Exploring Quadrangulations

CHI-HAN PENG
Arizona State University
MICHAEL BARTON and CAIGUI JIANG
King Abdullah University of Science and Technology
and
PETER WONKA
Arizona State University and King Abdullah University of Science and Technology

We present a framework for exploring topologically unique quadrangulations of an input shape. First, the input shape is segmented into surface patches. Second, different topologies are enumerated and explored in each patch. This is realized by an efficient subdivision-based quadrangulation algorithm that can exhaustively enumerate all mesh topologies within a patch. To help users navigate the potentially huge collection of variations, we propose tools to preview and arrange the results. Furthermore, the requirement that all patches need to be jointly quadrangulatable is formulated as a linear integer program. Finally, we apply the framework to shape-space exploration, remeshing, and design to underline the importance of topology exploration.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling

General Terms: Algorithms

Additional Key Words and Phrases: quadrilateral meshes, mesh optimization, topology, geometry processing, mesh connectivity

ACM Reference Format:

Peng, C-H., Barton, M., Jiang, C., and Wonka, P. 20xx. Exploring Quadrangulations. ACM Trans. Graph. dd, m, Article xxx (Month, YYYY), xx pages.
DOI = 10.1145/XXXXXXX.YYYYYYY
http://doi.acm.org/10.1145/XXXXXXX.YYYYYYY

This research was supported by the National Science Foundation and KAUST.

Authors' addresses: C-H. Peng, CIDSE, Arizona State University; email: pchihan@asu.edu; M. Barton, C. Jiang, and P. Wonka, GMSV, King Abdullah University of Science and Technology; email: michael.barton@kaust.edu.sa, caigui.jiang@kaust.edu.sa, pwonka@gmail.com.

1. INTRODUCTION

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.

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.

2. RELATED WORK

In the following, we briefly review recent quadrangulation methods in general and algorithms that explore quad mesh topologies in

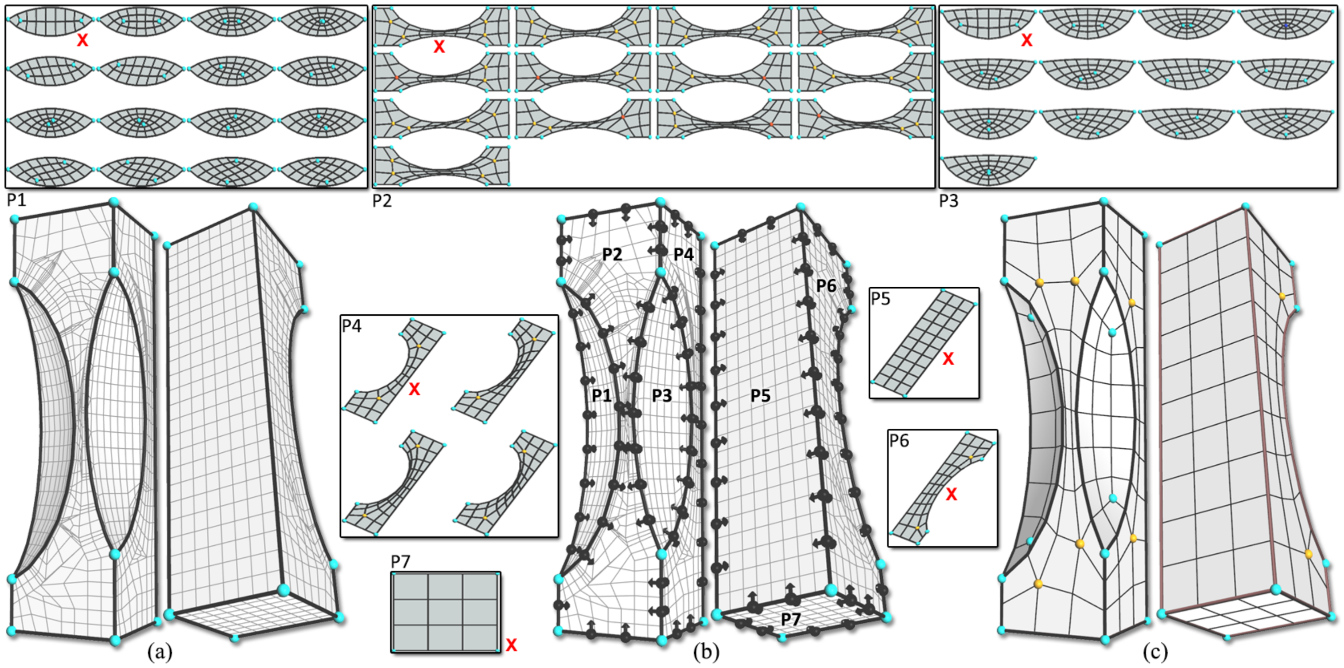


Fig. 1: Overview of our quadrangulation framework. (a) A control graph is generated by segmenting the underlying input mesh into a collection of surface patches. (b) The numbers of vertices on each edge can be computed by integer programming such that all patches are quadrangulatable by the minimum number 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.

more detail. For a broader background on polygon mesh processing and quad meshing, we suggest the book by Botsch et al. [Botsch et al. 2010] and the survey by Bommes et al. [Bommes et al. 2012].

Most existing algorithms for quad mesh generation are geared towards computing a single optimized quad mesh. Many popular algorithms generate a quad mesh from a field by streamline tracing, e.g., [Alliez et al. 2003; Marinov and Kobbelt 2004; Dong et al. 2005]. While fields can be edited by specifying locations of singularities or by controlling parameters of an optimization function, e.g., [Zhang et al. 2006; Tong et al. 2006; Palacios and Zhang 2007; Ray et al. 2008; Ray et al. 2009], there is no direct relationship between a field and a resulting quad mesh, because the algorithms used to derive a quad mesh from a field are quite involved. One such tool is global parametrization [Ray et al. 2006; Kälberer et al. 2007; Bommes et al. 2009; Zhang et al. 2010], which is used in most recent methods. Several quad meshing algorithms use a two-stage approach similar to our framework. First, a rough patch layout is generated and then the patches are quadrangulated [Dong et al. 2006]. Our algorithm is complementary in the sense that we assume that the input model is already subdivided in patches (or we compute a simple patch segmentation semi-automatically).

Besides the aforementioned algorithms, there are other specialized patch quadrangulation algorithms, e.g., those that attempt to find a topology with the fewest irregular vertices possible [Nasri and Yasseen 2009; Schaefer et al. 2004]. The three most related concepts for the exploration of quad mesh topologies are curve sampling [Marinov and Kobbelt 2006], connectivity editing [Maza et al. 1999; Peng et al. 2011], and advancing fronts (paving) [Blacker and Stephenson 1991; White and Kinney 1997; Park et al. 2007]. Marinov and Kobbelt [2006] connect the boundary vertices of a patch by curves and propose an algorithm to generate a layout and another algorithm to mutate an existing layout. The purpose of the algorithm is quad mesh generation, but it could also be used to explore quad mesh topologies. Compared with our algorithm, this approach generates a much larger set of invalid and duplicate meshes. Advancing-front algorithms incrementally grow quad elements from the patch boundaries and they could be used for enumerating topologies. However, one problem is that the algorithm does not terminate without a geometric heuristic and can grow fronts indefinitely.

3. OVERVIEW

3.1 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 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 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 sub-paths, which we denote as* sides. *A patch is* convex *if it does not contain any concave corners; otherwise, it is* concave.

DEFINITION 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 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 5.1.5.

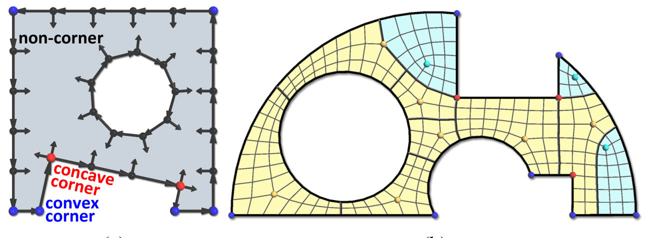
3.2 Framework Overview

An overview of our framework is shown in Figure 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 4). 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). 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 5.4). In Section 6, we show that the task of finding boundary constraints that make all patches jointly quadrangulatable can be formulated as a linear integer programming problem.

4. 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



(a) (b)

Fig. 2: (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.

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 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. A comparison between different parameterization methods is described in the additional materials. 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. 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 2b). The enumeration problem is thus greatly reduced to enumerating subdivisions into simple patches.

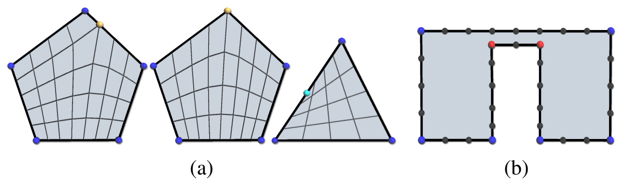


Fig. 4: (a) Left and middle: Two examples of boundary-case pentagons. Right: A boundary-case triangle. (b) A patch with a boundary that is incompatible with its potential embedding in a 4-by-8 parallelogram.

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 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 5.1. The algorithm relies on the efficient enumeration of subdivisions, which is described in Section 5.2. Strategies to filter redundantly generated subdivisions are described in Section 5.3. For larger examples, the enumeration might be time consuming. We therefore propose a fast sampling strategy to generate interesting topology variations early in Section 5.4. Finally, we propose tools to preview and arrange the topological variations in Section 5.5.

5.1 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.

5.1.1 *Simple Convex Patches.*

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

DEFINITION 5. *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 3).*

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.

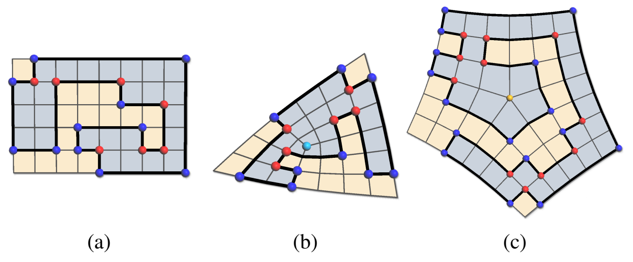


Fig. 5: Simple concave patches embedded inside a (a) parallelogram, (b) triangle, and (c) pentagon. P is shown in gray and the embedding into $\hat{P}$ is shown in yellow.

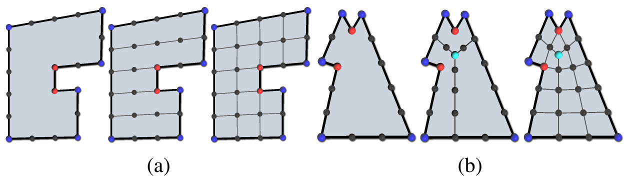


Fig. 6: Closed form solutions for quadrangulating simple concave patches embeddable inside a (a) parallelogram and (b) triangle.

Quadrangulation of Simple Convex Patches: A parallelogram is recursively subdivided into smaller parallelograms along the longer pair of opposing sides (Figure 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 3b, 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 4). In practice, we pre-calculate the quadrangulations of simple convex patches of various side lengths to accelerate the algorithm.

5.1.2 *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 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 4b.

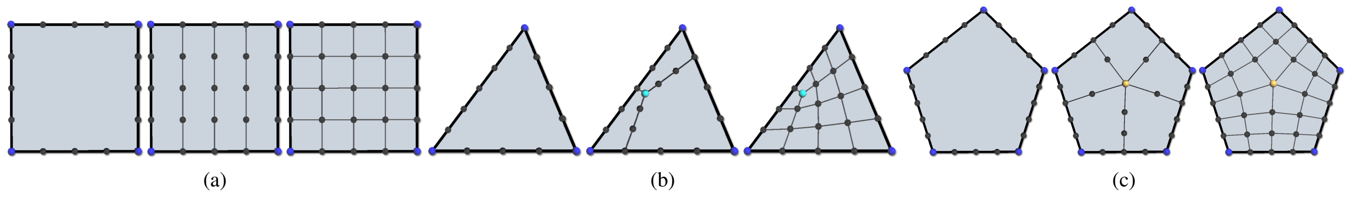


Fig. 3: Subdivision steps to quadrangulate a (a) parallelogram, (b) triangle, and (c) pentagon. Inner $v3$s are shown in cyan and inner $v5$s are shown in orange. For a triangle and a pentagon, the topological position of the irregular vertex is uniquely obtained by solving a linear system. For visualization 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.

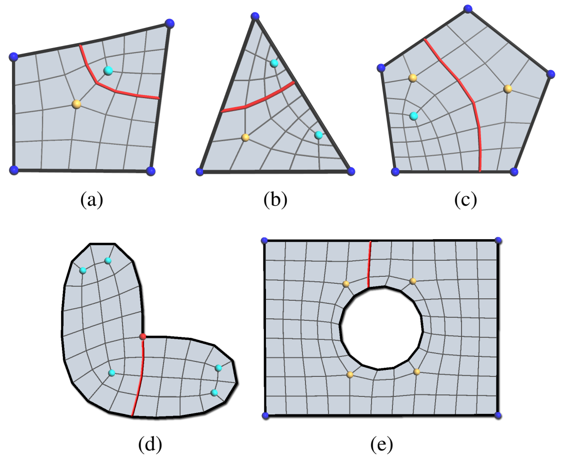


Fig. 7: (a) A non-simple patch with $TVD = 0$ is subdivided into a triangle and a pentagon. (b) A non-simple patch with $TVD = 1$ is subdivided into a triangle and a general 4-gon. (c) A non-simple patch with $TVD = -1$ is subdivided into a pentagon and a general 4-gon. (d) A patch with $TVD = 5$ is 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.

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 6).

5.1.3 *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 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 quadrangulated in closed form, and a general 5-gon, which is subsequently quadrangulated (Figure 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 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 7c).

For the first and third kinds of patch, we simply subdivide it heuristically by the scoring function described in Section 5.4.

5.1.4 *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 7d where a patch with $TVD = 5$ is subdivided into two sub-patches of $TVD = 3$ and 2.

5.1.5 *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 5.4. An example is shown in Figure 7e.

5.2 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 6.

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.

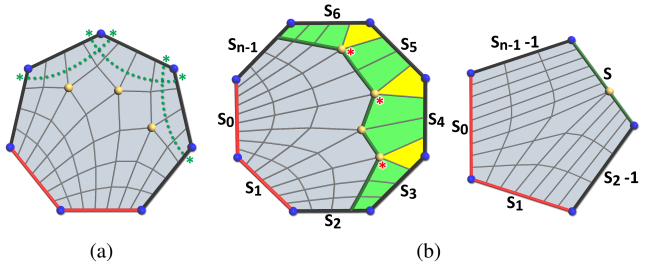


Fig. 8: (a) A quadrangulation of a 7-sided convex patch with the inequality in Theorem 6 being exactly satisfied, i.e., the longest consecutive pair of sides (red) cannot be any longer if the lengths of other sides are fixed. The $2|TVD|$ edges on the other sides and their emanating polychords that can never reach the longest pair are marked in green. (b) Left: quadrangulation of an 8-sided convex patch with the inequality in Theorem 6 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 6. *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 $\iff$ the sum of the lengths of the longest consecutive pair of sides $\leq$ the sum of the lengths of all other sides minus $2|TVD|$.*

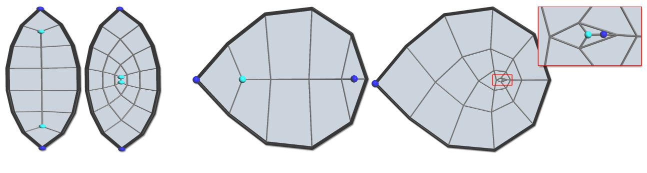


Fig. 9: The quadrangulations with the minimal and maximal numbers of quads for a 2-sided patch with side lengths $(7, 7)$ (left) and a 1-sided patch with side length 10 (right). Note that there are two $v3$ collocated as a $v2$ in the quadrangulations of the 1-sided patch.

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 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 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 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 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 7. *When quadrangulated with two $v3$s 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 8. *When quadrangulated with three $v3$s 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 7 and 8 are based on analyzing the decomposition of the patches into triangles and are given in the additional materials. A 2-sided and a 1-sided patch quadrangulated with the maximal and minimal numbers of quads are shown in Figure 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

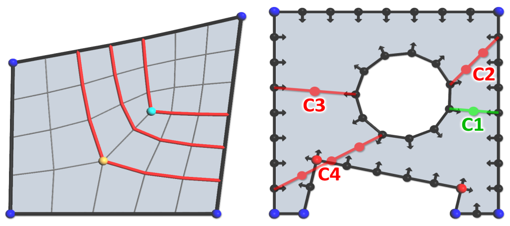


Fig. 10: (a) Three parallel connections are shown in red. All of them subdivide the patch into a triangle and a pentagon and result in equivalent topologies. Note that the top one leads to a boundary-case triangle and the bottom one leads to a boundary-case pentagon. (b) Comparing the ranking of connections. 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.

$v5$). 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 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 9. *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 5.4 and discard the rest. Examples of parallel connections are shown in Figure 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 7.

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 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 11b for an example.

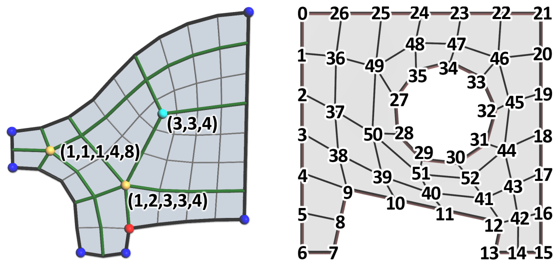


Fig. 11: (a) A quadrangulation of which the topology profile is $\{(3, 3, 4), (1, 1, 1, 4, 8), (1, 2, 3, 3, 4)\}$. (b) Ordering of vertices in a patch for the graph isomorphism test.

5.4 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 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.

5.5 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 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.

6. INTEGER PROGRAMMING

Assigning numbers of vertices lying on each boundary edge of a control graph path, i.e., their (topological) length, can be a nontrivial 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 5.2.

—For a triangle ($TVD = 1$), parallelogram ($TVD = 0$), and pentagon ($TVD = -1$), side lengths are constrained by Definition 5.

—According to Theorem 6, 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 5.1.5 and 5.4 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.

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

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. 2004].

7. 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, 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 paper, 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.

patch	TVD	$v3$-$v5$	boun. length	unique topo.	cluster	total topo.	time (total)	time (first)
Figure 7a	0	1	22	12	12	36	0.17	0.01
Figure 7e,5%	-4	0	56	536	525	568	198.62	0.06
Figure 8b	-3	0	34	7	5	77	0.28	0.00
Figure 14,1	-2	0	66	4	4	10	0.43	0.08
Figure 14,2	2	0	36	57	35	120	4.34	0.05
Figure 14,3	-4	0	66	559	558	3064	79.89	0.06
Figure 13	1	1	30	30	30	232	1.59	0.03
Figure 18	4	0	16	254	52	5166	23.09	0.01

Table I. : Timing statistics 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.

Statistics: Table I 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 the paper, 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 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 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 Figure13. 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.

Exploring Alternative Requadrangulations: We show that our enumeration framework enables users to explore multiple distinctive requadrangulations of an input mesh. In Figure 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 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 17, we requadrangulate the Accessory model in [Zhang et al. 2010] to remove all the redundant $v3$-$v5$ pairs. In Figure 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.

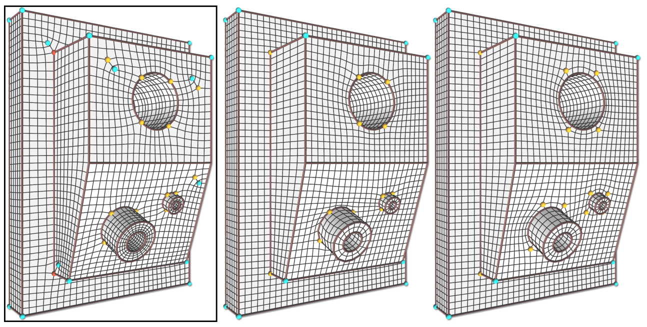


Fig. 17: Requadrangulations of the Accessory model. Left: the input mesh from [Zhang et al. 2010] with several redundant $v3$-$v5$ pairs and unnecessary $v6$. Middle: a requadrangulation with the same configuration of irregular vertices on patch boundaries. All patches are now quadrangulated without redundant irregular vertices. Right: an alternative with different allocations of the $v5$ on the top facades.

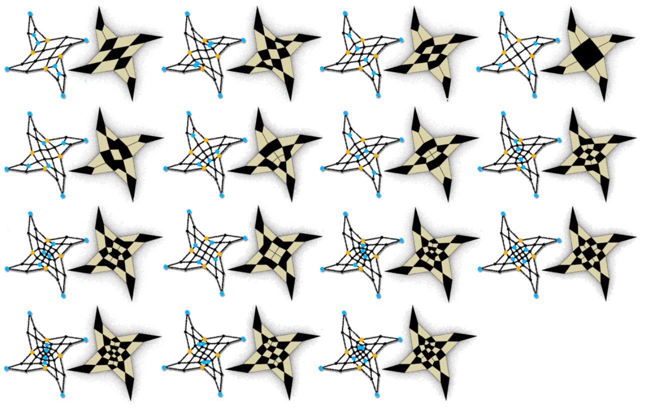


Fig. 18: A gallery of different quadrilateral meshes for a Shuriken. The quadrilaterals of the model were colored in a post-process. Topological variations have distinctive, interesting patterns of mesh lines.

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 18). In Figure 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

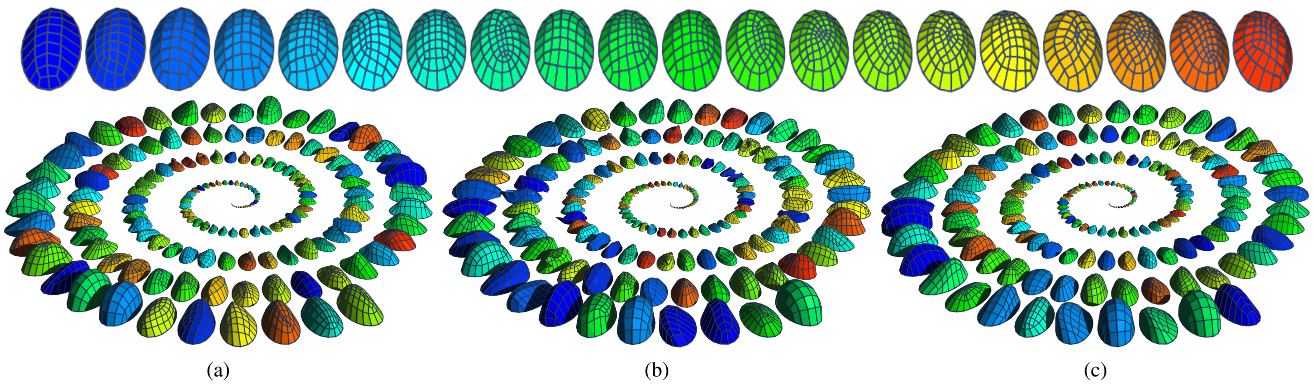


Fig. 12: Shape-space exploration on different topologies. The cap of an elliptic paraboloid is PQ meshed using twenty different topological patterns (top row) and planarity-preserving shape-space exploration is applied on each topology such that the boundary is fixed while the interior vertices are allowed 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.

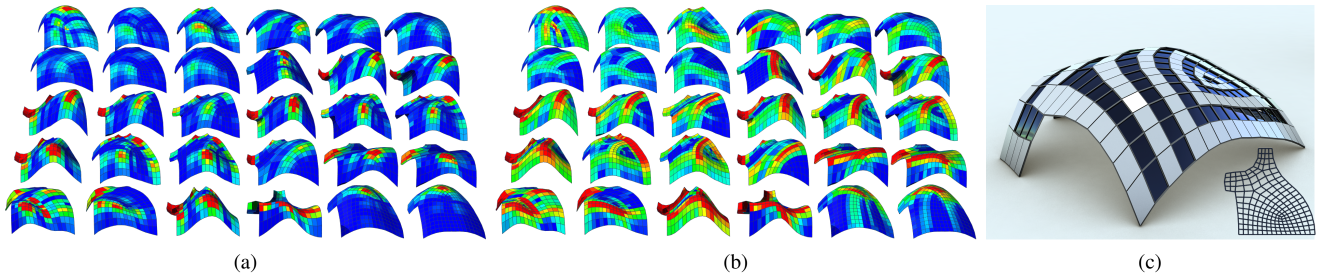


Fig. 13: Shape-space exploration of PQ meshes. For a 2D PQ mesh in the plane, six corner points are kept fixed and the remaining vertices move freely to form 3D shapes. From thirty topological patterns, the best "bumping" shape was selected. The color coding reflects the discrete Gaussian curvature (a) 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.

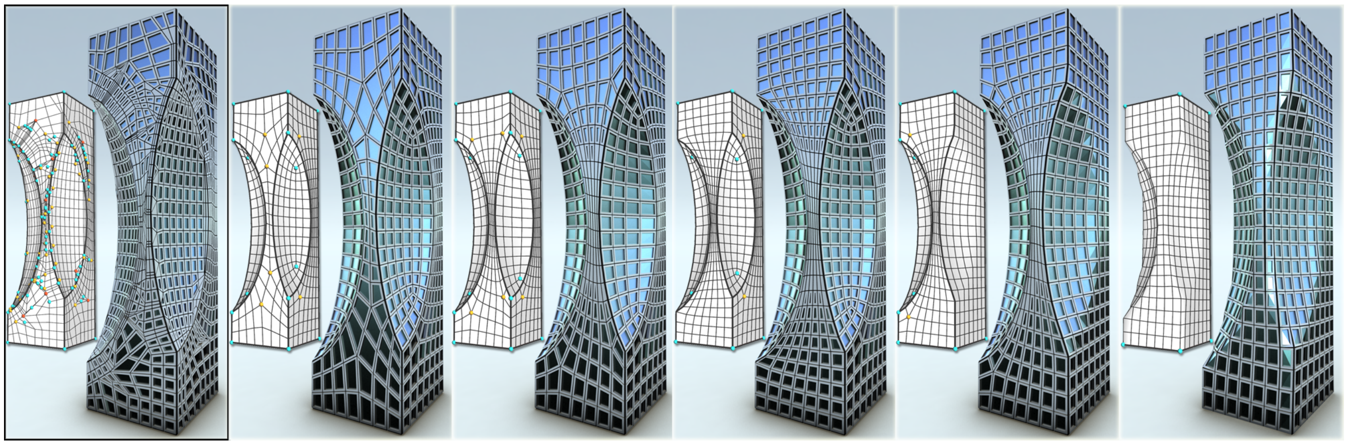


Fig. 14: Exploring alternative requadrangulations of an architectural tower model. Left: the input tower model with a large number of irregular vertices and small faces due to the computation of intersections by a professional modeling package. Second left to right: five requadrangulations explored with 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.

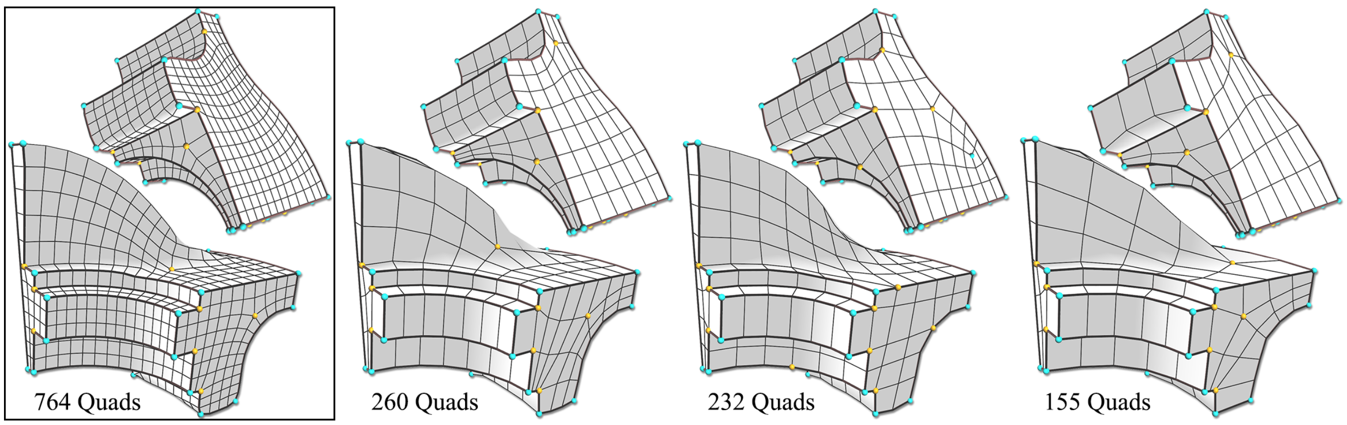


Fig. 15: Coarse requadrangulations of the Fandisk model. Left: the input mesh from [Bommes et al. 2009]. Second left to right: coarser requadrangulations with exact sharp feature fidelity. Note that for the one with 232 quads, we allow irregular vertices to appear on the patch boundaries and achieve a more regular flow of mesh lines.

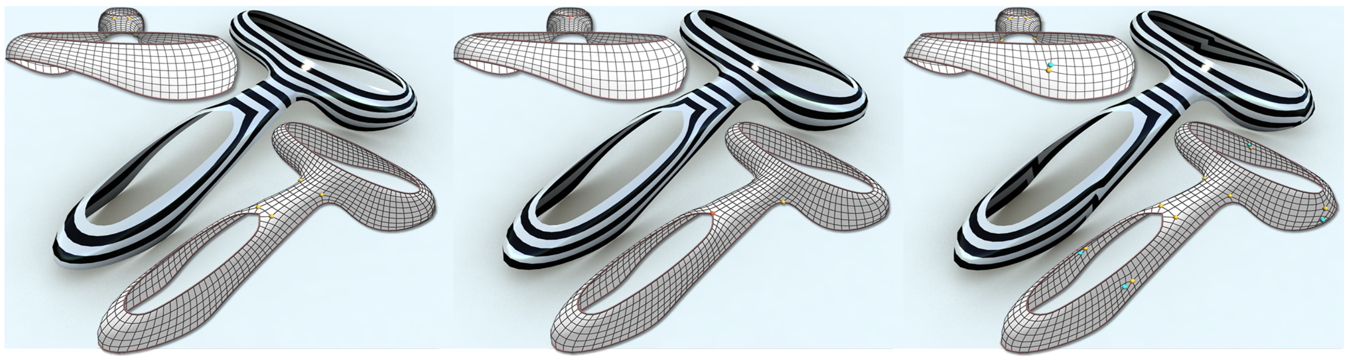


Fig. 16: Requadrangulations of a single curved patch with two inner holes. Left: a standard requadrangulation with four $v5$. Middle: an alternative in which the interior meshing is completely regular by moving the $v5$ to the boundaries (two $v5$ are collocated as a $v6$). Right: another alternative 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.

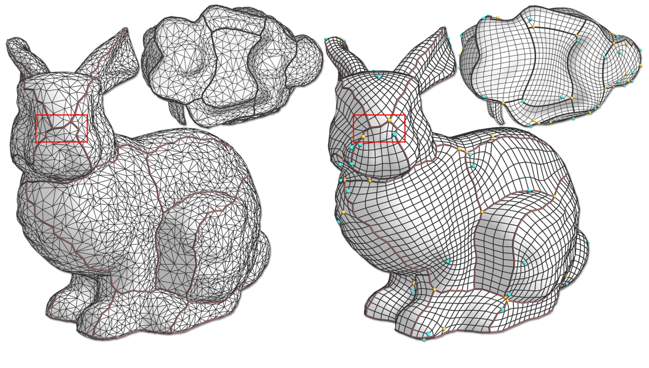


Fig. 20: Quadrangulating an organic model. Left: A triangular Bunny mesh and a manually generated control graph based on an initial segmentation by Variational Shape Approximation [Cohen-Steiner et al. 2004]. Right: a quadrangulation. The marked region shows that the quality of the control graph depends on the tessellation of the input mesh. Here, unnecessary corners are caused by jagged-edge strips.

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 20), 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 high-order surface fitting, but we hope that our work can also have impact on these areas.

8. CONCLUSION

Here we presented 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.

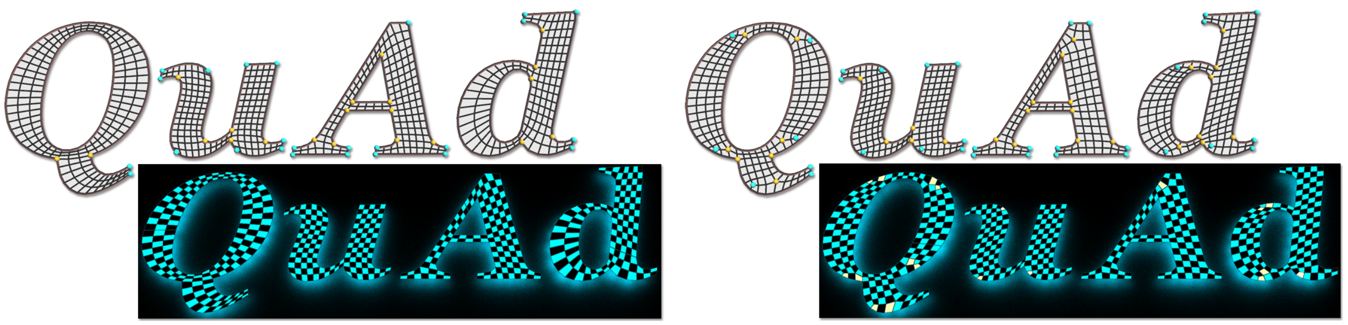


Fig. 19: Quadrangulations of font interiors. We choose Georgia (Bold, Italy) as a challenging example for its curved, asymmetric outlines. Left: a quadrangulation with regular interiors. However, some parts such as the left of the letter *d* can never be uniformly quadrangulated without irregular vertices. 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.

ACKNOWLEDGMENTS

We thank Helmut Pottmann and the anonymous reviewers for insightful comments, Alexander Schiftner and authors of [Bommes et al. 2009] and [Zhang et al. 2010] for providing datasets, Dongming Yan for an implementation of [Cohen-Steiner et al. 2004], Yoshihiro Kobayashi and Christopher Grasso for renderings, and Virginia Unkefer for the proofreading.

Appendix A: The Cave-Filling Algorithm

The goal of this cave-filling algorithm is to find the *extended patch* of an N-sided polygon ($N \geq 0$) P, $\hat{P}$, which is the minimal simple convex patch that encloses P. $\hat{P}$ is defined under the assumption that the exterior of P, $\bar{P}$, is regular. We assume that $\bar{P}$ is regular unless otherwise specified. For brevity, the proofs of the following theorems are provided in the additional materials.

DEFINITION 10. *A boundary edge of P is* projected *onto an edge in $\bar{P}$ if we can find a rectangular part of $\bar{P}$ with the two edges as its two opposite sides.*

Note that, due to the regularity of $\bar{P}$, a boundary edge of P either projects onto another boundary edge of P on the opposite side, or it projects onto an edge of $\hat{P}$.

THEOREM 11. *A patch P is free of consecutive concave corners $\Longleftrightarrow$ every boundary edge of P projects onto an edge of $\hat{P}$.*

According to Theorem 11, if a patch, P, is free of consecutive concave corners, then every boundary edge of P is projected onto an edge of $\hat{P}$, and boundary edges of P's parallel sides are projected onto parallel sides of $\hat{P}$ (see Figure 5). Since $\hat{P}$ is simple, it cannot have multiple sides that are parallel to each other. Thus, $\hat{P}$ must be a simple N-sided polygon (with N being the number of effective sides of P) with the same lengths as the lengths of P's effective sides.

Our goal is thus to transform the given patch, P, into another patch, P', that is free of consecutive concave corners and has the same extended patch as P. The process is realized by a *cave-filling algorithm* that removes all *caves* from P.

DEFINITION 12. *A path γ is* straight *if every interior vertex in γ is regular. Furthermore, every interior vertex's 1-ring neighborhood is evenly divided by γ, i.e., there are two quadrilaterals on both sides.*

DEFINITION 13. *A* cave *of P is a subset of quadrilaterals in $\bar{P}$ that can be completely separated from $\bar{P}$ by one straight path from a convex corner to another boundary vertex of P. We denote such a straight path as the* closure *of the cave.*

THEOREM 14. *There exists a cave $\Longleftrightarrow$ there exist consecutive concave corners for P.*

Note that identifying and removing arbitrary caves is a nontrivial task. Caves may contain smaller sub-caves, and there is no simple way to find the other end of the closure emanating from an arbitrary convex corner. Instead, we take an incremental approach.

DEFINITION 15. *A* simple cave *is a cave that is equivalent to a simple rectangular part of $\bar{P}$. Its four corners are identified as a convex corner of P, followed by two consecutive concave corners of P, followed by an arbitrary vertex of P.*

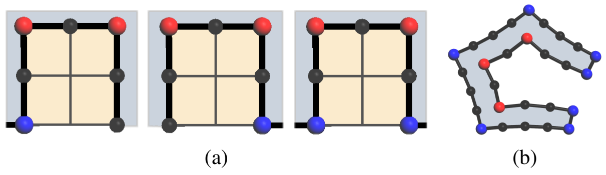


Fig. 21: (a) Examples of possible cases of a simple cave shown as yellow parts of $\bar{P}$. Convex corners of P are shown in blue, concave corners in red, and non-corners in black. (b) A patch with unremovable caves.

The three possible cases of a simple cave are shown in Figure 21a. In the first two cases, the closure emanating from the convex corner hits a non-corner on the opposing side. In the third case, the closure connects two convex corners. There are no other possible cases. Note that the closure of the simple cave can be of zero length.

THEOREM 16. *A patch P has a simple cave $\Longleftrightarrow$ P has a cave.*

Due to its simple rectangular shape, a simple cave can be easily identified, removed from $\bar{P}$, and added to P by a *simple cave-filling* operation. According to Theorem 16, all caves of a patch can be removed by iteratively exhausting simple cave-filling operations in arbitrary order. An example is shown in Figure 22. Finally, we note

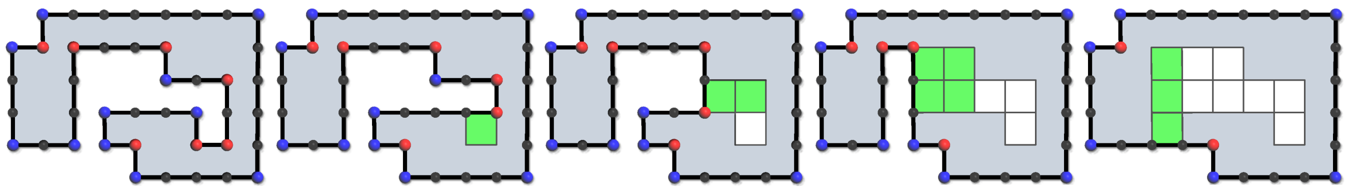


Fig. 22: Removing a complex cave of a patch by iteratively applying simple cave-filling operations. Green faces denote the latest simple cave being filled.

that $\hat{P}$ will not change during the simple cave-filling operations according to the following theorem.

THEOREM 17. *Applying simple cave-filling operations on P will not affect $\hat{P}$.*

In summary, the algorithm removes all caves of P by repeating the simple cave-filling operations in an exhaustive manner and, if successful, returns a cave-free patch, P'. P' is free of consecutive concave corners according to Theorem 14 and has the same extended patch as P according to Theorem 17. $\hat{P}'$ (the same as $\hat{P}$) is easily retrieved from P' since these two patches share the same boundary structure (the number of effective sides and their lengths). If P still has consecutive concave corners after all simple cave-filling operations are exhausted, then the regularity assumption of $\bar{P}$ is violated and $\hat{P}$ is undefined (Figure 21b). In such a case, our algorithm reports that P cannot be extended into a regular patch.

REFERENCES

ALLIEZ, P., COHEN-STEINER, D., DEVILLERS, O., LÉVY, B., AND DESBRUN, M. 2003. Anisotropic polygonal remeshing. *ACM Trans. Graph. 22,* 3, 485–493.

BERKELAAR, M., EIKLAND, K., AND NOTEBAERT, P. 2004. *lpsolve : Open source (Mixed-Integer) Linear Programming system*.

BESSMELTSEV, M., WANG, C., SHEFFER, A., AND SINGH, K. 2012. Design-driven quadrangulation of closed 3d curves. *ACM Trans. Graph. 31,* 5.

BLACKER, T. D. AND STEPHENSON, M. B. 1991. Paving: A new approach to automated quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering 32,* 4, 811–847.

BOMMES, D., LVY, B., PIETRONI, N., PUPPO, E., A, C. S., TARINI, M., AND ZORIN, D. 2012. State of the art in quad meshing. In *Eurographics STARS*.

BOMMES, D., ZIMMER, H., AND KOBBELT, L. 2009. Mixed-integer quadrangulation. *ACM Trans. Graph. 28,* 3, 77.

BOTSCH, M., KOBBELT, L., PAULY, M., ALLIEZ, P., AND LEVÝ, B. 2010. *Polygon Mesh Processing*. A K Peters, Natick, Massachusetts.

COHEN-STEINER, D., ALLIEZ, P., AND DESBRUN, M. 2004. Variational shape approximation. *ACM Transactions on Graphics. Special issue for SIGGRAPH conference*, 905–914.

DANIELS, J., SILVA, C. T., SHEPHERD, J., AND COHEN, E. 2008. Quadrilateral mesh simplification. *ACM Trans. Graph.*, 148:1–148:9.

DONG, S., BREMER, P.-T., GARLAND, M., PASCUCCI, V., AND HART, J. C. 2006. Spectral surface quadrangulation. *ACM Trans. Graph.*, 1057–1066.

DONG, S., KIRCHER, S., AND GARLAND, M. 2005. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Comput. Aided Geom. Des. 22*, 392–423.

KÄLBERER, F., NIESER, M., AND POLTHIER, K. 2007. Quadcover - surface parameterization using branched coverings. *Computer Graphics Forum 26,* 3, 375–384.

LÉVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. 2002. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph. 21,* 3 (July), 362–371.

LIU, Y., POTTMANN, H., WALLNER, J., YANG, Y.-L., AND WANG, W. 2006. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph. 25,* 3, 681–689.

MARINOV, M. AND KOBBELT, L. 2004. Direct anisotropic quad-dominant remeshing. *12th Pacific Conference on Computer Graphics and Applications (PG)*, 207–216.

MARINOV, M. AND KOBBELT, L. 2006. A robust two-step procedure for quad-dominant remeshing. *Computer Graphics Forum 25,* 3, 537–546.

MAZA, S., NOEL, F., AND LEON, J. 1999. Generation of quadrilateral meshes on free-form surfaces. *Computers and Structures 71*.

NASRI, A., S. M. AND YASSEEN, Z. 2009. Filling n-sided regions by quad meshes for subdivision surfaces. *Computer Graphics Forum 28*, 1644–1658.

PALACIOS, J. AND ZHANG, E. 2007. Rotational symmetry field design on surfaces. *ACM Trans. Graph. 26,* 3, 55.

PARK, C., NOH, J.-S., JANG, I.-S., AND KANG, J. M. 2007. A new automated scheme of quadrilateral mesh generation for randomly distributed line constraints. *Comput. Aided Des. 39,* 4 (Apr.), 258–267.

PENG, C.-H., ZHANG, E., KOBAYASHI, Y., AND WONKA, P. 2011. Connectivity editing for quadrilateral meshes. *ACM Trans. Graph. 30,* 6, 141.

RAY, N., LI, W. C., LÉVY, B., SHEFFER, A., AND ALLIEZ, P. 2006. Periodic global parameterization. *ACM Trans. Graph. 25,* 4, 1460–1485.

RAY, N., VALLET, B., ALONSO, L., AND LEVY, B. 2009. Geometry aware direction field design. *ACM Trans. Graph. 29,* 1, 1:1–1:11.

RAY, N., VALLET, B., LI, W. C., AND LÉVY, B. 2008. N-symmetry direction field design. *ACM Trans. Graph. 27,* 2, 10:1–10:13.

SCHAEFER, S., WARREN, J., AND ZORIN, D. 2004. Lofting curve networks using subdivision surfaces. In *Proceedings of the second Eurographics symposium on Geometry processing*. SGP '04. 103–114.

TONG, Y., ALLIEZ, P., COHEN-STEINER, D., AND DESBRUN, M. 2006. Designing quadrangulations with discrete harmonic forms. In *Proceedings of the fourth Eurographics symposium on Geometry processing*. SGP '06. 201–210.

WHITE, D. AND KINNEY, P. 1997. Redesign of the paving algorithm: Robustness enhancements through element by element meshing. In *Proc. 6 th Int. Meshing Roundtable*. 323–335.

YANG, Y.-L., YANG, Y.-J., POTTMANN, H., AND MITRA, N. J. 2011. Shape space exploration of constrained meshes. *ACM Trans. Graph. 30,* 6, 124:1–124:12.

ZHANG, E., MISCHAIKOW, K., AND TURK, G. 2006. Vector field design on surfaces. *ACM Trans. Graph. 25*, 1294–1326.

ZHANG, M., HUANG, J., LIU, X., AND BAO, H. 2010. A wave-based anisotropic quadrangulation method. *ACM Trans. Graph.*, 118:1–118:8.