Computers & Graphics 109 (2022) 1–14

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Structure simplification of planar quadrilateral meshes☆

Muhammad Naeem Akram *, Kaoji Xu, Guoning Chen

Department of Computer Science, College of Natural Sciences and Mathematics, University of Houston, Houston, TX, USA

ARTICLE INFO

Article history:
Received 19 July 2022
Received in revised form 1 October 2022
Accepted 6 October 2022
Available online 12 October 2022

Keywords:
Geometry computation
Quad mesh
Structure simplification
Separatrix-based

ABSTRACT

In this paper, we present a structure simplification framework for planar all-quad meshes with open boundaries. Our simplification framework can handle quad meshes with complex structures (e.g., quad meshes obtained via Catmull–Clark subdivision of the triangle meshes) to produce simpler meshes while preserving the boundary features. To achieve that, we introduce a set of separatrix-based semi-global operations and combine them with existing local operations to develop a new simplification framework. Additionally, we organize and order the individual simplification operations into groups and employ ranking strategies for each group to sort these operations to produce quad meshes with better quality and simpler structure. We provide a comprehensive evaluation of our framework using different input parameters on a number of representative planar quad meshes with various boundary configurations. To demonstrate the advantages of our method, we compare it with a few existing frameworks. Our comparison shows that our simplification framework usually produces simpler structure with faster computation than the state-of-the-art methods.

1. Introduction

Quadrilateral (or quad) meshes are preferred over triangle meshes in many engineering applications, such as modeling and simulation of elastic materials and fluid behaviors, due to their desired numerical properties [1] that usually leads to more accurate and faster simulations than triangle meshes. In addition, quad meshes with a higher overall quality produce better simulation results.

A (pure) quad mesh is composed entirely of quadrilateral elements, and its overall quality depends on the connectivity and shapes of the individual quad elements. An ideal quad mesh consists of regular quadrilateral elements, i.e., the interior angles of the quad elements are all 90°. The regularity of quad elements can be expressed in terms of *valence* of vertices in the quad mesh. The valence of a vertex is defined as the number of quad elements adjacent to that vertex. A vertex is *regular* if it has a valence of 4 (or 2 on the boundary), otherwise, it is *irregular*. Irregular vertices are also called *singularities* (Fig. 1(c)). Given a valence-n singularity ($n \neq 4$), n *separatrices* can be traced out along the n mesh edges adjacent to the singularity, respectively. These separatrices, that either end at other singularities or mesh boundary, partition the quad mesh into individual quadrilateral (or quad) patches (e.g., the colored patches in Fig. 1(c)). This partitioning is referred to as the *base complex* of the quad mesh, which is the structure of the quad mesh that this work is focused on. A quad mesh has a simple structure if it has a low number of singularities and a low number of quad patches.

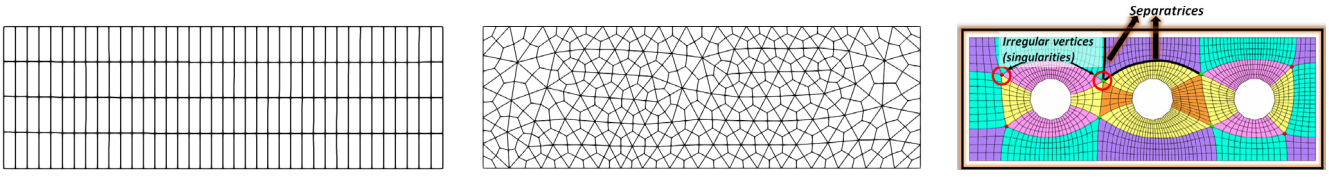
Fig. 1 shows examples of quad meshes for the same domain with a simpler structure (a) and a complex structure (b), respectively. Quad meshes with fewer and aligned singularities are regular or semi-regular, which are very useful for texturing and high-order modeling and simulations [1]. In the meantime, fewer singularities in turn improve the element quality as the ideal angles surrounding a vertex is determined by the valence of that vertex (i.e., 90° is achieved when its valence is 4). Therefore, the goal of quad mesh generation is to produce quad meshes with good structure (or fewer singularities) and high element quality (i.e., all elements have a close-to-regular shape). However, despite numerous efforts, automatic generation of high-quality quad meshes for complex and arbitrary domains remains a challenging task [1,2].

Existing quad mesh generation techniques take an arbitrary input and process geometric information from the input to generate a quad mesh. However, for many complex inputs, these techniques can exhibit certain limitations and fail to produce a quad mesh with good structure due to the lack of explicit control of the positions and valences of the singularities. An alternative to direct quad mesh generation methods is the structure simplification of a highly unstructured quad mesh. Two simplification strategies are typically employed, i.e., local simplification [3] and global simplification [4], to procedurally cancel or merge singularities in the quad mesh. However, local simplification cannot

☆ This article was recommended for publication by Prof L. Barthe.

* Correspondence to: Department of Computer Science, Philip Guthrie Hoffman Hall, 3551 Cullen Blvd, Room 501, Houston, TX 77204-3010, USA.
E-mail address: makram2@uh.edu (M.N. Akram).

https://doi.org/10.1016/j.cag.2022.10.001
0097-8493/



(a) Quad mesh with a simple structure (b) Quad mesh with a complex structure (c) Base complex of a quad mesh

Fig. 1. An example illustrating same mesh with different structures: (a) a simple and ideal structure, (b) a complex structure. (c) shows the base complex of a quad mesh. Different color patches illustrate the quad layout structure of the mesh.

guarantee to generate a globally optimal structure or a near optimal one, while global simplification may not be able to simplify the structure due to the constraint of boundary/feature preservation.

Recently, Xu et al. [5] presented the preliminary results of a new simplification approach for planar quad meshes. Their method uses *semi-global operations* to simplify the structure of the quad meshes with complex structure. The semi-global structure simplification aims to address the limitations of local and global simplification methods by performing simplification operations in the regions bounded by separatrices. Despite the promising preliminary results, there are a number of limitations of their pipeline. First, the simplification operations in their pipeline are not organized and sorted, leading to sub-optimal output and slow performance. Second, there is no local smoothing and refinement during simplification, thus, the resulting mesh could be too coarse for feature preservation and often leads to overlapping elements at boundaries that are hard to fix with existing optimization.

In this work, we address the limitations of the work in [5] and present a more robust and efficient simplification framework that integrates the semi-global operations with the local operations. In particular, we propose a strategy to organize or group different types of simplification operations. For each type of the semi-global simplification operations, a ranking strategy is developed to sort the individual operations. In addition, we include a local refinement and smoothing process in-between simplification operations to mitigate the reduction of vertices caused by simplification to better preserve boundary features. The boundary features of the mesh encompass sharp edges and corners. The new pipeline can produce better results i.e valid all-quad meshes, than those shown in [5] with much faster computation (e.g., up to **18x speedup** for complex models). It can also handle much more complex models than the pipeline in [5]. Furthermore, we provide an option for the user to produce inversion free quad meshes with some sacrifice in the simplicity of the output structure. We have applied our framework to a number of quad meshes with various boundary configurations and compare it with the local simplification [3] and Xu et al.'s method [5] to demonstrate its advantages. A parameter study is also reported to demonstrate the impact of different parameter values.

2. Related work

In the meshing literature, extensive research has been carried out for the generation of quad meshes with good structure and quality. There are two different strategies to generate quad meshes as described in the introduction, i.e., direct quad mesh generation and structure simplification of quad meshes. In this section we review some of these techniques.

2.1. Direct quad mesh generation

Direct quad mesh generation methods produce quad meshes from a complex and arbitrary input, such as a triangle mesh. The most straightforward method of generating a quad mesh from a triangle input is through Catmull–Clark subdivision scheme [6]. The subdivision scheme is robust and simple but introduces a large number of singularities which are difficult to optimize (Fig. 1(b)). Other methods, like the triangle-to-quad conversion methods
[3,7] and Voronoi diagram based approaches [8,9], usually generate unstructured quad meshes or quad-dominant meshes that are also difficult to optimize with existing methods. A group of direct mesh generation methods considers the geometric information and configurations of the input to generate quad meshes, such as quad layout construction approaches [10], the parameterization based approaches and the field aligned approaches. The quad layout generation initially produces coarse quadrilateral components on the polygonal mesh using either field line tracings [11,12] based on certain surface directional fields [13], base domain simplification [14], curve skeleton [15], dual loops [16,17], parameterization [18,19], polycube-maps [20], or T-meshes from motorcycles [21–23]. From these coarse quad patches, the refined quad meshes can be obtained via subdivision. In [24], a coarse quad layout of input mesh is first obtained by tracing separatrices of the cross fields and then optimized through the global simplification using chord collapsing. This hybrid approach directly simplifies the quad layout of the mesh and aims to reduce the T-junctions, however, for certain complex inputs, this approach may not be able to completely optimize the quad layout and eliminate all T-junctions. Parameterization-based methods [18, 25–28] construct a mapping from the curved surface to a flat 2D domain. Cutting is typically needed so that the surface patches are equivalent to a topological disk to enable a low distortion mapping. With this mapping, quad mesh can be obtained trivially in the 2D domain before mapping back to the surface. In [29], quadrangulations of 2D patches are obtained by subdividing the surface patches into simpler basic cases with the objective of keeping the number of irregular vertices as little as possible. That technique was applied to mostly simple models with simple cuts. In summary, the quad meshes generated using the quad layout computation and the parameterization based methods are usually semi-regular and need little structure optimization. However, the local element quality may be sub-optimal near the boundaries of the individual quad domains, and the quad layout can still be very complex and too fine if the boundary of the domain is complex.

A recent popular line of methods for quad mesh generation is the field-aligned methods [30,31], starting with the seminal mixed-integer quadrangulation technique [32]. This method explicitly controls element quality via certain guiding fields (e.g., frame fields or cross fields). Quadriflow [33], extends the instant field-aligned method to produce meshes with fewer singularities by enforcing certain constraints. For the frame-field based approaches, given any random input, the output quad mesh may not have an all-quad structure resulted from the generated field. Furthermore, the field-aligned method fails to produce optimal results and preserve sharp features for planar domains or open boundary meshes (Section 8).

2.2. Structure simplification of quad meshes

To address the limitations and difficulty of directly generating high quality quad meshes with simpler structure from the input triangle meshes, an alternative is to simplify the structure of a highly unstructured quad mesh to achieve the ideal configuration and simpler structure. For this purpose, various simplification techniques have been proposed, such as the local simplification approaches [3,34,35] and the global simplification methods [4,36,37]. These strategies have been demonstrated effective for initial valid quad meshes with reasonably good structures. Nonetheless, local simplification cannot guarantee to generate an optimal structure or a near optimal one, while global simplification may not produce a boundary-conformal structure or may not be able to simplify the structure due to the constraint of boundary/feature preservation (Fig. 9(b)).

Local quad mesh simplification [3,34,35,38] aims to reduce the amount of quad elements in a small local region via coarsening operations (e.g., edge rotation, diagonal collapsing, and doublet removal), cleaning operations, or other optimization operations. It produces quad meshes with fewer quad elements but may not reduce the amount of singularities significantly. In addition, these simplification operations can easily lead to the loss of surface features if not treated properly. This is probably part of the reasons that most examples shown in those works are smooth surfaces with few sharp features (e.g., edges or corners) except for the recent work by Docampo-Sanchez and Haimes [38] that reports results on simple CAD models. To explicitly reduce the singularities in the quad meshes, Peng et al. [39] proposed an editing framework to modify the connectivity of a quad mesh within a local region that usually contains a pair of singularities. Nonetheless, this is a manual process that cannot be applied to large quad meshes with many singularities. It also does not address the preservation of sharp features.

Global quad mesh simplification methods aim to reduce both structure and element complexity of the input quad meshes via global operations. Tarini et al. [37] introduced the singularity alignment technique to connect the mis-matched singularities, which cause complex structure, e.g., helical or even tangled configurations. Similar issue in the quad meshes has been reported with the field-guided quad mesh generation approaches and was partially addressed in [40]. However, these singularity alignment methods cannot be applied to a quad mesh split from a triangle mesh because the singularities there are already aligned. Daniels et al. [4] used the poly-chord collapsing operation to remove a sequence of quad elements at once, which not only removes singularities but also simplifies the mesh structure. However, it cannot handle certain complex poly-chords (e.g., poly-chords that are self-intersecting) and can easily introduce high-valence singularities during collapsing. In addition, it does not preserve surface features. To overcome this global issue, an adaptive simplification method [36] is proposed to perform poly-chord collapsing in a local region or surface with boundaries. However, for certain inputs, the complexity of poly-chords can still pose an additional challenge for simplification due to constraints of feature preservation.

Recently, Kaoji et al. [5] proposed a **semi-global simplification** strategy which uses semi-global simplification operations to resolve singularities in the local regions bounded by separatrices for planar quad meshes with boundaries. Their approach can address complex mesh structure configurations, like those often seen in the quad meshes obtained via Catmull–Clark subdivision from triangle meshes, and significantly reduce them to much simpler ones. However, the framework reported in [5] is slow and the involved simplification operations are not well-organized (or ordered), leading to sub-optimal outcomes. To address these limitations, in this work, we propose a strategy to organize different types of operations and order the individual operations of each type. This new strategy significantly reduces the simplification time while achieving comparable and (many times) better outcomes than the previous framework. In addition, we include additional smoothing and refinement operations to mitigate the overlapping elements in the output simplified meshes due to the loss of vertices after simplification. The new framework also provides an option for the user to achieve an inversion-free output that the previous works cannot.

3. Method overview

3.1. Features of an ideal planar quad mesh

As already described earlier, a high-quality planar quad mesh should have a simpler structure (i.e., with fewer singularities and fewer quad patches), while having as-regular-as-possible quad elements. To achieve the latter, we need to constrain the valence of singularities (i.e., either 3 or 5 for the interior singularities). In addition, a high-quality planar quad mesh should preserve the boundary features (e.g., corners) and the shape of the boundary. To achieve that, the following additional criteria are needed.

1. The ideal structure should place singularities at corners (at the boundaries) with valence either 1 or 3, depending on the spanning angle (or discrete curvature) around them.
2. It should ensure boundary conformality, i.e., boundary quads should align to the boundary, and each boundary vertex has an ideal valence based on the angle around it. The sector angle around a boundary vertex, v_b, is computed as $\theta = \sum_{i=1}^{n} \theta_i$, where n is the valence of v_b and θ_i represent angle between two adjacent edges of v_b. The ideal valence of v_b is calculated as $\text{val}_{ideal} = \lfloor \frac{\theta}{90} \rfloor + \lfloor \frac{\theta \% 90}{45} \rfloor$. The ideal number of edges connected to the boundary vertex should be its $\text{val}_{ideal} + 1$ (see Fig. 2).

3.2. Overview of our framework

The above criteria motivate the design of our simplification framework. Our framework takes a valid all-quad mesh as input and the output is a valid all-quad mesh with simplified structure (i.e. fewer singularities and simpler base complex) that preserves the open boundary (features identified according to the criteria in 3.1) of the planar input quad mesh. Here open boundary refers to the collection of edges that are only shared by one quad as opposed to closed boundary surface meshes where each boundary/feature edge is shared by two quads. As a very first step towards simplification, the simplification framework extracts the singularities and base complex of the mesh, then identifies the sharp boundary features (corners) that need to be preserved during simplification. Next, it enters an iterative process, and each iteration checks whether there are simplification operations that can be performed. The simplification operations that our framework provides include the semi-global simplification based on separatrices (Section 4) and some local operations (Section 5). If no simplification can be performed, our framework outputs the current mesh. To determine the operations to perform, a number of criteria and constraints are checked. These constraints can be classified into two groups (Section 7), i.e., the valence constraint (e.g., valence should be between 3 and 5) and the boundary constraint (e.g., boundary conformality and shape preservation). In general, the local operations that achieve boundary conformality and cleaning operations (to remove degeneracy) are performed before all semi-global operations. The semi-global operations are grouped and performed based on different types of separatrices

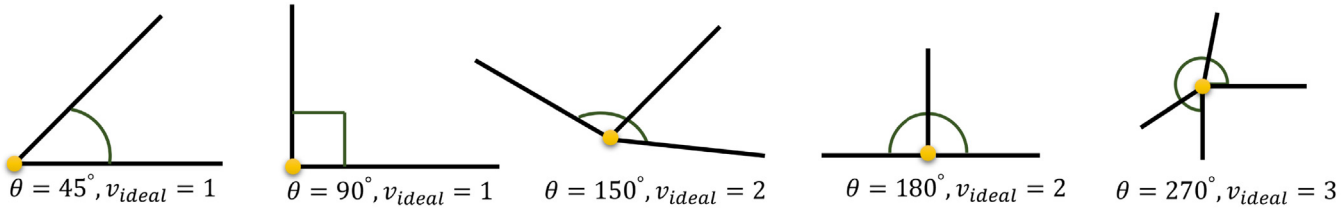
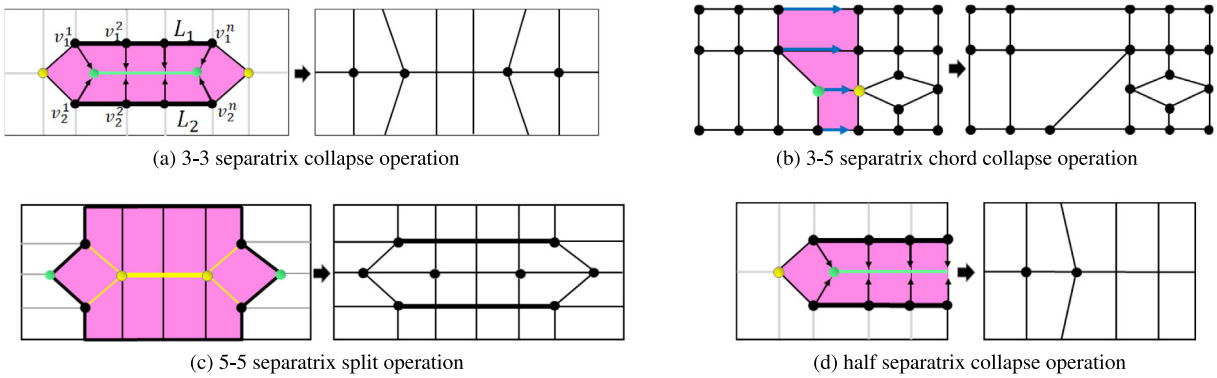


Fig. 2. Illustration of ideal valence for different boundary configurations.



(a) 3-3 separatrix collapse operation

(b) 3-5 separatrix chord collapse operation

(c) 5-5 separatrix split operation

(d) half separatrix collapse operation

Fig. 3. Semi-global operations based on different separatrix connections.

(Section 4). Our framework continues to identify and perform possible semi-global and local simplification operations until no more singularities can be removed without violating the valence and boundary constraints. The detailed pipeline of our framework will be provided in Section 6. In the following, we will first introduce the semi-global operations, followed by the utilized local operations.

4. Semi-global simplification

Our semi-global simplification strategy is based on the following knowledge. There is a topological index of each singularity of a quad mesh, v, that is defined as $\text{idx}_v = 1 - \frac{\text{val}_v}{4}$, where val_v is the valence of v. The index of a regular vertex is 0. Given a local region of the quad mesh, its total index is the sum of the individual indices of all the interior vertices. In particular, if the total index of a local region is zero, it can be re-meshed such that the new mesh within it is singularity-free according to the Poincaré index theorem [41]. One way to construct such a local region with trivial index is via separatrices as done in vector field simplification [42,43], since separatrices usually connect nearby singularities.

In a quad mesh with boundary, four types of separatrices can be constructed by connecting singularities of different valences. Generally, we have separatrices that connect two low valence singularities, separatrices that connect a low valence singularity to a high valence singularity, separatrices that connect two high valence singularities, and separatrices that end at a boundary. To simplify our discussion and utilize the Poincaré index theorem, we assume a quad mesh that consists of only valence-3 and valence-5 singularities in the interior. This assumption results in four basic types of separatrices: 3–3, 3–5, 5–5, and half-separatrices, and allows us to construct local regions surrounding them with trivial Poincaré index. We then design their respective simplification operations accordingly.

4.1. Separatrix collapsing for 3–3 connections

These separatrices connect a pair of valence 3 singularities directly. Fig. 3(a) (left) shows such an ideal configuration. In this ideal configuration, two valence 5 singularities (yellow dots) share the same quads as valence 3 singularities (green dots) respectively, which enables the construction of the semi-global region enclosing those 4 singularities whose total index is zero.

To remove these singularities, we perform a collapsing within the region surrounding the separatrix. There are three parallel lines (including the separatrix) as shown in Fig. 3(a) which have same number of vertices. We denote the separatrix as $L_s = \{v_s^i\}$, where $i \in \{1, 2, \ldots, n\}$ and n is the number of vertices associated with the separatrix. Assume $L_1 = \{v_1^i\}$ and $L_2 = \{v_2^i\}$ are the other two lines parallel to the separatrix. We collapse the vertices lying on L_1 and L_2 towards the corresponding vertices on L_s. In this way, the four singularities associated with the separatrix are canceled.

4.2. Chord collapsing for 3–5 connections

These separatrices connect a pair of valence 3 and valence 5 singularities directly. Since valence 3 and valence 5 singularities have opposite indices, a semi-global region enclosing such singularities can be constructed with trivial total index for simplification.

To handle this configuration, a chord collapsing is performed (Fig. 3(b)). The chord is formed by tracing parallel edges from the 3–5 separatrix edge until boundary is reached or a cycle is formed. The chord can be collapsed by merging the vertices of the parallel edges which cancels the two singularities associated with the chord as shown in Figure Fig. 3(b). The details on the chord collapsing can be found in [4]. We avoid the chord collapse operation if it does not satisfy the valence and boundary preservation constraints (Section 7) or if it self-overlaps.

4.3. Separatrix splitting for 5–5 connections

These separatrices connect pairs of valence-5 singularities. Fig. 3(c) (left) illustrates an ideal 5–5 separatrix configuration. Similar to the 3–3 connection described above, the two valence-5 singularities (yellow dots) are connected directly via the separatrix while two valence 3 singularities share the quads with them,

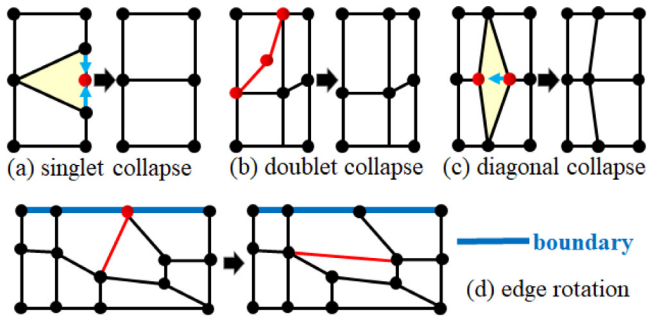


Fig. 4. Illustration of the utilized local operations.

enabling the construction of semi-global region with trivial total index.

To simplify the 5–5 separatrix, we perform splitting in the region surrounding a 5–5 connection. In Fig. 3(c), the region (purple) surrounding the separatrix (the yellow line) contains a pair of valence-5 singularities (yellow dots) and two valence-3 singularities (green dots). The splitting operation inserts two lines (dark lines in Fig. 3(c) (right)) parallel to the original separatrix, which removes the four singularities involved.

4.4. Collapse operations for half-separatrices

For a quad mesh with boundaries, separatrices may not always connect two singularities. As illustrated in Fig. 3(d) (left), a singularity connects with a regular vertex on the boundary thus creating a half-separatrix. Half-separatrices containing valence 3 singularity can be collapsed like the 3–3 collapsing operation defined above (Fig. 3(d)), while half-separatrix containing valence 5 singularity can be resolved using the split operation similar to the above 5–5 split.

4.5. Generalized separatrix operations

What we described so far are simplification operations in the ideal semi-global regions. However, in practice we may not always have such an ideal region, i.e., not having the two ending vertices with the desired valences as shown in Fig. 3(b) or Fig. 3(c). For example, in the quad meshes obtained by subdividing a triangle mesh, while 3–3 separatrices are prevalent, 3–5 and 5–5 are not due to the high valences at the original vertices of the triangles. In general, 3–n (n>5) connections are more common. To enable their simplification, we relax the constraint of valid valence range (i.e., all vertex valences fall in the range [3, val_{max}]) where val_{max} is provided as an input parameter to the simplification framework.

5. Local operations

In addition to the semi-global operations for separatrices, local operations are needed to address any degeneracy in the mesh that might arise during simplification and to achieve boundary conformality (Section 3.1). The simplification in such regions can be achieved using the available local simplification operations introduced in the previous works [3,34,35,38]. The local operations incorporated in our simplification pipeline are as follows.

Operations for removing degeneracy.

- *Doublet collapse.* During the simplification, doublets may be introduced in the mesh which are detected and removed immediately. Since a doublet is created due to a valence-2 vertex, we merge the two involved quads to remove the doublet as shown in Fig. 4b.
- For singlets on the flat portion of the boundary, we perform a *singlet collapse* operation by collapsing the involved two edges into one vertex, as shown in Fig. 4a.

Operations for achieving boundary conformality We perform *edge rotation* operation to achieve boundary conformality during simplification such that the boundary has as many regular vertices as possible. For each boundary vertex, based on its sector angle, we estimate its ideal valence, val_{ideal} (see Section 3). Consequently, the number of redundant edges that need rotation is calculated as $n = \text{val}_v - \text{val}_{ideal}$. Each edge rotation operation reduces the valence of boundary vertex by 1, therefore, we perform n edge rotations to achieve ideal valence (Fig. 4d).

Operations for mesh simplification. During simplification, certain local regions in the mesh might not be encompassed by any of the separatrices. Therefore, we need to perform other local operations, such as *diagonal collapsing*. An ideal diagonal collapse operation is illustrated in Fig. 4(c), where two valence-3 and two valence-5 singularities shared by a single quad are eliminated by collapsing the diagonal containing valence-3 singularities.

6. The complete simplification pipeline

Our simplification framework combines the operations described in Sections 4 and 5 to build the complete pipeline necessary for carrying out simplification. We illustrate our entire simplification pipeline in Fig. 5. The input to the pipeline is the initial irregular quad mesh obtained through Catmull–Clark subdivision of the original triangle mesh. Prior to simplification, sharp features and corners on the boundary of the mesh are detected for boundary preservation. The singularities and their separatrices are also extracted. Next, simplification is applied in an iterative manner to gradually simplify the structure of the mesh. Each simplification iteration is carried out by three sets of operations in the following order:

i **Degeneracy Handling Operations:**
These operations optimize the mesh connectivity by eliminating the degenerate elements, i.e. doublets and singlets using the local operations, such as doublet removal and singlet removal.

ii **Boundary Optimization Operations:**
We perform the edge rotation operation to optimize the boundary of the mesh. Element regularity at the boundary is achieved through the criteria describe in Section 5.

iii **Simplification Operations:**
In this step, we perform the simplification (local, semi-global and global) operations to reduce the singularities in the mesh. *We organize the simplification operations in the following order*: diagonal collapse, separatrix collapse, separatrix split, chord collapse and half-separatrix collapse. The ideal configurations of diagonal collapse and separatrix operations remove four singularities, whereas, the ideal configurations of chord collapse and half-separatrix operations remove two singularities. The greedy aspect of our framework in terms of singularity removal calls for prioritization of diagonal collapse and separatrix operations before chord collapse and half-separatrix operations. We order the diagonal collapse operation before the separatrix operations since its local nature allows for the singularity removal without modification of large areas of the mesh, thus allowing further simplification by the semi-global operations in the later stages. Figs. 6(a) and 6(b) show the simplification outputs obtained by performing diagonal collapse before and after separatrix operations respectively. In addition, we perform the half-separatrix

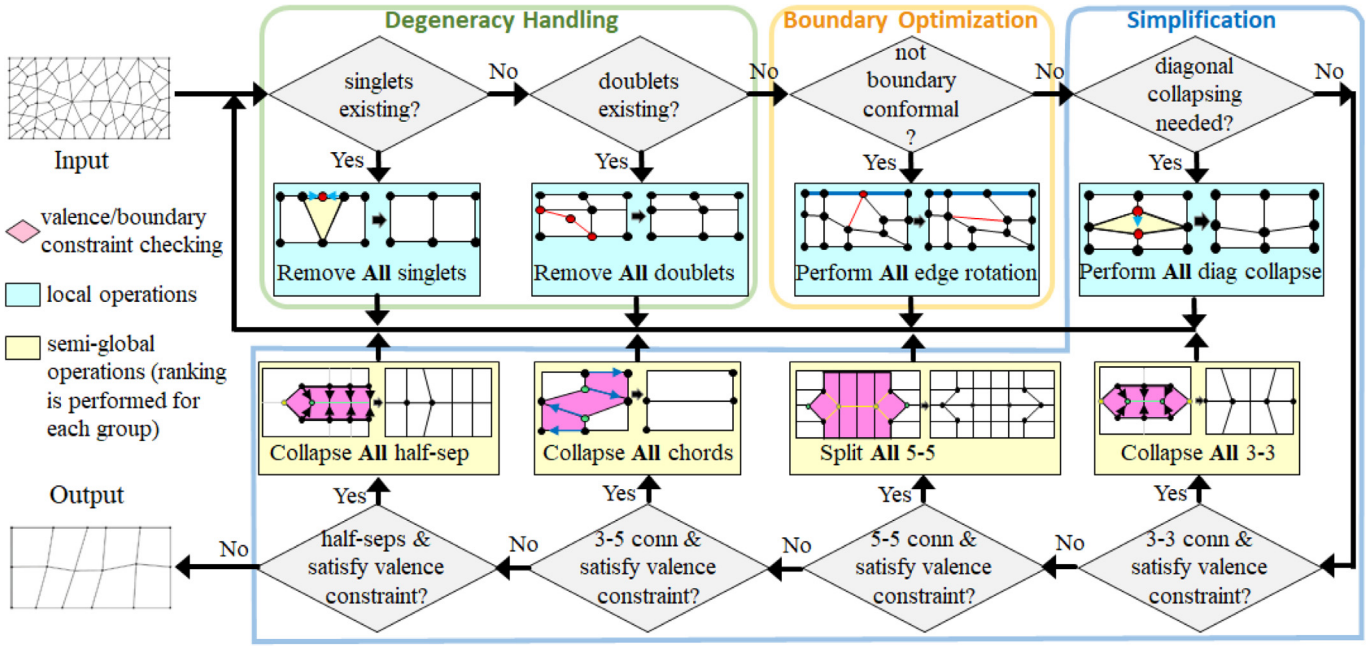


Fig. 5. Detailed pipeline of our simplification framework.

operation once all other operations have been exhausted for the sake of preserving the boundary as much as possible. In each simplification iteration, we identify and *group only one type of simplification operations*. For instance, the mesh is first checked for all candidate diagonal collapse operations. If there are no diagonal collapse operations found in the current mesh, it is checked for the prospective separatrix collapse operations and so on.

Once a group of simplification operations is identified, we employ a length-based ranking strategy to prioritize the operations. The ranking strategy is heuristic in nature and analogous to the ranking criteria for simplifying sheets in hex-mesh simplification [44]. It aims to prioritize operations that cause the smaller regions to be simplified first. The ranking of an operation is calculated as: $r_{op} = \sum length(e)$, where e belongs to the set of edges that make up the separatrix and chord links. In case of diagonal collapse, the diagonal length is used for the ranking strategy. We sort the candidate operations in a group in an ascending order according to their respective rankings. During sorting, we exclude an operation from the sorted list if its bounded region overlaps with the bounded region of another operation already present in the sorted list. In this way, all simplification operations in a group which are disjoint in terms of their bounded regions are performed in a single simplification iteration as opposed to [5], thus avoiding large geometric distortions in the mesh and reducing the time complexity of our framework.

Certain operations in the above process may distort the meshes by a great extent. If left untreated, the mesh may become tangled and get worse during further simplification. To address this, we perform local smoothing and refinement as described next. Note that both of these two processes are optional in our pipeline.

6.1. Local smoothing

The simplification of semi-global regions may introduce a coarse configuration in the interior and boundary which may introduce inverted elements. We identify the vertices involved in the simplification operations and add one ring neighborhood of those vertices to cover the region bounded by the group of operations executed in each simplification iteration. We then perform the local smoothing [45] at the end of each simplification iteration to fix any inverted elements. We offer local smoothing in our framework as an optional step and leave the decision to the user, since it may increase the computation time for large meshes.

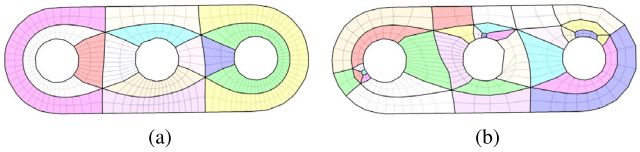


Fig. 6. An example illustrating the effect of ordering local operation (diagonal collapse) before (a) and after (b) semi-global operations respectively. (a) has 4 singularities while (b) has 15 singularities for the same input model.

6.2. Optional local refinement

As the simplification progresses, the semi-global regions bounded by the separatrices in the simplified mesh may cover a large area. Simplifying such regions can introduce distorted elements in the mesh especially after collapse operations. The removal of overlapping elements introduced during simplification is challenging since some overlapping elements can have a positive Jacobian measure which can be regarded as non-inverted elements by the optimization algorithms. One solution to mitigate this problem is to refine the mesh (e.g., splitting some chords of the mesh that become too coarse for boundary preservation) during simplification before the introduction of overlapping elements. However, it is challenging to determine when refinement should be executed, since refinement at an early stage during simplification can increase the complexity of the mesh, increasing the running time for simplification pipeline, whereas, performing refinement when the simplified mesh is too coarse may not always be helpful since presence of overlapped elements can worsen the quality of the mesh. Therefore, we refine

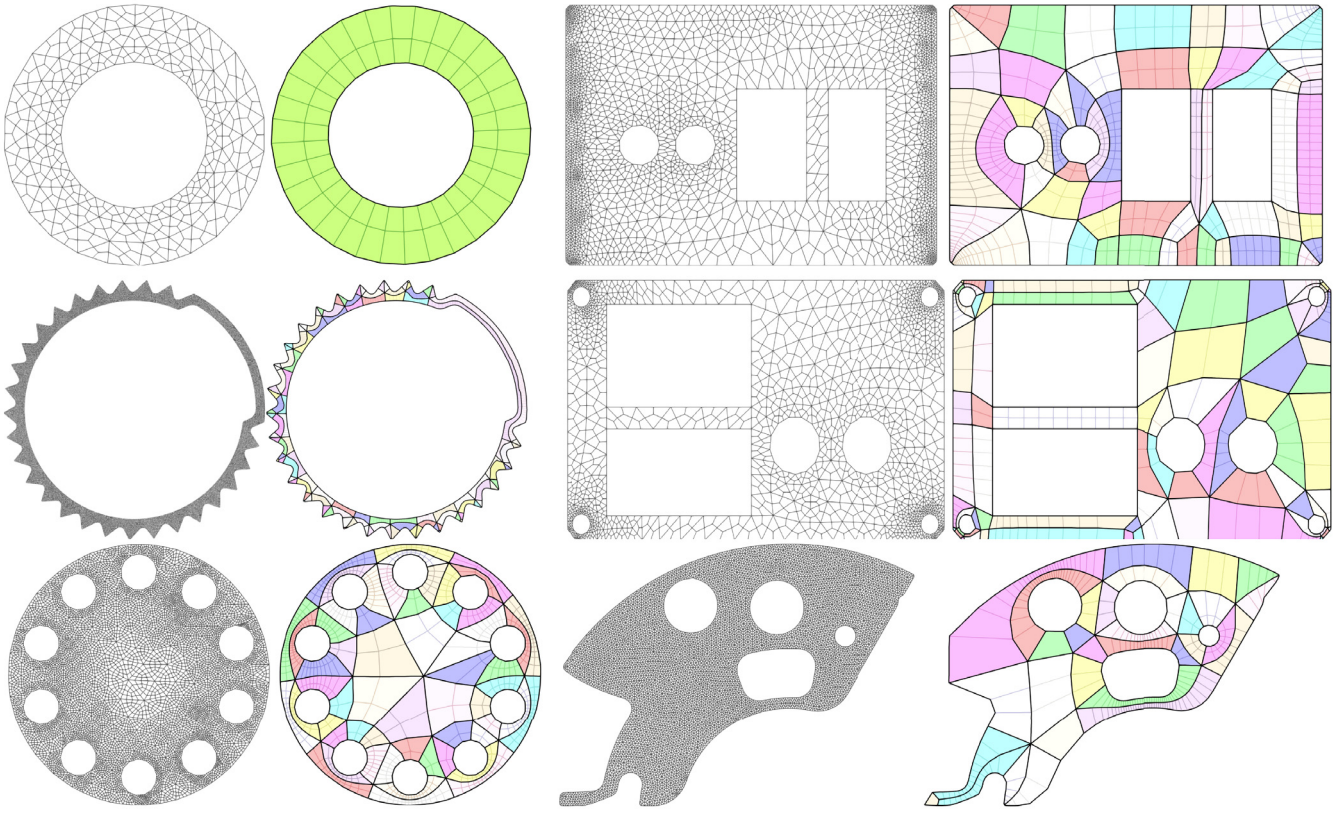


Fig. 7. Some results from our simplification framework. For each pair, the left image shows the input mesh and the right is the output. Different color regions correspond to different base complex components, and the dark curves show the structure of the base complexes.

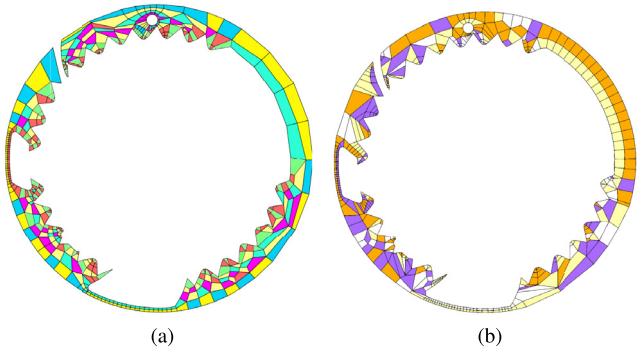


Fig. 8. Comparison of previous semi-global simplification work [5] with our semi-global simplification strategy. (a) shows the result from previous work. The reported singularities for this result are 175 whereas, the result from our simplification pipeline (b) has 145 singularities.

the mesh if total number of quad elements in current mesh falls below 25% of the number of elements in input mesh. In our experiments, refinement can sometimes lead to inversion free simplified meshes but in some other cases it leads to worse quality elements. Again, we leave this as an option for the user to decide.

After the above structure simplification, we perform a feature preserved smoothing and re-sampling, adapted from a previous work [45]. This smoothing optimizes the angles between edges in the mesh and preserves the boundary features by fixing vertices located at the corner-like sharp features. It also improves the Jacobian measures of the meshes.

A reference implementation of the proposed simplification pipeline can be found at https://github.com/DaViM-Lab-Repository/CotrikMesh.

7. Valence and boundary constraints

During the simplification, a number of constraints need to be checked in addition to the ranking to ensure the output of a high quality mesh with simpler structure.

7.1. Singularity valence constraint

In order to ensure that the valences of all singularities fall in the range of $[3, \text{val}_{max}]$ after the simplification operations, we calculate the prospective valence of the vertices involved in the separatrix simplification operations. For the 3–3 separatrix, the valences of the singularities diagonal to the two valence 3 singularities (Fig. 4(a)) reduces by 1. For 5–5 separatrix, the valence of singularities connected by the separatrix reduces by 1 (Fig. 4(c)), whereas the singularities diagonal to the separatrix singularities increases by 1. Generally, the valence of any two vertices being collapsed in 3–3 separatrix collapse and chord collapse is calculated as $\text{val}_1 + \text{val}_2 - 4$. If the prospective valences (val_p) satisfy the condition: $3 <= \text{val}_p <= \text{val}_{max}$, the simplification operation is performed, otherwise it is skipped. However, even after satisfying the constraints, doublets or singlets may be introduced after simplification, leading to the violation of minimum valence requirement. The doublet and singlet removal operations can be performed to remove these degenerate elements. A detailed discussion on the undesired configurations can be found in the supplemental document (Section 1).

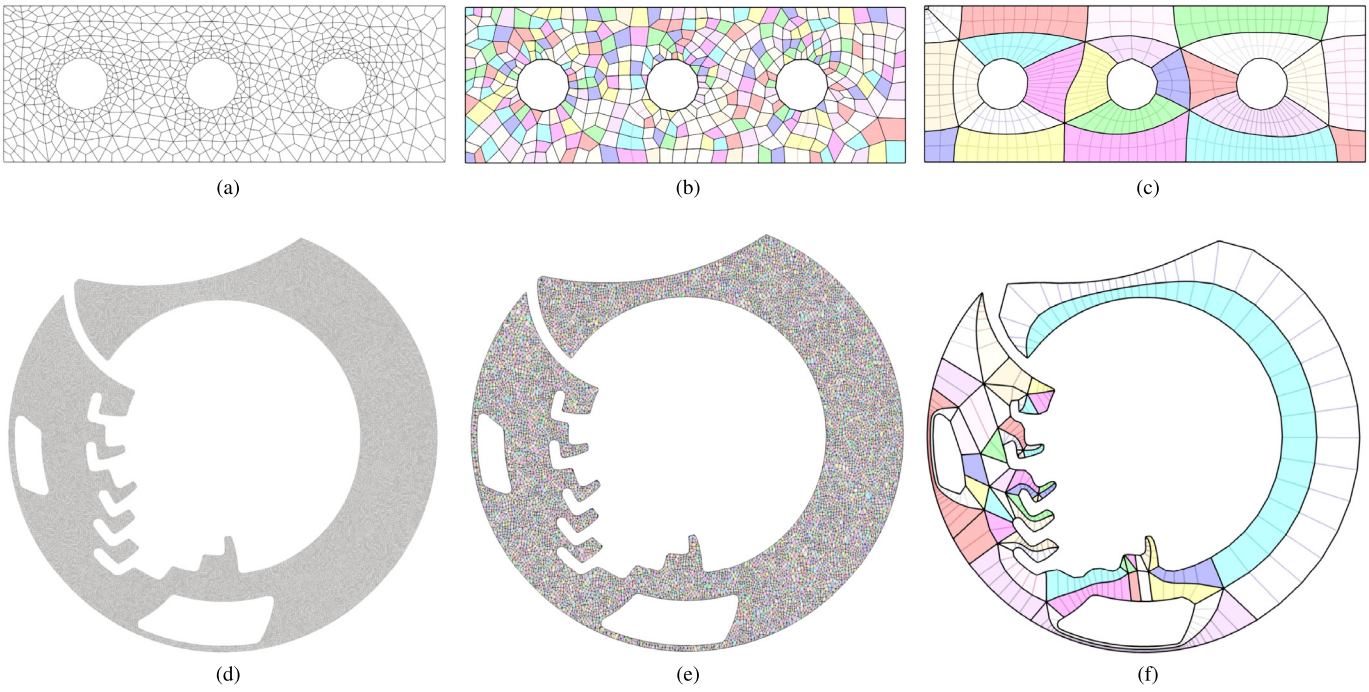


Fig. 9. Comparison of local simplification [3] with our semi-global simplification strategy. (a) and (d) show the input quad-meshes split from respective triangle meshes, (b) and (e) represent outputs with base complex from local simplification and (c) and (f) represent the outputs with base complex obtained through our simplification framework.

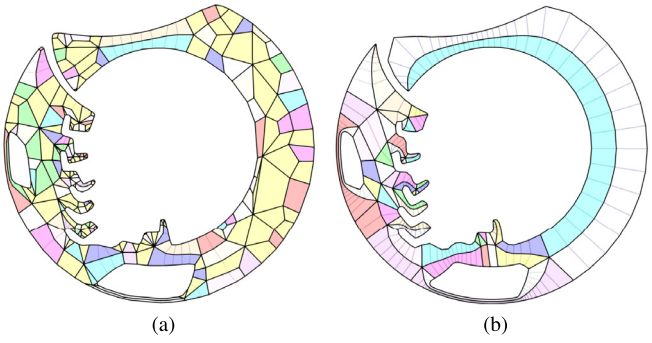


Fig. 10. Comparison of global simplification [4] with our semi-global simplification strategy. (a) represents output with base complex from the global simplification and (b) represents the output with base complex obtained through our simplification framework.

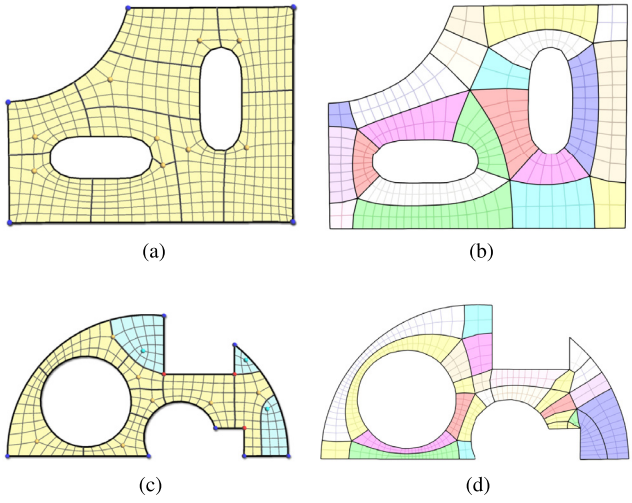


Fig. 11. Comparison of the quadrangulation produced by [29] of two patches with our result. (a) has 14 singularities while our result (b) has 12; in the meantime, (c) has 20 singularities and our result (d) has 18 singularities.

7.2. Boundary feature preservation

Before simplification, we identify the feature vertices (e.g., corners) and extract the boundary feature lines. Vertices at the corners (i.e., boundary singularities or with angles outside a range, $[\theta, 360^\circ - \theta]$, θ is set by the user) are fixed. Vertices on sharp edges can only move along the sharp edges. Operations that alter a corner or a sharp edge are prevented to achieve a boundary configuration as close to the input boundary as possible. A detailed description on the handling of different scenarios for boundary preservation is provided in the supplemental document (Table 1, Section 2).

7.3. Optional inversion-free constraint

The above introduced semi-global and local operations may introduce inverted quads (i.e., with negative Jacobian [2]) into the resulting meshes, which may be hard to correct. To prevent the introduction of inverted quads, similar to the work [2], after each collapsing operation (including 3–3 collapse, chord collapse and diagonal collapse), we check whether the affected quads become inverted or not. If an inverted quad is found, the operation is reversed. In practice, we leave this as an option for the user to choose. The default setting has this constraint turned off. The reason for disabling this constraint by default is to ensure maximum simplification in terms of singularity reduction as most inverted elements can be fixed by the local smoothing, however, certain input mesh configuration may become too coarse after simplification and inverted elements in such places may not be able to be improved using geometry optimization, therefore, simplification can be avoided in such regions of the mesh with the

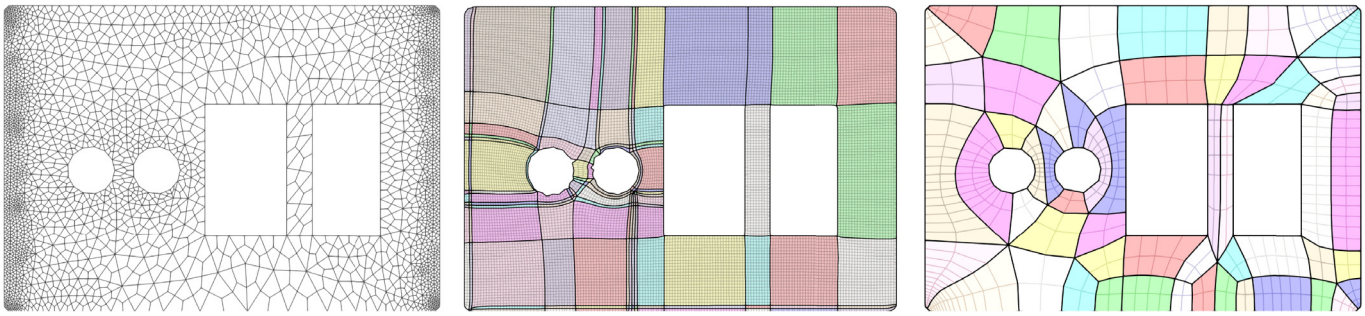


Fig. 12. Comparison of Quadriflow result (b) with our semi-global simplification strategy (c) given an input mesh (a).

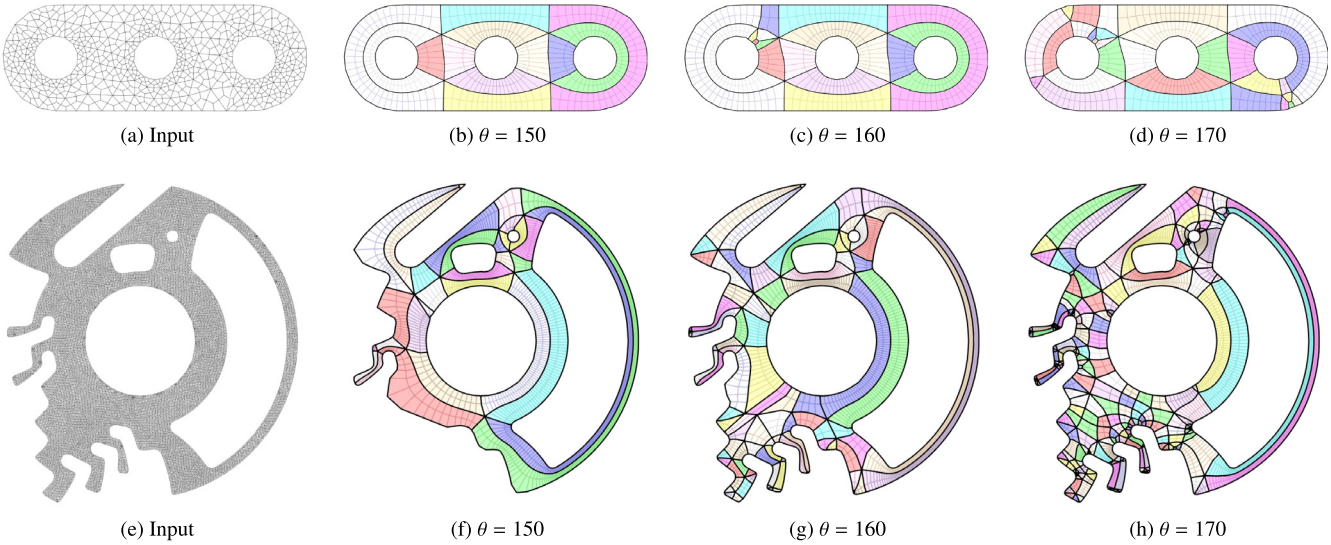


Fig. 13. Comparison of outputs produced using different values of θ for sharp feature (or corner) extraction. The 3 hole model (top row) and the mazewheel model (bottom) are used. As can be seen, with a small θ value (i.e., (b) and (f)), the structure is the simplest, yet important features may be lost ((f)). With increasing θ value, the features of the shape are better preserved (i.e., (h)), but the simplified structure is more complex than the one with a small θ.

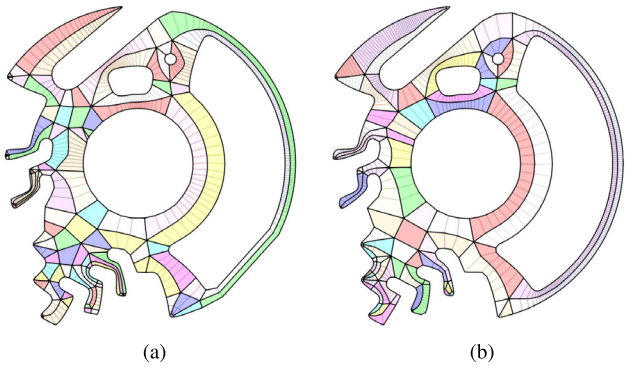


Fig. 14. Comparison of the results obtained with and without ranking strategy for simplification operations. The number of singularities for the result in (a) without any ranking, are 65, whereas, the result (b) using the proposed ranking has 53 singularities.

aid of inversion free constraint. This strong guarantee of inversion free outcome may affect the level of simplification as shown in Section 8.

8. Results

We applied our simplification method to a number of quad meshes with different geometry and topology configurations. Fig. 7 shows the results of our simplification framework applied to a number of representative quad meshes. In particular, we achieve over **96%** reduction of the singularities for these models, and the structures in the simplified meshes are close to their ideal structures given their boundary configurations. Table 1 provides the statistics of the results of our method and two other most relevant methods.

8.1. Comparison with other simplification methods

A comprehensive study of the comparison between the results obtained from our simplification framework and other mesh generation methods is as follows:

Previous semi-global simplification: Our simplification pipeline has three key differences from the previous semi-global simplification work [5]. *(1)* The ordering of operations in our pipeline (Section 6) is different than [5]. We perform all cleaning operations ahead of all other operations and diagonal collapse operation is ordered before semi-global operations as opposed to [5].

Table 1
Performance of our simplification method.

Model	Input	Local method [3]				Kaoji et al. [5]					Ours				
	#S	#S	Min. Scaled Jacobian	Avg. Scaled Jacobian	Hausdorff Dist.	#S	Min. Scaled Jacobian	Avg. Scaled Jacobian	Hausdorff Dist.	Time (s)	#S	Min. Scaled Jacobian	Avg. Scaled Jacobian	Hausdorff Dist.	Time (s)
Patch 1	195	64	0.435062	0.813715	0.00016	**4**	0.956264	0.992745	0.00735	0.47	**4**	0.997924	0.999526	0.00140	**0.39**
Patch 2	269	95	0.636886	0.886676	0	**4**	0.999351	0.999887	0	0.65	**4**	1	1	0	**0.43**
Patch 3	2428	1238	0.631782	0.876612	0.000170	18	0.352553	0.918613	0.00046	29.49	**10**	0.0647535	0.909919	0.03793	**7.93**
Patch 4	1791	781	0.627532	0.898935	0.00055	13	0.368196	0.832329	0.00962	15.96	**9**	0.348614	0.870761	0.00818	**2.41**
Patch 5	919	413	0.607521	0.886647	0.00046	17	−0.0742107	0.843955	0.01132	5.08	**14**	0.138939	0.802023	0.02975	**1.8**
1 hole	230	99	0.524566	0.861475	0.00197	**0**	0.989097	0.99407	0.00400	1.08	**0**	0.991053	0.994188	0.00527	**0.31**
2 holes	539	300	0	0.849616	0.00304	**4**	0.630725	0.910972	0.01636	4.04	**4**	0.748057	0.971484	0.010686	**0.89**
3 holes	417	148	0.676705	0.880316	0.00215	**4**	0.661058	0.942534	0.00432	3.07	**4**	0.685901	0.946452	0.00550	**0.63**
3 holes square	523	208	0.652941	0.887485	0	**12**	0.695252	0.915355	0	2.06	**12**	0.677243	0.928474	0	**0.85**
2 holes 2 squares	1825	922	0.479384	0.863472	0.00067	**20**	−0.134864	0.834113	0	15.46	**20**	0.230834	0.926504	0.03026	**4.33**
6 holes 2 squares	1112	435	0.520637	0.87084	0.00045	**23**	0.224303	0.87921	0.00558	8.78	**23**	0.280634	0.898758	0.00234	**2.99**
8 holes	2109	1089	0.484588	0.862253	0.00056	18	0.253169	0.90472	0.00079	19.10	**16**	0.620603	0.903053	0.00151	**5.5**
10 holes	1137	456	0.647599	0.913045	0.00027	22	0.0956317	0.83831	0.00377	65.68	**18**	0.469836	0.943661	0.00140	**5.5**
Mazewheel 1	23628	8484	0.0531367	0.891381	0	175	0.076333	0.809203	0	834.18	**145**	0.0451114	0.827077	0.00662	**259.54**
Mazewheel 6	11446	4093	0.62016	0.907099	0.00011	84	0.378568	0.924225	0.00055	1064.22	**78**	0.376601	0.916187	0.00172	**68.22**
Mazewheel 8	14315	5023	0.617715	0.915379	0.00028	103	−0.368346	0.90156	0.02407	1584.72	**77**	0.0258166	0.970718	0.00546	**114.22**
Mazewheel 19	5618	1996	0.670187	0.91499	0.00024	23	0.454705	0.908444	0.00197	199.95	**16**	0.106829	0.927656	0.05070	**26.23**

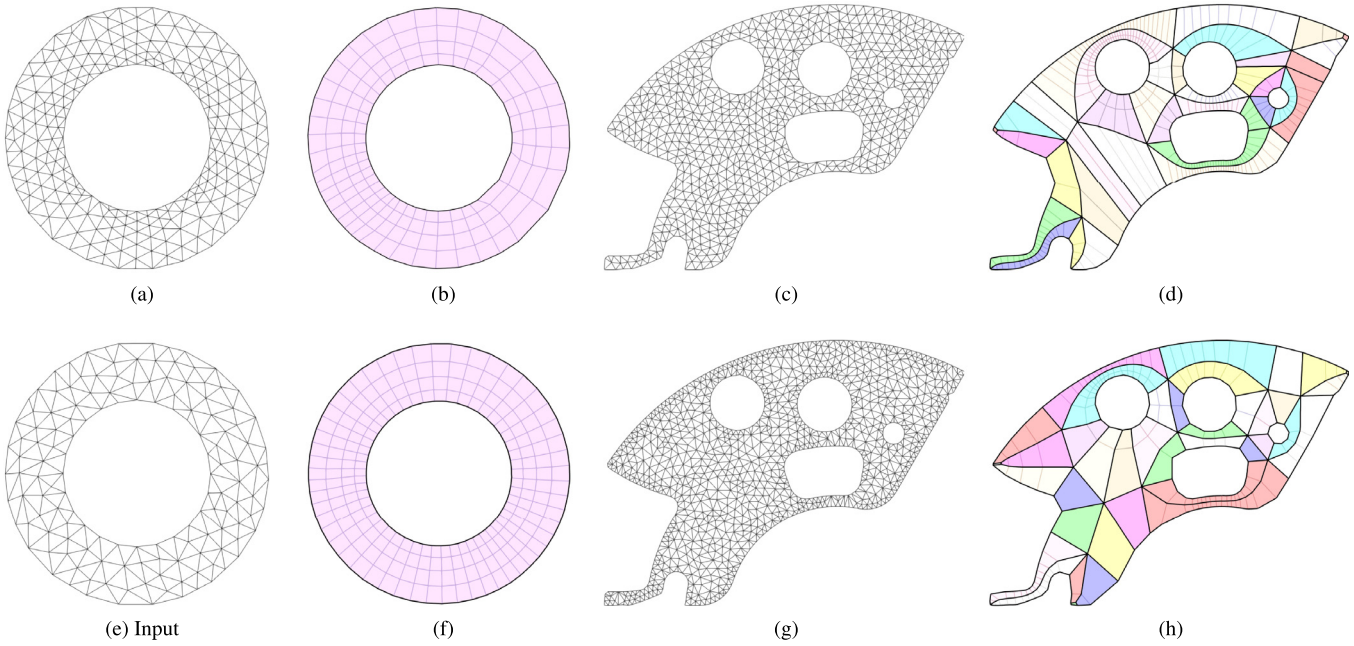


Fig. 15. Comparison of outputs produced using different triangle inputs as sources for our simplification framework. As can be seen, our framework produces similar result for simple models (e.g., (a), (b), (e), (f)), while it may lead to different structures for complex shapes if different triangulations are used (e.g., (c), (d), (g), (h)).

(2) We group each type of operation (Section 6) such that independent regions (non-overlapping regions) bounded by each type of operation are performed at once in contrast to the simplification pipeline in [5] where only one type of operation is performed in each simplification iteration. In this way, the time complexity of our pipeline is reduced to a great extent, see Table 1. *(3)* We employ a length based ranking strategy to order simplification operations which achieves increased singularities reduction. Fig. 8(a) shows a simplified result reported in [5] and the result obtained using our simplification pipeline Fig. 8(b). While exhibiting similar structure, all resulting meshes with our new method have equal or fewer singularities than those with the method in [5]. The new method also improves the Jacobian metric for some resulting meshes compared to the ones produced by the method of [5].

Local simplification: Usually, local operations can be applied to a quad mesh that is split from a triangle mesh, while the global operations, such as chord collapsing, cannot because they may produce singularities with valences not falling in the range $[3, \text{val}_{max}]$. Singularity alignment is usually applied to a closed surface, while our method targets an open surface with boundary. Also, most singularities in our input meshes are aligned (i.e., connecting with other singularities). Therefore, it is not suitable to compare our results to the simplification obtained using singularity alignment. Nonetheless, we compare the results generated by our simplification strategy to local simplification strategy proposed by [3]. Since their original method does not focus on meshes with open boundaries, we made some adjustments so it can preserve boundary features and achieve boundary conformality. We calculate the total singularities for both simplification strategies and report the minimum scaled Jacobian values. As seen in Fig. 9, the obtained structure is not as simple as our results. Table 1 reports the statistics for the comparison between the results generated through local simplification and those with our simplification framework. It is observed that our method outperforms local simplification in terms of singularity reduction to a great extent. Our method also improves the Jacobian metric of the individual elements, while having a slightly larger Hausdorff distance to the input shapes. This is understandable as our method significantly reduces the number of singularities, leading to fewer elements (or samples) at the boundaries.

Global simplification: Global simplification usually performs the chord collapse operation along with other operations to optimize the structure of the mesh. However, as mentioned earlier, performing global operations on a highly unstructured mesh may produce singularities outside optimal range. In [4], a ranking strategy is defined to improve the singularity valence while optimizing the structure. We modify their strategy for the planar meshes and pair it with edge rotation for boundary conformality. Fig. 10 shows the result obtained through global simplification and our result using the mazewheel_1 mesh.

Quadrangulation of 2D patches: In [29], quadrangulation of 2D patches is obtained through subdivision into simple basic patches with an emphasis on introducing as few singularities as possible within those basic patches. We demonstrate the effectiveness of our framework in Fig. 11, where Figs. 11(a) and 11(c) show the quadrangulation of two 2D patches as presented in [29], while Figs. 11(b) and 11(d) show the structures of the patches obtained through our simplification framework. From the comparison, we see that our simplification achieves simpler structures with better placement of singularities. This is because the subdivision strategy used by [29] may lead to a locally optimal result (i.e., optimal within a sub-patch) but not a globally optimal one.

Quadriflow: While Quadriflow [33] works well on surface meshes, results for the planar cases (Fig. 12) are sub optimal and may fail to capture the boundary features of the input meshes. Moreover, results from our framework exhibit element size adaptivity as opposed to uniform element sizes in Quadriflow (14 400 faces to preserve input features). Fig. 12(b) shows the result from Quadriflow through parameter tuning that almost captures the features of the input mesh with 32 singularities, whereas our result (Fig. 12(c)) preserves the features with similar singularities but lesser element count.

Table 2
Impact of the angle threshold θ for boundary preservation.

Model	#Singularities				
	Input	$\theta = 150°$	155°	160°	165°
Square	269	4	4	4	4
1 hole	230	0	0	0	6
2 holes	539	4	24	24	25
3 holes	417	4	4	4	7
2 holes 2 squares	1825	19	49	49	49
6 holes 2 squares	1112	21	54	54	54
8 holes	2109	16	62	62	62
10 holes	1137	18	72	72	72

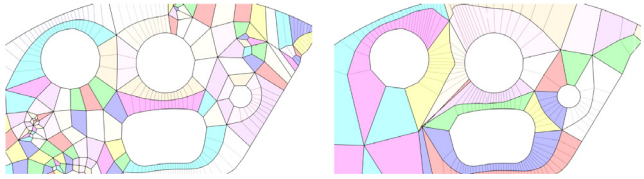


Fig. 16. Comparison of results produced with (a) and without (b) inversion-free guarantee. The result in (a) has 103 singularities, while (b) has 17.

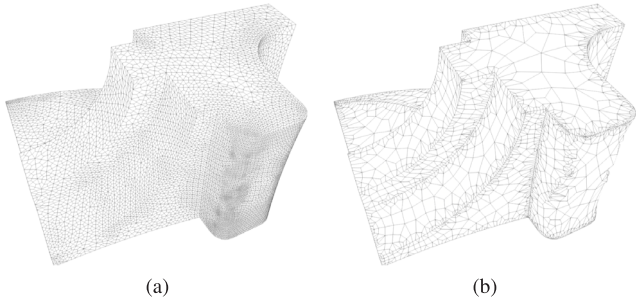


Fig. 17. Simplification result for a surface quad mesh using our simplification framework. (a) represents the input and (b) shows the simplified mesh.

8.2. Impact of parameters

The most crucial parameter of our framework that can be tuned by the user is the angle threshold θ for feature preservation. Higher values for the angle threshold can put a limit on the simplification process since the boundary is preserved as much as possible while lower angle threshold values can oversimplify certain regions in the mesh. Table 2 represents the singularities in the output mesh for different angle threshold θ in different models. Fig. 13 shows the effect of different values for θ on simplification extent. For the 3 holes model, it can be seen that the number of singularities in the result using $\theta = 150$ is the least compared to other values of θ (sub- Figs. 13(c) and 13(d)). For the patch extracted from mazewheel model, using the lower value for θ causes some features to be lost. Therefore, θ can be tuned by users to achieve the best simplification results while preserving the features.

8.3. Impact of ranking the operations

In our implementation, separatrix connections are built in the order in which singularities in the mesh are encountered. We iterate through the mesh vertices, identify singularities and build separatrix connections. Therefore, without a ranking strategy, the groups of simplification operations are sorted in a random order [5]. In our pipeline, we order the operations according to a length based ranking strategy as described in Section 6. Fig. 14 shows the simplification results obtained for a model with and without the ranking strategy. It is evident that with ranking in place, the number of singularities in the final result 14(b) is less than the result obtained without any ranking for the simplification operations 14(a).

8.4. Impact of different triangle inputs

We study the impact of different triangle inputs for the same model on the simplification of quad meshes obtained through Catmull–Clark subdivision using our framework. Some example results are shown in Fig. 15. In general, for models with simple boundary configurations, the input triangle meshes have little impact to the simplification result as shown by the one hole example in Fig. 15. For more complex models, the triangulation has some impact to the simplification, as shown by the mechanic model in Fig. 15. In this example, the two simplification results (Figs. 15(d) and 15(h)) have slightly different structures, while having similar numbers of singularities. The difference of the two structures is mostly caused by the slight difference of the positions and valences of some singularities.

8.5. Option of inversion-free output

As mentioned in Section 7.3, our framework offers an option for the user to achieve a guaranteed inversion-free output. However, the resulting mesh using this strategy may have more singularities than the results produced without inversion-free guarantee. We have applied this strategy to produce inversion free simplified meshes for certain complex examples. Fig. 16 shows a region of the mazewheel-19 model, produced with (a) and without (b) inversion free guarantee. It can be seen that 16(a) contains convex quad elements with a minimum and average scaled Jacobian values of **0.107** and **0.928**. In contrast, there are some overlapping elements in the simplified mesh with the inversion-free strategy turned off 16(b), which are difficult to fix using optimization, hence, the minimum and average scaled Jacobian values are calculated as **-0.265** and **0.896**, respectively.

In summary, our hybrid simplification framework addresses the limitations of global and local operations and can simplify a quad mesh to a near optimal structure. In addition, singularity alignment is implicitly achieved via the above simplification operations without an explicit treatment.

9. Summary and future work

We proposed an effective simplification method for planar quad meshes with boundaries that is mainly based on semi-global separatrix operations and a few local operations. These operations are organized into groups and are sorted to achieve an optimal result with fast computation. Our framework preserves the boundary features and produces singularities in the user-specified valence limit. Our framework also provides option

for the user to achieve an inversion-free output. We have applied our simplification framework to a number of planar quad meshes with different boundary configurations to demonstrate its effectiveness. We also compared our results with the previous semi-global simplification framework and other quad mesh simplification and quad mesh generation methods to demonstrate its advantages.

With the proposed method, the practitioners (or engineers) can start with designing their (CAD) models using triangular meshes or other simpler quad-mesh generation approaches that are robust. They can then use our method to convert these initial meshes into high quality quad-meshes with simpler structures (and with fewer singularities) that are often preferred by higher-order finite element methods and isogeometric analysis [23].

Limitations. There are a few places in which our framework can be improved. First, the utilized simple ranking strategy based on the width of the regions for collapsing may not be optimal. Thus, some simplification results may not be optimal (e.g., the upper-left corner of Fig. 9(c)). Also, as the simplification progresses, some boundary features may still be lost due to fewer elements around those features (e.g., the lower right example in Fig. 7). Second, our algorithm may not place the resultant singularities to their optimal locations, which may be addressed using certain guidance fields (e.g., cross or frame fields) derived from boundaries. One way to utilize the frame fields for optimal singularity placement would be to prioritize and select operations that contain singularities that most align with the frame field. For each type of operation, e.g., collapse, split, rotate, etc., the singularities with most optimal placement can be retained while other singularities involved in such operations can be eliminated. Third, our current framework concentrates mainly on 2D meshes with open boundaries and does not directly apply to closed surface meshes. For surface quad mesh inputs, the half separatrix operations are not valid, however, a normal separatrix that crosses the boundary can be considered as two half separatrices on each side of the boundary region connected via boundary edges. To preserve the mesh features, we avoid the semi-global and global operations involving boundary regions which restricts the extent of simplification on the mesh. In addition, the curved regions of the surface mesh can become distorted following simplification and additional processing may be required to recover the surface shape. Fig. 17(b) shows the simplified mesh obtained through our simplification framework. It can be observed that the resultant mesh contains fewer singularities as compared the input but is still quite unstructured nevertheless. In the future, we plan to address the above limitations and extend our simplification pipeline to handle surface quad meshes and hex-meshes.

CRediT authorship contribution statement

Muhammad Naeem Akram: Conceptualization, Methodology, Software, Visualization, Writing, Reviewing. **Kaoji Xu:** Conceptualization, Methodology. **Guoning Chen:** Conceptualization, Supervision, Writing, Reviewing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The link for the code repository is included in the manuscript.

Acknowledgments

We would like to thank Lei Si (University of Houston) for helping to generate the results using Quadriflow. This work was partially supported by National Science Foundation (NSF) IIS 1553329.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cag.2022.10.001.

References

[1] Bommes D, Lévy B, Pietroni N, Puppo E, Silva C, Tarini M, et al. Quad-mesh generation and processing: A survey. Comput Graph Forum 2013;32(6):51–76. http://dx.doi.org/10.1111/cgf.12014.
[2] Gao X, Huang J, Xu K, Pan Z, Deng Z, Chen G. Evaluating hex-mesh quality metrics via correlation analysis. Comput Graph Forum 2017;36(5):105–16. http://dx.doi.org/10.1111/cgf.13249.
[3] Tarini M, Pietroni N, Cignoni P, Panozzo D, Puppo E. Practical quad mesh simplification. Comput Graph Forum 2010;29(2):407–18. http://dx.doi.org/10.1111/j.1467-8659.2009.01610.x, URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2009.01610.x.
[4] II JD, Silva CT, Shepherd J, Cohen E. Quadrilateral mesh simplification. ACM Trans Graph 2008;27(5):148.
[5] Xu K, Akram MN, Chen G. Semi-global quad mesh structure simplification via separatrix operations. In: SIGGRAPH Asia 2020 technical communications. New York, NY, USA: Association for Computing Machinery; 2020, http://dx.doi.org/10.1145/3410700.3425436.
[6] Catmull E, Clark J. Recursively generated B-spline surfaces on arbitrary topological meshes. In: Seminal graphics: Pioneering efforts that shaped the field. New York, NY, USA: Association for Computing Machinery; 1998, p. 183–8.
[7] Remacle JF, Lambrechts J, Seny B, Marchandise E, Johnen A, Geuzainet C. Blossom-quad: A non-uniform quadrilateral mesh generator using a minimum-cost perfect-matching algorithm. Internat J Numer Methods Engrg 2012;89(9):1102–19.
[8] Boier-Martin I, Rushmeier H, Jin J. Parameterization of triangle meshes over quadrilateral domains. In: Proceedings of the 2004 Eurographics/acm SIGGRAPH symposium on geometry processing. ACM; 2004, p. 193–203.
[9] Lévy B, Liu Y. Lp centroidal voronoi tessellation and its applications. ACM Trans Graph 2010;29(4):119:1–119:11.
[10] Campen M. Partitioning surfaces into quadrilateral patches: A survey. In: Proceedings of the European association for computer graphics: Tutorials. Goslar, DEU: Eurographics Association; 2017, http://dx.doi.org/10.2312/egt.20171033.
[11] Alliez P, Cohen-Steiner D, Devillers O, Lévy B, Desbrun M. Anisotropic polygonal remeshing. ACM Trans Graph 2003;22(3):485–93.
[12] Pietroni N, Puppo E, Marcias G, Scopigno R, Cignoni P. Tracing field-coherent quad layouts. Comput Graph Forum 2016;35(7):485–96.
[13] Vaxman A, Campen M, Diamanti O, Panozzo D, Bommes D, Hildebrandt K, et al. Directional field synthesis, design, and processing. Comput Graph Forum 2016;35(2):545–72.
[14] II JD, Silva CT, Cohen E. Semi-regular quadrilateral-only remeshing from simplified base domains. Comput Graph Forum 2009;28(5):1427–35.
[15] Usai F, Livesu M, Puppo E, Tarini M, Scateni R. Extraction of the quad layout of a triangle mesh guided by its curve skeleton. ACM Trans Graph 2015;35(1):6.
[16] Campen M, Bommes D, Kobbelt L. Dual loops meshing: Quality quad layouts on manifolds. ACM Trans Graph 2012;31(4):110:1–110:11. http://dx.doi.org/10.1145/2185520.2185606, URL http://doi.acm.org/10.1145/2185520.2185606.
[17] Campen M, Kobbelt L. Dual strip weaving: Interactive design of quad layouts using elastica strips. ACM Trans Graph 2014;33(6):183:1–183:10. http://dx.doi.org/10.1145/2661229.2661236, URL http://doi.acm.org/10.1145/2661229.2661236.
[18] Dong S, Bremer PT, Garland M, Pascucci V, Hart JC. Spectral surface quadrangulation. ACM Trans Graph 2006;25(3):1057–66.
[19] Campen M, Kobbelt L. Quad layout embedding via aligned parameterization. Comput Graph Forum 2014;33(8):69–81.
[20] Tarini M, Hormann K, Cignoni P, Montani C. PolyCube-maps. ACM Trans Graph 2004;23(3):853–60.
[21] Campen M, Bommes D, Kobbelt L. Quantized global parametrization. Acm Trans Graph (Tog) 2015;34(6):1–12.
[22] Lyon M, Campen M, Kobbelt L. Quad layouts via constrained T-mesh quantization. Comput Graph Forum 2021;40(2):305–14.

[23] Couplet M, Reberol M, Remacle J. Generation of high-order coarse quad meshes on CAD models via integer linear programming. 2021, CoRR abs/2108.02635, URL https://arxiv.org/abs/2108.02635, arXiv:2108.02635.
[24] Viertel R, Osting B, Staten ML. Coarse quad layouts through robust simplification of cross field separatrix partitions. 2019, ArXiv arXiv:1905.09097.
[25] Ray N, Li WC, Lévy B, Sheffer A, Alliez P. Periodic global parameterization. ACM Trans Graph 2006;25(4):1460–85.
[26] Kalberer F, Nieser M, Polthier K. QuadCover - surface parameterization using branched coverings. Comput Graph Forum 2007;26(3):375–84.
[27] Huang J, Zhang M, Ma J, Liu X, Kobbelt L, Bao H. Spectral quadrangulation with orientation and alignment control. ACM Trans Graph (TOG) 2008;27(5):147.
[28] Fang X, Bao H, Tong Y, Desbrun M, Huang J. Quadrangulation through morse-parameterization hybridization. ACM Trans Graph 2018;37(4):92.
[29] Peng CH, Barton M, Jiang C, Wonka P. Exploring quadrangulations. ACM Trans Graph 2014;33(1). http://dx.doi.org/10.1145/2541533.
[30] Bommes D, Campen M, Ebke HC, Alliez P, Kobbelt L. Integer-grid maps for reliable quad meshing. ACM Trans Graph 2013;32(4):98.
[31] Jakob W, Tarini M, Panozzo D, Sorkine-Hornung O. Instant field-aligned meshes. ACM Trans Graph 2015;34(6):189.
[32] Bommes D, Zimmer H, Kobbelt L. Mixed-integer quadrangulation. ACM Trans Graph 2009;28(3).
[33] Huang J, Zhou Y, Nießner M, Shewchuk J, Guibas L. QuadriFlow: A scalable and robust method for quadrangulation. 37, 2018, http://dx.doi.org/10.1111/cgf.13498,
[34] II JD, Silva CT, Cohen E. Localized quadrilateral coarsening. Comput Graph Forum 2009;28(5):1437–44.
[35] Bozzo A, Panozzo D, Puppo E, Pietroni N, Rocca L. Adaptive quad mesh simplification. In: Eurographics Italian chapter conference. Citeseer; 2010, p. 95–102.
[36] Shepherd JF, Dewey MW, Woodbury AC, Benzley SE, Staten ML, Owen SJ. Adaptive mesh coarsening for quadrilateral and hexahedral meshes. Finite Elem Anal Des 2010;46(1):17–32. http://dx.doi.org/10.1016/j.finel.2009.06.024, Mesh Generation - Applications and Adaptation.
[37] Tarini M, Puppo E, Panozzo D, Pietroni N, Cignoni P. Simple quad domains for field aligned mesh parametrization. ACM Trans Graph 2011;30(6):142.
[38] Docampo-Sanchez J, Haimes R. Towards fully regular quad mesh generation. In: AIAA scitech 2019 forum. 2019, p. 1988.
[39] Peng CH, Zhang E, Kobayashi Y, Wonka P. Connectivity editing for quadrilateral meshes. ACM Trans Graph 2011;30(6):141.
[40] Bommes D, Lempfer T, Kobbelt L. Global structure optimization of quadrilateral meshes. Comput Graph Forum 2011;30(2):375–84.
[41] Viertel R, Osting B. An approach to quad meshing based on harmonic cross-valued maps and the Ginzburg–Landau theory. SIAM J Sci Comput 2019;41(1):A452–79.
[42] Chen G, Mischaikow K, Laramee RS, Pilarczyk P, Zhang E. Vector field editing and periodic orbit extraction using morse decomposition. IEEE Trans Vis Comput Graphics 2007;13(4):769–85.
[43] Skraba P, Wang B, Chen G, Rosen P. Robustness-based simplification of 2D steady and unsteady vector fields. IEEE Trans Vis Comput Graphics 2015;21(8):930–44.
[44] Gao X, Panozzo D, Wang W, Deng Z, Chen G. Robust structure simplification for hex re-meshing. ACM Trans Graph 2017;36(6):1–13.
[45] Akram MN, Si L, Chen G. An embedded polygon strategy for quality improvement of 2D quadrilateral meshes with boundaries. In: Proceedings of the 16th international joint conference on computer vision, imaging and computer graphics theory and applications - Volume 1: GRAPP. SciTePress, INSTICC; 2021, p. 177–84. http://dx.doi.org/10.5220/0010209101770184.