- For a triangle ($TVD = 1$), parallelogram ($TVD = 0$), and pentagon ($TVD = -1$), side lengths are constrained by Definition 5.3.1.

- According to Theorem 5.3.2, for a patch with $TVD < -1$, the sum of the lengths of every consecutive pair of sides $\leq$ the sum of the lengths of all other sides minus $2|TVD|$.

The above constraints are applicable only under the assumption that a patch is convex and has a single boundary loop. Therefore, patches with multiple boundary loops and concave patches need to be split with the heuristics described in Section 5.3.1 and 5.3.4 first. In practice, we apply the above constraints to all patches initially. If the system is overconstrained, we heuristically drop constraints for selected patches until the system becomes feasible.

Feasible solutions to the above problem guarantee that all patches are jointly quadrangulatable; however, they may be geometrically undesirable, e.g., geometrically long edges are segmented sparsely and vice versa. To address this problem, every edge L_i is given an optimal topological length O_i as the rounded ratio of its geometric length to a desired edge length. We first impose the following constraints: $|L_i - O_i| \leq C, 0 \leq i < n$, where C is a (non-negative integer) value thresholding the feasible range of each L_i. Furthermore, we formulate the cost function as $\sum_{i=0}^{n-1} |L_i/O_i - 1|$ under the assumption that the length of each optimally subdivided segment is one. Each nonlinear term ($|L_i/O_i - 1|$) is approximated as a linear function within the feasible range of L_i in a least-squares sense, e.g., as a linear equation passing through $(O_i - C, |(O_i - C)/O_i - 1|)$ and $(O_i + C, |(O_i + C)/O_i - 1|)$. A mathematical formulation of the integer programing problem is as follows.

$$\sum_{i=0}^{n-1} |L_i/O_i - 1| \rightarrow \min \quad \text{such that} \tag{5.1}$$

for every edge

$$L_i > 0 \quad \text{and} \quad |L_i - O_i| \leq C,\ i = 0, \ldots, n-1$$

and for every patch

$$\sum_{L_i \in P_j} L_i - 2S_j = 0.$$

Additionally, depending on the type of each individual patch, P_j,

- patches with non-zero $TVD : L_i \geq 2$,
- parallelograms : $L_{i_0} = L_{i_2}$,
- triangles : $L_{i_0} + 1 \leq L_{i_1} + L_{i_2}$,
- pentagons : $L_{i_0} + L_{i_1} + 1 \leq \sum_{i \neq i_{\{0,1\}}} L_i$,
- N-gons with $TVD < -1 : L_{i_0} + L_{i_1} + 2|TVD| \leq \sum_{i \neq i_{\{0,1\}}} L_i$

need to hold for all admissible $L_i \in P_j$. We denote L_{i_k} as the length of the k-th next side after L_i in P_j in a counter-clockwise order. For example, L_{i_0} denotes L_i itself, L_{i_1} denotes the length of the side next to L_i in P_j, and so on. We solve the problem by lpsolve Berkelaar *et al.* (2008).

5.5 Results and Applications

Complexity of the Enumeration: The only competing approach, that we are aware of, is the brute-force enumeration of all possible connections of inward edges, e.g., Marinov and Kobbelt (2006). For simplicity, we present an example where the number of inner irregular vertices of the patch is minimal, a 100 by 100 2-sided patch ($TVD = 2$). In the brute-force approach, there are a staggering $198!/2$ possible connections of the 198 inward edges, even before considering the effect of inner irregular vertices. In our approach, we need to enumerate all first-level subdivisions between the i-th ($1 \leq i \leq 99$) inward edge of the first side and the j-th ($1 \leq j \leq 99$) inward edge of the second side. By filtering, we just need to enumerate one connection among all pairs with $i-j = d$, where $-98 \leq d \leq 98$. After the first level of subdivision,

		$v3$-	boun.	unique		total	time	time
patch	TVD	$v5$	length	topo.	cluster	topo.	(total)	(first)
Figure 5.7a	0	1	22	12	12	36	0.17	0.01
Figure 5.7e,5%	-4	0	56	536	525	568	198.62	0.06
Figure 5.8b	-3	0	34	7	5	77	0.28	0.00
Figure 5.14,1	-2	0	66	4	4	10	0.43	0.08
Figure 5.14,2	2	0	36	57	35	120	4.34	0.05
Figure 5.14,3	-4	0	66	559	558	3064	79.89	0.06
Figure 5.13	1	1	30	30	30	232	1.59	0.03
Figure 5.18	4	0	16	254	52	5166	23.09	0.01

Table 5.1: Timing Statistics.

Timing table showing: 1) the number of unique topologies after filtering, 2) clusters of rotationally equivalent topologies, 3) total enumerated topologies, and 4) the time spent for complete enumerations and retrieving the first topology in seconds.

we have two simple triangles that are uniquely quadrangulated. In total, we just need to enumerate 197 possible cases.

Comparison to Peng *et al.* (2011): In the Peng et al. paper, an exhaustive enumeration of all possible topologies involving up to two irregular vertices ($v3$ and $v5$) is provided. In this chapter, we extend the idea to enumerate all possible topologies of a given patch exhaustively, upper bounded by the number of irregular vertices. Moreover, their method requires an initial mesh to work on.

Statistics: Table 5.1 shows the time spent on enumerating all topologies with the minimal number of irregular vertices (or capped at a reasonable number of quads if such topologies are infinite) for patches shown in this chapter, recorded on a standard 2.66GHZ PC. In most cases, the first topology is retrieved instantly. For patches with excessive amounts of variations such as Figure 5.7e, we only sample a fraction of all possibilities. We find that the time complexity does not depend on the number of vertices on the boundaries; instead, it is proportional to the number of non-parallel connections enumerated during the subdivisions.

Shape-Space Exploration: Shape-space exploration is a powerful tool Yang *et al.* (2011) to modify a planar quad (PQ) mesh, while faithfully preserving the planarity of the faces and satisfying additional constraints like proximity to a reference object, fairness, etc. Each mesh corresponds to a point in a high-dimensional *shape space*, and the planarity constraint forms a certain manifold in that space, called the planarity manifold. We apply shape-space exploration to investigate the most natural behavior of different topological patterns and to understand the significance of topology in the genesis of PQ meshes. While the original shape-space exploration itself gives rise to a large variety of meshes with one fixed topology, we can now explore an even wider spectrum of possible shapes by varying the topology and geometry together.

We demonstrate how shape-space exploration interacts with various topological patterns. In Figure 5.12, we show the variations of a 3D mesh whose planar elliptic boundary has been fully fixed. The variations are ranked according to three different geometric objectives (planarity, fairness, circularity) and color coded in the visualization such that we can observe which topologies are well suited for a specific geometric objective. A shape-space exploration that starts with a fully planar mesh is shown in Figure5.13. Six feature vertices are selected to remain in the plane while the remaining vertices are freely movable. The goal of this exploration is to see the natural shapes of each topology. As it is difficult to compare general shapes, we apply a geometric heuristic to select the best bump shape for thirty different topologies. This example illustrates that some topologies are more suitable for a selected target shape.

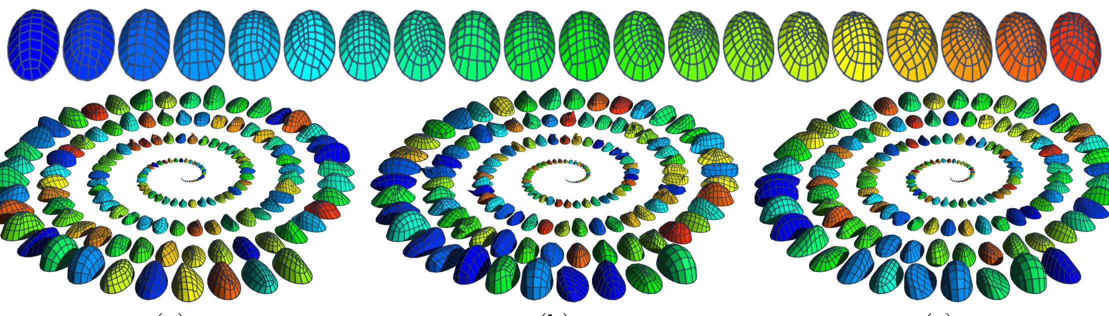

(a) (b) (c)

Figure 5.12: Shape-space Exploration on Different Topologies.
The cap of an elliptic paraboloid is PQ meshed using twenty different topological patterns (top row) and planarity-preserving shape-space exploration is applied on each topology such that the boundary is fixed while the interior vertices are allowed to move. The best eight eigendirections are sampled for each topology. The results (clockwise) are ranked in decreasing fashion according to planarity (a), circularity (b) and fairness (c). The color shows the affiliation with the topological pattern.

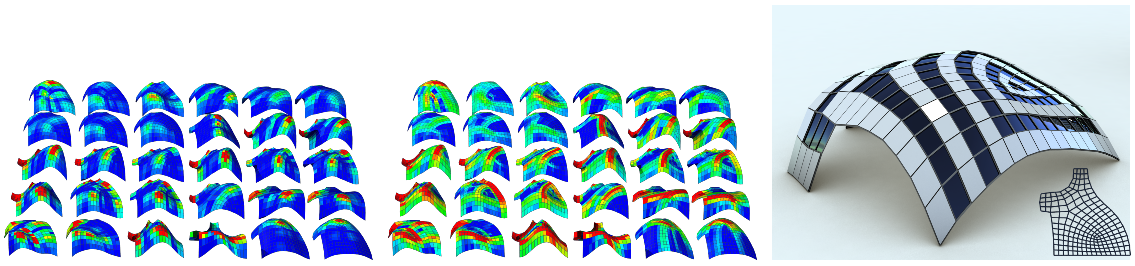

(a) (b) (c)

Figure 5.13: Shape-space Exploration of PQ Meshes.
For a 2D PQ mesh in the plane, six corner points are kept fixed and the remaining vertices move freely to form 3D shapes. From thirty topological patterns, the best "bumping" shape was selected. The color coding reflects the discrete Gaussian curvature (a) and the average kink angle between the neighboring faces (b). (c) The best explored mesh in terms of minimal kink angle together with its planar input is shown.

Exploring Alternative Requadrangulations: We show that our enumeration framework enables users to explore multiple distinctive requadrangulations of an input mesh. In Figure 5.14, we explore alternative requadrangulations of an architectural tower model. Each alternative has its own advantages and disadvantages. In general, there is a trade-off between sharp feature fidelity and the number of irregular vertices. In Figure 5.15, we requadrangulate the Fandisk model at resolutions that are coarser than the ones in Bommes *et al.* (2009) and Zhang *et al.* (2010) while maintaining all sharp features. In Figure 5.17, we requadrangulate the Accessory model in Zhang *et al.* (2010) to remove all the redundant $v3$-$v5$ pairs. In Figure 5.16, we explore alternative requadrangulations of a single curved patch with two inner holes. Throughout the examples, explorations are done not only by enumerating topologies of patches, but also by exploring different configurations of the control graphs. The vertex positions are optimized by methods desrcribed in Liu *et al.* (2006) as a post-process.

Art and Design: Topology is important for art and design when the mesh lines or quad faces are visible. We show designs of planar Shuriken (Japanese dart) patterns inspired by the enumerated topologies (Figure 5.18). In Figure 5.19, we show font designs by quadrangulating the interiors inspired by Bessmeltsev *et al.* (2012).

Limitations and Future Work: One major limitation of our work is that we restrict our analysis to pure quad meshes. A fruitful avenue of future work is to analyze the topology of mixed quadrilateral and triangular meshes. A limitation of the enumeration algorithm is that we cannot filter all redundant subdivisions early in the process. It would be interesting to explore if there were an optimal algorithm to detect all redundant efforts. We focus on examples from architecture, art, and design, in which control graphs can be trivially generated. While our framework is applicable to organic models, such as the bunny (Figure 5.20), manual efforts are required to

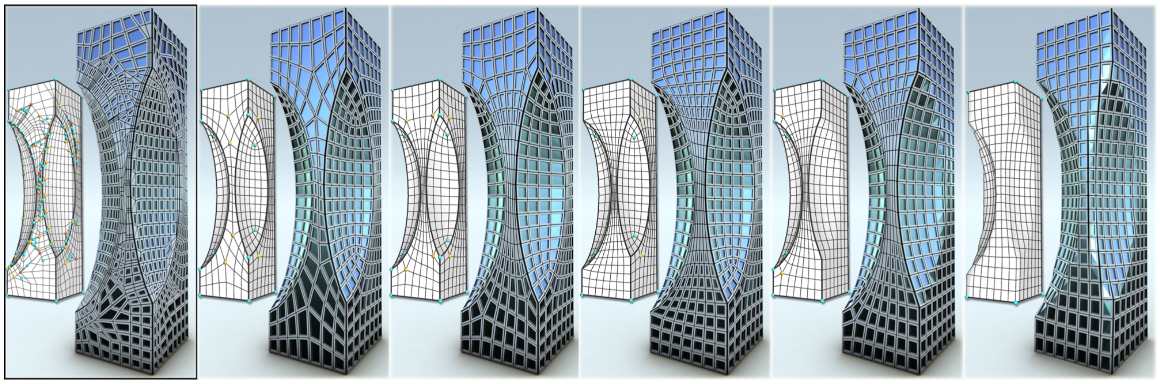


Figure 5.14: Exploring Alternative Requadrangulations of an Architectural Tower Model.

Left: the input tower model with a large number of irregular vertices and small faces due to the computation of intersections by a professional modeling package. Second left to right: five requadrangulations explored with our enumeration framework. The first and second both consider the hyperboloid-shaped facade on the front as an 8-sided patch, while the former favors uniform quad size and the latter favors alignment of irregular vertices. The third alternatively considers the front facade as a 4-sided patch, and four $v3$s are removed. The fourth no longer preserves the sharp feature on the right, and the two adjacent patches are merged. It has fewer irregular vertices at the cost of less sharp feature fidelity. The fifth discards all sharp features on the front in exchange for even fewer irregular vertices.

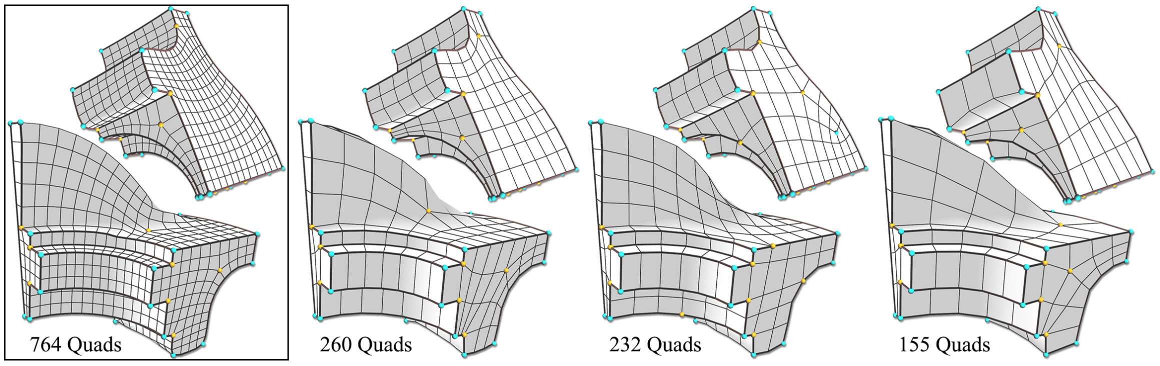


Figure 5.15: Coarse Requadrangulations of the Fandisk Model.

Left: the input mesh from Bommes *et al.* (2009). Second left to right: coarser requadrangulations with exact sharp feature fidelity. Note that for the one with 232 quads, we allow irregular vertices to appear on the patch boundaries and achieve a more regular flow of mesh lines.

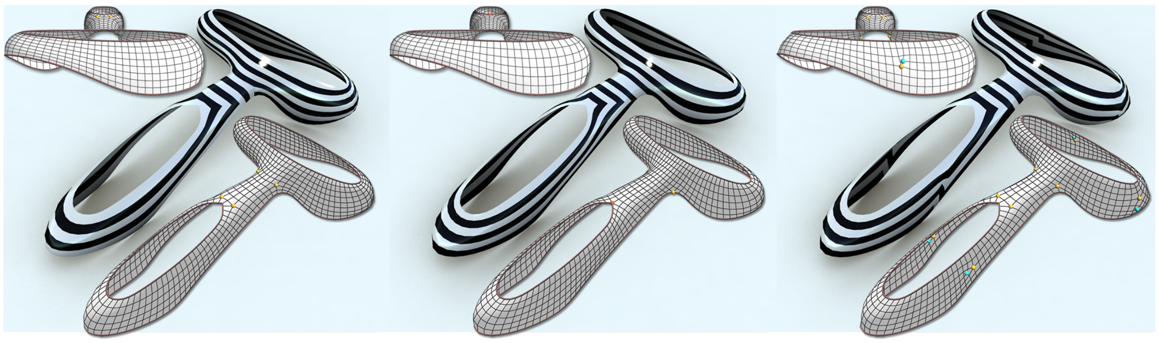


Figure 5.16: Requadrangulations of a Single Curved Patch with Two Inner Holes. Left: a standard requadrangulation with four $v5$. Middle: an alternative in which the interior meshing is completely regular by moving the $v5$ to the boundaries (two $v5$ are collocated as a $v6$). Right: another alternative that favors uniform quad size at the cost of additional irregular vertices. The arrangement of irregular vertices significantly influences the overall grid layout as observed from the zebra pattern rendering.

generate the control graph. A general and fast requadrangulation method may be achievable by combining mesh segmentation methods and our patch quadrangulation algorithm. Finally, we did not investigate the impact of topology in simulations, e.g., FEM and high-order surface fitting, but we hope that our work can also have impact on these areas.

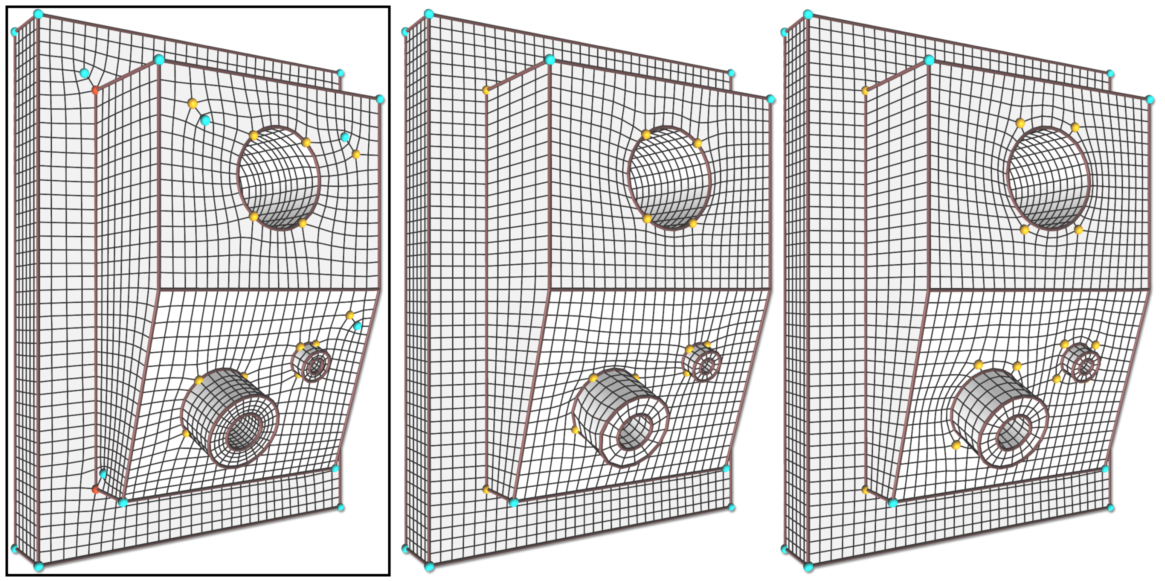


Figure 5.17: Requadrangulations of the Accessory Model.

Left: the input mesh from Zhang *et al.* (2010) with several redundant $v3$-$v5$ pairs and unnecessary $v6$. Middle: a requadrangulation with the same configuration of irregular vertices on patch boundaries. All patches are now quadrangulated without redundant irregular vertices. Right: an alternative with different allocations of the $v5$ on the top facades.

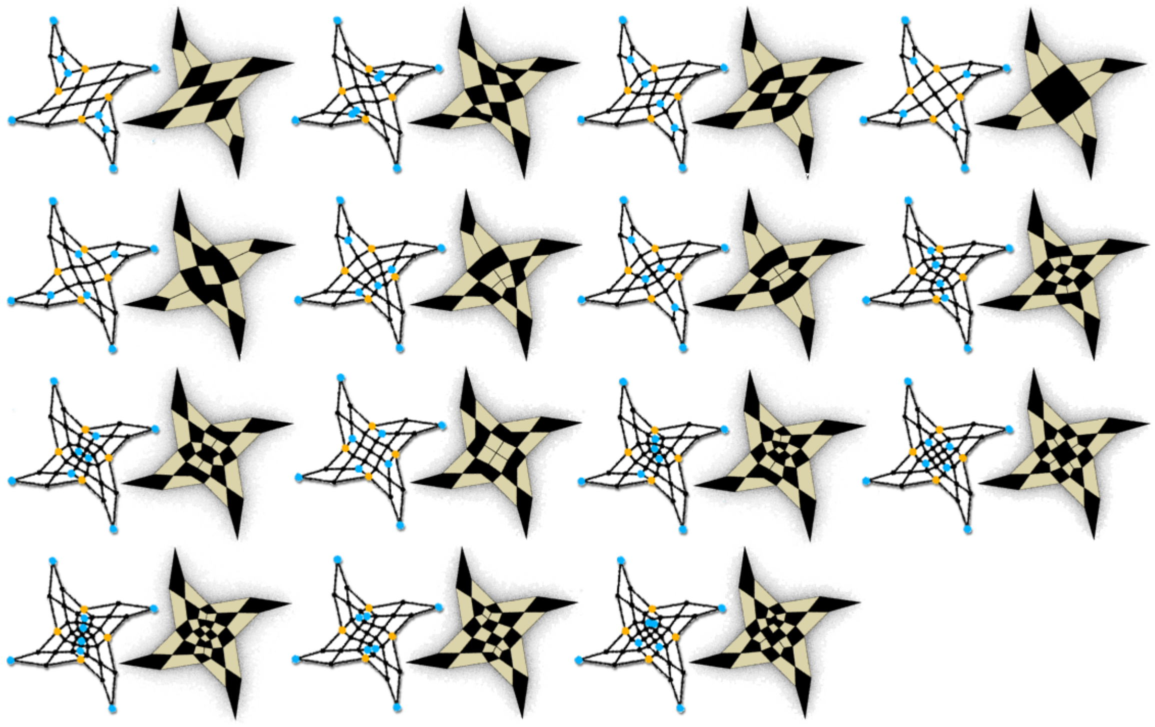


Figure 5.18: A Gallery of Different Quadrilateral Meshes for a Shuriken.
The quadrilaterals of the model were colored in a post-process. Topological variations have distinctive, interesting patterns of mesh lines.

Figure 5.19: Quadrangulations of Font Interiors.
We choose Georgia (Bold, Italy) as a challenging example for its curved, asymmetric outlines. Left: a quadrangulation with regular interiors. However, some parts such as the left of the letter d can never be uniformly quadrangulated without irregular vertices. Right: an alternative that favors uniform quad size at the cost of additional irregular vertices. Bottom left: 2-coloring of the quads is possible because the meshing is regular (quads are merged when necessary). Bottom right: a third color is necessary for quads adjacent to the $v3$ and $v5$ vertices.

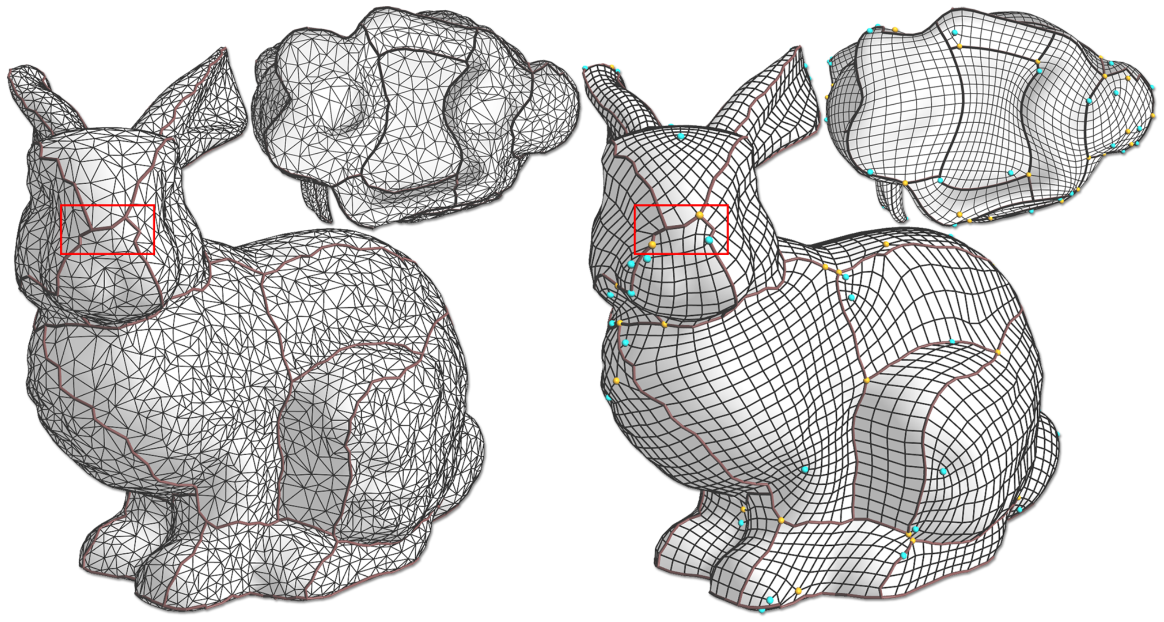


Figure 5.20: Quadrangulating an Organic Model.

Left: A triangular Bunny mesh and a manually generated control graph based on an initial segmentation by Variational Shape Approximation Cohen-Steiner *et al.* (2004). Right: a quadrangulation. The marked region shows that the quality of the control graph depends on the tessellation of the input mesh. Here, unnecessary corners are caused by jagged-edge strips.

Chapter 6

CONCLUSION AND FUTURE WORK

We conclude this thesis by outlining goals we have achieved, and outlooks for future work.

First, we propose editing operations for quad meshes to explicitly control the location, orientation, type, and number of irregular vertices. We analyzed what edits are possible and what edits are impossible in a quad mesh with sharp feature edges. Based on our theoretical analysis, we can conclude that the movement of a pair of irregular vertices is the best possible compound editing operation. These operations and their effects on the mesh are also discussed. We believe that these editing operations are essential in applications such as quadrangular mesh optimization and pattern design. We underline our argument by showing how our editing operations can improve the output of important state-of-the-art remeshing algorithms. We hope that our research can make a contribution to the communities of 3D artists, designers, and architects. Here, quad meshes are often created manually, even in a quad-by-quad fashion, instead of automatically generated by remeshing techniques.

Second, we present a connectivity editing framework for QD meshes that realizes the fundamental editing operations to control the location, type, and number of irregular elements. While the framework is useful for the manual control of QD mesh connectivity, for QD meshes with an excessive number of irregular elements, it may become impractical. We thus expect that a global framework that automates the local edits, possibly driven by topological and geometric heuristics (e.g. curvature), can be useful. Besides the given examples, we envision other uses for QD mesh editing. In certain situations, regular vertices are very important and it is beneficial to convert irregular vertices to irregular faces. For example, it may be preferable to have regular vertices at the expenses of irregular faces, like in conical meshes Liu *et al.* (2006) where the definition of a conical vertex is applicable only to regular vertices (the four adjacent faces need to be tangent to a common sphere). However, irregular vertices can help to combine multiple patches of conical meshes. We have spoken with multiple applied mathematicians about possible advantages of quad-dominant meshes for solving PDEs. We would like to pursue related research questions in future work.

Third, we present a framework to explore quad mesh topologies. The core of our work is a systematic enumeration algorithm that can generate all possible quadrangular meshes inside a defined boundary with an upper limit of $v3$-$v5$ pairs. The algorithm is orders of magnitude more efficient than previous work. The combination of topological enumeration and shape-space exploration demonstrates that mesh topology has a powerful influence on geometry. The results illustrate that mesh topology has an impact on the quality of the requadrangulation, especially when the mesh lines are visible. We focus on examples from architecture, art, and design, in which control graphs can be trivially generated. While our framework is applicable to organic models, such as the bunny, manual efforts are required to generate the control graph. A general and fast requadrangulation method may be achievable by combining mesh segmentation methods and our patch quadrangulation algorithm. Finally, we did not investigate the impact of topology in simulations, e.g., FEM and higher-order surface fitting, but we hope that our work can also have impact on these areas.

REFERENCES

"Cgal, Computational Geometry Algorithms Library", Http://www.cgal.org (2014).

Akleman, E. and J. Chen, "Practical polygonal mesh modeling with discrete gaussian-bonnet theorem", Geometric Modeling and Processing pp. 287–298, Springer-Verlag (2006).

Alliez, P., D. Cohen-Steiner, O. Devillers, B. Lévy and M. Desbrun, "Anisotropic polygonal remeshing", ACM Transactions on Graphics **22**, 3, 485–493, ACM (2003).

Alliez, P., M. Meyer and M. Desbrun, "Interactive geometry remeshing", ACM Transactions on Graphics **21**, 3, 347–354, ACM (2002).

Bauer, U., K. Polthier and M. Wardetzky, "Uniform convergence of discrete curvatures from nets of curvature lines", Discrete & Computational Geometry **43**, 4, 798–823, Springer-Verlag (2010).

Berkelaar, M., K. Eikland and P. Notebaert, *lpsolve : Open source (Mixed-Integer) Linear Programming system* (2008).

Bessmeltsev, M., C. Wang, A. Sheffer and K. Singh, "Design-driven quadrangulation of closed 3d curves", ACM Transactions on Graphics **31**, 6, 178:1–178:11, ACM (2012).

Blacker, T. D. and M. B. Stephenson, "Paving: A new approach to automated quadrilateral mesh generation", International Journal for Numerical Methods in Engineering **32**, 4, 811–847, John Wiley & Sons (1991).

Bommes, D., M. Campen, H.-C. Ebke, P. Alliez and L. Kobbelt, "Integer-grid maps for reliable quad meshing", ACM Transactions on Graphics **32**, 4, 98:1–98:12, ACM (2013).

Bommes, D., T. Lempfer and L. Kobbelt, "Global structure optimization of quadrilateral meshes", Computer Graphics Forum **30**, 2, 375–384, Blackwell Publishing Ltd (2011).

Bommes, D., B. Lévy, N. Pietroni, E. Puppo, C. Silva, M. Tarini and D. Zorin, *State of the Art in Quad Meshing*, Blackwell Publishing Ltd (2012).

Bommes, D., H. Zimmer and L. Kobbelt, "Mixed-integer quadrangulation", ACM Transactions on Graphics **28**, 3, 77:1–77:10, ACM (2009).

Botsch, M., L. Kobbelt, M. Pauly, P. Alliez and B. Levý, *Polygon Mesh Processing*, A K Peters (2010).

Campen, M., D. Bommes and L. Kobbelt, "Dual loops meshing: quality quad layouts on manifolds", ACM Transactions on Graphics **31**, 4, 110:1–110:11, ACM (2012).

Cohen-Steiner, D., P. Alliez and M. Desbrun, "Variational shape approximation", ACM Transactions on Graphics **23**, 3, 905–914, ACM (2004).

Daniels, J., C. T. Silva, J. Shepherd and E. Cohen, "Quadrilateral mesh simplification", ACM Transactions on Graphics **27**, 5, 148:1–148:9, ACM (2008).

D'Azevedo, E. F., "Are bilinear quadrilaterals better than linear triangles?", SIAM Journal on Scientific Computing **22**, 1, 198–217, Society for Industrial and Applied Mathematics (2000).

Desbrun, M., M. Meyer, P. Schröder and A. H. Barr, "Implicit fairing of irregular meshes using diffusion and curvature flow", Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH) pp. 317–324, ACM (1999).

Dong, S., P.-T. Bremer, M. Garland, V. Pascucci and J. C. Hart, "Spectral surface quadrangulation", ACM Transactions on Graphics **25**, 3, 1057–1066, ACM (2006).

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

Doo, D. and M. Sabin, "Behaviour of recursive division surfaces near extraordinary points", Computer-Aided Design **10**, 6, 356–360 (1978).

Eigensatz, M., M. Kilian, A. Schiftner, N. J. Mitra, H. Pottmann and M. Pauly, "Paneling architectural freeform surfaces", ACM Transactions on Graphics **29**, 4, 45:1–45:10, ACM (2010).

Frey, P. J. and L. Marchal, "Fast adaptive quadtree mesh generation", Proceedings of the Seventh International Meshing Roundtable pp. 211–224, Springer-Verlag (1998).

Glymph, J., D. Shelden, C. Ceccato, J. Mussel and H. Schober, "A parametric strategy for free-form glass structures using quadrilateral planar facets", Automation in Construction **13**, 2, 187–202, Elsevier (2004).

Harary, F., *Graph theory*, Addison-Wesley series in mathematics, Addison-Wesley (1969).

Hoppe, H., "Progressive meshes", Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH) pp. 99–108, ACM (1996).

Hull, D. and D. Bacon, *Introduction to dislocations*, Butterworth-Heinemann (2001).

Jiang, C., J. Wang, J. Wallner and H. Pottmann, "Freeform honeycomb structures", Computer Graphics Forum **33**, 5, 185–194, Blackwell Publishing Ltd (2014).

Kälberer, F., M. Nieser and K. Polthier, "Quadcover-surface parameterization using branched coverings", Computer Graphics Forum **26**, 3, 375–384, Blackwell Publishing Ltd (2007).

Kobbelt, L., “Interpolatory subdivision on open quadrilateral nets with arbitrary topology”, Computer Graphics Forum **15**, 3, 409–420, Blackwell Publishing Ltd (1996).

Kobbelt, L., “$\sqrt{3}$-subdivision”, Proceedings of the 27th annual conference on Computer graphics and interactive techniques (SIGGRAPH) pp. 103–112, ACM (2000).

Lai, Y.-K., L. Kobbelt and S.-M. Hu, “An incremental approach to feature aligned quad dominant remeshing”, Proceedings of the 2008 ACM symposium on Solid and Physical Modeling pp. 137–145, ACM (2008).

Lévy, B., S. Petitjean, N. Ray and J. Maillot, “Least squares conformal maps for automatic texture atlas generation”, ACM Transactions on Graphics **21**, 3, 362–371, ACM (2002).

Li, Y., E. Zhang, Y. Kobayashi and P. Wonka, “Editing operations for irregular vertices in triangle meshes”, ACM Transactions on Graphics **29**, 6, 153:1–153:12, ACM (2010).

Liu, Y., H. Pottmann, J. Wallner, Y.-L. Yang and W. Wang, “Geometric modeling with conical meshes and developable surfaces”, ACM Transactions on Graphics **25**, 3, 681–689, ACM (2006).

Liu, Y., W. Xu, J. Wang, L. Zhu, B. Guo, F. Chen and G. Wang, “General planar quadrilateral mesh design using conjugate direction field”, ACM Transactions on Graphics **30**, 6, 140:1–140:10, ACM (2011).

Marchal, L., “Advances in octree-based all-hexahedral mesh generation: Handling sharp features”, Proceedings of the 18th International Meshing Roundtable pp. 65–84, Springer-Verlag (2009).

Marinov, M. and L. Kobbelt, “Direct anisotropic quad-dominant remeshing”, 12th Pacific Conference on Computer Graphics and Applications (PG) pp. 207–216, IEEE Computer Society (2004).

Marinov, M. and L. Kobbelt, “A robust two-step procedure for quad-dominant remeshing”, Computer Graphics Forum **25**, 3, 537–546, Blackwell Publishing Ltd (2006).

Maza, S., F. Noel and J. Leon, “Generation of quadrilateral meshes on free-form surfaces”, Computers and Structures **71**, 5, 505–524, Elsevier (1999).

Myles, A., T. Ni and J. Peters, “Fast parallel construction of smooth surfaces from meshes with tri/quad/pent facets”, in “Proceedings of the sixth Eurographics symposium on Geometry processing”, SGP ’08, pp. 1365–1372, Eurographics Association (2008).

Myles, A., N. Pietroni, D. Kovacs and D. Zorin, “Feature-aligned t-meshes”, ACM Transactions on Graphics **29**, 4, 117:1–117:11, ACM (2010).

Nasri, S. M., A. and Z. Yasseen, “Filling n-sided regions by quad meshes for subdivision surfaces”, Computer Graphics Forum **28**, 1644–1658, Blackwell Publishing Ltd (2009).

Nieser, M., J. Palacios, K. Polthier and E. Zhang, “Hexagonal global parameterization of arbitrary surfaces”, in “ACM SIGGRAPH ASIA 2010 Sketches”, pp. 5:1–5:2, ACM (2010a).

Nieser, M., C. Schulz and K. Polthier, “Patch layout from feature graphs”, Computer Aided Design **42**, 3, 213–220, Butterworth-Heinemann (2010b).

Palacios, J. and E. Zhang, “Rotational symmetry field design on surfaces”, ACM Transactions on Graphics **26**, 3, ACM (2007).

Panozzo, D. and E. Puppo, “Adaptive lod editing of quad meshes”, in “Proceedings of the 7th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa”, AFRIGRAPH ’10, pp. 7–16, ACM (2010).

Panozzo, D., E. Puppo, M. Tarini and O. Sorkine-Hornung, “Frame fields: Anisotropic and non-orthogonal cross fields”, ACM Transactions on Graphics **33**, 4, 134:1–134:11, ACM (2014).

Park, C., J.-S. Noh, I.-S. Jang and J. M. Kang, “A new automated scheme of quadrilateral mesh generation for randomly distributed line constraints”, Computer Aided Design **39**, 4, 258–267, Elsevier (2007).

Peng, C.-H., M. Barton, C. Jiang and P. Wonka, “Exploring quadrangulations”, ACM Transactions on Graphics **33**, 1, 12:1–12:13, ACM (2014a).

Peng, C.-H. and P. Wonka, “Connectivity editing for quad-dominant meshes”, Computer Graphics Forum **32**, 5, 43–52, Blackwell Publishing Ltd (2013).

Peng, C.-H., Y.-L. Yang and P. Wonka, “Computing layouts with deformable templates”, ACM Transactions on Graphics **33**, 4, 99:1–99:11, ACM (2014b).

Peng, C.-H., E. Zhang, Y. Kobayashi and P. Wonka, “Connectivity editing for quadrilateral meshes”, ACM Transactions on Graphics **30**, 6, 141:1–141:12, ACM (2011).

Pottmann, H., A. Asperl, M. Hofer, A. Kilian and D. Bentley, *Architectural Geometry*, Bentley Institute Press (2007a).

Pottmann, H., Y. Liu, J. Wallner, A. Bobenko and W. Wang, “Geometry of multi-layer freeform structures for architecture”, ACM Transactions on Graphics **26**, 3, ACM (2007b).

Pottmann, H., A. Schiftner, P. Bo, H. Schmiedhofer, W. Wang, N. Baldassini and J. Wallner, “Freeform surfaces from single curved panels”, ACM Transactions on Graphics **27**, 3, 76:1–76:10, ACM (2008).

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

Ray, N., B. Vallet, L. Alonso and B. Levy, “Geometry-aware direction field processing”, ACM Transactions on Graphics **29**, 1, 1:1–1:11, ACM (2009).

Ray, N., B. Vallet, W. C. Li and B. Lévy, “N-symmetry direction field design”, ACM Transactions on Graphics **27**, 2, 10:1–10:13, ACM (2008).

Schaefer, S., J. Warren and D. Zorin, “Lofting curve networks using subdivision surfaces”, in “Proceedings of the second Eurographics symposium on Geometry processing”, SGP ’04, pp. 103–114, ACM (2004).

Schiftner, A. and J. Balzer, “Statics-sensitive layout of planar quadrilateral meshes”, Advances in Architectural Geometry 2010 pp. 221–236, Springer Vienna (2010).

Sederberg, T. W., D. L. Cardon, G. T. Finnigan, N. S. North, J. Zheng and T. Lyche, “T-spline simplification and local refinement”, ACM Transactions on Graphics **23**, 3, 276–283, ACM (2004).

Sederberg, T. W., J. Zheng, A. Bakenov and A. Nasri, “T-splines and t-nurccs”, ACM Transactions on Graphics **22**, 3, 477–484, ACM (2003).

Shepherd, J. F. and C. R. Johnson, “Hexahedral mesh generation constraints”, Engineering with Computers **24**, 3, 195–213, Springer-Verlag (2008).

Southern, G., “The shape of things: understanding topology”, 3D World (2011).

Stam, J. and C. Loop, “Quad/triangle subdivision”, Computer Graphics Forum **22**, 1, 79–85, Blackwell Publishing, Inc (2003).

Takayama, K., D. Panozzo, A. Sorkine-Hornung and O. Sorkine-Hornung, “Sketch-based generation and editing of quad meshes”, ACM Transactions on Graphics **32**, 4, 97:1–97:8, ACM (2013).

Tang, C., X. Sun, A. Gomes, J. Wallner and H. Pottmann, “Form-finding with polyhedral meshes made simple”, ACM Transactions on Graphics **33**, 4, 70:1–70:9, ACM (2014).

Tarini, M., N. Pietroni, P. Cignoni, D. Panozzo and E. Puppo, “Practical quad mesh simplification”, in “Computer Graphics Forum (Special Issue of Eurographics 2010 Conference)”, vol. 29, pp. 407–418, Blackwell Publishing Ltd (2010).

Tarini, M., E. Puppo, D. Panozzo, N. Pietroni and P. Cignoni, “Simple quad domains for field aligned mesh parametrization”, ACM Transactions on Graphics **30**, 6, 142:1–142:12, ACM (2011).

Taubin, G., “Detecting and reconstructing subdivision connectivity”, The Visual Computer **18**, 5–6, Springer-Verlag (2002).

Tchon, K.-F. and R. Camarero, “Quad-dominant mesh adaptation using specialized simplicial optimization”, Proceedings of the 15th International Meshing Roundtable pp. 21–38, Springer-Verlag (2006).

Tong, Y., P. Alliez, D. Cohen-Steiner and M. Desbrun, "Designing quadrangulations with discrete harmonic forms", in "Proceedings of the fourth Eurographics symposium on Geometry processing", SGP '06, pp. 201–210, Eurographics Association (2006).

White, D. and P. Kinney, "Redesign of the paving algorithm: Robustness enhancements through element by element meshing", in "Proceedings of the sixth International Meshing Roundtable", pp. 323–335, Springer-Verlag (1997).

Yang, Y.-L., Y.-J. Yang, H. Pottmann and N. J. Mitra, "Shape space exploration of constrained meshes", ACM Transactions on Graphics **30**, 6, 124:1–124:12, ACM (2011).

Zadravec, M., A. Schiftner and J. Wallner, "Designing quad-dominant meshes with planar faces", Computer Graphics Forum **29**, 5, 1671–1679, Blackwell Publishing Ltd (2010).

Zhang, E., J. Hays and G. Turk, "Interactive tensor field design and visualization on surfaces", IEEE Transactions on Visualization and Computer Graphics **13**, 1, 94–107, IEEE Computer Society (2007).

Zhang, E., K. Mischaikow and G. Turk, "Vector field design on surfaces", ACM Transactions on Graphics **25**, 4, 1294–1326, ACM (2006).

Zhang, M., J. Huang, X. Liu and H. Bao, "A wave-based anisotropic quadrangulation method", ACM Transactions on Graphics **29**, 4, 118:1–118:8, ACM (2010).

BIOGRAPHICAL SKETCH

Chi-Han Peng received PhD in Computer Science at Arizona State University. His dissertation, Connectivity Control for Quad-Dominant Meshes, was supervised by Prof. Peter Wonka. His research is about polygonal mesh processing, geometric modelling, and architecture and urban environment design, with multiple publications (as first author) in Siggraph/TOG and EG Symposium on Geometry Processing (SGP). During his PhD studies, he also took summer intern at Adobe ATL (supervised by Dr. Nathan Carr and Radomir Mech) and visited KAUST, Saudi Arabia, TU Vienna, Austria, and UCL, UK, for research collaborations.

Prior to his PhD study, he received B.S. and M.S. in Computer Science at National Chiao Tung University (NCTU), Taiwan. He worked for CyberLink, the maker of PowerDVD video players, as a software engineer for four years. He was also the co-founder of an online taxi carpooling service serving the Taipei metropolitan area.