博士論文

Feature-aware Partitioning of Quadrilateral Meshes for Reverse Engineering
(リバースエンジニアリングのための特徴を考慮した四辺形メッシュ分割)

Doctoral Thesis

2013

Gunpinar Erkan

(グンプナ　エルカン)

学生番号 37-107380

精密機械工学専攻博士課程

ABSTRACT

Reverse engineering is a tool that transforms a real-world object into digital world to perform assessments and improvements on the object particularly using CAD/CAE/CAM applications. B-spline surfaces are the most common representation that is used for representing free-form shapes mainly in ship building, automotive and airplane industry. Conversion of scanned data (obtained using 3D scanning machines) to B-spline surfaces is nontrivial and a little work has been done on this conversion. The research in this dissertation aims to generate B-spline surfaces from a given quadrilateral mesh which is preferable than triangular mesh because of its good fit with surface parameterization, its oriented elements according to the principal curvature directions, etc. Highly-curved regions (feature curves) inside partitions cannot be represented well with smooth B-spline surfaces, therefore feature curves in the model is appropriate to be located on the surface boundaries which is the basic approach in this dissertation. Three different algorithms which uses the concept of placement of feature curves on the surface boundaries are proposed in order to generate high-quality B-spline surfaces.

The bi-monotone approach generates *bi-monotone* partitions that have ability to capture the surface details of an object easier than the quad-like partitions. Therefore, they are appropriate for use in reverse engineering applications since an appropriate number of partitions can determine feature curves within the boundaries which is not always possible when quad partitions are used. The proposed algorithm also includes sequence of methods that generates bi-monotone partitions which are at the end fitted with B-spline surfaces.

The Path Flipping approach generates quadrilateral partitions from a given quadrilateral mesh which is based on the motorcycle graph (MCG) algorithm of Eppstein et al. As the MCG algorithm does not take surface geometry into account during partitioning, the partitions produced are not appropriate to fit by B-spline surfaces. Initial partitioning is first performed using a speed control algorithm identical to the original algorithm except that it assigns variable rather than constant speed to particles. Partition boundaries are then improved via local path flipping operations. However there are still highly-curved regions existing inside the partitions after these two steps. For this reason, feature curves are extracted and integrated into the proposed framework by using some methodology. Some of flat partition boundaries are then removed. The generated partitions of this algorithm are in quad-shape and fitted with B-spline surfaces without surface trimming.

The MCG enumeration approach, the third approach, also intends to generate quadrilateral partitioning from motorcycle graph and it is motivated by the weak point of local path flipping operation (of the second algorithm). This algorithm is locally performed and therefore the results

produced converge to the local optimum. In order to reach global optimum, all possible motorcycle graphs of a given quadrilateral mesh are intended to be listed to find the global optimum one. Even though this is possible, it has high computational cost. In order to reduce the computation cost, mesh is divided into several sub-meshes and optimum motorcycle graph is found separately by enumerating all motorcycle graph combinations of these sub-meshes. Finally a quasi-optimum motorcycle graph of the whole mesh is obtained by combining the separate solution in each sub-meshes. The motorcycle graph obtained using this method is appropriate to fit by B-spline surfaces.

The results of each algorithm are demonstrated and comparison is made by checking visually as well as using surface fitting quality of the generated surfaces. In terms of capturing feature curves not aligned with parameter *uv* directions, the bi-monotone approach is much more advantageous than other two approaches. The Path Flipping approach can generate better results (more highly-curved regions can be placed on partition boundaries) than the bi-monotone approach and it can generate similar results in a shorter time compared to the MCG enumeration approach for the models having smooth features. The MCG enumeration enumerates many motorcycle graphs in order to find the optimum one, and therefore the partitions generated using this method are possibly more appropriate to fit by B-spline surfaces than that of the bi-monotone and Path Flipping approaches. All tests in this dissertation has been made with semi-regular quadrilateral mesh models generated using the mixed integer quadrangulation technique of Bommes et al.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Introduction

1.1.1 Reverse Engineering

As computer-aided technologies are highly evolved in the last decade, there is a need for digitizing the real-world products in order to perform assessments or improvements on these products. Reverse engineering is a method that transforms an existing physical object into a 3D virtual model so that the model can be used in business or engineering softwares for several purposes such as analyze, evaluation, cost estimation, etc. Starting from a measured object using 3D scanning technologies, point cloud lacking of topological information is obtained. This data is processed in order to generate much more useful format such as mesh, surface or CAD model.

Figure 1.1 illustrates a reverse engineering application where parametric surface model is generated from a scanned point cloud. The common process for the reverse engineering is as follows:

1. **Data acquisition:** 3D scanners collects data (point clouds) on the shape of a given physical object which is used digital, three dimensional models in the next step. There exist various technologies (such as CMM, laser scanners, structured light digitizers, CT scanners) for capturing 3D data of the object. The technique that is used for the measurement depends on the application and the measurement environment.

2. **Triangulation:** Triangular mesh models represent a shape using small triangular flat faces and they include the topology information between these faces. These models are suitable for many applications such as data visualization, CAE, CAM, etc. Constructing surfaces from point clouds has been studied well so far and existing techniques [4, 6, 15, 28] can be used for this. In order to generate better quality triangular meshes (such as isotropic or adaptive), several remeshing methods [3, 10, 47] exist.

3. **Partitioning:** Starting from a triangular mesh data (also possible from point cloud), partitioning process aims to partition the data into several meaningful groups each of which is fitted with surfaces. This is done using some geometrical constraints and it is good to perform surface fitting and partitioning iteratively.

4. **Surface fitting:** A CAD model should embody the *design intent* such that each partitioned data in the previous step should be fitted with an appropriate feature such as basic primitives or NURBS surfaces. Moreover, features and their relationship to other features should be defined properly in order to obtain a precise CAD model.

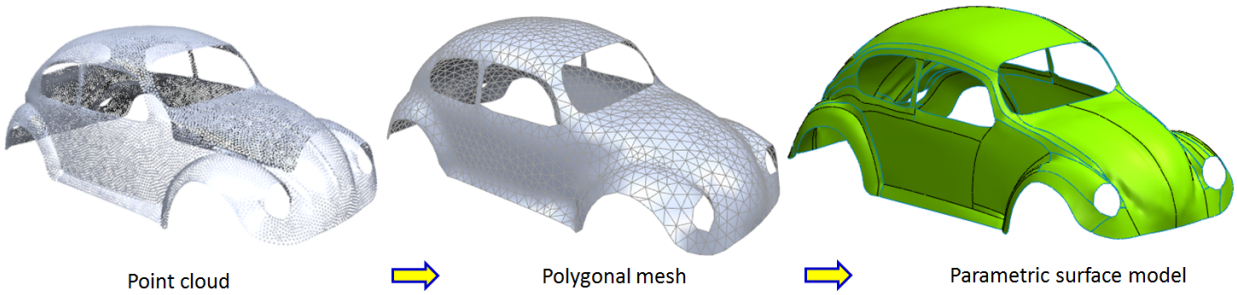


Figure 1.1: A reverse engineering application: A 3D scanning instrument measures a given model and generates point cloud from which polygonal mesh model can be obtained. Finally parametric surface model is generated so that the digital model is ready for business or engineering applications.

The ultimate aim of the reverse engineering systems is to develop a fully automatic system from data acquisition to CAD model generation so that CAD model is directly created after measuring the physical object [50, 51, 17]. Data partitioning is one of the most critical process and it must be carefully performed according to the data representation type that will be used in the final CAD model. Otherwise, the CAD model generated does not represent the scanned real model accurately since each representation has its drawbacks. In case of B-spline surface representation, sharp edges in the model should be identified and located on the surface boundaries. For this reason, the chosen partitioning algorithm should decompose a given data set into groups each of which is fitted with smooth surfaces so that smooth connections should exist between these surfaces.

1.1.2 Basic Motivation: Quadrilateral Partitioning for Reverse Engineering

Before going into the research details of this dissertation, basic motivation and idea are listed as follow:

- From the point of view of surface fitting, quadrilateral partitions are preferred because of their parameterization easiness in surface modeling applications. It is called quadrilateral partitioning.

- Some early works have presented methods for the generation of quadrilateral partitioning from a give triangular meshes [17, 38, 33]. The proposed methods in these works partition the mesh like tiling square tiles and thus cannot capture the *feature curves* (highly-curved parts in the model). In order to capture these feature curves in the partitioning, the partitions

are not of square shapes. Some of them try to make the boundaries of the partitions as convex as possible. But most of the cases, the boundaries of the partitions cannot be guaranteed to be quadrilateral.

- Quadrilateral partitioning problem is considered hard when triangular meshes are utilized. However, recent developments in quadrangulation techniques allow us to work on high quality quadrilateral meshes (mesh of quads). Figure 1.3 (a) shows quadrilateral partitioning from a quadrilateral mesh. Drawing straight lines starting from outward direction of extraordinary vertices in a quadrilateral mesh (left), quadrilateral partitions (right) can easily be obtained.

- In the papers published on quadrangulation, it is suggested to use such quadrilateral meshes for the reverse engineering, but there are only a few studies utilizing quadrilateral meshes for quadrilateral partitioning.

- In this dissertation, we have presented some novel algorithms for generating partitions from quadrilateral meshes (not from triangular meshes as is usually the case of other partitioning approaches) suitable for the reverse engineering.

1.1.3 Quadrilateral Meshes

In computer graphics and geometric modeling, polygonal meshes are used to represent the shape of an object using vertices, edges and faces. Triangular meshes are the most common polygonal mesh representation where the faces are in triangle shape. Many software packages can operate on them since these meshes have ability to capture the shape of the object well by triangle faces. Quadrilateral mesh is another type of polygonal meshes such that a set of quadrilateral faces defines the object. These meshes are preferable than triangular meshes in various applications such as texture mapping, subdivision surfaces, FEM applications. Several researches [16, 24, 31, 34, 56] have pointed out that quadrilateral meshes are much more favorable in FEM analysis (such as automobile crash simulation and sheet-metal forming simulation) because of solution accuracy and they can be degenerated robustly and efficiently during the simulation.

Quadrilateral meshes have a high-level global structure such that their elements intrinsically cover much more global information of surface geometry than those of triangular meshes because they are well oriented based on the principal curvature directional fields of the surfaces involved (shown by the light green and yellow lines in Figure 1.2 (b)). A vertex in quadrilateral mesh is ordinary vertex if the number of edges incident to it is four (for interior vertices) or two or three

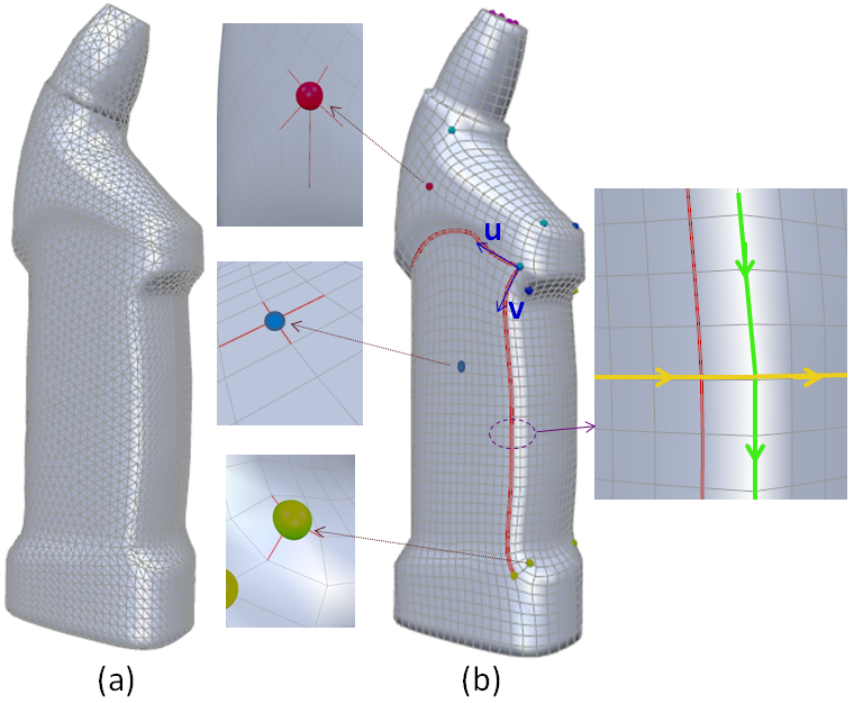


Figure 1.2: (a) Triangular mesh (b) Quadrilateral mesh with regular (blue dot) and irregular vertices (green and red dots). The parametric $u - v$ directions can be defined starting from irregular vertices, and quadrilateral elements are oriented according to the maximum (yellow) and minimum (light green) principal curvature directions.

(for boundary vertices) (see the blue dot in Figure 1.2 (b)). Vertices that are not ordinary are called extraordinary vertices (see the green and red dots in Figure 1.2 (b)). Starting from extraordinary vertices, parametric *u*-*v* directions (the blue arrows in Figure 1.2 (b)) can be defined at each regular vertex, and parametric lines (the red edges in Figure 1.2 (b)) can be generated based on these directions. For this reason, quadrilateral meshes are also appropriate for surface parameterization.

1.1.4 Quadrilateral Partitioning

Automatic mesh partitioning [1, 46] has been studied in the last few years and various techniques have been proposed for different applications such as modeling, morphing, etc. Even though ”What is a good partitioning?” depends on the application, many of these partitioning techniques strive for capturing the design intent. To do so, several evaluation criteria are used to determine the resulting partitioning which reflects the human intuition. Most of mesh partitioning methods uses triangular meshes as input, however only a few proposed works benefit from the structural beauties of quadrilateral meshes.

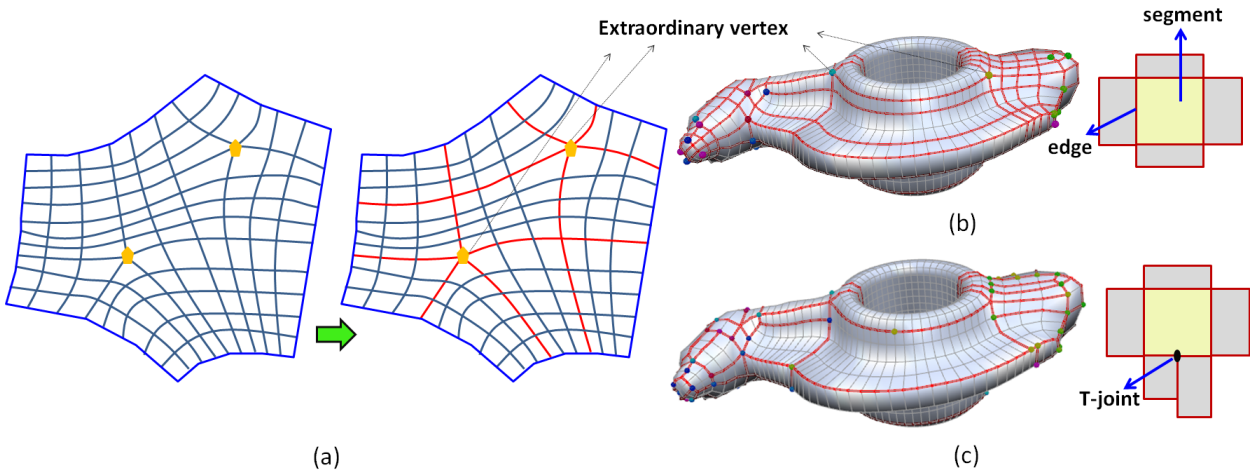


Figure 1.3: (a) Drawing straight lines starting from outward direction of extraordinary vertices in a quadrilateral mesh (left), quadrilateral partitions (right) can easily be obtained. Quad partitioning (b) without T-joints (c) with T-joints.

With the improvement of quad meshing techniques [3, 8], quad partitioning recently becomes popular which can easily be obtained using quadrilateral meshes. Such partitioning produces domains appropriate for surface modeling applications, texture mapping, subdivision surfaces, etc. Since the extraordinary vertices in the mesh are removed by placing them on the partition boundaries, surfaces for the obtained partitions can be generated without parameterization step. Figure 1.3 (b) and (c) shows two kinds of quadrilateral partitioning without and with T-joints. In the former type, each partition has always four edges and neighbor partitions. In the latter one, a partition may have more than four neighbor partitions so that T-joints can be seen on the partition edges. The latter one offer much more flexibility than the former one since T-joints are allowed [39] and therefore a few quadrilateral partitions can represent the shape.

1.1.5 Thesis Motivation

Each reverse engineering application has its own specific purpose and therefore various partitioning algorithms are utilized. In this dissertation, our aim is to partition a given quadrilateral mesh into groups each of which are appropriate to fit by B-spline surfaces. Figure 1.5 shows this dissertation's motivation. The important requirements for reverse engineering can be listed as mesh partitioning for surface fitting, noise/error reduction and others. This dissertation focuses on the mesh partitioning requirement. There are many existing noise reduction techniques that reduces the noise or error in the mesh which can easily be applied to the input data if it has noise. Noise in a model also reduces during quadrangulation, and therefore the generated quadrilateral mesh

possesses less noise (see Figure 1.4 (a)). If the noise in the input model cannot be reduced using the existing techniques, the proposed methods in this dissertation may not work well.

Mesh partitioning for surface fitting prefers the following characteristics of partitioning:

R1. **No surface trimming:** As stated in previous section, quadrilateral partitions do not need surface trimming which is preferable in surface modeling applications. Performing trimming between surfaces may reduce surface qualities due to the errors coming from trimming computations.

R2. **Free of surface parameterization:** Surface parameterization involves high computational costs. As quadrilateral meshes intrinsically have parameter $u - v$ values, utilizing them eliminates parameterization step during surface fitting.

R3. **Feature curves on the partition boundaries:** *Feature* (said as *Feature curve* throughout the paper) is defined as a set of highly curved connected edges with an appropriate length (not short). Placing them to inner parts of partitions is problematic because they cannot be represented well with smooth surfaces. It is preferable to place these curves on the partition boundaries and these partitions are called as *feature-aware* partitions. And *feature-aware* partitioning of a mesh consists of such feature-aware partitions.

R4. **Other requirements:** There are other criteria (a small number of partitions, etc.) that can be taken into account during partitioning process. As many of these criteria has been considered by existing partitioning methods, they are not focus of this dissertation. Inclusion of these criteria to the proposed approaches can be possible, but they can make the partitioning problem of the dissertation more complex.

Recent quad-meshing techniques can generate *semi-regular quadrilateral meshes* where there are only a few extraordinary vertices. This is advantageous because such a few vertices will generate a small number of quadrilateral partitions after partitioning. The mixed-integer quadrangulation (MIQ) [9] method of Bommes et al. generates such quadrilateral mesh from a given triangular mesh. We utilize semi-regular quadrilateral meshes generated using MIQ because extraordinary vertices are generally placed on feature points (highly curved vertices) and $u-v$ curves are aligned along the feature curves in the model. Utilizing the mesh of MIQ can enable a partitioning that can satisfy above-mentioned three requirements.

Many works proposed for quadrilateral partitioning take the requirements R1 and R2 into account, however they do not consider the requirement R3. The research in this dissertation focuses on these three requirements, particularly on R3. Feature curves in the model may not be aligned

with $u - v$ directions of quadrilateral mesh or they are aligned. The bi-monotone approach is proposed for the former one and it splits motorcycle graph partitions to be bi-monotone. Eppstein et al. defines motorcycle graph (MCG) concept in the paper [19]. The Path Flipping and MCG enumeration approaches are based on the motorcycle graph and they are proposed for the models whose feature curves are aligned with $u - v$ directions. These two approaches find quadrilateral partitioning (motorcycle graph) that is appropriate for reverse engineering.

Brief Summary of Closely Related Works with the Proposed Methods Figure 1.4 (b) shows summary of related works. Eck et al. [17] generates quadrilateral partitions from a given quadrilateral mesh, however feature curves of the model are located inner part of the partitions unlike that of this dissertation. There are many quad-meshing techniques; one of them is mixed-integer quadrangulation of Bommes et al [9] which is also the input for the proposed methods in the dissertation. The methods of Epsstein et al. [19] and Myles et al. [39] can effectively generate quadrilateral partitions (with T-joints) from a given mesh like the proposed methods (the bi-monotone, path flipping and MCG enumeration approaches). Comparison between these quadrilateral partitioning can be seen at the results section of chapter 3 and 4.

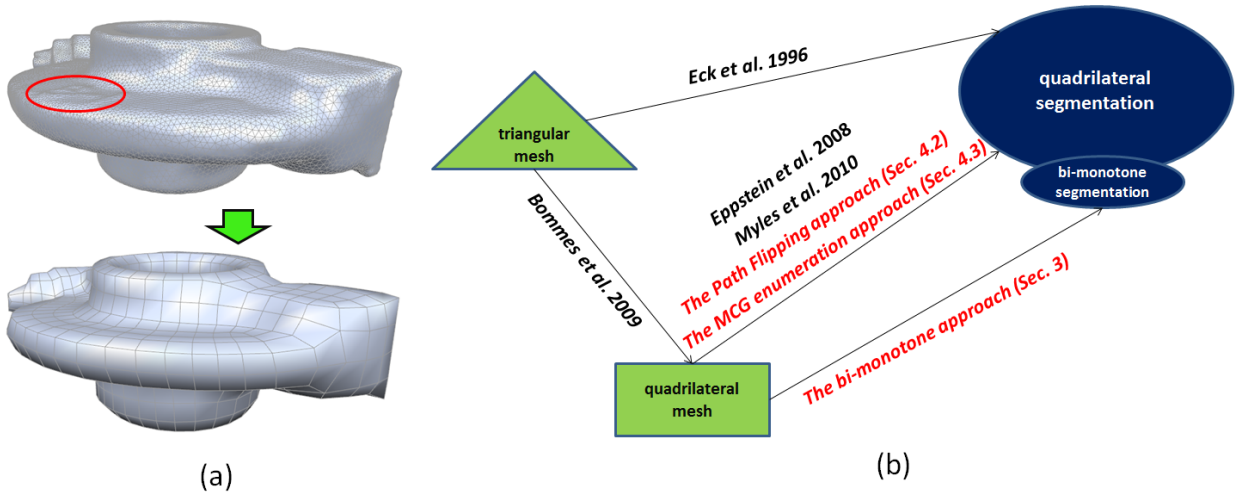


Figure 1.4: (a) Noise in a model (red part in top) reduces during quadrangulation, and therefore the generated quadrilateral mesh (bottom) possesses less noise. (b) Brief summary of related works.

1.2 Research Objectives and Proposed Approaches

Parametric surfaces are commonly used in computer graphics and computer-aided design to define curves and surfaces. B-spline surfaces are the most common type that is used in industry to represent free-form shapes such as ship hulls, car bodies, etc. This representation is preferable because

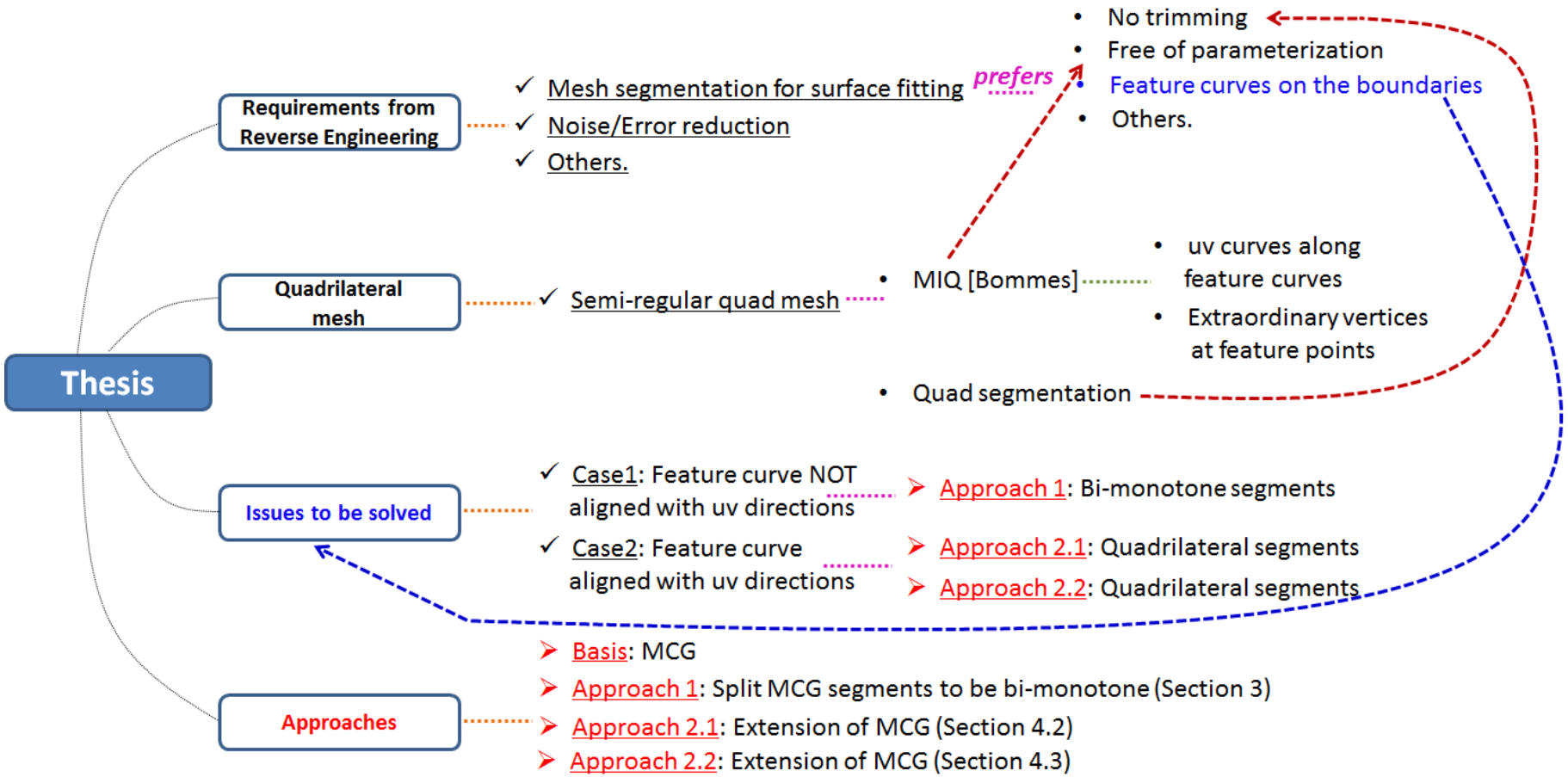


Figure 1.5: Thesis Motivation

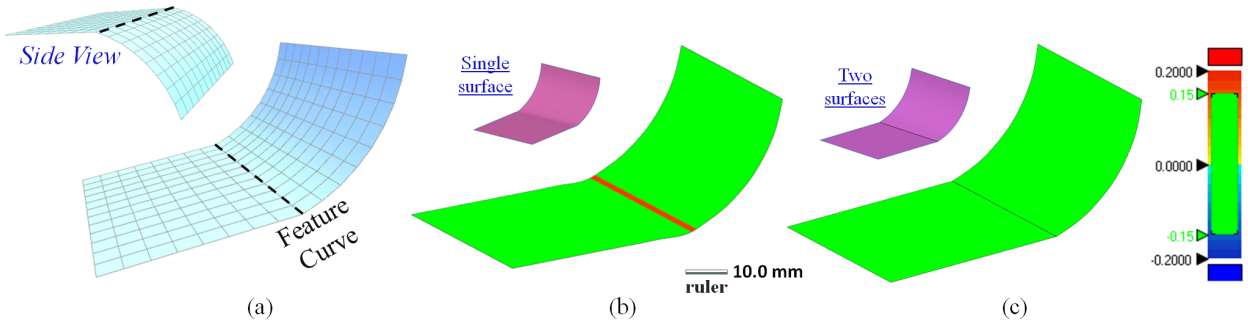


Figure 1.6: A shape having quadrilateral mesh data is fitted with B-spline surfaces. Once feature curve is located on the surface boundary, better quality B-spline surface can be obtained. In case of fitting with single surface, deviations from the original data occurs along the feature curve.

of its flexibility in design with the help of control parameters, evaluation easiness in CAE/FEM applications and many more reasons. However, feature curves where geometric discontinuities exist cannot be represented by smooth B-spline surfaces, therefore positioning these feature curves on the surface boundaries will increase the quality of surfaces generated. The research objective in this dissertation is to generate a partitioning from a given quadrilateral mesh which is appropriate for B-spline surface fitting. In order to generate high quality surfaces where each partition can be represented by smooth B-spline surfaces, feature curves of the model are located on the surface boundaries wherever possible. Figure 1.6 illustrates a shape where planar and cylindrical parts meet at a feature curve. A single surface fitted to this shape has deviations from the original data as seen on the red area. Utilizing two different B-spline surfaces where the feature curve is on their boundaries decreases this deviation. For this purpose, three following approaches (see Figure 1.7) have been proposed:

- **Approach 1** - The bi-monotone approach (Section 3): Quad partitioning approach [19, 39] has been used so far because of its shape simplicity and ease of trimming in surface modeling applications. However, determining feature curves within the boundaries with an appropriate number of partitions is not always possible with quad partitions. Therefore bi-monotone partitions that can capture feature curves in the model much more effectively are defined.

- **Approach 2.1** - The Path Flipping approach (Section 4.2): Motorcycle graph algorithm [19] can generate quad partitions and these partitions may not have a smooth geometry since the algorithm does not take the mesh geometry into account. Using same rules as in the original algorithm, highly curved regions in the model are located on the motorcycle graph edges as many as possible. By doing this, smooth geometry inside partitions can be obtained so that

B-spline surfaces fitted to these partitions have lower surface fitting error.

- **Approach 2.2** - The MCG enumeration approach: Huge amount of motorcycle graphs can be generated from a given quadrilateral mesh one of which is the best (optimum) one as it has lower cost than other graphs based on a defined cost function. Enumerating all motorcycle graphs of a quadrilateral mesh is possible however it has high computation cost. The problem is simplified by dividing it into several sub-problems (sub-meshes). In each sub-mesh, optimum motorcycle graph is found separately by enumerating all motorcycle graph combinations. Finally solutions in each sub-meshes are combined in order to obtain a quasi-optimum motorcycle graph in the whole mesh.

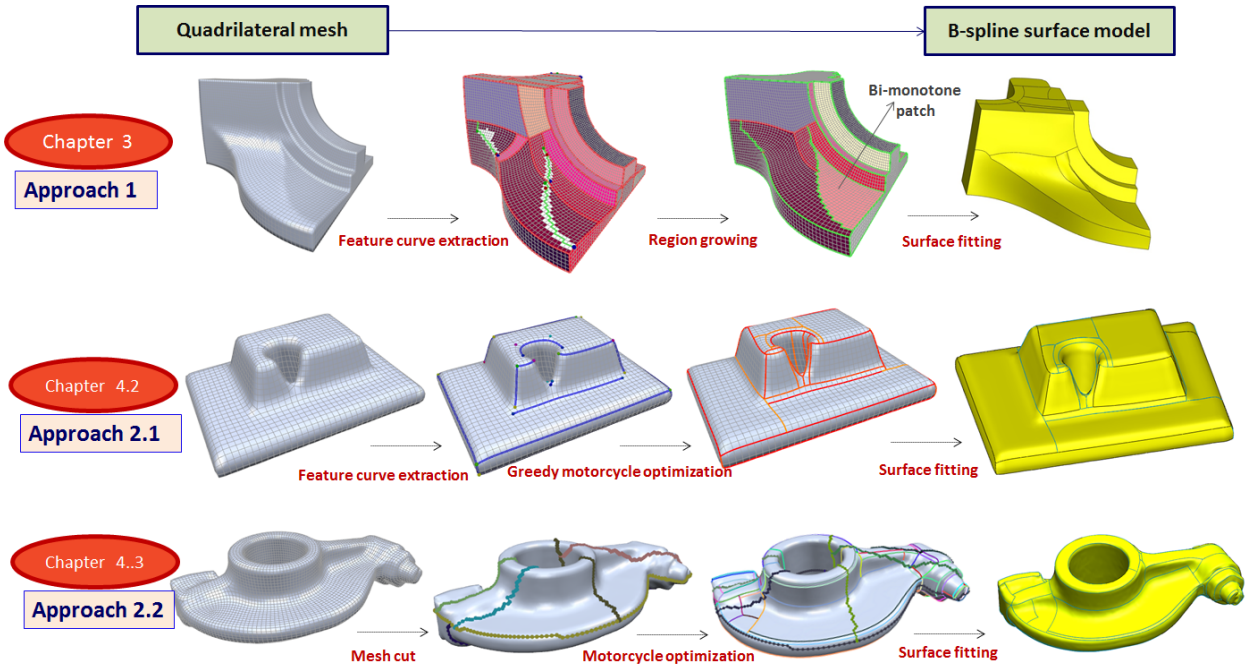


Figure 1.7: Proposed approaches and thesis outline.

1.2.1 Evaluation of Proposed Approaches

The proposed approaches are intended to generate partitioning appropriate for reverse engineering (B-spline surface fitting). Surface modeling problems are actually nontrivial because there are many criteria that should be taken into account to obtain a high quality surface model such as fitting error, $G0 - G1$ continuities on the boundaries, number of surfaces, design intent, etc. Considering all these criteria during partitioning will have a very high computational cost. Therefore, a simplified criterion is utilized which is intended to favor placing highly curved parts in the model

on surface boundaries. We believe that the proposed approaches can capture designer's intent to some degree because when modeling a part ab-initio, the geometric discontinuities are generally placed on the surface boundaries like the proposed approaches. We also propose algorithms for merging partitions to reduce their number in the post-processing steps. Evaluation of results are generally performed by visually checking the locations of highly curved parts (i.e., whether they are placed on partition boundaries or not) as well as checking deviations of the generated surfaces from the input mesh by computing surface fitting errors.

1.3 Thesis Outline

This dissertation includes five chapters. Related works are outlined in chapter 2. Chapter 3 then explains a method which detects feature curves and generates bi-monotone partitions. In chapter 4, two quadrilateral partitioning algorithms that extract and locate highly curved parts wherever possible are proposed. The Section 4.2 and 4.3 detail the Path Flipping algorithm and the MCG Enumeration algorithms. Conclusions and future directions of this dissertation will be introduced in chapter 5.

Chapter 2

Related Works

2.1 Quad Meshing

Many studies have explored methods of quad-remeshing and a number of significant surveys in the field have been performed. A brief review of these techniques is given here [3, 8]. Some early methods [2, 37] made use of model anisotropy for quadrangulation. In these approaches, the principal curvature direction fields of the model are used to guide tracing isocurves, from which quadrilateral mesh elements are extracted. These methods involve the generation of quad-dominant meshes (with some triangles) that include many extraordinary vertices. Parameterization-based methods [9, 27, 44, 40, 30] generate pure triangle-free quad meshes that fit the guiding fields as smoothly as possible. Unlike early methods [2, 37], the approach of Bommes et al. [9] utilizes only certain principal curvature directions, known as meaningful curvatures, to smoothly interpolate the cross fields. Huang et al. [25] find the optimum Morse-Smale complex, which is later used to generate a quadrilateral mesh. The method proposed in [13] involves the use of Catmull-Clark subdivision to initially generate a quad-only mesh, which is then simplified to create a base domain homeomorphic to the original mesh. Two recent methods [54, 29] also involve using model anisotropy in the same way as some early techniques [2, 37]. Zhang et al. [54] aim to generate anisotropically sized quads, unlike the approach outlined in [9], using a wave-based quadrangulation method by which quad size can be controlled. The recently proposed method of Panozzo et al. [41] takes generalized symmetries of the shape into account during quadrangulation. Zhang et al. [55] proposes a divide-and-conquer quadrangulation approach that uses the global structure information of the object such as symmetries, primitives, etc.

2.1.1 Mixed-integer Quadrangulation [9]

Bommes et al. proposes a novel method that quadrangulates a given triangular mesh. The output is optimized with respect to the following quality aspects: Quad faces should be close to square, quad edges should be aligned along the principal curvature directions and quad edges should be aligned with the sharp features in the model. Figure 2.1 shows the algorithm flow of the mixed-integer quadrangulation technique. As principal curvature directions are not possible to define everywhere (flat or spherical areas) in the mesh, only the directions in the highly-curved regions (see Figure 2.1 (a), (b)) are detected. Then cross fields are globally interpolated using a global optimization procedure. During this optimization, positions and numbers of extraordinary vertices are determined. Globally smooth parametrization which is oriented according to the cross fields is computed by cutting open the mesh. The parameter lines then forms the quad edges whose vertices are on the intersection of these lines. Both cross field and parametrization steps are formulated

as a mixed-integer problem which is solved by an adaptive greedy solver. The resolution (coarse or fine) of generated mesh can be manually set using target edge length parameter. In contrast to previous methods, mixed-integer quadrangulation generates more high quality quad meshes. Some details about the proposed quadrangulation method are listed in the below.

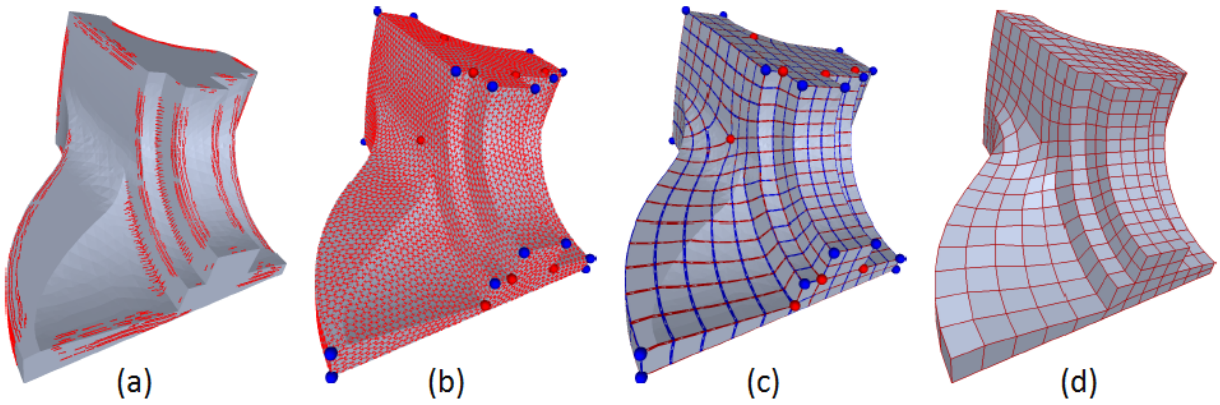


Figure 2.1: Mixed-integer quadrangulation [9]: Starting from a sparse set of orientation and alignment constraints (a), cross fields are smoothly interpolated which are used in the parametrization step to generate iso-parameter lines. Finally quad mesh is extracted using these parameter lines (image taken from [9]).

Interpolating Cross Fields: An angle-field $\theta : F \rightarrow R$ is defined on a triangle mesh M by assigning a real number to each face ($\subset F$) and a period-jump field $p : E \rightarrow Z$ assigning an integer to each edge ($\subset E$). The angles θ is used to determine a single unit length vector-field and it then extends to a symmetric cross field by applying three rotations $\pi/2$ (see Figure 2.2 (a)). The smoothness of a unit vector-field $E_{smooth} = \sum_{e_{ij} \in E} (\theta_i - \theta_j)^2$ can be computed simply the sum of all squared angle differences between neighboring triangles where e_{ij} is the edge between the faces i and j. E_{smooth} is minimized so that p_{ij} per edge and θ_i per face are found. Left-top of Figure 2.2 (b) shows a smooth cross field in the vicinity of a cube corner, where the red arrows reflect the corresponding period jumps. And left image shows a triangle whose cross field directions u_T (θ), v_T ($\theta + \pi/2$) are extracted with the angle θ w.r.t. the local reference direction (green) and the linear scalar functions u, v (used for the parametrization) are found whose gradients are oriented consistently with the cross field directions. At the end of this interpolation step, positions of extraordinary vertices are determined (see red/blue spheres on the right image of Figure 2.2 (b)). Note that noise in a given triangular mesh reduces during mixed-integer quadrangulation because cross fields are smoothly interpolated as explained in the above.

Global Parametrization: A global parametrization is computed mapping from a given mesh

to some disk-shape parameter domain. A u, v value is assigned to each vertex of the mesh. The following energy is minimized to find a smooth global parametrization: $E_T = (\|h\nabla u - u_T\|)^2 + (\|h\nabla v - v_T\|)^2$ where h is a global scaling parameter representing the edge target length of the resulting quad mesh. u_T and v_T are the cross fields that are interpolated using selected user-defined constraints. To compute a proper parametrization minimizing the E_T, the mesh is cut open and topologically disk-shape is obtained. The right image of Figure 2.2 (c) shows the model after cut open and finally obtained quadrilateral mesh is also shown (left).

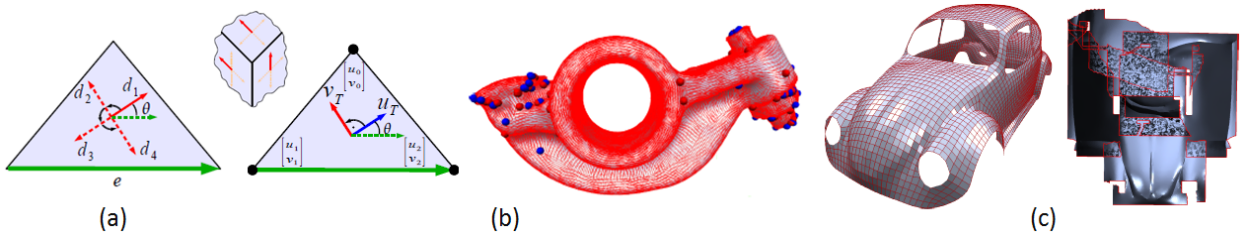


Figure 2.2: (a, b) Interpolating cross fields (c) Global parametrization (images taken from [9]).

2.2 Mesh Segmentation

Mesh segmentation techniques have a long tradition in the graphics community and a considerable body of literature surveys [1, 46] on the topic exists. Some methods [32, 36, 52] involve utilizing surface curvature characteristics to segment a mesh into sub-meshes. [36] applies morphological watersheds (an image segmentation technique) to 3D surfaces, and [52] partitions the mesh while iterating between region growing and surface fitting. Other methods [20, 5, 12, 53] involve using a geometric measure or filter for clustering or approximating mesh elements. [20] proposes a hierarchical clustering methodology utilizing edge contraction on a dual graph of the mesh, while [5] separates the mesh into sub-meshes, each approximated by geometrical primitives. [53] extends the optimization technique of geometric surface approximation [12] by allowing spheres, cylinders and rolling-ball blend patches in addition to planes.

A limited amount of work has been done on direct segmentation of quadrilateral meshes. Eppstein et al. [19] propose a motorcycle graph algorithm to segment a given quadrilateral mesh into regular sub-meshes (i.e., without extraordinary vertices). Myles et al. [39] utilize local operators to optimize the layout obtained after initial quad patch layout construction. A greedy optimization is used to generate a T-mesh patch layout appropriate for T-spline fitting. There are also other works [23, 43, 12, 35] exploring methods for the generation of high-order surfaces.

2.2.1 Motorcycle Graph Algorithm [19]

Motorcycle graph algorithm of Eppstein et al. [19] segments a given quadrilateral mesh into sub-meshes where there is no extraordinary vertices. The algorithm uses a very simple strategy to generate quad partitions (with T-joints) from the mesh. As recent quad-meshing techniques [9, 8] generate high-quality quadrilateral meshes, using this algorithm for such partitioning is much more useful with such high-quality meshes and therefore, high quality segmentation can be achieved.

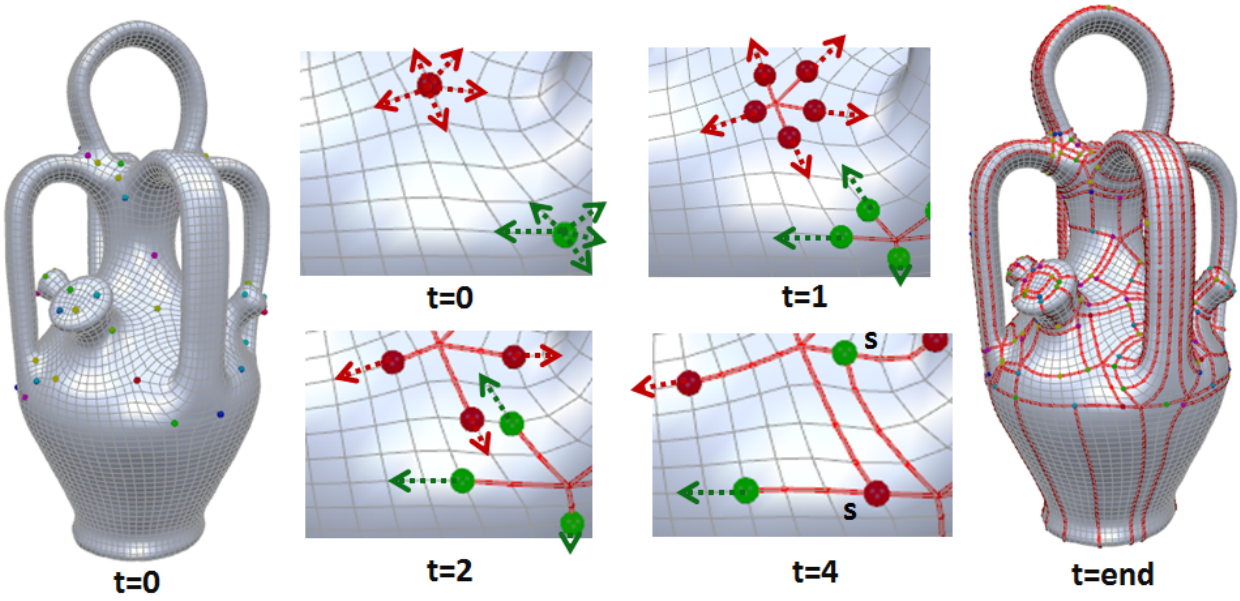


Figure 2.3: Motorcycle graph [19] segmentation: Particles are placed at extraordinary vertices and they start moving the outward direction with the same speed. When particles meet others' tracks or the mesh boundary, they stop. As a result of these traced tracks quad partitions are formed.

Figure 2.3 illustrates the motorcycle graph algorithm. Particles are first placed on extraordinary vertices. The number of particles placed at each one is equal to the valence of that vertex. The same speed is given to all particles, and tracing starts. A particle stops when it meets an interior vertex of another's path, its own path, or a boundary vertex. If two particles meet perpendicularly at a vertex, the right-hand rule is used; one stops and the other keeps going. If two particles moving in opposite directions meet, they also stop. If three or four particles meet at a vertex simultaneously, they all stop. Eventually, the tracks of the particles form partition boundaries and quad partitions are generated.

2.2.2 Feature-aligned T-meshes [39]

This study aims to construct a patch layout which is suitable to fit by T-splines [45] which has advantages over B-spline surfaces. The proposed algorithm generates patches while satisfying some quality requirements. Well-shaped patches that have minimal skew and bounded aspect ratio are desirable. Patches should also be well approximated with T-splines after surface fitting. The number of patches should be as few as possible. And finally, should be aligned with principal curvature directions and highly curved regions in the mesh. After obtaining a initial patch layout (see left image in Figure 2.4 (a)), it is improved using mesh modification operators (see right image in Figure 2.4 (a)). Operations is performed only if the defined cost decreases which is based on the above-mentioned patch quality requirements. Figure 2.4 (b) shows three modification operators that modifies the patch layout in a greedy way. Refinement decreases the patch size and increase the the number of patches. After extension process, size of one patch increases, while other patches shrink or sometimes they are eliminated. Edge set moves in a direction by relocation operator. Even though the results obtained by this algorithm is not guaranteed to be the global optimum, significant reduction in cost is achieved. And final layout is fitted by T-splines.

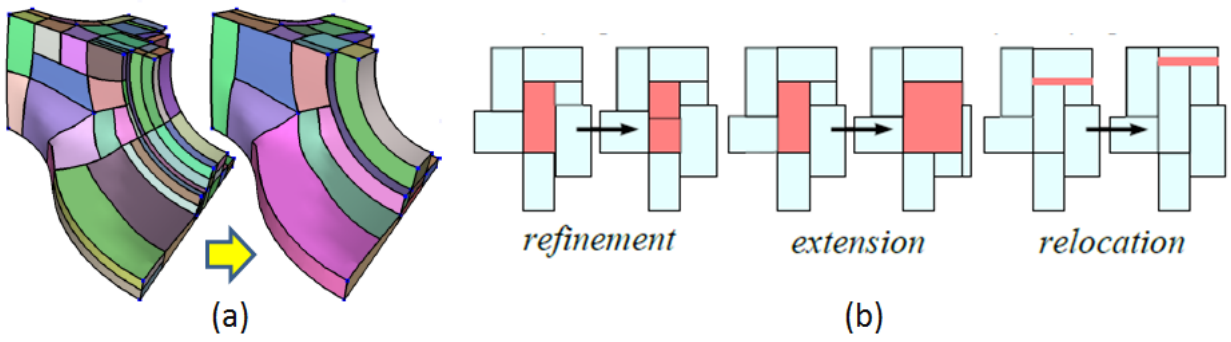


Figure 2.4: Feature-aligned t-meshes [39]: The initial patch layout is improved using three layout modification operators. Modification is done if the patch-based defined cost decreases. Final layout is suitable to fit by T-splines (image taken from [39]).

2.3 Structure Optimization of Quadrilateral Meshes

Placing an appropriate number of extraordinary vertices in suitable positions (generally regions with high Gaussian curvature) is crucial for obtaining high-quality quadrilateral meshes. A quadrilateral mesh yielding a simple base complex (sequences of straight quadrilateral edges starting and ending at extraordinary vertices) is also desirable. In this regard, several mesh simplification

methods [14, 7, 48, 42, 49, 39] have been proposed recently. Myles et al. [39] detail a singularity alignment problem for which misaligned pairs of extraordinary vertices are detected and then aligned on the parametric domain. Bommes et al. [7] propose an algorithm to detect helices and remove them using local connectivity operators that do not change the number or position of singularities. The quadrilateral mesh in thus generated implies a simpler base complex. Campen et al. [11] recently outlined a novel algorithm that first builds a dual layout from curvature-guided crossing loops on the surface. Then quad mesh with a high-level patch structure is generated using this layout.

2.3.1 Global Structure Optimization of Quadrilateral Meshes [7]

Many quad-meshing techniques (such as [9]) generates high quality meshes in terms of good orientation and alignment of mesh elements by placing irregular vertices generally on highly curved regions without consideration of the positions of other irregular vertices. This study strives for the generation of a quadrilateral mesh which yields simple patch layout where helical configurations do not exist (see blue and orange line-sets of left image in Figure 2.5 (a)) and new irregular vertices are not introduced. The algorithm first detects these helical structures and removes them by applying three simplification operators as seen in Figure 2.5 (b). In a shift left step (red arrow), a triangle and a pentagon are generated by releasing the vertex on the left side of the dual half-edge and shifting it towards the next vertex. The shift right (green) is similar to the shift left. A collapse step (yellow) collapses the edge is into a single vertex. A valid set of simplification operators removes helices (see middle image of Figure 2.5 (a)). By doing this, underlying patch layout of the mesh is simplified such that the number of patches drastically decreases (see right image in Figure 2.5 (a))).

The proposed approaches in this dissertation are not intended to change the global structures of quadrilateral meshes, extracted feature curves are placed on the quadrilateral partition boundaries. In order to support the extracted features, the method [7] probably needs more extraordinary vertices which should be placed on appropriate positions to capture these features. This is problematic to have such a large number of extraordinary vertices because they produce a more complex base complex and thereby the number of quadrilateral patches in the base complex would be also very high. Therefore, the proposed approaches can more effectively place highly curved regions on the partition boundaries compared to the recent works [7, 11, 49]. Recall that T-junctions do not exist in the partitions of [7, 11, 49], whereas they exist in those of the proposed approaches.

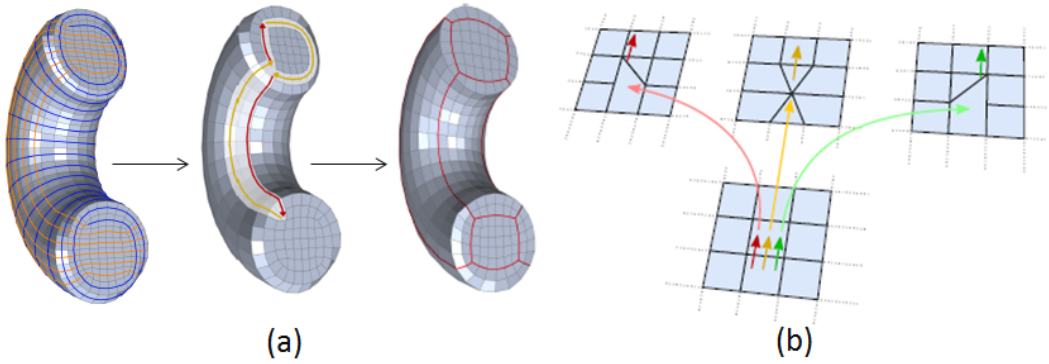


Figure 2.5: Global structure optimization of quadrilateral meshes [7]: Patch layout of a given quadrilateral mesh is simplified (b) using three simplification operators (image taken from [7]).

Chapter 3

The Bi-monotone Approach

3.1 Introduction

The ability to capture the surface details of mechanical CAD objects accurately and robustly is still a major issue in reverse engineering. Compared to using general meshes, the adoption of parametric surfaces to represent objects provides a much more efficient and compact solution. Although many studies have focused on the use of triangular meshes to generate surface models, little work with quadrilateral meshes as input has been performed even though their utilization simplifies the conversion process. The elements of quadrilateral meshes are well arranged in such a way that they capture the principal curvature directional fields in a natural way, unlike triangular meshes. Moreover, quadrilateral meshes intrinsically have surface parameter values, enabling surface model generation with no parameterization step.

Using quad partitions to represent an object is a popular approach today, because of their appealing properties (e.g., shape simplicity and ease of trimming) in surface modeling applications. The basic concept involves partitioning a quadrilateral mesh by removing irregular vertices, whose valences are fewer or more than four valences, so that all partitions are four-sided and form a regular quadrilateral grid structure. Such partitioning is beneficial for surface fitting because parameterization can be achieved efficiently using a regular grid structure, and because there is no need for surface trimming. However, it is not always possible to determine the feature curves of objects within the boundaries formed by such partitioning which leads to quality problems with the generated surfaces. To capture all feature curves with an appropriate number of segments (see Figure 3.1 (a)), it may be necessary to allow partitions with arbitrary shapes. However, this is problematic because it results in an increased number of trimmed curves.

We take the concept in between quad partitions and partitions with arbitrary shapes which is called as *bi-monotone* partitions. *Monotone* is a concept defined on a regular mesh of quadrilaterals. If the number of quadrilateral segments (i.e., continuous sequences of quadrilaterals) is one for every column or row, the mesh is called *monotone*. A mesh that is monotone in both the column and the row directions is referred to as bi-monotone (see Fig. 3.1 (b, c)).

Our target is to produce good segmentation for fitting surfaces in the post processing step. In practice, the data points are preferred to have as uniform gaps as possible to each other so as to generate a better surface. In a bi-monotone partition, there are no holes or large inward curved regions and data points corresponding to an iso-parametric curve are always in one segment. In other words, there is no big gap in the intervals of the data points. Therefore, it is expected to be able to determine a good parameterization of the data points in the bi-monotone partition.

This study aims to generate large-size bi-monotone partitions where the feature curves reside

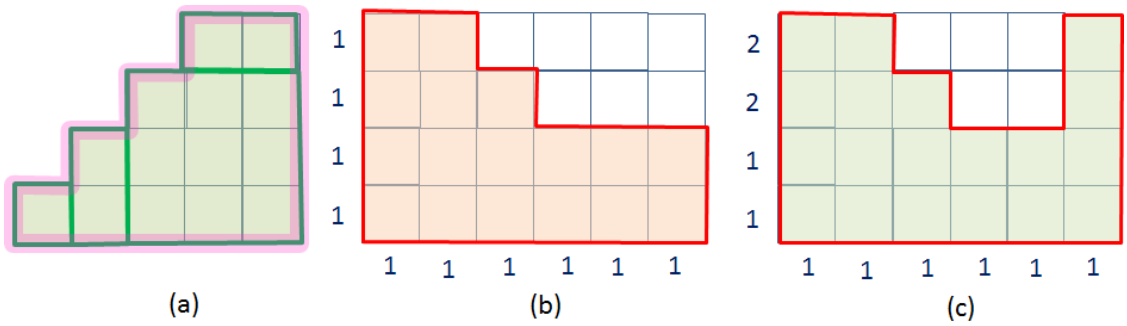


Figure 3.1: (a) Four quad partitions (in green) capture the shape, however same shape can be captured by a single arbitrary-shaped partition (in pink). (b) Bi-monotone partition, (c) Non-monotone partition (Numbers in blue shows the number of quadrilateral segments for rows or columns).

on the boundary. We first outline the partitioning a quadrilateral mesh into four-sided partitions, then describe the detection of feature curves inside these partitions. If a partition has feature curves, we split it into bi-monotone partitions along these curves. Finally, partition configuration is improved to generate large partitions. As it is undesirable to obtain partitions whose shapes are far away from quadrilaterals, we control the shape using a *shape control* parameter that can be adjusted by the user during the partition generation process. In this study, we mainly target shape models of mechanical parts which consisting of major smooth surfaces with feature curves between them.

3.2 Method Overview

Figure 3.2 illustrates the main steps of the proposed algorithm. We adopt a quadrilateral mesh generated using a mixed integer quadrangulation algorithm of Bommes et al. [9] to generate large-size bi-monotone partitions. Section 3 describes an algorithm which partitions the mesh into quadrilateral partitions. Then a feature detection algorithm will be explained in Section 4. Section 5 and 6 details the generation of the bi-monotone partitions.

3.3 Extended Motorcycle Graph Algorithm

The first part of the proposed mesh decomposition involves tracing the edges of a given quadrilateral mesh using the motorcycle graph algorithm proposed by [19]. The mesh is then partitioned into several sub-meshes with boundaries formed by the motorcycle graph edges. In this step, the

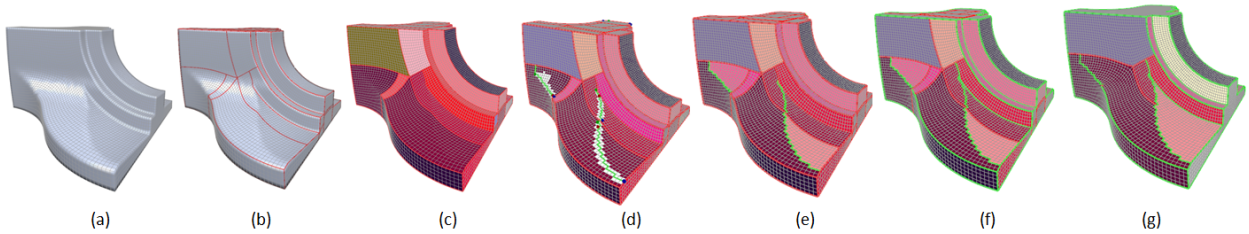


Figure 3.2: Flow of the proposed algorithm. (a) A quadrilateral mesh model generated using the mixed integer quadrangulation technique of Bommes et al. [9]. (b) Mesh is partitioned into sub-meshes using an extended version of the motorcycle graph (EMG) algorithm [19]. (c) partitions are generated inside the EMG edges (shown in red). (d) Surface features inside the partitions (edges shown in green) are extracted. (e) Bi-monotone regions are generated inside the partitions via a quadrivial region growing (QRG) process. (f) The EMG edges are screened, and some are tagged as surface features (edges shown in green). (g) Planar partitions (shown in gray) are merged, and the partition configuration is improved to produce large bi-monotone partitions via a partition configuration improvement (PCI) process.

extraordinary vertices (valence≠4) are removed and located at the corners of these sub-meshes.

The motorcycle graph algorithm decomposes a quadrilateral mesh into several sub-meshes without extraordinary vertices; these sub-meshes are referred to as *structured meshes*. However, the resulting sub-meshes are not necessarily flat; they may have non-flat faces because the motorcycle graph algorithm does not take mesh geometry into account. In this chapter, we propose an extended motorcycle graph (EMG) algorithm and promote the generation of sub-meshes with fewer non-flat faces.

The aim of the study outline in [19] is to decompose semi-regular quadrilateral meshes into structured quadrilateral sub-meshes based on the motorcycle graph concept of [18]. To achieve this, particles are placed on each extraordinary vertex and moved outward from the extraordinary vertices along one edge in each time step. When a particle reaches an ordinary vertex, it continues to move along its opposite edge. If a particle meets a boundary or a previously traversed vertex, it stops. When two particles meet at a vertex perpendicularly, the *right-hand rule* is used and one stops, while the other keeps going. Moreover, if three or four particles meet at a vertex simultaneously, they all stop.

The concept of the extended motorcycle graph algorithm is the same as that of the original algorithm [19], except the EMG algorithm checks the angle of two consecutive edges on the path during tracing. If it meets an angle α whose value (in degrees) exceeds the user-defined threshold T_{ψ}, it stops tracing and branches into right and left paths (see Figure 3.3). In this way, paths can

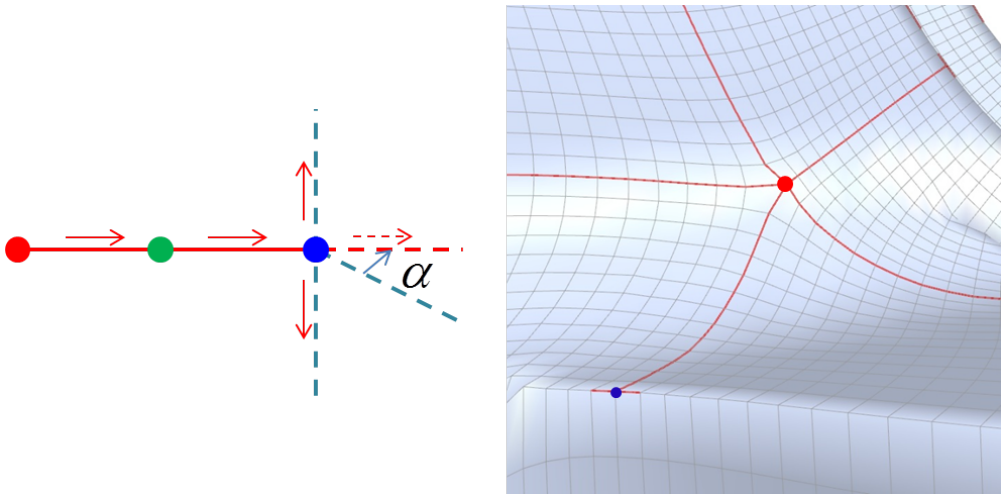


Figure 3.3: The EMG (extended motorcycle graph) algorithm. Particles move outward from an extraordinary vertex (shown in red). When a particle meets a crease with a large angle between its consecutive edges at a vertex (shown in blue), its direction of movement changes as it branches into the left and right paths.

go through highly curved features.

All sub-meshes generated by the original motorcycle graph algorithm are structured. First, all extraordinary vertices of the original mesh are split into corner vertices of sub-meshes, each of which is incident to two edges and therefore ordinary. The algorithm then does not generate any extraordinary vertices (incident with four edges) on the traced boundary of the sub-meshes. As the proposed EMG algorithm clearly provides these characteristics, it is guaranteed that the generated sub-meshes will be structured. A proof of the proposed algorithm is given below.

Theorem: If a quadrilateral mesh homeomorphic to an orientable two-manifold with or without a boundary is not a structured torus, the proposed method will cut the mesh into structured disks or structured annuli.

Proof: The proposed method cuts a mesh into sub-meshes without extraordinary vertices - that is, each sub-mesh is structured. As shown in [19], there are only three types of (orientable) structured meshes: structured disks, structured annuli and structured tori. Furthermore, based on the assumption that the input is not a structured torus, each sub-mesh generated using the proposed method has a boundary and so is not a structured torus.

We refer to a sub-mesh inside the EMG edges as a *partition*, and each partition has a regular quadrilateral mesh structure. As the final step, partitions with annulus topology are partitioned to create two partitions with disk-shaped topology to facilitate the operation of the next processes.

Figure 3.4 shows the results of motorcycle and extended motorcycle graph algorithms for

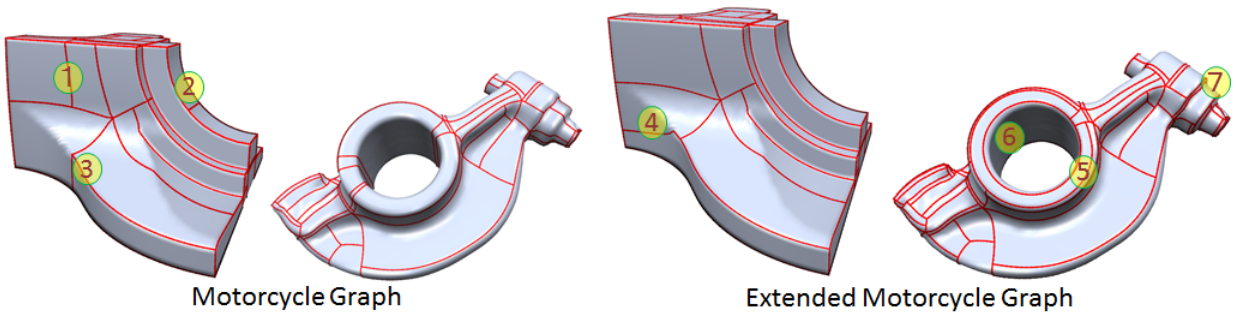


Figure 3.4: Comparison of motorcycle and extended motorcycle graph algorithms

quadrilateral mesh models of a fandisk and a rockerarm. The red lines show the motorcycle edges, and improvements are marked with numbers from 1 to 7. The motorcycle graph algorithm produces over-segmented partitions (1, 2), which is not desirable. Conversely, the extended motorcycle graph algorithm not only avoids over-splitting but also easily captures geometric details such as planar (5) and cylindrical (6, 7) parts. It also avoids sharp changes during tracing (3) and continues tracing in the cross direction (4).

3.4 Feature Detection

To generate a high-quality surface model from an object, it is crucial to capture geometric discontinuities (i.e., normal, curvature) on its mesh model. Surface features consist of feature vertices that violate surface continuity and have high curvature values. As the EMG process does not necessarily capture such features, some may reside inside the generated partitions. In particular, the sharp features inside the partitions cannot be represented by smooth B-spline surfaces.

This section details the extraction of surface features constrained to be boundaries of partitions generated after the QRG and the PCI process. As these extracted features directly affect the quality of the finally generated surface model, they should have the following characteristics:

- **Shape:** Their shape must be simple so that the surface can be split into several well-shaped sub-partitions. It must also be elongated through the surface.
- **Position:** They must go through highly curved partitions to enable splitting into sub-partitions with *low curvature*.

To extract such features, we first detect regions that intrinsically hide them and call them *feature-regions*. The features residing in these areas are then extracted.

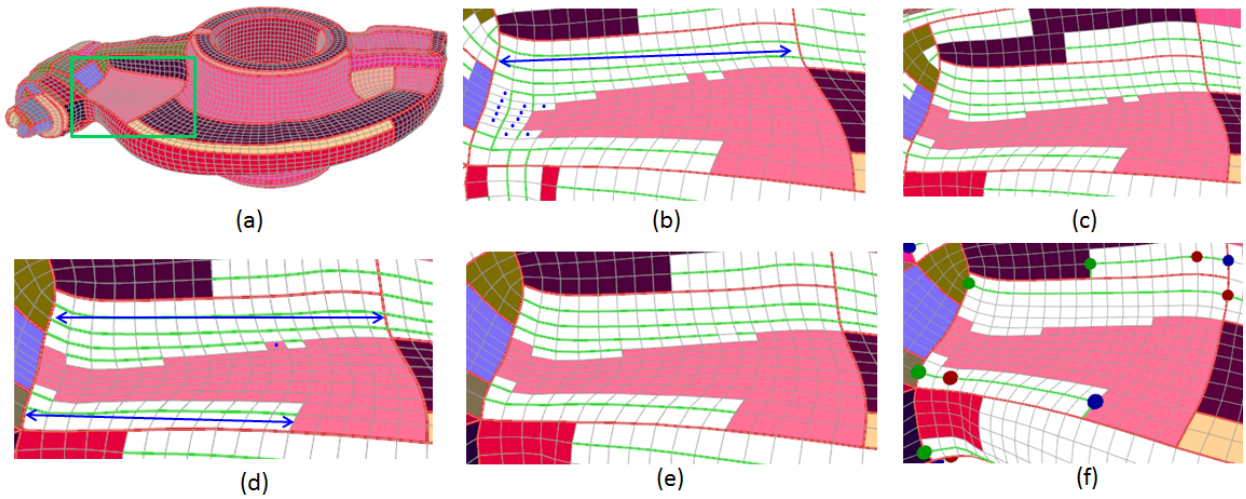


Figure 3.5: Feature detection algorithm. (a) Rockerarm model (regions inside the EMG edges shown in the same color). The next images show close-ups of the partition enclosed in the green rectangle. Red curves are partition boundaries. (b) Detection of non-flat regions (non-flat faces are shown in white, and *non-flat edge*s are shown in green). There is one non-flat region in the partition; its elongation direction is described by the blue arrow. (c) Discarding of non-flat faces with no non-flat edges in the feature-elongation direction. These are marked with blue dots in (b). After deletion, the non-flat region is split into two. (d) Detection of feature regions by checking their size in the feature elongation directions shown by the blue arrows. (e) Generation of bi-monotone quasi-hulls. The face marked with the blue dot in (d) is made non-flat in (e). (f) Extraction of feature curves.

3.4.1 *Feature region* detection

Let Q be a partition obtained via the EMG process. Each quadrilateral face $q \in Q$ is connected to its eight surrounding faces in an *8-connectivity* manner. The *8-neighborhood* $N(q)$ of q is defined as a set of these faces and q itself. For the sake of simplicity, here we ignore faces on the boundary of Q in the following discussion. In Q, an *8-path* between faces q_1 and q_m is defined as a sequence of faces from q_1 to q_m, that is, $< q_1, q_2, \cdots, q_m >$, where no faces are repeated, and q_i and q_{i+1} are *8-connected* for all instances of $i \in [1, m-1]$. A *connected component* is a subset of Q such that there is an *8-path* between every pair of its faces. Here we use the term *region* to represent connected components in partitions.

To compute the *non-flatness* F_i of a face q_i, the formula $F_i = \max_{q_j \in N(q_i)} \|n_i - n_j\|$ is used where n_i is the unit normal vector of q_i. To compute n_i, we cut the quadrilateral face along one of its two diagonals to obtain two triangles, and the average of these triangles' normals is assigned as the

normal vector n_i. Similarly, other criteria such as *developable charts* [26] can also be applied to detect the unflat regions.

The term *non-flat face* is used to describe a face whose non-flatness value exceeds the user-defined non-flatness threshold T_Γ. Otherwise the face is deemed to be flat. The term *non-flat region* is also introduced to describe a region consisting of only non-flat faces. The boundary of a non-flat region is a set of faces that have flat faces in its 8-neighborhood. Accordingly, there are flat to non-flat 8-connections on such boundaries.

On the partition Q, a face consists of four edges. In the inner part of the partition, an edge is shared by two incident faces, and four edges meet at a vertex. We use the term *non-flat edge* to describe an edge whose corresponding adjacent faces' normal difference exceeds the non-flatness threshold T_Γ.

A quadrilateral mesh may have many non-flat regions scattered all over it. However, as not all of these regions will correspond to surface features themselves, it is necessary to detect the non-flat regions that shows signs of surface features. These are referred to as *feature-regions*.

Figure 3.5 illustrates the proposed feature region detection algorithm. We first detect non-flat regions by looking for non-flat faces connected to each other using the depth-first search technique. In Fig. 3.5 (b), only one non-flat region is detected in the partition. The maximum number of non-flat faces (shown in white) in each u, v direction of the rectangular grid structure is then computed. The higher number is regarded as the length of the non-flat region, and the smaller one is taken as the width. We call the direction of the length in which the feature is expected to be aligned the *feature elongation* direction. The non-flat faces of non-flat regions with no non-flat edges (shown in green) in the feature elongation direction are then discarded. These faces are marked with blue dots in Fig. 3.5 (b), and are deleted in (c).

As the non-flat faces of non-flat regions may be disconnected due to the elimination of non-flat faces, it is necessary to check whether regions need to be split. Thus, some non-flat regions are split into two or more sub-non-flat regions at the end of this process (see Fig. 3.5 (c)). Next, we eliminate isolated non-flat regions which have short lengths in the feature elongation direction. To do this, the number of faces in the feature elongation direction is calculated. If the total exceeds the user-defined integer threshold T_φ, the non-flat region is called as *feature region*. Otherwise, it is no longer considered a non-flat region (see Fig. 3.5 (d)). If a feature region encircles flat faces (holes) in the partition, the shape of the eventually generated surface feature may not be simple or bi-monotone. In order to avoid such cases, holes are filled by expanding the region to form a *bi-monotone quasi-hull*, which is a bi-monotone region enclosing the non-flat region with the smallest area and perimeter. In Fig. 3.5, the face marked with the blue dot in (d) is made non-flat

in (e) after a bi-monotone quasi-hull has generated for it.

3.4.2 Feature curve detection

Feature regions contain many non-flat faces that hide surface features. We split a feature region in its feature elongation direction along a *feature curve*. The proposed feature curve detection method involves the use of a greedy algorithm similar to the single-source Dijkstra's algorithm. A source vertex s is selected as one of the vertices of the edge with the maximum dihedral angle in the feature region R. A path consisting of a sequence of connected edges is first found in R from s to the farthest vertices in one elongation direction (the *target* vertices) with the minimum cost. Another path is then generated in the other direction in the same way. The feature curve is given by the union of these two paths.

A cost function $f(C_k)$ is defined for the path $C_k =< e_1, e_2, \ldots, e_k >$, where s is incident to e_1 as follows:

$$f(C_k) = w(C_k) + b(C_k) + d(C_k)$$

This function is designed for the path to be bi-monotone. If C_k is not bi-monotone, $w(C_k)$ has a value of 1. Otherwise, the value is 0. $b(C_k)$ is the number of such vertices that the path violates the bi-monotone condition specified above by making turns at these vertices. It is also desirable to generate a path corresponding to the high-curvature zones of the model. The dihedral angle of the edge e_k is maximized during tracing. $d(C_k)$ is determined based on the average θ of the unsigned dihedral angles of all edges of C_k. If the average θ exceeds $\frac{\pi}{2}$, then $d(C_k)$ is 0. Otherwise, it is $1 - \theta/\frac{\pi}{2}$.

Although no theoretical proof is given here for the termination of the proposed algorithm, experiments showed that termination occurs when the two target vertices are reached starting from a source vertex. Dijkstra's algorithm is guaranteed to terminate after finding the shortest path from a vertex to all other vertices if a graph does not contain negative-cost edges. The proposed algorithm is different from Dijkstra's in that we dynamically quote graph edges with a cost $f(C_k)$, which can be either 0 or positive.

Fig. 3.6 (a) depicts the feature curve detection algorithm. Tracing starts from a vertex (the red dot) incident to the edge with the maximum dihedral angle in the feature region. Two shortest paths are found from the source vertex to one of the leftmost vertices (the green dot) and one of the rightmost vertices (the blue dot) in the feature region, and are taken as feature-curves (the green edges). As a post-processing step, these two curves are extended to the closest EMG edges without violating the bi-monotone condition. The eventually generated feature curve is a union of these

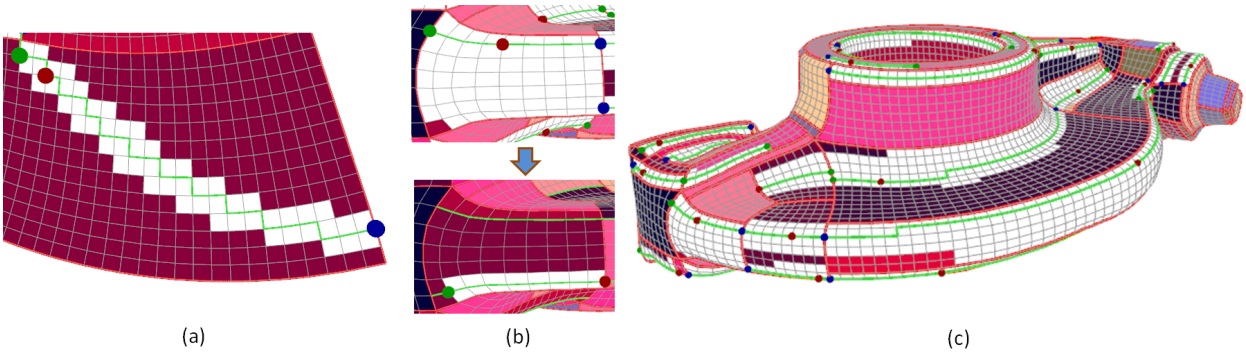


Figure 3.6: (a) Feature curve detection using Dijkstra's algorithm. Two shortest paths (feature curves shown in green) are found from the source vertex (the red dot) to one of the leftmost vertices (the green dot) and one of the rightmost vertices (the blue dot) in the feature region. (b) Extraction of two feature curves from a feature region (upper image: after the first run of the algorithm; lower image: after the second run of the algorithm). (c) Rockerarm model after the first run of the feature detection algorithm.

two curves and their extensions. Figure 3.6 (c) presents several feature curves of the rockerarm model (also see 3.5 (f)) after the feature detection process.

To prevent the generation of many feature curves close to each other, we tag the parallel edges (residing inside the partition) of the feature curves as *feature neighbors*, and no tagged edge can be an element of a feature curve. We adopt the user-defined integer threshold T_ρ for this. Figure 3.6 (b) shows a feature region hiding two feature curves. With the threshold T_ρ set to four, the upper feature curve is extracted and its parallel edges in the neighborhood are tagged as feature neighbors. Then in the second run, the feature curve in the lower image is extracted.

3.5 The Quadrivial Region Growing (QRG) Process

In this step, we grow bi-monotone partitions between the generated feature curves and the EMG edges. A *shape control* parameter that can be adjusted by the user is also provided to support the creation of partitions with better shapes. Readers should refer to Figure 3.7 for the illustration of the QRG algorithm.

After the EMG and feature detection processes, the quadrilateral mesh M is decomposed into several partitions. Each partition Q can be represented as a grid array of quadrilateral faces $Q = \{q[i, j], i = 1, 2, \cdots, H, j = 1, 2, \cdots, W\}$ where H and W are the numbers of columns and rows in the array.

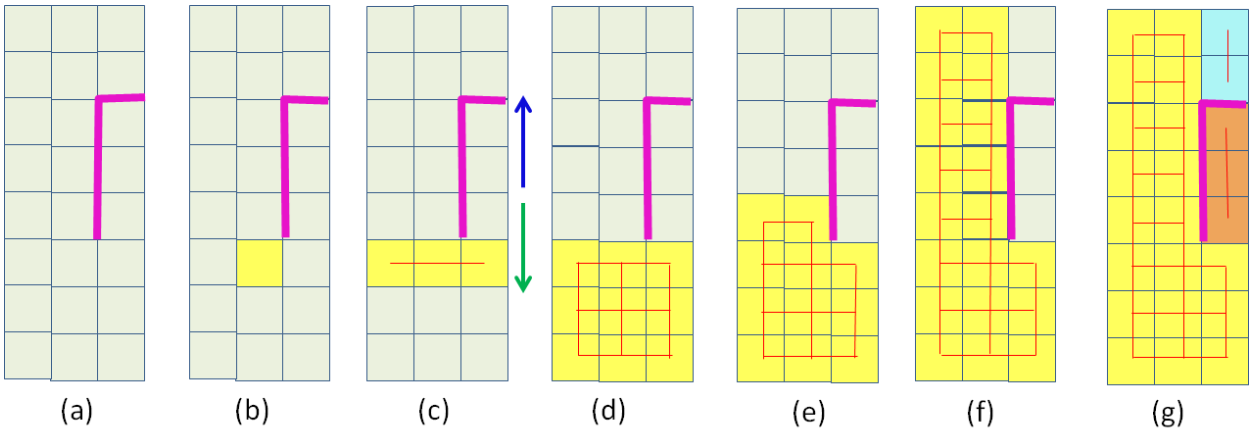


Figure 3.7: Illustration of the QRG algorithm: (a) The quad partition here contains a feature curve (shown in green). (b) A random face is selected in the partition (shown in yellow). (c) A poly-chord (shown in red) is generated (downward/upward direction shown with green/blue arrow). (d) The partition grows in the downward direction. (e) While growing in the upward direction, the region meets an edge of the feature curve, and the longest poly-chord is selected. (f) Growth in the upward direction ends. (g) Two more partitions (shown in green and orange) are generated by repeating the process from (a) to (f).

The first step of the QRG process involves the generation of a poly-chord, which is a connected sequence of faces in the same column j or the same row i. For simplicity, only poly-chords in the column direction are discussed here. Such a poly-chord j can be represented as $ch[a, b; j] = \{q[a, j], q[a + 1, j], \cdots, q[b, j]\}$, where $1 \leq a \leq b \leq N$ so that there are no feature curve edges between consecutive faces $q[i, j]$ and $q[i + 1, j]$. Both ends of the poly-chord $q[a, j]$ and $q[b, j]$ must therefore be either on the boundary or next to a feature curve edge.

To obtain a poly-chord, an unconquered face $q[i, j] \in Q$ is first randomly selected, and the poly-chord $ch[a, b; j]$ including $q[i, j]$ is generated. Note that $a \leq i \leq b$, and such a poly-chord is uniquely determined by searching faces from $q[i, j]$ in both directions of the column. Next, all quadrilateral faces of $ch[a, b; j]$ are set to be conquered and are added to partition R which is initialized as empty.

The second step of the QRG process involves growing the partition R by generating poly-chords in the next columns $j + 1$ and $j - 1$. Here, we first outline the case of $j + 1$. We try to find a new poly-chord $ch[c, d; j + 1]$ in the $j + 1^{th}$ column that is connected to $ch[a, b; j]$; in other words, $[a, b] \cap [c, d] \neq \oslash$. There should also be no feature curve edges between the connected quadrilateral faces of $ch[a, b; j]$ and $ch[c, d; j+1]$ or inside the poly-chord $ch[c, d; j+1]$. Based on these criteria, there may be no poly-chords, one poly-chord or multiple poly-chords. If there are

none, the algorithm switches to the $j-1$ direction and the same procedure is followed. If there are multiple poly-chords, the longest one is selected and is added to the partition R. This process is repeated while increasing the column number j up to W, and the algorithm then switches back to the $j-1$ direction as mentioned above. Finally, R is obtained as a partition of quadrilateral faces, and is added to the partition group G. If an unconquered faces remain in Q, we repeat the above region-growing algorithm to obtain other regions.

A straightforward method of finding $ch[c,d;j+1]$ is described here. It is important to note that in order to make the partition monotone in the poly-chord direction, we have to restrict the ranges of the lower and the upper bounds of the poly-chord in the growing process. To achieve this, we introduce two guard parameters for the lower bound LB and upper bound UB. Initially, LB and UB are set at the end of poly-chord generation as a and b, respectively after the first poly-chord $ch[a,b;j]$ is obtained.

Then, $ch[c,d;j+1]$ can be obtained via the following procedure:

1. Define a general poly-chord $ch[a,b;j+1]$ with the same span as $ch[a,b;j]$.

2. If there are feature curve edges between $ch[a,b;j]$ and $ch[a,b;j+1]$ or inside $ch[a,b;j+1]$, split $ch[a,b;j+1]$ at the quadrilaterals with the feature curve edges to generate poly-chords $ch[a_k,b_k;j+1], k=1,2,\cdots,W$, where $a_k < b_k$ for $k=1,2,\cdots,W$, and $b_k < a_{k+1}$ for $k=1,2,\cdots,W-1$.

3. The first and last poly-chords can be extended in the column direction. That is, if $a_0 = LB$, extend $ch[a,b_0;j+1]$ to be $ch[a',b_0;j+1]$ so that $a' \le a$. Then if $b_k = UB$, extend $ch[a_k,b;j+1]$ to be $ch[a_k,b';j+1]$ so that $b \le b'$.

4. Select the longest poly-chord from $ch[a',b_0;j+1], ch[a_1,b_1;j+1], \cdots, ch[a_k,b';j+1]$ to be $ch[c,d;j+1]$.

5. If lower bound c is smaller than LB, set LB as c. Similarly, upper bound d is bigger than UB, set UB as d.

Readers should refer to the pseudo code in Algorithm 1 for further details of the QRG process.

Theorem: The QRG process always produces bi-monotone partitions.

Proof: The proposed algorithm prevents the growing of a non-monotonic partition in the poly-chord direction based on the concepts of lower bound LB and upper bound UB explained in the pseudo code. Similarly the following operation ensures monotonicity in the poly-chord cross direction. During the QRG process, the poly-chord obtained is split into several sub-poly-chords

Algorithm 1 The QRG process

while all smooth quadrilateral elements $q[i, j] \in F^h \subset M$ are not conquered **do**
  region $R = \oslash$
  // 1. poly-chord generation
  select an unconquered $q[i, j] \in Q$
  generate a poly-chord $ch[a, b; j]$ including $q[i, j]$
  set faces of $ch[a, b; j]$ to conquered
  add faces of $ch[a, b; j]$ to R
  set lower bound LB and upper bound UB as a and b respectively
  // 2. region growing from a poly-chord
  while true **do**
    define a general poly-chord $ch[a, b; j + 1]$
    split $ch[a, b; j+1]$ at the quadrilaterals having the edges of *feature curve*s to generate poly-chords $ch[a_k, b_k; j+1], k = 1, 2, \cdots, W$ if there are some edges of *feature curve*s between $ch[a, b; j]$ and $ch[a, b; j + 1]$ or inside the $ch[a, b; j + 1]$
    if $a_0 = a$ and $a_0 = LB$ **then**
      extend $ch[a, b_0; j + 1]$ to be $ch[a', b_0; j + 1]$
    end if
    if $b_k = b$ and $b_k = UB$ **then**
      extend $ch[a_k, b; j + 1]$ to be $ch[a_k, b'; j + 1]$
    end if
    select the longest poly-chord, set as $ch[c, d; j + 1]$
    if $ch[c, d; j + 1]$ is empty **then**
      switch back to the $j - 1$ direction or add partition R to partition group G
      break
    end if
    set faces of $ch[c, d; j + 1]$ to conquered
    add faces of $ch[c, d; j + 1]$ to partition R
    if $LB < c$ **then**
      $LB = c$
    end if
    if $UB > d$ **then**
      $UB = d$
    end if
  end while
end while

if there are feature curve edges between the poly-chord and its neighboring poly-chord so that only one of them is selected as a poly-chord.

Shape-control: A shape-control parameter P_β is introduced to control the shape of the bi-monotone partitions during the QRG process. The bi-monotonic shape value ζ of a poly-chord $ch[c, d; j + 1]$ generated from poly-chord $ch[a, b; j]$ is computed as follows: $\zeta = \frac{\|(d-c)-(b-a)\|}{b-a}$. In short, it is relative difference of the lengths of two successive poly-chords. The value ζ is constrained to be smaller than the user-defined P_β. While assigning higher values to the parameter P_β results in the generation of much more arbitrary shapes, shapes close to quad topology are generated with smaller values. As many neighbor poly-chords can be generated from a single poly-chord, we select the longest one whose ζ value is smaller than the user-defined value P_β. Additionally, during the poly-chord extension process, we check the ζ value after each extension and stop extending the poly-chord once ζ exceeds the user-defined value P_β.

3.6 The Partition Configuration Improvement (PCI) Process

In this section our goal is to construct a configuration with large partitions in which all the extracted surface curves lie on the partition boundaries. To achieve this, planar regions are first detected and merged. Next, some partition boundaries are screened to determine whether they represent feature curves, as only surface features inside the partitions are detected in the procedure described in Section 4. partition growing is then performed between the surface features.

3.6.1 Detection of planar regions

After the QRG process, there may be planar partitions that need to be separated from other partitions as they can be simply represented by plane primitives. To detect and merge such partitions, we first calculate and assign a normal vector for each planar quadrilateral element. Randomly selecting a quadrilateral face in a partition Q, we compute the differences of its normal vector from those of all the other quadrilateral faces in Q. If all these differences are smaller than the user-defined planarity threshold T_{Γ_0}, whose value is set as 0.001, the partition Q is defined as planar. Furthermore, if two neighboring partitions are planar and the difference between their normal vectors is smaller than T_{Γ_0}, they are merged.

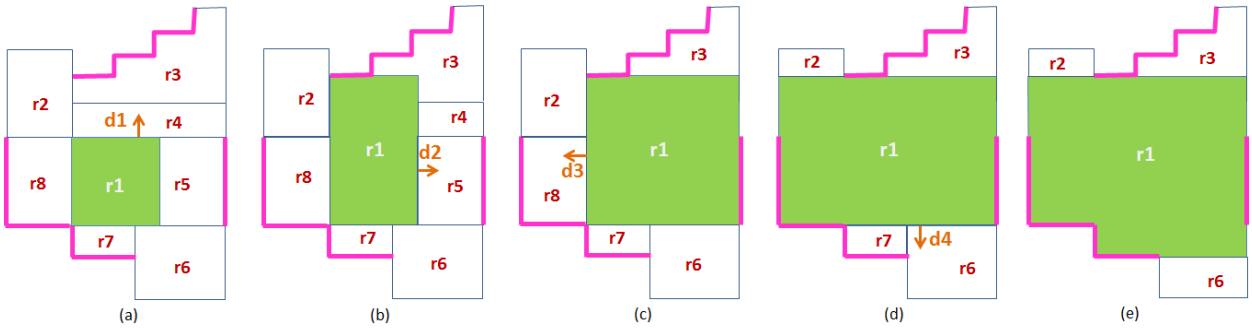


Figure 3.8: Illustration of the PCI algorithm]: (a) Growing starts from partition r1 in direction d1. (b) Growing stops when a feature curve is met. partition r1 becomes larger and covers parts of partitions r3 and r4. Growing direction d2 is set. (c) After growing stops at a feature curve, partitions r4 and r5 disappear, and partition r3 becomes smaller. Growing direction d3 is set. (d) Growing in direction d3 produces the large partition r1, partition r2 shrinks and partition r8 disappears. Growing direction d4 is set. (e) In line with the shape control parameter P_β values, partitions in a non-quad shape can be generated. Growing in direction d4 results in a bi-monotone partition.

3.6.2 Screening of partition boundaries

The partition boundaries consisting of the EMG edges need to be screened to ascertain whether they represent surface features. For each one, the number of edges for which the normal difference between corresponding two adjacent faces is larger than the non-flatness threshold T_Γ (from the feature detection step) is determined. If such edges dominate more than half the boundary, it is treated as a surface feature (see Fig. 3.2 (f)).

After the screening process as a post processing step, it is possible to discard some detected surface features which are close to each other. The elimination of such features may result in the generation of fewer bi-monotone partitions, but this should be performed carefully to produce surface models having good quality. In this study, we respect all detected features of the feature detection and partition boundary screening processes, and constrain them to act as the partition boundaries.

3.6.3 Partition Growing

The partition layout after the QRG algorithm contains unnecessary boundaries that need to be removed to create large partitions. The proposed growing strategy is different from those in which process starts from a face and regions are grown from it. As the partitions observed after the QRG

algorithm have good shapes and all feature curves reside on partition boundaries, the PCI process improves partition configuration by growing these existing partitions. The growing method of this process is similar to that of the QRG process in that it is also quadrivial. Starting from a partition, growing is performed in a one-by-one manner in four directions and stops when the partition meets a surface feature. Readers should refer to Figure 3.8 for the illustration of the PCI algorithm.

When growing starts from a randomly selected partition, the PCI process may result in skinny partitions, and the number of partitions in the configuration may be large. For this reason, priority is given to growing certain partitions first. An energy function E is utilized and computed for each partition so that growing starts from the partition with the lowest energy. After growing is completed in four directions, the partition and its boundaries are set as having been processed and do not change further. For each partition, the same growing process is applied. The algorithm terminates when all partitions in the configuration have been processed. Even though optimum partition configuration cannot be guaranteed, the number of partitions is seen to decrease, many of them become larger, some disappear and a few shrink.

The energy E of a partition is calculated as: $E = \frac{F*A}{N}$ where F is the partition non-flatness, A is the shape aspect ratio of the partition, and N is the number of the quadrilaterals in the partition. The partition non-flatness F is computed by averaging the non-flatness values of the faces in the partition. The shape aspect ratio is $A = L/W$, where L is the length and W is the width of the partition in the rectangular grid structure. This formula encourages flat partitions, partitions with more faces and compact (i.e., square) partitions to grow first. As a result, flat partitions in the model are covered first. Despite the simplicity of the proposed energy formula, it works well because partitions at the beginning of the PCI process have good shapes and all feature curves reside on partition boundaries. It is also possible to utilize different energy formulas or optimization algorithms in this step.

The PCI process takes a number of topological constraints into account:

- **Bi-monotonicity:** While advancing partition boundaries, the algorithm checks whether the region growing process generates non-monotone partitions. If the bi-monotonicity of a partition itself is violated, it is made bi-monotone by splitting its last-added poly-chord. A grown partition may also make its neighboring partitions non-monotone. In such cases, these partitions are split into several bi-monotone partitions.

- **Mesh regularity:** The valences of the vertices in the partition are monitored. If an extraordinary vertex is included in the partition, splitting at that vertex is performed.

- **Shape control:** As with the QRG process, the shapes of bi-monotone partitions are con-

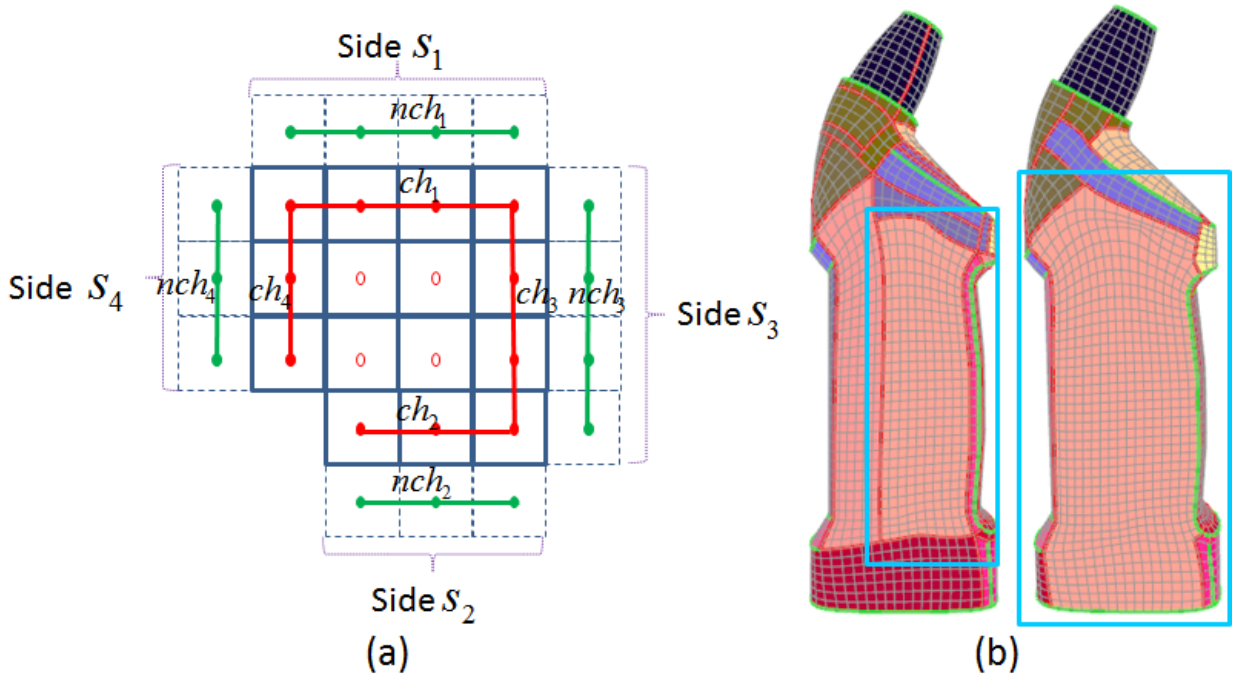


Figure 3.9: (a) partition (R) preliminaries. (b) Bottle model after the QRG process (left) and the PCI process (right).

trolled using the shape-control parameter P_β.

This section describes the PCI process in further detail. The pseudo code in Algorithm 2 summarizes the PCI process. For simplicity, only growing in one direction is illustrated. Here, we have a set of partitions G obtained from the QRG process. For each one, the energy E is computed. First, the algorithm sorts the partitions according to their energy values and grows partition R of the G with the lowest energy value. partition R has four sides $s_k, k = 1, 2, 3, 4$ with four poly-chords ch_k and four neighboring poly-chords nch_k as depicted in Fig. 3.9 (a). If nch_k contains faces that have extraordinary vertices or feature curve edges, it is split into poly-chords such that nch_k is revised as the poly-chord having more number of faces. The un-flatness value S_k for each neighboring poly-chord is calculated by averaging the flatness values of all faces in nch_k. To enable the start of growing from a poly-chord, the algorithm selects the flattest neighboring poly-chord nch_k residing at s_k.

Fig. 3.9 (b) shows bottle model after the QRG process (left) and the PCI process (right). Note how the partition enclosed by the blue rectangle has grown after the process.

Algorithm 2 The PCI process

sort G by energy E
while any partition of G is unprocessed **do**
  get *unprocessed* partition R from G
  while all side s_k of R are processed **do**
    get all *unprocessed* side poly-chords ch_k
    get neighbor poly-chords nch_k attached to side poly-chords ch_k
    check nch_k whether it has some *processed* faces or have extraordinary vertices and revise nch_k
    calculate un-flatness S_k of neighbor poly-chords nch_k
    get the flattest neighbor poly-chord $ch[m, n; j] \in nch_k$
  end while
  while true **do**
    define a general poly-chord $ch[m, n; j + 1]$
    split $ch[m, n; j + 1]$ if any feature curve edges exist and generate poly-chords $ch[a_k, b_k; j + 1], k = 1, 2, \cdots, L$
    select the longest poly-chord and set as $ch[c, d; j + 1]$
  end while
  if $ch[c, d; j + 1]$ is empty **then**
    set side s_k of R as *processed*
    break
  end if
  calculate *bi-monotonic* shape value ζ for poly-chord $ch[c, d; j + 1]$
  if $\zeta \geq P_\beta$ **then**
    break
  end if
  if *non-monotone* neighbor partitions Nr exist **then**
    split *non-monotone* partitions Nr into several bi-monotone partitions
  end if
  erase faces of $ch[c, d; j + 1]$ from its previous belonging region
  add faces of $ch[c, d; j + 1]$ to partition R
  set partition R and all its faces as *processed*
  sort G by energy E
end while

3.7 Results and Discussion

The method proposed in this chapter consists of four separate consecutive processes: EMG, feature detection, QRG and PCI. The proposed algorithm is automatic once the thresholds of the EMG process (T_ψ) and the feature detection process (T_Γ, T_φ, T_ρ) are set. At the same time, the shapes of the generated partitions can be controlled by adjusting the shape control parameter P_β.

T_φ (feature elongation size) and T_ρ (feature neighbor) are adjusted depending on the resolution of the mesh. T_ψ, angle threshold used in the EMG process, is an angle which is set to a value between 30 and 70 degrees. T_Γ is used to detect unflat regions which we set between 0.15 and 0.4. The shape control parameter P_β can be set to 0.2 to generate better bi-monotone shapes. However, in some cases such as the fandisk model, setting it to a value close to 1.0 generates much desirable results to capture the surface features of the model with a single bi-monotone partition. From our experience, our parameters are easy to tune.

T_φ determines the length of the feature curve that will be extracted. Assigning small values to this parameter may produce many feature curves especially for models having noise. While constraining these curves on partition boundaries, large number of partitions will be generated which is not desirable. Therefore, determining the value of T_φ according to the mesh resolution using heuristics will generate appropriate feature curves. T_ρ avoids the generation of feature curves parallel to each other. Otherwise, many feature curves can be produced for models having noise that results in a large number of partitions. The angle threshold T_ψ is used in the EMG process to go through the highly curved regions when its value is high. When it is high, the particle stops at its current position and two particles are placed which go on left and right paths. Therefore, if small values are assigned for T_ψ, large number of paths can be produced which results in a large number of partitions. T_Γ is used to detect the unflat regions and setting it to small values will detect many unflat edges, and therefore large number of partitions will be generated. Setting shape control parameter P_β to 0.0 will favor generation of quadrilateral partitions and assigning highler values result in bi-monotone partitions. We believe that appropriate values can easily be set to our parameters after several value assignment attempts done by the user.

For the rockerarm model shown in Fig. 3.11, the proposed algorithm generates 103 bi-monotone partitions. Large flat parts are captured successfully, because the PCI process gives flat partitions priority to grow first. The cylindrical and fillet geometry of the model is also retained by the bi-monotone partitions.

Figure 3.12 shows a beetle model whose whole roof is a single partition. The proposed algorithm captures almost all symmetries of the model such that the front part shown in dark blue is

represented by two partitions whose common boundary (shown in green) is found during the feature detection process. For different values of the shape control parameter P_β, differently shaped bi-monotone partitions are generated. When smaller values are assigned to P_β, the shape of partitions becomes more quad-like, but their number increases.

Figure 3.13 shows a bottle model whose geometry is captured with large bi-monotone partitions. In the rightmost image in (a), there are two small quadrilateral faces representing a partition. These are not included in the large partition next to them during the QRG and the PCI processes because the partition with them contains irregular vertices, which is not desirable. Even though using a larger value for P_β reduces the number of bi-monotone partitions, it may result in the generation of partitions that are not appropriate for B-spline surface fitting. For instance, the the light-orange partition in (c) covers the whole right, left and rear sides of the model, which may not be desirable.

The smooth surfaces of the fandisk model shown in Fig. 3.10 can be represented by large bi-monotone partitions as shown in (a). As setting smaller values for P_β produces smaller fragments with quad shapes (b, c), the smooth surfaces cannot be captured by a single partition. The motorcycle graph of Eppstein et al. [19] (d), the method of Myles et al. [39] (e), and the method of Bommes et al. [7] utilize quad partitions to recover the model, but these partitions hide surface features inside themselves. The method of Myles et al. method may capture such features with many quad fragments which is undesirable.

Our algorithm gets quadrilateral mesh as input and generates bi-monotone partitions from it. Therefore, the quality of the bi-monotone partitions is directly dependent on the quality of the quadrilateral mesh (singularity distribution and anisotropic element alignment). In case there are too many singular points, the mesh will be segmented into many bi-monotone partitions which may not be desirable.

It is possible to improve the segmentation results. A few tiny partitions residing between two feature curves can be seen in the beetle and the rockerarm models. It is possible to eliminate one of these feature curves so that the tiny partitions will automatically disappear. However, this process should be performed carefully to not generate partitions having high curvatures in the inner parts.

Capturing the Design Intent. When modeling a part ab-initio, the geometric discontinuities are generally placed on the boundaries between surfaces (partitions). Similarly, our algorithm detects the surface features and constrains them to be the boundaries of the generated (bi-monotone) partitions. Therefore, the partitions generated by our algorithm can capture designer's intent to some degree.

Fitting of B-splines. To fit bi-monotone partitions with B-splines, we use the *global approx-*

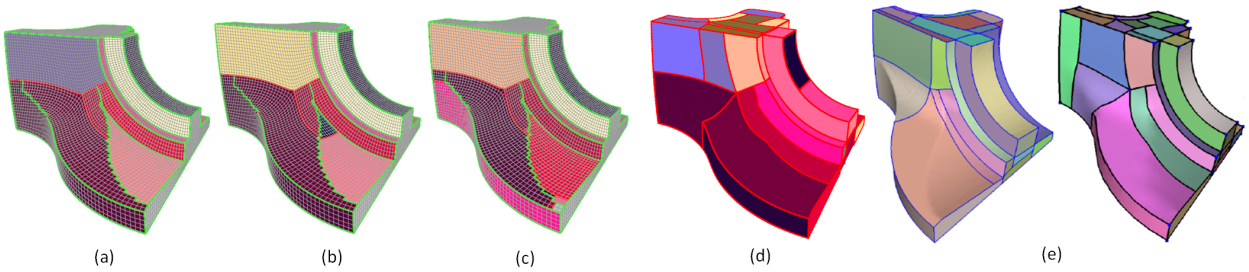


Figure 3.10: A fandisk model with 13,181 faces and bi-monotone partitions generated by the proposed algorithm ($T_\psi = 30$, $T_\Gamma = 0.15$, $T_\varphi = 8$, $T_\rho = 4$) Changing the shape-control parameter P_β affects the shape of the model. Numbers of bi-monotone partitions: (a) 46 ($P_\beta = 0.6$) (b) 51 ($P_\beta = 0.4$) (c) 59 ($P_\beta = 0.2$). Quad partitions generated by other algorithms: (d) motorcycle graph (55 partitions) [19] (e) feature-aligned T-meshes [39] (images taken from [39]).

imating surfaces [51]. First, a four-sided surface that contains all partition points is chosen. The distances between the surface points and the partition points are then minimized using least-squares method. By iterating these two processes, a good fit for the surface can be obtained. Finally, trimming curves are constructed by intersecting the surfaces with their neighboring surfaces. Figure 3.14 shows surface models for fandisk and beetle models as observed after the EMG and the PCI processes. Compared to the post-EMG surface models, those after the PCI process have large partitions and capture the surface features better.

Performance. A standard use 3.4 GHz PC was used for the experiments in this study. The EMG processing time is negligible in all cases. The feature detection process takes less than 0.15 seconds for all test models except that of the rockerarm model which has many non-flat quadrilateral faces scattered over its mesh. For this model, feature detection takes 1.09 seconds. The QRG process takes less than 0.8 seconds for all test models. The processing times for the PCI process are 2.861 seconds for the fandisk model of Fig.3.10 (a), 13.666 seconds for the rockerarm model, 1.85 seconds for the beetle model of Fig.3.12 (a) and 1.934 seconds for the bottle model of Fig.3.13 (a). The time taken for the PCI process depends on resolution of the model since the number of quadrilateral faces that should be processed increases. We also observe that starting with a fewer partitions or more feature curves on the partition boundaries (as seen with fandisk model) reduces the processing time of the PCI process.

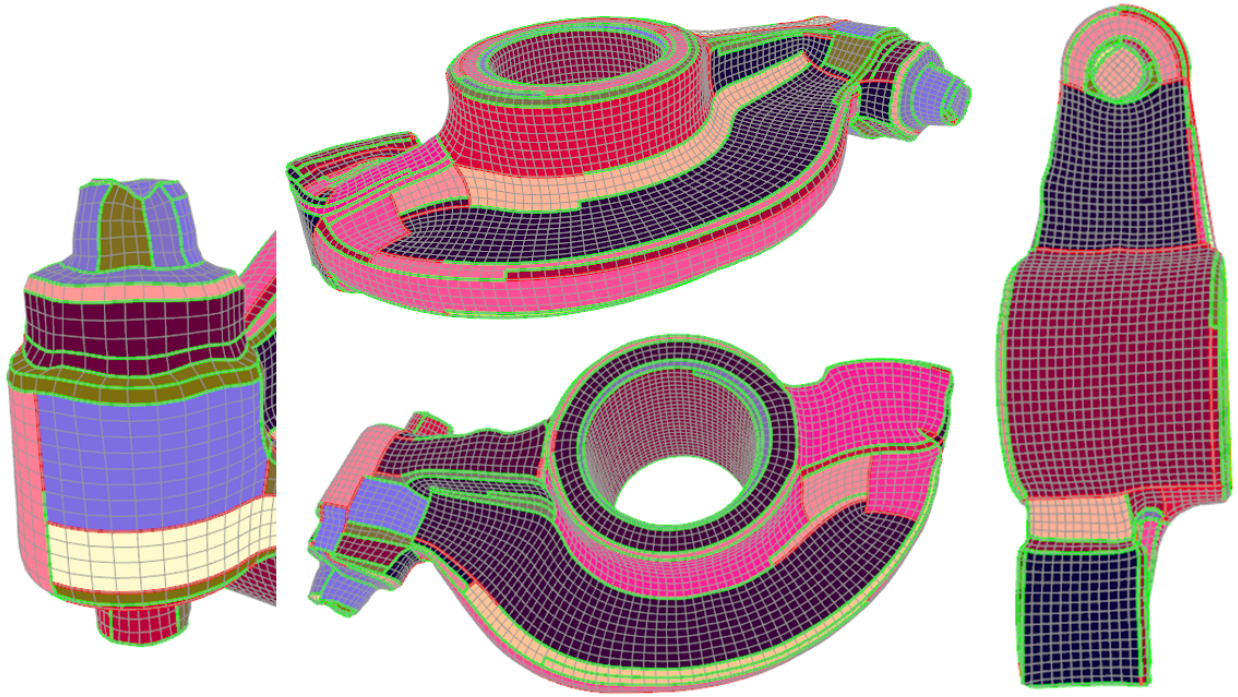


Figure 3.11: A rockerarm model with 9,413 faces. A total of 103 bi-monotone partitions are generated by the proposed algorithm (T_ψ = 40, T_Γ = 0.2, T_φ = 8, T_ρ = 4, P_β = 0.2).

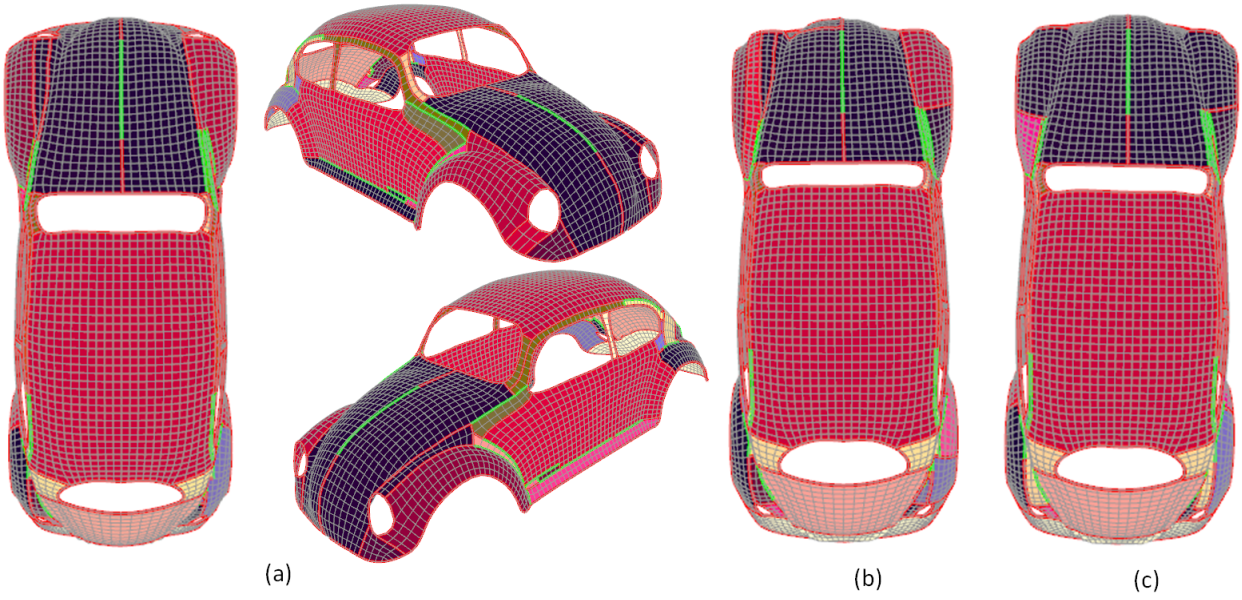


Figure 3.12: A Beetle model with 3,779 faces and bi-monotone partitions generated by the proposed algorithm (T_ψ=70, T_Γ=0.4, T_φ=7, T_ρ=2). Changing the shape-control parameter P_β affects the shape of the model. Numbers of bi-monotone partitions: (a) 45 (P_β = 0.2) (b) 51 (P_β = 0.0) (c) 34 (P_β = 0.6). When smaller values are assigned to P_β, the shape of the partitions becomes more quad-like, but their number increases.

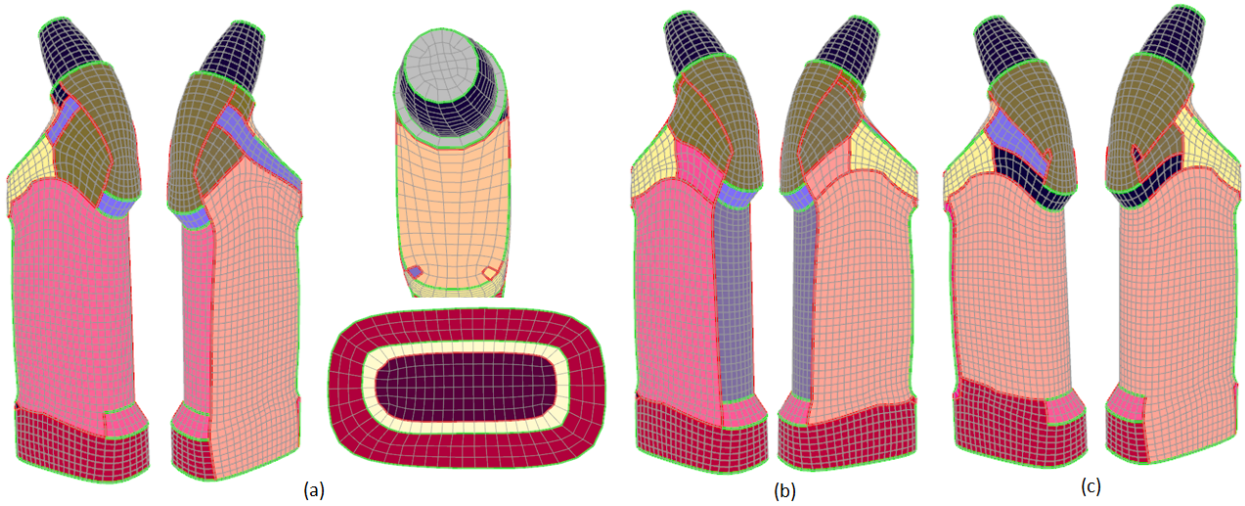


Figure 3.13: A bottle model with 3,117 faces and bi-monotone partitions generated by the proposed algorithm ($T_\psi = 70$, $T_\Gamma = 0.4$, $T_\varphi = 6$, $T_\rho = 1$). Changing the shape-control parameter P_β affects the shape of the model. Numbers of bi-monotone partitions: (a) 38 ($P_\beta = 0.2$) (b) 40 ($P_\beta = 0.0$) (c) 35 ($P_\beta = 0.6$).

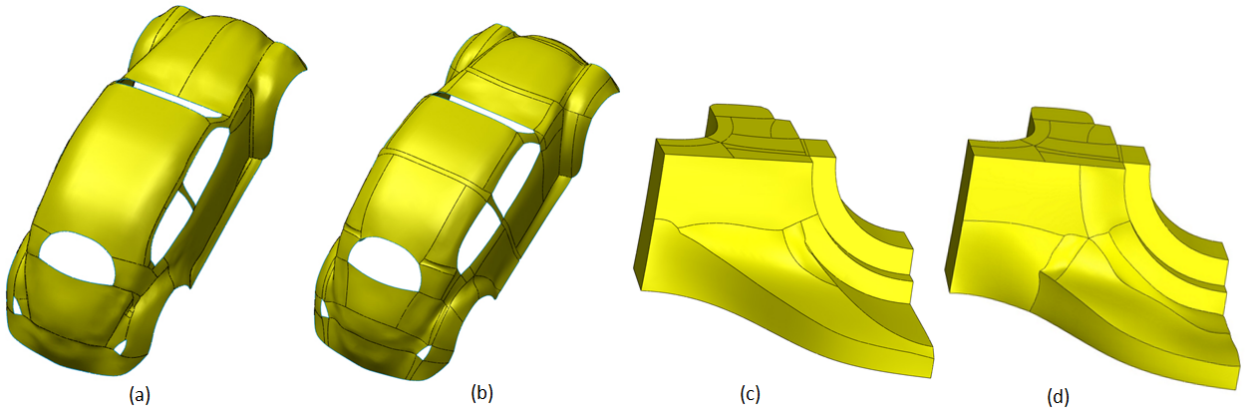


Figure 3.14: B-spline surfaces generated for: (a) the bi-monotone partitions of the beetle model in Fig. 3.12 (a); (b) the quad partitions of the beetle model after the EMG process; (c) the bi-monotone partitions of the fandisk model in Fig 3.10 (a); (d) the quad partitions of the fandisk model after the EMG process; Surface models in (a) and (c) have larger partitions and capture the surface features better.

Chapter 4

MCG-based Approaches for Quadrilateral Partitioning

4.1 Introduction

Reverse engineering can be used to generate a CAD model (i.e., a set of parametric surfaces) from a set of scanned points. A polygonal mesh is first obtained from these points, and partitioning is then applied to divide the mesh into regions for individual parametric surface fitting. The quality of the CAD model heavily depends on this partitioning. partitions ideally have two important features:

- **A. Quadrilateral partitions:** Partitions are quadrilateral regions with four corners and four boundaries. In other words, they are isomorphic to an a, b-grid which is a mesh of unit squares in the rectangle $\{(x, y)|0 \leq x \leq a, 0 \leq y \leq b\}$. The image on the left of Fig. 4.1 (a) shows partitioning featuring several quadrilateral partitions with boundaries in red and corners in blue. The image on the right of the figure depicts a quadrilateral partition consisting of regularly arranged quadrilateral faces.

- **B. Capture of features:** Partition boundaries should pass through feature regions (such as highly curved parts) of the mesh.

Quadrilateral partitions are preferable for surface fitting in the reverse engineering because they do not require surface trimming or surface parameterization. When partitions in a mesh are all quadrilateral partitions, they are are called as *structured partitions*. The main objective of the research reported here is to find a set of structured partitions considered optimal in terms of reverse engineering. A set of structured partitions can be easily generated using a semi-regular quadrilateral mesh. A semi-regular quadrilateral mesh consists of quadrilateral elements (quads) whose vertices mostly of valence four (*ordinary vertices*), with the others being non-valence four (*extraordinary vertices*). Such a mesh can be generated from a triangular mesh using recently proposed quadrangulation methods [9, 11, 42, 49]. There can be many structured partitions for a semi-regular quadrilateral mesh. We use *motorcycle graph* of Eppstein et al. [19] to generate structured partitions from a semi-regular quadrilateral mesh created using the method of Bommes et al. [9].

The requirement for B listed in above is also important. The introduced cost formula produces small values for motorcycle graphs whose edges trace a large number of highly curved regions of the model. Additionally, feature curves (i.e., sets of highly curved edges) that are very difficult to capture with motorcycle graph edges are also extracted and integrated into the proposed framework.

Definitions: The main component of the MCG algorithm is a path on the mesh traced by a moving particle. A path for the particle R is defined from the seed edge e_1 and the seed vertex v_1, which is an end vertex of e_1. The pair $< v_1, e_1 >$ is the seed of the path.

Tracing using the rules of the MCG algorithm is applied which is referred to as *MCG tracing* hereafter. In MCG tracing, R starts from v_1 along e_1 toward its other end vertex. R continues tracing the edges and vertices of the mesh to form the path W from the seed $< v_1, e_1 >$ to R. In the proposed algorithms, it is necessary to represent only situations in which R is on a vertex, so W can be represented as a sequence of vertices and edges described by $W =< v_1, e_1, v_2, e_2, \ldots, v_m >$, where R is located on v_m. When R stops, v_m represents the end vertex of the path. For the sake of simplicity, a path, which is not a seed $W =< v_1, e_1 >$, can also be represented by its vertices $W =< v_1, \ldots, v_m >$ or by its edges $W =< e_1, \ldots, e_m >$. In the MCG algorithm, seeds are defined for all edges incident to extraordinary vertices using them as seed vertices.

Problem Setting and Cost Function: A structured partition can be obtained from a set of paths $\{W_k\}, k = 1, 2, \ldots, N_W$ based on a given set of seeds, where N_W is the total number of paths. The goal here is to find such a set of paths that pass through as many feature edges as possible, and thereby generate feature-aware structured partitions. The unsigned dihedral angle $\theta(e)$, $(0 \leq \theta(e) \leq \pi)$ of the edge e is the angle between the normal vectors of its adjacent faces. $\theta(e)$ is 0 for a flat edge with two flat faces, and it gets large values for sharp edges. We have chosen dihedral angle as a geometric criterion to represent features because it has low computation cost and can capture feature curves of the model well.

To find such feature-aware partitions, the cost function $F(W_k)$ of the path W_k is defined so as to give a low-cost value for a path of edges with large dihedral angles. $F(W_k)$ is defined as $F(W_k) = -C(W_k) + \alpha D(W_k)$, where α is a user-defined value. $C(W_k) = \sum_{e_j \in W_k} \theta(e_j)$ is simply the sum of all the dihedral angles for the edges of W_k and $D(W_k) = \sum_{e_j \in W_k} d(e_j)$, where

$$d(e_j) = \begin{cases} 0 & \text{if } \theta(e_j) > \epsilon; \\ 1 & \text{if } \theta(e_j) \leq \epsilon. \end{cases}$$

$D(W_k)$ is introduced because paths that are less curved and longer have a lower cost without it (see Fig. 4.10 (a). Setting α to an appropriate value reduces the number of such paths (b)). A set of paths $\{W_k\}$ that minimizes the global cost $G_W = \sum_k F(W_k)$ is sought. It is also possible to utilize other cost functions including different path or partition criteria (such as number of paths, path or partition size, surface fitting error, etc.), but we consider it as a future work to investigate these functions. Note that utilizing surface-related criteria (such as surface fitting error) in the cost function may increase the quality of generated surfaces but its computational cost will be very

high.

The following sections outline two quadrilateral partitioning techniques, the Path Flipping (Sec. 4.2) and MCG Enumeration approaches (Sec. 4.3), which are intended to solve the same problem that is explained in the above.

4.2 The Path Flipping Approach

Fig. 4.1 (b) shows a quadrilateral mesh partitioned by the MCG algorithm and highly-curved regions are seen inside the generated partitions. Our approach is based on the MCG algorithm but can locate feature curves in these highly curved regions (marked with blue circles in Fig. 4.1 (c)) on the partition boundaries, therefore better surface quality is expected when fitting parametric surfaces to the partitions generated using our method. Figure 4.2 (b) also shows the partitions (with boundaries in different colors) obtained using the MCG algorithm, similarly not all portions (as shown by the black circle in Fig. 4.2 (b)) of highly-curved regions (as shown in blue in Fig. 4.2 (c)) are captured by the MCG algorithm. The proposed algorithm is a practical and useful algorithm that can generate quad partitioning (with T-junctions). The generated curvature-aware partitions are appropriate to be used for surface modeling applications (B-spline, T-spline fitting), texture mapping, subdivision surfaces, etc.

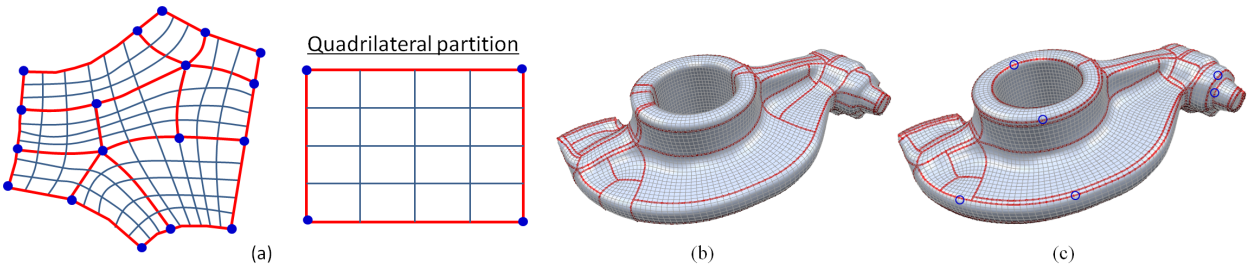


Figure 4.1: A quadrilateral partition ((a), right) has four boundaries (in red) and four corners (in blue) consisting of regularly arranged quadrilateral faces. Quadrilateral partitions in a quadrilateral mesh are illustrated in the image on the left of (a). Quadrilateral mesh model partitioned by: (b) the motorcycle graph algorithm of Eppstein et al. (c) our algorithm. Our method locates the highly-curved regions on the partition boundaries, therefore better surface quality is expected.

Algorithm Flow: Figures 4.2 (c) to (f) show these algorithms. First, feature curves are extracted and particles (spheres in colors) are placed on ordinary seeds (from where tracing starts) in addition to extraordinary ones (see Fig. 4.2 (c)). A speed control algorithm that starts tracing from these seeds is applied with rules identical to those of the MCG algorithm, except a constant

speed is assigned to particles (see Fig. 4.2 (d)). The partition layout is then improved using local path flipping operations (see Fig. 4.2 (e)). Partitions obtained using the proposed algorithms are still structured as no extraordinary seeds are introduced during the particle placement step. To form a much more compact partition layout while respecting feature curves, smooth boundaries are removed (see Fig. 4.2 (f)). The difference between paths obtained using the MCG algorithm and those obtained after each step of the proposed algorithm are shown by circles in different colors in Fig. 4.2. In this study, we have utilized quadrilateral mesh generated using mixed integer quadrangulation method [9].

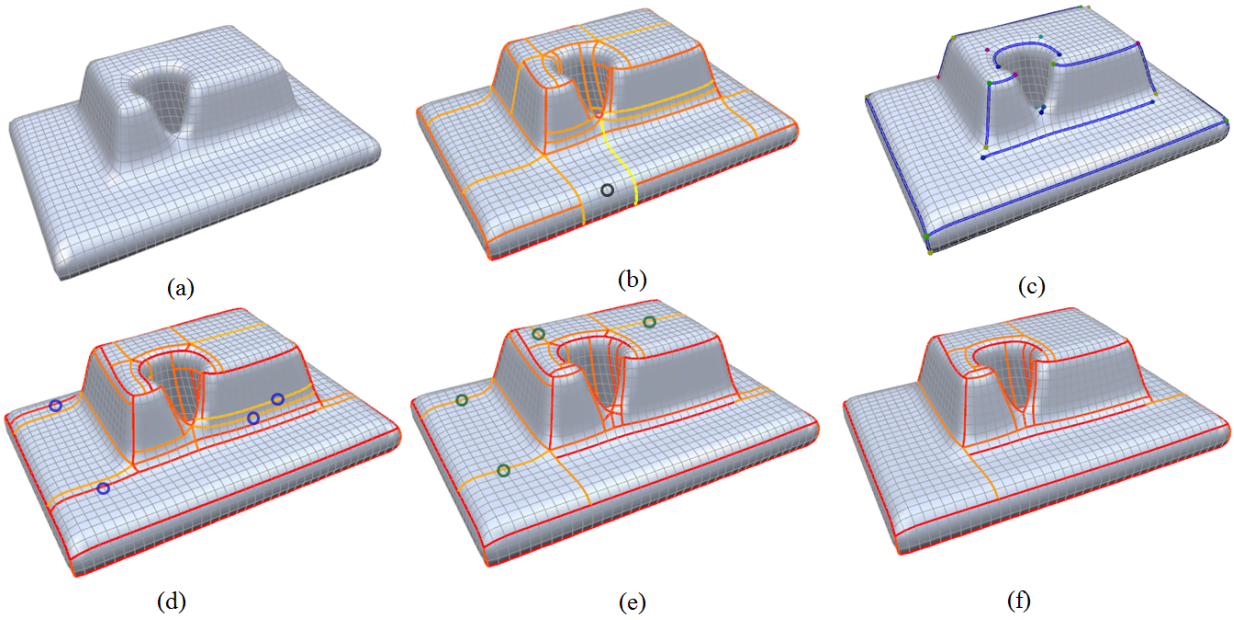


Figure 4.2: (a) Input quadrilateral mesh generated using the mixed integer quadrangulation [9] (b) Motorcycle graph partitioning using the technique of Eppstein et al. [19] The following images show the steps of feature-aware algorithms: (c) Particle (spheres in colors) placement for extraordinary vertices and end vertices (regular) of extracted feature curves (shown in blue) (d) Speed control algorithm (particle speeds are controlled according to the surface geometry) (e) Path flipping operation ($\tau = 15$) (f) Removal of flat partition boundaries ($\eta = 0.1$)

4.2.1 Approach for Feature-aware Motorcycle Graph

Here, the motorcycle graph algorithm [19] is first summarized, then the proposed approach and basic terminology are described in relation to problem setting.

Motorcycle Graph (MCG) Algorithm: Partitions obtained using the MCG algorithm have no extraordinary vertices because tracing starts from such vertices, which are thus constrained to be

on partition corners. From a topological aspect, the MCG algorithm produces partitions (patches) having favorable shapes. As shown in [19], if a quadrilateral mesh is homeomorphic to a manifold with or without a boundary and is not itself already structured, the motorcycle graph algorithm partitions the mesh into structured disks (i.e., rectangular grids). The proposed geometry-aware algorithms also have this property.

The MCG algorithm is simple and effective when used for quadrilateral mesh partitioning. However, the partitions it creates may not be appropriate for use in certain geometric applications such as surface modeling because surface geometry is not taken into account. In this context, MCG algorithm-based partitions may not have smooth geometry; that is, geometric discontinuities (i.e., normal, curvature) of the model may reside inside partitions, making them unsuitable for surface modeling applications.

Eppstein et al. [19] suggested methods to generate smaller partitions. One such approach involves is adding one path at a time rather than adding numbers of valence paths simultaneously to extraordinary vertices. This technique cannot be applied to the proposed application for numerous reasons. In Eppstein et al.'s work, it is mentioned that priority is given first to paths that extend from one extraordinary vertex to another. For a mesh without a boundary, there is always a path from an extraordinary vertex to another. Due to the singularity alignment problem [39, 7, 11], this path may have an ivy-like form that turns around the model several times, causing undesirable partitioning. Moreover, if all edges at an extraordinary vertex must be required to capture feature curves, their use should be encouraged rather than using only one of two consecutive edges at that vertex.

Basic Concepts: In this study, an edge having large angle between normal vectors of its adjacent faces is considered as feature. Such feature edges are often aligned to form feature curves. Figure 4.3 shows an example obtained using the MCG algorithm for a quadrilateral mesh generated via the mixed integer quadrangulation method [9]. Due to the sophisticated performance of this quadrangulation, extraordinary vertices are well located on the mesh, and paths generated by the MCG algorithm capture feature curves to a greater or lesser extent.

However, we have found some problems after partitioning using the MCG algorithm (see Fig. 4.3):

1. *Blocked feature curve:* The path p starting from the extraordinary vertex v is blocked by another path q and thus is prevented from tracing p', which is the remaining part of the feature curve p.

2. *Regular feature curve:* The feature curve r around the hole cannot be traced because there

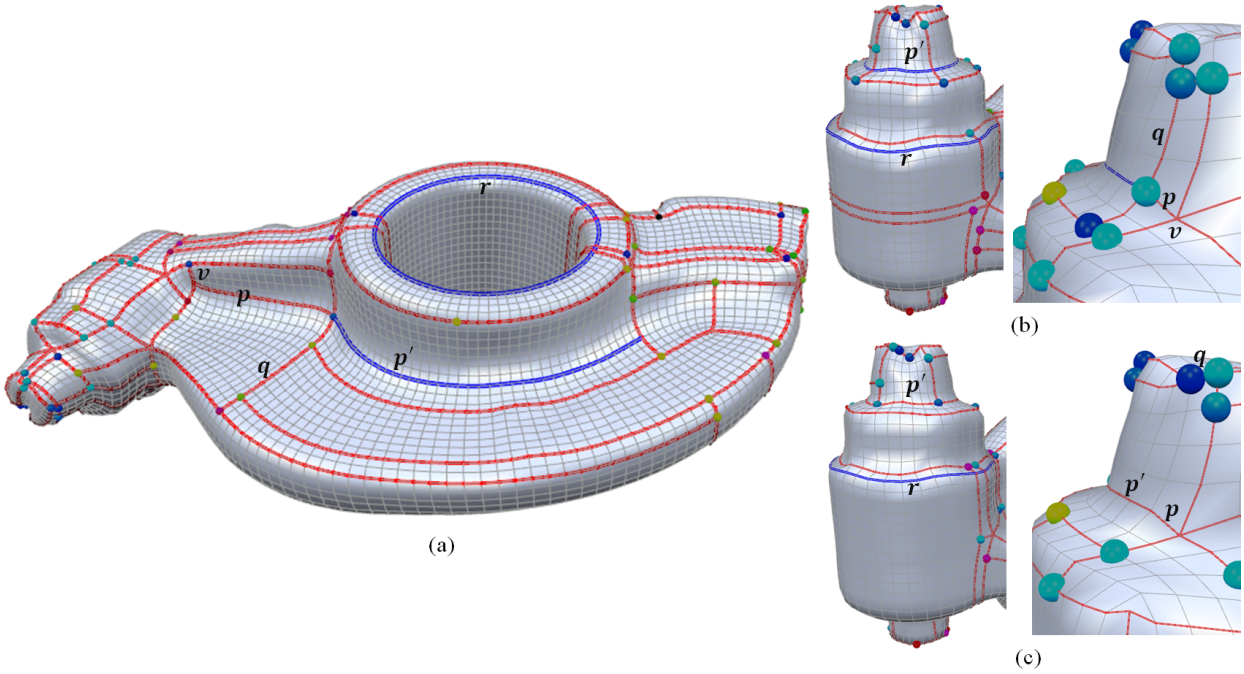


Figure 4.3: Red paths are traced by the MCG algorithm. It does not necessarily trace all feature curves (in blue). Only some portions of the feature curve p (not p') are traced. As there is no extraordinary vertex on the feature curve r, it is not traced either. After applying speed control algorithm, some feature curves (in blue) are not traced (b). After path flipping operations, the feature curve p' is captured and the path q becomes shorter (c).

is no extraordinary vertex on r.

The aim here is to involve in the partition as many feature curves as possible that are not necessarily captured by the MCG algorithm. At the same time, rather than losing the main structure (concepts) in the MCG algorithm, we devise algorithms consistent with it. The following three synergetic approaches are proposed:

1. **Speed Control:** In the MCG algorithm, particles are given initial speeds and maintain them. If the particle starting from v in Figure 4.3 (a) moves faster, it will not be blocked and will reach p'. This suggests that feature curves can be better captured by speeding up particles going through feature curves. Changing the speed of particles in this way still produces structured partitions as Eppstein et al. also mentioned [19].

2. **Path Flipping:** A particle can reside on a non-feature edge whose adjacent edges are feature edges. Figure 4.3 (b) illustrates such a case in which the feature curve (in blue) is not captured after applying the speed control algorithm. This algorithm therefore flips intersecting paths to trace such feature curves. After path flipping operations, the feature curve p' is

traced by a particle originating from the vertex v, and the path q becomes shorter (see Figure 4.3 (c)).

3. **Regular Seed Insertion:** The feature curves (depicted as r) in Figure 4.3 are not captured by the existing paths after applying speed control algorithm and path flipping operations. To enable capturing all feature curves of the model, the insertion of regular seeds (vertices) on feature curves in addition to extraordinary ones is suggested.

4.2.2 Feature-aware MCG Algorithms

In this section we propose algorithms to extend the MCG algorithm to be feature-awareness without damage to its structure. After initial partitioning with paths using the speed control algorithm, path flipping operations are performed on these paths. A method for integrating feature curves into the proposed framework is also outlined here.

4.2.2.1 Speed Control (SC) Algorithm:

The SC algorithm is essentially the same as the MCG algorithm except that variable rather than constant speeds are assigned to particles. The basic concept involves speeding up particles on feature curves to prevent blockage by other particles in their way. Speed is controlled based on the dihedral angle of the edge.

A particle moves on an edge e at a speed of f. It is assumed that all edges have a length of one, and the speed is given by f in this unit. Thus, a particle with a speed of f moves a distance of $100 \times f$ percent of the edge e in each time step. f is defined as $f = T(\frac{\theta(e)}{\pi/2})$, and the function $T(x)$ is defined as follows:

$$T(x) = \begin{cases} \gamma & \text{if } x \leq \gamma; \\ x & \text{if } \gamma < x \leq 1; \\ 1 & \text{if } x > 1. \end{cases}$$

where γ is a user-defined constant set as 0.01 in this study to avoid very low speeds in highly smooth regions. The computational time taken for motorcycle graphs is higher when smaller values are set for γ. The speed f is also set to 1 for highly curved regions whose θ value exceeds $\pi/2$. Thus, the particle can move up to one edge in each time step.

Figure 4.4 shows a fandisk model partitioned using (a) the MCG algorithm and (b) the SC algorithm. As the SC algorithm takes surface geometry into account and assigns higher speed to particles (shown as colored dots) in highly curved areas (c), it fully traces the feature curves shown (shown by blue circles), whereas the MCG algorithm does not. In contrast to the MCG algorithm,

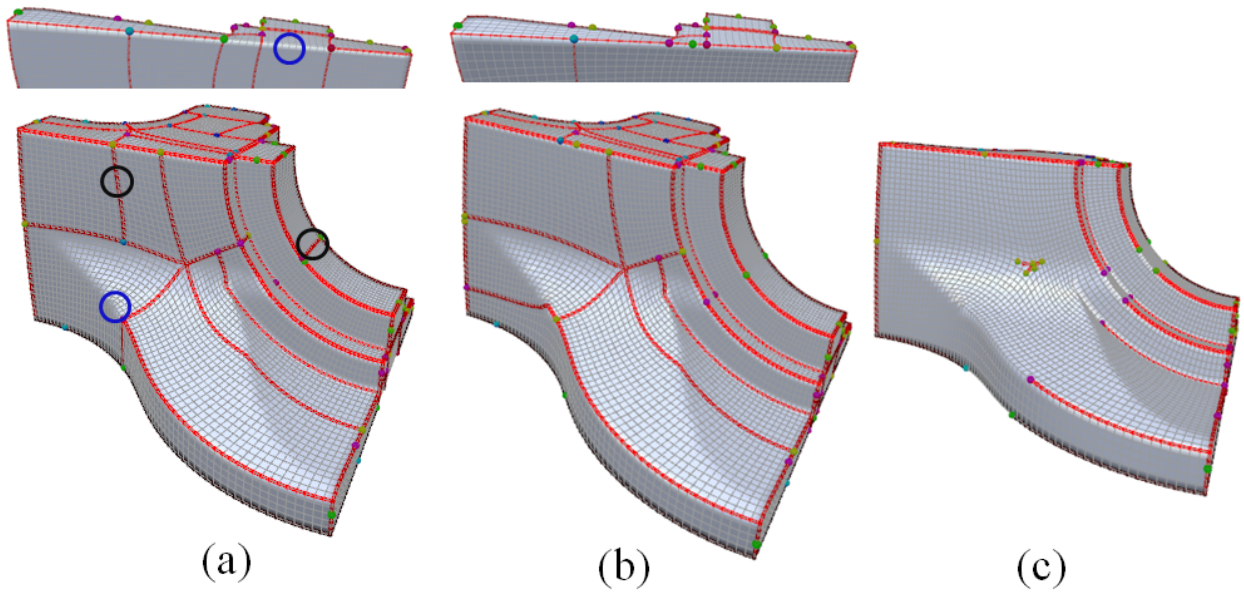


Figure 4.4: The SC algorithm ((b), cost $G_W = -1337.37$) generates partitioning different to that of the MCG algorithm ((a), $G_W = -1230.94$). Particles (shown as colored dots) move faster in highly curved areas and more slowly in smooth regions (c).

the SC algorithm does not produce undesirable partitioning shown by black circles in (a). The global cost G_W after applying the SC algorithm is also lower than that of the MCG algorithm.

Implementation of MCG tracing. Particles are placed on extraordinary vertices and moved one by one sequentially in each time step. The order to move particles is determined by the value of the "*distance remaining to the next vertex*/*particle speed*" ratio. Particles with smaller ratios are moved first. At the initial step, the order is random. During each time step, if a particle reaches the vertex of the edge, the speed is revised using the next edge, and the particle continues tracing on the next edge. A particle that meets already-traced vertices or mesh boundaries is assigned as *stopped* and positioned on the vertex. If two particles collide on an edge, both of them are stopped at the edge vertex nearer to the colliding position.

4.2.2.2 Path Flipping (PF) Algorithm:

Paths are often blocked by others to form junctions. This section describes improvement of paths obtained using the SC algorithm based on local path flipping (PF) operations for the blocked and blocking paths. Note that intersections of paths occur only at ordinary vertices to form either T or + junctions.

Several paths are shown in Fig. 4.5 (a). The vertically running path $\hat{W}$ can be represented as

a sequence of vertices along the path $\hat{W} =< \hat{v}_0, \hat{v}_1, \ldots, \hat{v}_m >$, where $\hat{v}_0$ is the seed vertex and $\hat{v}_m$ is the end vertex. As paths between two extraordinary vertices are not considered, $\hat{v}_m$ should not be an extraordinary vertex. Several paths with end vertices on $\hat{W}$ are blocked by it. Such paths are denoted with reference to the indexes of their end vertices. For instance, path W_i is blocked by $\hat{W}$ at $\hat{v}_i$.

Figure 4.5 (b) shows PF operation acting at a T-junctions on the k-th vertex $\hat{v}_k$ of $\hat{W}$. It truncates $\hat{W}$ at $\hat{v}_k$ and unlocks its blockage against W_k and W_j. MCG tracing is then applied to these unlocked paths, which will subsequently be extended. However, W_l and W_l' in the figure will not extend because they collide even after PF operation.

PF operation for the path $\hat{W} =< \hat{v}_0, \hat{v}_1, \ldots, \hat{v}_m >$ at its vertex $\hat{v}_k (\neq \hat{v}_0)$ is defined by the following procedure:

1. $\hat{W}$ is truncated to $\hat{W} =< \hat{v}_1, \hat{v}_2, \ldots, \hat{v}_k >$.

2. MCG tracing is applied to paths whose end vertices are included in $\{\hat{v}_k, \ldots, \hat{v}_m\}$.

It should be noted that $\hat{v}_k$ can be the end vertex ($\hat{v}_m$) but not the seed vertex $\hat{v}_0$. This PF operation is general enough to be applied to all junction types rather than being limited to T-junctions. Below, PF operations for these junctions are outlined. We first classify the cases into (a) those involving flipping at an intermediate vertex ($\hat{v}_k \in (\hat{v}_1, \ldots, \hat{v}_{m-1})$) and (b) those with flipping at the end vertex ($\hat{v}_k = \hat{v}_m$).

Flipping at the Intermediate Vertex. The PF operation discussed above is typical to this case. Another possible case involves a + junction where two paths have the same end vertex $\hat{v}_k$ on $\hat{W}$, as shown in Fig. 4.5 (c). In this case, the same PF operation can be applied (Fig. 4.5 (d)) where two colliding paths will not move after flipping operation.

Flipping at the End Vertex $\hat{v}_m$. In PF operation, only paths whose end vertices are on $\hat{W}$ are handled. Accordingly, cases to be considered at the end vertex are those where the path $\hat{W}$ meets the end vertices of other paths. Such cases arise when the end vertices of two or more paths coincidentally collide during SC tracing.

There is no need to consider the case of two paths because they do not form a junction when they meet from opposite directions and because one of the end vertices will move away when they meet perpendicularly in line with the right-hand rule of SC tracing.

Figure 4.6 (a) shows a case involving three paths where two vertical paths collide at a vertex and block another path coming into the vertex from the left. PF operation can also be applied to such coincident junctions with the condition $\hat{v}_m = \hat{v}_k$. Here, the end vertex of $\hat{W}$ is infinitesimally truncated to prevent blocking at $\hat{v}_k$ by $\hat{W}$ (Fig. 4.6 (b)).

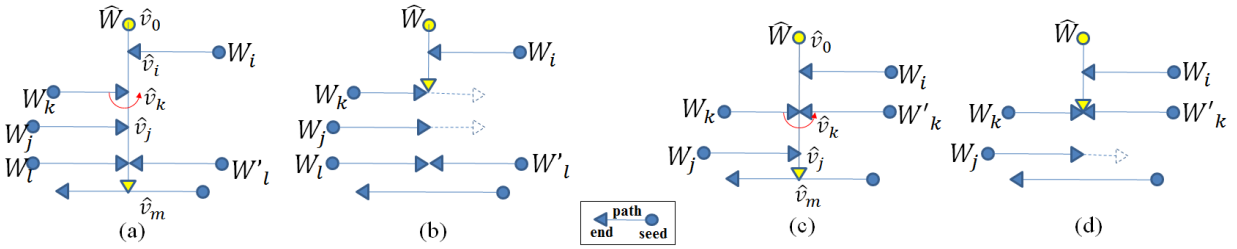


Figure 4.5: Paths (a) before and (b) after a PF operation at the vertex $\hat{v}_k$. Paths (c) before and (d) after PF operation at a + junction on the vertex $\hat{v}_k$.

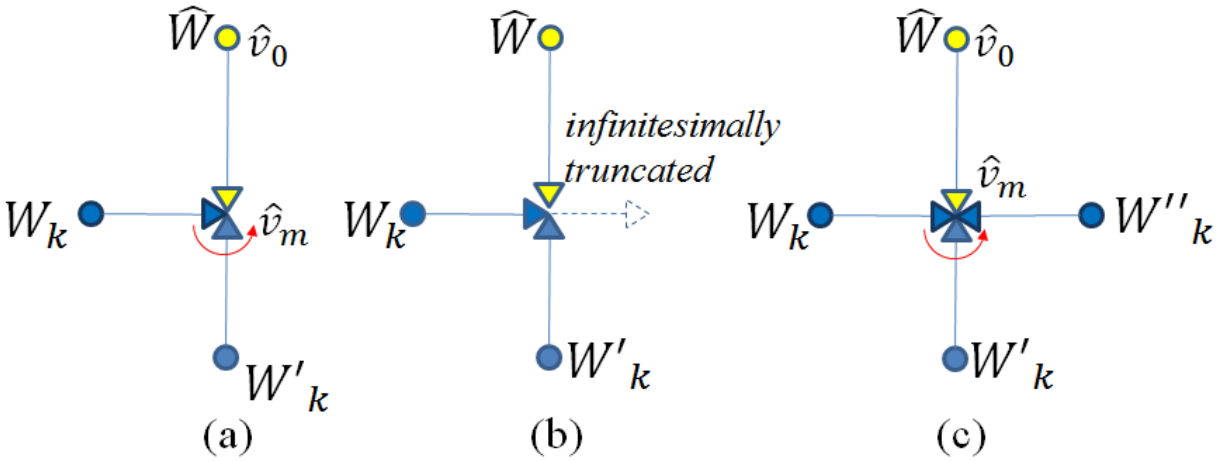


Figure 4.6: Paths (a) before and (b) after PF operation at a coincident T-junction on the vertex $\hat{v}_k$. The end vertex of $\hat{W}$ is infinitesimally truncated to prevent blocking at $\hat{v}_k$ by $\hat{W}$. (c) Four paths colliding at a vertex. PF operation can be applied, but no movement along these paths is possible because the blocking relationship cannot be resolved.

Figure 4.6 (c) shows a case involving four paths colliding at the end vertex. In this case, PF operation can be applied as in the three-path case, but movement along these four paths is not possible afterward because the operation cannot resolve their blocking relationship. Consequently, this case can be ignored.

Figure 4.7 shows examples of PF operations for both T-junction (b, c, d) and + junction (e, f, g) cases. After PF operation at the vertex $\hat{v}_{k1}$, the path $\hat{W}_1$ becomes shorter, while the paths W_{j1} and W_{k1} becomes longer. PF operation at the vertex $\hat{v}_{k2}$ results in extension of the path W_{j2} and shortening of the path $\hat{W}_2$.

Optimization with PF Operations. The proposed PF algorithm repeatedly applies the PF operations to junctions. The problem here is to find the order of PF operations that will yield partitioning with minimum cost. However, motorcycle graphs have very restricted structures and

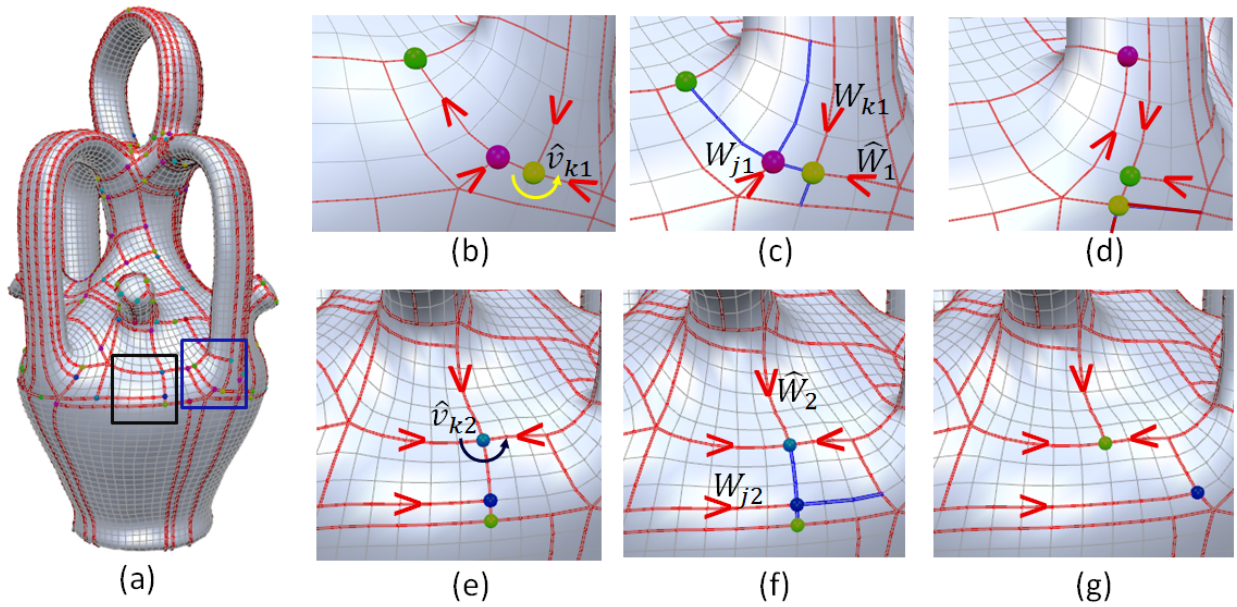


Figure 4.7: (a) Quadrilateral mesh model of a botijo container showing the traced paths (in red) and placed particles (colored dots). Images (b) to (d) show close-ups of the portion enclosed in the blue rectangle for PF operation at a T-junction. (b) Path configuration before PF operation at v_{k1} (c) Paths affected by the operation are shown in blue. $\hat{W}_1$ becomes shorter, while W_{j1} and W_{k1} become longer. (d) Path configuration after PF operation. Images (e) to (g) show close-ups of the portion enclosed in the black rectangle for PF operation at a + junction. (e) Path configuration before PF operation at v_{k2} (f) Paths affected by the operation are shown in blue. $\hat{W}_2$ becomes shorter, while W_{j2} becomes longer. (g) Path configuration after PF operation.

optimizing them is inherently a difficult problem. The change in the partitioning resulting from a single PF operation is often large which reduces PF performance in intensifying the search. Consequently, it is often necessary to go through bad solutions when transforming a good solution into a better one through a sequence of PF.

We have tested two naive approaches; random search and greedy search. In the former, a junction is randomly selected and PF operation is applied. If the post-operation cost G_W is higher, the result is discarded and another junction is tried. In the latter, all possible PF operations are first enumerated, and those that reduce the cost the most are selected. The process is reiterated until no further cost reduction is possible. According to our tests, greedy search results in convergence with a smaller number of PF iterations, but both approaches produce almost similar partitions with similar costs. Accordingly, we use greedy search.

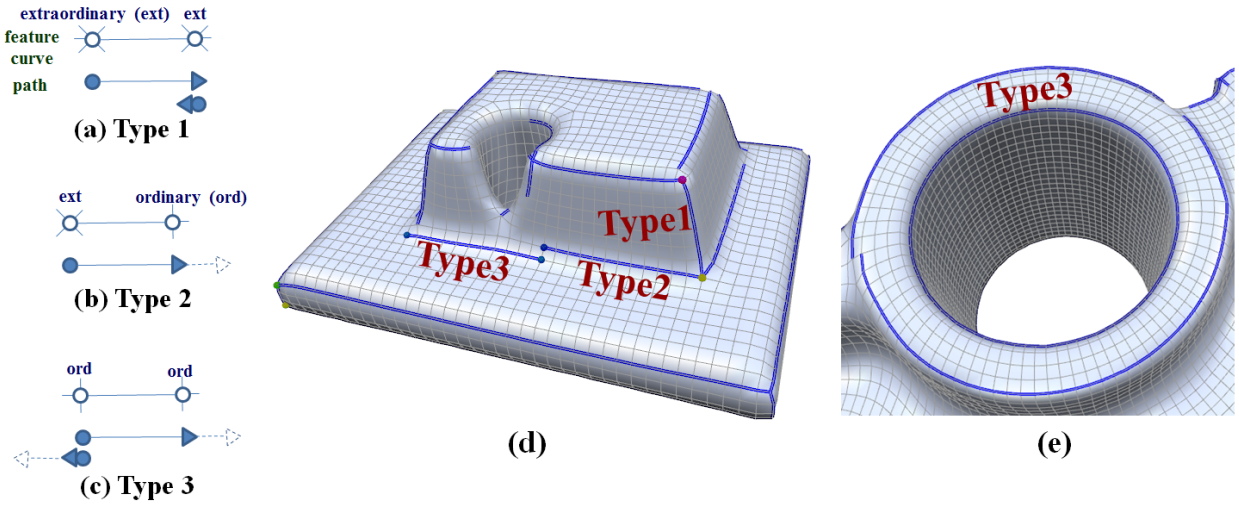


Figure 4.8: Feature curves. (a) **Type 1**: Starting from an extraordinary vertex and ending at an extraordinary vertex. (b) **Type 2**: Starting from an extraordinary vertex and ending at an ordinary vertex. (c) **Type 3**: Starting from an ordinary vertex and ending at an ordinary vertex. (d) The three feature curve types shown with added paths (in blue) and particles (colored dots) (e) The closed feature curve of Type 3.

4.2.2.3 Feature Curve (FC) Algorithm:

A feature curve is a sequence of edges with large dihedral angles. It has the same characteristics as a path generated by the MCG algorithm, i.e., it goes straight along the mesh. However, while each path created by the MCG algorithm is connected to one or two extraordinary vertices at its terminals, feature curves may not be connected to any extraordinary vertices, meaning that both terminal vertices of such curves can be regular. Accordingly, not all feature curves can be involved in the MCG algorithm.

The proposed approach involves the extraction of feature curves and their integration to paths using the SC algorithm. With this algorithm, MCG tracing is begun from extraordinary vertices to create a structured partition. Such partitions can be generated even by adding ordinary vertices as seeds to the SC algorithm because no other extraordinary seeds are introduced during the particle placement step. Accordingly, if the extracted feature curves are given to the initial state of the SC algorithm, and if MCG tracing is applied to them and to seeds at extraordinary vertices, the curves will be automatically included in the paths forming a structured partition. A method of extracting feature curves and details of their integration are outlined below.

Feature Curve Extraction. All edges whose dihedral angles are greater than the flatness threshold ρ are first collected. This set, denoted as H, is sorted by dihedral angle in descending

order so that priority is given to edges with larger dihedral angles and feature curves are grown from them first. Priority is also given to edges connected to extraordinary vertices by collecting them in the front part of H. The first edge e is selected from H, and a path including this edge is generated by tracing edges from it, as with the MCG algorithm. If the two end vertices of e are v_1 and v_2, MCG tracing is applied with the two seeds $< v_1, e >$ and $< v_2, e >$, but is restricted to the edges of H. The resulting two paths are merged into a single one to define a feature curve. If its length (i.e., the number of edges) is less than the threshold τ, it is discarded. To avoid the generation of too many feature curves close to each other, edges of H sharing a quad element with feature curve edges can be tagged. The edges of the traced feature curve and the tagged edges are then erased from H. A set of feature curves is obtained by repeating this procedure until H is empty.

There exist several robust feature detection methods which can be used. For a triangle mesh with severe noise, mixed integer quadrangulation (MIQ) is not be able to generate a quadrilateral mesh with sufficient quality to be used in our method. In other words, smoothing the noisy mesh is needed before quadrangulation so that MIQ can generate a quadrilateral mesh with sufficient quality for our method. It means that we can assume no severe noise remain in the input quadrilateral mesh. MIQ also itself has some smoothing effect on the mesh, which even reduce the noise level of the mesh.

Feature Curve Integration. Each feature curve can be represented as a sequence of vertices and edges $< v_1, e_1, \ldots, e_{m-1}, v_m >$. Paths are added to the seeds of the SC algorithm depending on the curve type as discussed below. Once the particles are placed on the seeds and the ends of the feature curves, the SC and PF algorithms are applied in the same way. This is referred to as feature curve (FC) algorithm.

Below is a summary of the three feature curve types (1, 2, and 3) and the paths added for each type (see Fig. 4.8 (a, b, c)).

1. **[Type 1]** v_1 and v_m are both extraordinary: the path $< v_1, e_1, \ldots, e_{m-1}, v_m >$ is added. This path is not extended by MCG tracing because both terminals are at extraordinary vertices.

2. **[Type 2]** v_1 is extraordinary and v_m is ordinary: the path $< v_1, e_1, \ldots, e_{m-1}, v_m >$ is added. Particle tracing starts from v_m in MCG tracing.

3. **[Type 3]** v_1 and v_m are ordinary: two paths, $< v_1, e_1, \ldots, e_{m-1}, v_m >$ and $< v_m, e_{m-1} >$, are added. Two particles start tracing from v_1 and v_m.

Type 3 can include a closed feature curve for which v_1 is defined as an edge where tracing

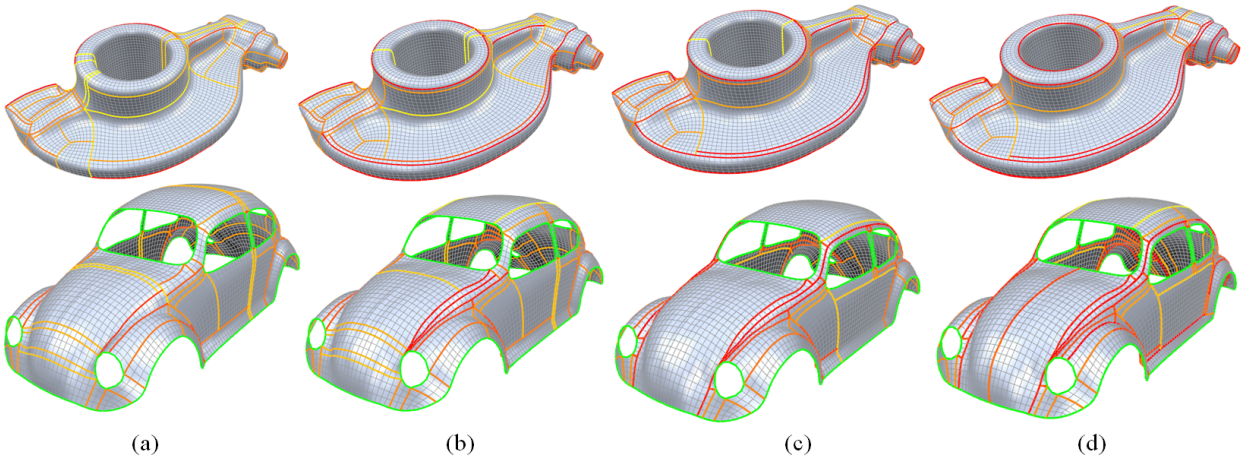


Figure 4.9: Partitions after applying (a) the MCG algorithm ($G_W/N_W = -0.286$ (top, N_W = the total number of paths), $G_W/N_W = 1.170$ (bottom)), (b) the SC algorithm ($G_W/N_W = -1.648$ (top), $G_W/N_W = 0.833$ (bottom)), (c) the PF algorithm ($G_W/N_W = -2.656$ (top), $G_W/N_W = -0.235$ (bottom)), and (d) the FC algorithm ($\tau = 15$ and $G_W/N_W = -3.126$ (top), $\tau = 8$ and $G_W/N_W = -0.965$ (bottom).

was started. No special care is needed, but MCG tracing from the seed $< v_m, e_{m-1} >$ does not proceed because it faces another path $< v_1, e_1, \ldots, e_{m-1}, v_m >$. Figure 4.8 (d, e) shows examples of extracted feature curves.

4.2.3 Results and Discussion

The proposed algorithms are implemented, and experiments are conducted using four quadrilateral meshes generated based on the mixed integer quadrangulation method [9] for a drill hole (see Fig. 4.2), a rockerarm, a Beetle and a bottle. Figure 4.9 shows a comparison of the rockerarm and the Beetle models with (a) the MCG algorithm, (b) the SC algorithm, (c) the PF algorithm and (d) the FC algorithm. It can be seen that the path cost G_W decreases from (a) to (d). Highly curved parts of the models are also captured better in this order. The paths in (d) are considered suitable for reverse engineering because they capture almost all character lines and they are located in highly curved areas such as fillet regions.

Thresholds: The proposed algorithm is automatic once the user tunes the thresholds. The flatness threshold ρ is set to 0.3 for the bottle model, and to 0.4 for the others. The minimum feature curve length τ is adjusted depending on the mesh resolution (see the figure captions for the assigned values of τ).

The threshold α is set to 0.4 for all the models. ϵ is set to 1.0 for models those with sharp

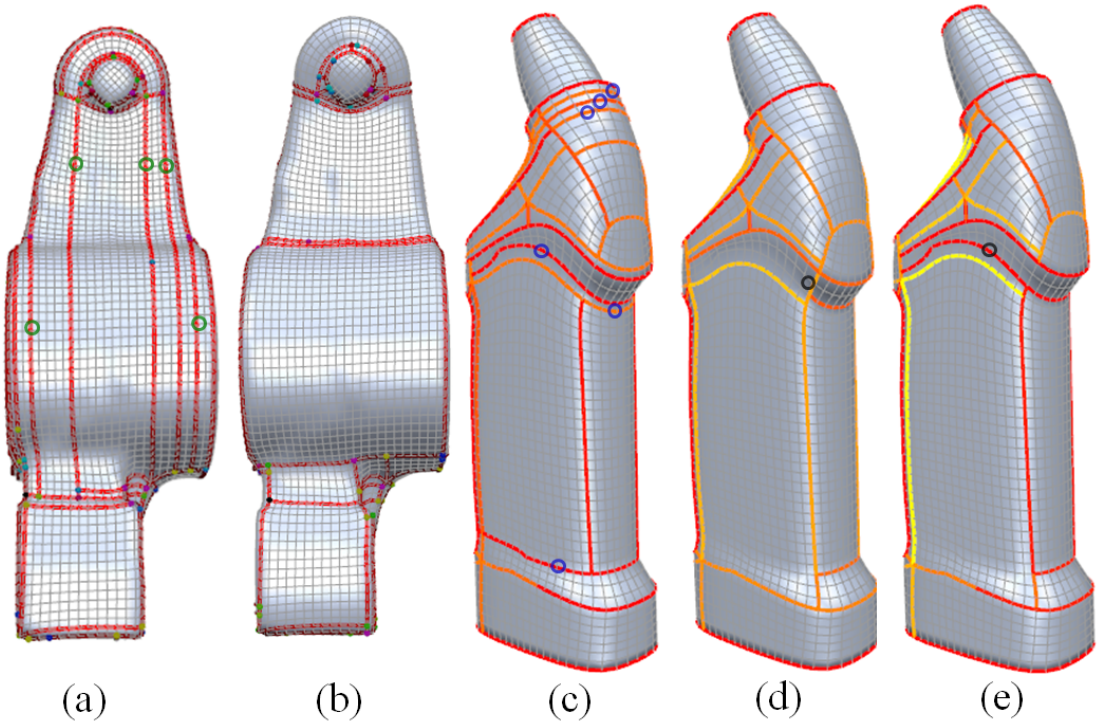


Figure 4.10: Paths of rockerarm model after path flipping operations when (a) $\alpha = 0.0$ (b) $\alpha = 0.4$, $\epsilon = 0.2$. To minimize the sum of edge curviness, undesirable low curved long paths are produced for $\alpha = 0.0$. Setting α to 0.4 reduces the number of such paths in (b); Partitioning of the bottle model ($\tau = 15$) with the parameters (c) $\alpha = 0.0$, (d) $\alpha = 0.4$, and (e) $\alpha = 0.8$.

features such as the drill hole, the fan disk and the bottle. For the Beetle and the rockerarm, it is defined as 0.2. Figure 4.10 (c, d, e) shows partitioning for the bottle model with the parameter α (for computation of path costs) set to 0.0, 0.4 and 0.8. Undesired low curved long paths (depicted by blue circles) are produced with $\alpha = 0.4$. According to our experience, fine-tuning does not lead to further improvement and parameters are easy to tune.

τ determines the length of the feature curve that will be extracted. Assigning small values to this parameter may produce many feature curves especially for models having noise. While constraining these curves on partition boundaries, large number of partitions will be generated which is undesirable. Therefore, determining the value of τ according to the mesh resolution using heuristics will generate appropriate feature curves. ρ is used to detect the unflat regions and setting it to small values will detect many unflat edges, and therefore large number of partitions will be generated. The use of parameter ϵ discourages the generation of paths that are less curved and longer. Otherwise, thin partitions may occur which is undesirable. Setting to very large values will avoid such partitioning. The parameter α adjusts the relation between the *dihedral*

angle maximization and *non-generation of less curved paths* terms in the cost function. Setting it to an appropriate value will produce a partitioning balanced in terms of these two criteria. We believe that appropriate values can easily be set to our parameters after performing several value assignment attempts.

Path Coloring: A color is assigned for each path depending on the path cost in Figures 4.2, 4.9, 4.10 ((c) to (e)) and 4.11. The color is interpolated between yellow and red for path costs between -10.0 and 10.0.

Importance of Initial Path Layout: Starting with a high-quality initial path layout involving feature curves such as that of the SC algorithm (rather than the MCG algorithm) was found to lead to convergence with a low number of PF iterations. We have also observed (in some cases) that conjunct use of the SC and the PF algorithms generates better results. Starting a better quality path layout (generated by the SC algorithm) results in the partitions where as many as feature curves are located on the boundaries.

Performance: A 2.27 GHz PC was used for the experiments in this study. The PF algorithm and the FC algorithm take negligible amounts of time in all cases. The approximate processing times for the SC algorithm are 5 seconds for the rockerarm model, 1 second for the drill hole model, 2 seconds for the Beetle model, 2 seconds for the bottle model. The time taken for the SC algorithm depends on the number of edges of the model.

Removal of Flat Paths: By using a simple post processing, the partitioning generated with the proposed approach can be improved by removing paths whose costs exceeds the user-defined parameter η. Similar to the PF algorithm, a greedy approach is chosen such that the paths having higher costs are removed first. During this process, it is not desirable to regenerate extraordinary vertices. To achieve this, no two consecutive edges at an extraordinary vertex are removed. Moreover, removing a path may form a dangling path which is extended by applying MCG tracing. After this process if resulting configuration result with a higher cost, we do not remove the path and extend another one. Note that it is possible to propose much more sophisticated algorithms in this step. Figures 4.2 (f) and 4.11 show partitions seen after the removal process.

B-spline Surface Fitting: Quadrilateral mesh of [9] has a good match with B-spline surfaces since each quad edge generally has equal length meaning that the data is regularly sampled and therefore, uniform parameter $u - v$ values can be assigned to each quad points. Using this intrinsic property of quadrilateral meshes, each quad partition is fitted with *uniform* bi-cubic B-spline surfaces.

A quad partition has $(m + 1) \times (n + 1)$ data points $P_{i,j}$ where $0 \leq i \leq m$ and $0 \leq j \leq n$ representing indices of columns and rows respectively. *Two-stage curve blending* is performed

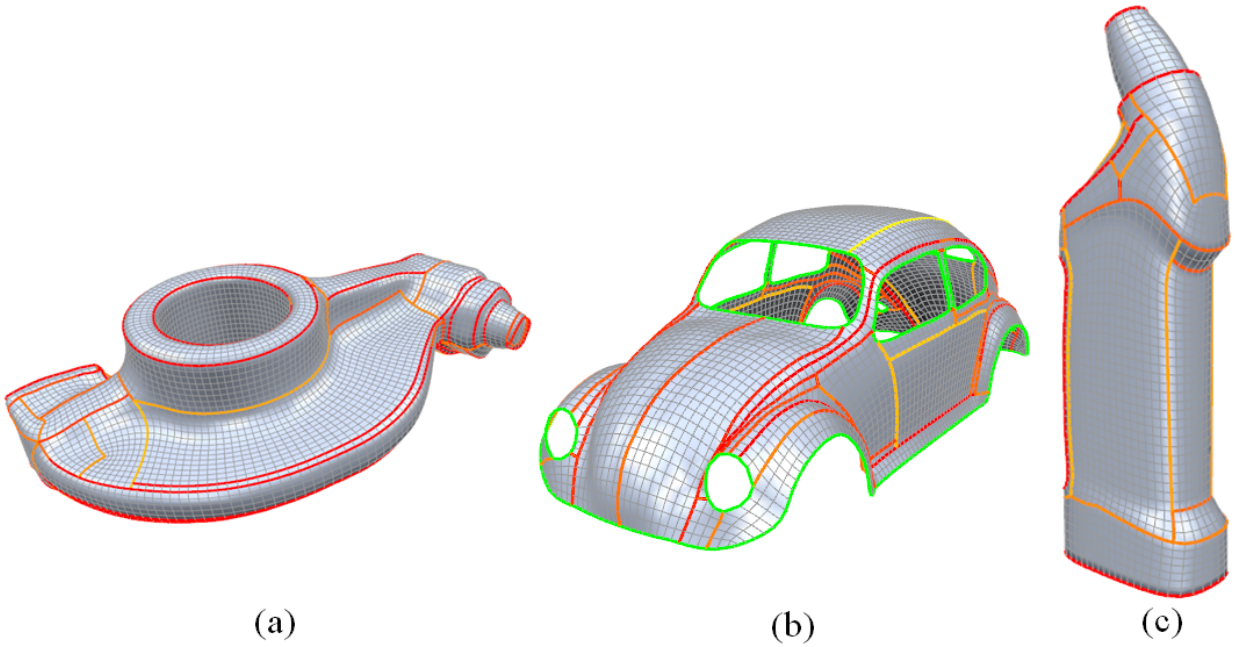


Figure 4.11: Removal of flat (less curved) paths: (a) rockerarm model ($\eta = 0.3$) (b) Beetle model ($\eta = 0.1$), (c) bottle model ($\eta = 0.1$).

to interpolate the data on these columns and rows. Each row is blended to form a curve, then these curves are blended to form a surface. *Uniform* cubic B-spline curves are fitted for all rows, and the following linear system is solved under the free-end condition to obtain control points $< C_{0,j}, C_{1,j}, C_{2,j}, \ldots, C_{m+1,j}, C_{m+2,j} >$:

$$\begin{bmatrix} 1 & -1 & 0 & 0 & \cdots & 0 \\ 1 & 4 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 4 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 4 & 1 \\ 0 & \cdots & 0 & 0 & -1 & 1 \end{bmatrix} \times \begin{bmatrix} C_{0,j} \\ C_{1,j} \\ C_{2,j} \\ \vdots \\ C_{m+1,j} \\ C_{m+2,j} \end{bmatrix} = \begin{bmatrix} 0 \\ 6P_{0,j} \\ 6P_{1,j} \\ \vdots \\ 6P_{m+1,j} \\ 0 \end{bmatrix}.$$

The obtained control points $C_{i,j}$ are again blended for all columns, and the following linear system is solved under the free-end condition to obtain final control points $< V_{i,0}, V_{i,1}, V_{i,2}, \ldots, V_{i,n+1}, V_{i,n+2} >$

of uniform cubic B-spline surface:

$$\begin{bmatrix} 1 & -1 & 0 & 0 & \cdots & 0 \\ 1 & 4 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 4 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 4 & 1 \\ 0 & \cdots & 0 & 0 & -1 & 1 \end{bmatrix} \times \begin{bmatrix} V_{i,0} \\ V_{i,1} \\ V_{i,2} \\ \vdots \\ V_{i,n+1} \\ V_{i,n+2} \end{bmatrix} = \begin{bmatrix} 0 \\ 6C_{i,0} \\ 6C_{i,1} \\ \vdots \\ 6C_{i,n} \\ 0 \end{bmatrix}.$$

Figure 4.12 shows B-spline surfaces, and better surface quality is expected when our partitions are fitted with T-splines.

Comparison: There are several existing methods [19, 39, 7, 11, 22] whose partitions can be suitable for B-spline surface fitting. Compared to other methods, the generated partitions of [22] are expected to have higher surface quality since feature curves are located on the partition boundaries as done in our method. In order to analyze the surface quality, we first fit the partitions of [22] and of our method with B-spline surfaces. Then these surface models are compared with the input quadrilateral mesh. Fig. 4.13 shows deviations from the input quadrilateral mesh and the resulting surface models. The surfaces for our partitions have less deviation from the input quadrilateral mesh than that of [22], since our method improves the initial partitioning by local path flipping operations.

Fig. 4.14 shows the surface deviations for the partitions of [19] (a) and for our partitions (b). Higher deviations are seen in the portions where feature curves reside inside the boundary (marked with black circles of Fig. 4.14 (a)). However, if these curves are located on the partition boundaries the surface fitting quality gets better and thus smaller deviations occur (see Fig. 4.14 (b)). The blue circle in Fig. 4.14 (b) shows a portion having high deviation in the generated surface of our partition. This can be eliminated by allowing several feature curves next to each other by not tagging the neighborhood of the feature curves (refer Section 4.3).

Our method generates surfaces with less deviation as motivated in Section 1. Fig. 4.15 shows the partitions generated using [39, 49, 7, 11]. Highly curved regions (marked with black circles) can be seen inside the partition boundaries which will be problematic for the B-spline surface fitting.

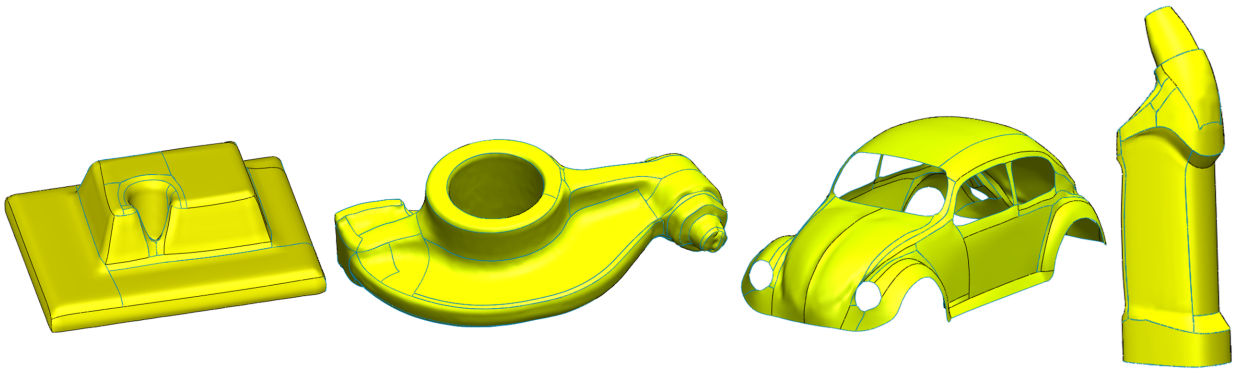


Figure 4.12: B-spline surfaces for the partitions generated using our method.

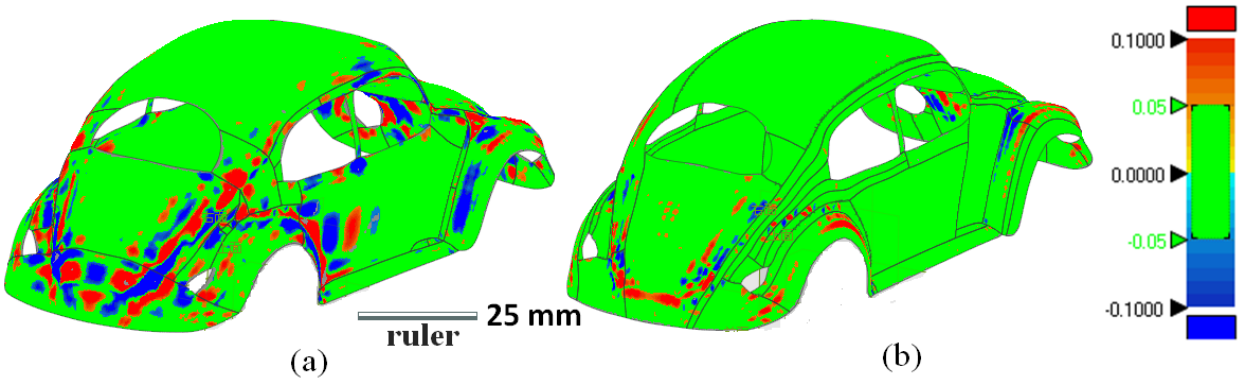


Figure 4.13: Deviations from the input quadrilateral mesh for the surfaces of partitions generated using: (a) the method of [22] (b) our method. Since our method captures the highly curved parts well, there is less surface fitting error compared to [22].

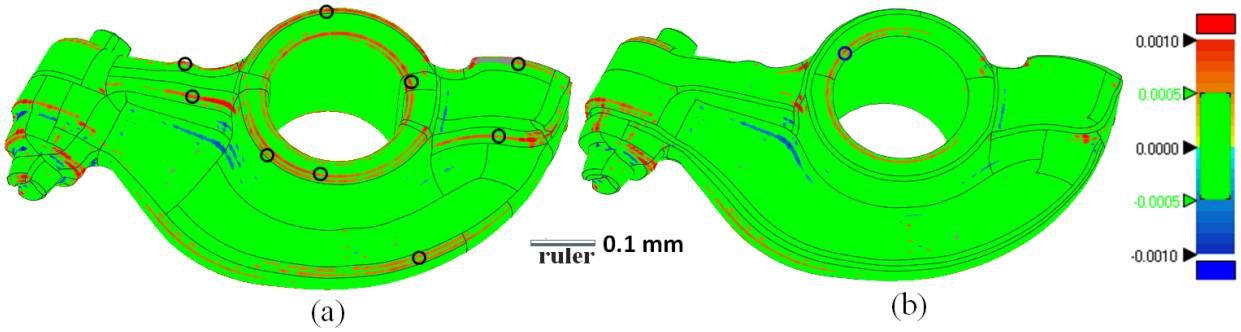


Figure 4.14: Deviations from the input quadrilateral mesh for the surfaces of partitions generated using: (a) the method of [19] (b) our method. Regions marked with black circles (top image in (b)) have large deviations since highly curved regions reside inside the B-spline surfaces and are not represented well with smooth surfaces.

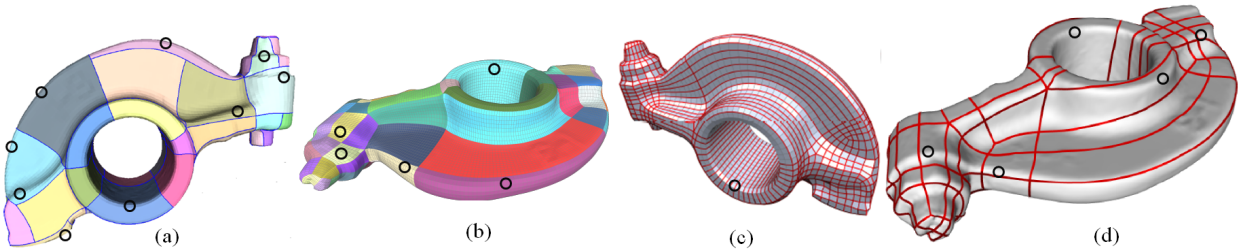


Figure 4.15: Partitions generated using the method of: (a) [39] (b) [49], (c) [7], (d) [11] (images taken from [39, 49, 7, 11]). Highly curved regions (marked with black circles) can be seen inside the partition boundaries which will be problematic for the B-spline surface fitting.

4.3 The MCG Enumeration Approach

The method proposed in this section is intended to enumerate all motorcycle graphs of a given quadrilateral mesh and then to find the optimum motorcycle graph for reverse engineering. However, this enumeration has very high computational cost because its complexity is quasi-exponential. As a result, enumerating all solutions in the case of Beetle model shown in Fig. 4.16 (a, b) takes more than a week according to our tests. Accordingly, the given mesh (a) is cut into several sub-meshes (cuts are shown in red) to reduce the computational time required and, all solutions are then enumerated for the motorcycle graphs in each sub-mesh to find the global optimum one. By applying successive enumeration steps in each sub-mesh separately, a motorcycle graph with proximity to the global optimum (quasi-optimum) is found. For the Beetle model, enumeration process takes about one second (see solution in Fig. 4.16 (b)) using such approach. Feature curve extraction and integration method is also proposed to trace highly curved parts that are difficult to capture with motorcycle graph edges.

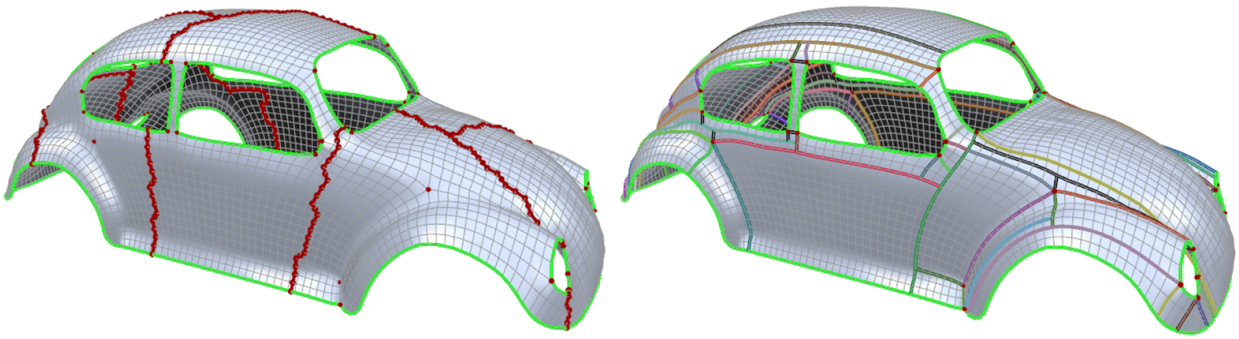


Figure 4.16: The computational cost of enumerating all motorcycle graphs that can be generated from a quadrilateral mesh model (with 48 extraordinary vertices) of the Beetle model is so high that it would take weeks. Accordingly, the mesh is cut into several sub-meshes (a) to reduce the time required. Enumeration is performed in each sub-mesh separately with successive steps, and a quasi-optimum graph (b) is obtained within about one second.

4.3.1 Enumerating Motorcycle Graphs on Quadrilateral Meshes

Base Complex vs. Motorcycle Graph: A quadrilateral mesh $Q < V, E, F >$ consists of a set of quadrilateral faces F with edges E and vertices V. In the approach by Eppstein et al. [19], edges are traced on Q by moving particles to generate tracks. The particles are placed on extraordinary vertices whose number is equal to the valence of the placed vertices. They move outward from the

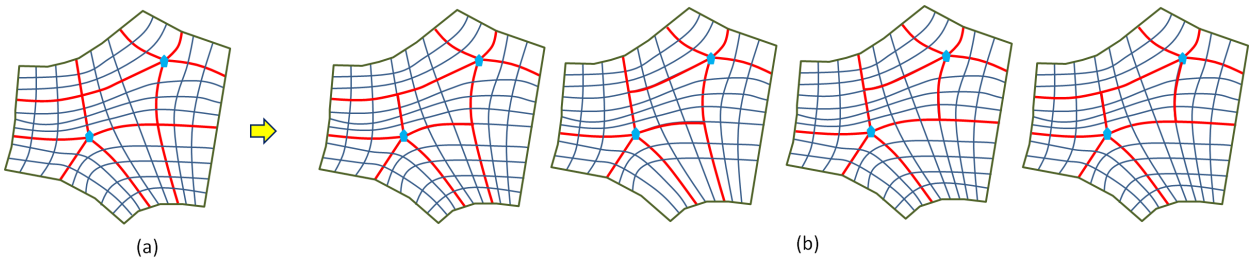


Figure 4.17: The base complex (a) of a given quadrilateral mesh is defined as the union of separatrices (in red) starting at an extraordinary vertex (in blue) and ending at an extraordinary vertex or a boundary vertex. Four different motorcycle graphs (b) with boundaries (in red) generated from a given mesh.

extraordinary vertices and trace edges in a straightforward manner, moving straight along edges and on to the opposite edge at (ordinary) vertices. Particles stop when they reach extraordinary vertices or at the mesh boundary, and the edges they trace generate tracks on Q. Such tracks form *separatrices*, and their union is known as a *base complex* (Fig. 4.17 (a), red). For meshes without boundaries, a particle starting at an extraordinary vertex v_1 ends at another extraordinary vertex v_2, while one starting at v_2 ends at v_1. These two particles trace two separatrices between v_1 and v_2 in opposite directions.

The following rules are added to the above particle tracing for motorcycle graph generation: Particles stop if they meet already traced vertices or a mesh boundary. The right-hand rule is also applied when two particles meet simultaneously at a vertex.

The tracks generate structured partitions under these rules. The graph structure of such partitions is known as *motorcycle graph* as proposed by Eppstein et al. [19]. These graphs have different topological structures to those of base complexes. T-joints can be seen in motorcycle graphs such that the intersection of two neighboring partitions is not the whole edge or vertex. Allowing T-joints in this way makes it possible to represent the mesh with a smaller number of partitions.

Notations and Definitions: The notations described here are based on a quadrilateral mesh that is homeomorphic to a two-manifold without a boundary. Figure 4.18 shows two separatrices and their local coordinate systems with notations. Here let the i-th separatrix in a base complex be called S_i (in blue). The vertex on S_i whose local coordinate on S_i is j is denoted as vertex (i, j), and opp(i) (in red) is the index of the opposite separatrix of S_i. left(i, j) (in green) and right(i, j) (in light blue) are the indexes of the left and right separatrices of S_i at vertex(i, j) which should be an internal vertex of S_i. The local coordinate on S_i of vertex v is denoted as $\varphi_i(v)$, and v should be

vertex S_i. length(S_i) is the length of S_i which is the total number of vertices in S_i.

The 1D local coordinate system for the separatrix S_i is based on the notations provided, and the position (coordinate) of a particle on S_i is represented by the integer x_i. The initial endpoint of the separatrix S_i has the coordinate 0 in the coordinate system of S_i, and its terminal endpoint has the coordinate length(S_i). The coordinate value of a point increases by one when it moves to the next vertex away from the origin.

Suppose the input base complex has n separatrices, and a particle is placed on each one. The particle z_i starts at the initial endpoint of the separatrix S_i and moves along S_i until it is stopped. The point at which a particle stops is called the *crashing point*. The crashing point of z_i is denoted as κ_i. The track of z_i, denoted as T_i, is the path on S_i starting at the initial endpoint of S_i and ending at the crashing point κ_i. If each particle track T_i (is defined as W_i in Section 4.1) satisfies the following conditions, the set of particle tracks $\{T_i\}^n{}_{i=1}$ is considered valid:

A1. (No overlapping tracks.) $\boxed{x_i \leq \text{length}(S_i) - x_{\text{opp}(i)}}$.

A2. (No crossing tracks.) For each internal vertex v of T_i, $\boxed{x_{\text{left}(i,j)} \leq \varphi_{\text{left}(i,j)}(v)}$ and $\boxed{x_{\text{right}(i,j)} \leq \varphi_{\text{right}(i,j)}(v)}$, where $j = \varphi_i(v)$.

A3. (No inactive particles.) If $x_i = 0$, then $\boxed{x_{\text{opp}(i)} = \text{length}(S_i)}$.

A4. (No dangling tracks, no L-junctions.) If $0 < x_i < \text{length}(S_i) - x_{\text{opp}(i)}$, then one of the following holds.

(1) $\boxed{x_{\text{opp}(i)} = \text{length}(S_i) - x_i}$.

(2) $\boxed{x_{\text{opp}(i)} < \text{length}(S_i) - x_i}$ and $\boxed{x_{\text{left}(i,j)} \geq \varphi_{\text{left}(i,j)}(\kappa_i) + 1}$.

(3) $\boxed{x_{\text{opp}(i)} < \text{length}(S_i) - x_i}$ and $\boxed{x_{\text{right}(i,j)} \geq \varphi_{\text{right}(i,j)}(\kappa_i) + 1}$.

(4) $\boxed{x_{\text{opp}(i)} < \text{length}(S_i) - x_i}$, $\boxed{x_{\text{left}(i,j)} = \varphi_{\text{left}(i,j)}(\kappa_i)}$ and $\boxed{x_{\text{right}(i,j)} = \varphi_{\text{right}(i,j)}(\kappa_i)}$.

Otherwise the set is considered *invalid*. Figure 4.19 shows the five types of invalid particle tracks. A graph formed by valid particle tracks is called a *motorcycle graph*, and induces quadrilateral partitioning with T-junctions. Different particle tracks can form the same motorcycle graph as seen in Fig. 4.20. Note that, if a base complex has a separatrix with self-crossing (self-intersection at an internal vertex of a separatrix), invalid motorcycle tracks can be generated. To avoid such cases, these separatrices should be detected and truncated to the intersection vertex.

Motorcycle Graph Enumeration Algorithm: Let particles move at different speeds. The order of arrival at vertices changes, and thus the resulting motorcycle graphs also change. Figure

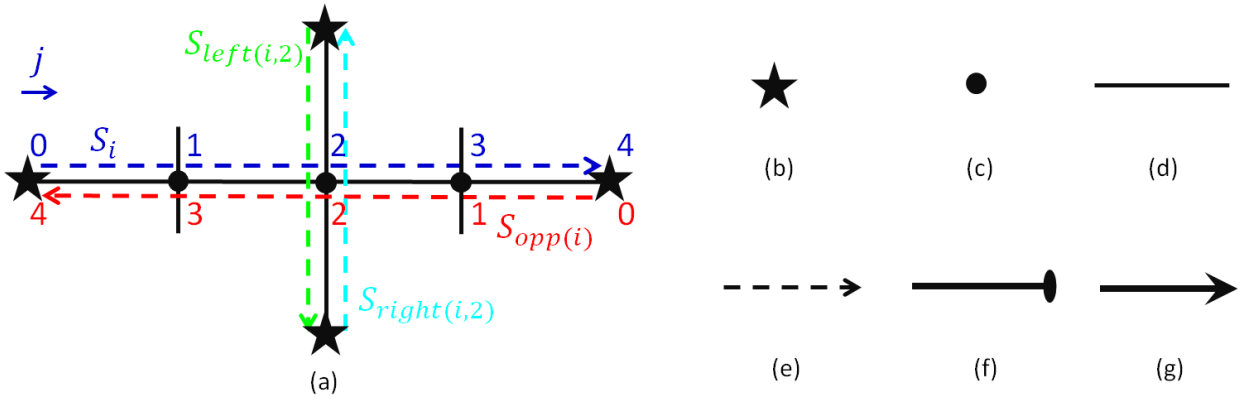


Figure 4.18: (a) Two separatrices and their local coordinate systems. The separatrix S_i shown in blue, the opposite separatrix opp(i) of S_i in red, the left separatrix (at $i = 2$) left($i, 2$) of S_i in green, and the right separatrix (at $i = 2$) right($i, 2$) of S_i in light blue. Symbols used in the figures: (b) Irregular vertex (c) Regular vertex (d) Undirected separatrix (e) Separatrix (f) Particle track (Arrow tip: crashing point) (g) Particle track (Crashing point not shown)

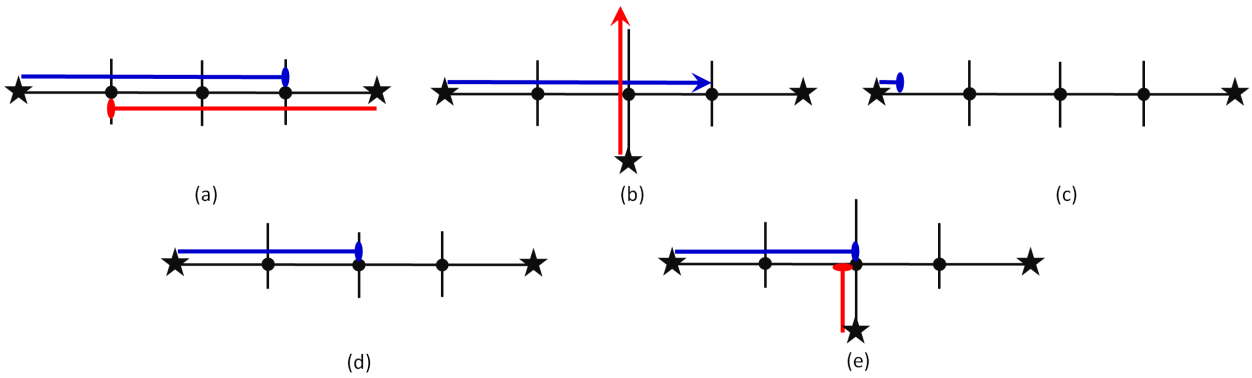


Figure 4.19: Invalid particle tracks: (a) Overlapping (b) Crossing (c) Inactive particle (d) Dangling track (e) L-junction

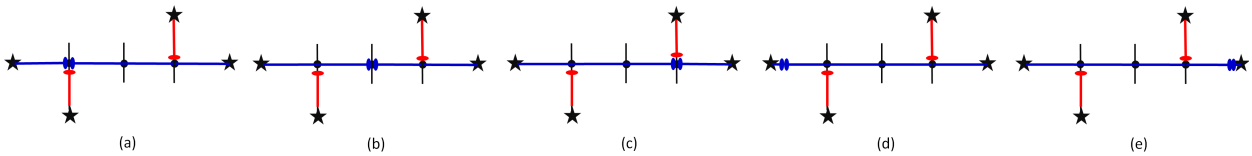


Figure 4.20: Different particle with the same motorcycle graph.

4.17 (b) shows four different motorcycle graphs generated from the same quadrilateral mesh (a), all of which are structured. These graphs can be enumerated so that the optimum one in terms of reverse engineering can be selected. As particle tracks determine this graph, particles need to be placed in appropriate positions to prevent invalid results such as non-motorcycle graphs. These positions must be at the vertices of the base complex.

Let the input base complex have n separatrices. As motorcycle graphs are formed by valid tracks that can be represented by the local coordinates of their crashing points, they can be enumerated by enumerating the coordinates $x_1, \ldots, x_n$. A naive method for enumerating valid tracks is to first enumerate all coordinate combinations and then filter out invalid tracks. This enumeration can be performed using a simple recursive procedure:

EnumerateCoordinates(Q) /* Q: a base complex with n separatrices. */

1. Call Enumerate(1, n).

Enumerate(i, n) /* Recursively enumerate coordinates $x_i, \ldots, x_n$. */

1. If $i \leq n$,

    For $j = 0, \ldots,$ length(S_i),

        Set x_i to j.

        Call Enumerate($i + 1$, n) recursively.

2. Otherwise,

    If the coordinates $x_1, \ldots, x_n$ represent valid tracks,

        Output them.

However, this naive method is inefficient because a large number of invalid tracks are unnecessarily enumerated before being eventually filtered out. The next section describes a method for avoiding such inefficiency.

Enumeration without Invalid Tracks: To efficiently enumerate motorcycle graphs, it is important to avoid invalid tracks. This is achieved by maintaining lower and upper bounds of coordinates to enforce validity conditions (A1)–(A4). The basic structure of this enumeration is the same as that of the naive method, and a recursive procedure is used to enumerate the coordinates. The difference is that the lower and upper bounds of the coordinates are maintained, and these bounds are used to prevent unnecessary enumeration.

Let $\underline{x}_i$ and $\overline{x}_i$ denote the lower and upper bounds of coordinate x_i, respectively. The modified versions of EnumerateCoordinates and Enumerate are as follows:

EnumerateCoordinates(Q) /* Q: a base complex with n separatrices. */

1. Initialize lower bounds:

   Set $\underline{x}_i$ to 0, for $i = 1, \ldots, n$.

2. Initialize upper bounds:

   Set $\overline{x}_i$ to length(S_i), for $i = 1, \ldots, n$.

3. Call Enumerate(1, n).

Enumerate(i, n) /* Recursively enumerate coordinates $x_i, \ldots, x_n$. */

1. If $i \leq n$,

   For $j = \underline{x}_i, \ldots, \overline{x}_i$,

   Set x_i to j.

   Update lower and upper bounds.

   Call Enumerate($i + 1$, n) recursively.

   Restore lower and upper bounds.

2. Otherwise,

   Output $x_1, \ldots, x_n$.

In the above pseudo-code of Enumerate, only one *enumeration branch* (i.e., only one recursive call to Enumerate) is described in the loop body for simplicity. However, the loop body actually contains multiple enumeration branches to enforce the validity conditions.

The lower and upper bounds are said to be *consistent* when each lower bound is lower than or equal to the corresponding upper bound. If the bounds are not consistent, there are no valid tracks in that enumeration branch. Whenever the bounds are updated, consistency is checked; if the check fails, enumeration is stopped for that branch. Maintaining the lower and upper bounds requires some care. When the coordinate x_i is set to j, $\underline{x}_i$ and $\overline{x}_i$ are also set to j to facilitate bound consistency checking. When a bound is updated, if the new bound is weaker than the old one,

the old one is kept. Restoring a bound means setting it back to the old one (i.e., that before the update). To efficiently support update and restore operations, stacks are used to represent bounds.

To enforce the validity conditions, the lower and upper bounds are updated in the following way, which is simply derived from (A1)–(A4):

B1. (Preventing overlapping tracks.) Suppose coordinate x_i is set to j. The upper bound $\overline{x}_{\mathrm{opp}(i)}$ is updated to $\mathrm{length}(S_i) - j$.

B2. (Preventing crossing tracks). Suppose coordinate x_i is setting to j. For $k = 1, \ldots, j-1$, $\overline{x}_l$ is updated to $\varphi_l(v)$ and $\overline{x}_r$ to $\varphi_r(v)$, where $v = \mathrm{vertex}(i,k)$, $l = \mathrm{left}(i,k)$ and $r = \mathrm{right}(i,k)$.

B3. (Preventing inactive particles.) Suppose the coordinate x_i is set to 0. The lower bound $\underline{x}_{\mathrm{opp}(i)}$ is updated to $\mathrm{length}(S_i)$.

B4. (Preventing dangling tracks and L-junctions). Suppose the coordinate x_i is set to j ($0 < j < \mathrm{length}(S_i)$). Let $v = \mathrm{vertex}(i,j)$, $l = \mathrm{left}(i,j)$ and $r = \mathrm{right}(i,j)$. There are four cases.

(1) Both $\underline{x}_{\mathrm{opp}(i)}$ and $\overline{x}_{\mathrm{opp}(i)}$ are updated to $\mathrm{length}(S_i) - j$.

(2) $\overline{x}_{\mathrm{opp}(i)}$ is updated to $\mathrm{length}(S_i) - j - 1$. $\underline{x}_l$ is updated to $\varphi_l(v) + 1$.

(3) $\overline{x}_{\mathrm{opp}(i)}$ is updated to $\mathrm{length}(S_i) - j - 1$. $\underline{x}_r$ is updated to $\varphi_r(v) + 1$.

(4) $\overline{x}_{\mathrm{opp}(i)}$ is updated to $\mathrm{length}(S_i) - j - 1$. Both $\underline{x}_l$ and $\overline{x}_l$ are updated to $\varphi_l(v)$, and both $\underline{x}_r$ and $\overline{x}_r$ are updated to $\varphi_r(v)$.

Note that any one of (B4-1), (B4-2), (B4-3) and (B4-4) guarantees the prevention of dangling tracks and L-junctions, and each of them generates a different set of tracks.

In Enumerate(i, n), the cases of $x_i = 0$ and $x_i = \mathrm{length}(S_i)$ are excluded from the for-loop iteration because the track T_i can neither be a dangling track nor have an L-junction. The bounds are updated according to (B3) for $x_i = 0$ and according to (B1) and (B2) for $x_i = \mathrm{length}(S_i)$. In the loop body, the update methods of (B1) and (B2) are first applied, and the bounds are then updated and a recursive call is made to Enumerate($i + 1$, n) using each of (B4-1), (B4-2-a), (B4-2-b) and (B4-2-c). The loop body has four enumeration branches.

Improved Algorithm: The previous algorithm succeeds in (duplication-free) enumeration of motorcycle tracks without invalid ones. However, the algorithm is still inefficient from the viewpoint of motorcycle graph enumeration: it generates duplicate motorcycle graphs, as different motorcycle tracks can form the same motorcycle graph (Fig. 4.20). To improve the efficiency, duplications are reduced by avoiding enumeration of motorcycle tracks with a certain property.

Note that, since our application does not require duplication-free enumeration, we do not aim to remove duplications completely.

Suppose a pair of opposite motorcycles m_i and $m_{\mathrm{opp}(i)}$ collide and stop at the same vertex v. If v is an internal vertex of separatrix S_i, we call it a *crack* between tracks T_i and $T_{\mathrm{opp}(i)}$. In Fig. 4.20 (a–c), there is a crack between the horizontal tracks. But, in Fig. 4.20 (d–e), since the common crashing point of the two motorcycles on the horizontal separatrices is not an internal vertex of the separatrix, there is no crack. Suppose there is a crack between T_i and $T_{\mathrm{opp}(i)}$, and let j be the local coordinate of the crack on S_i. If motorcycle $m_{\mathrm{left}(i,j)}$ (or $m_{\mathrm{right}(i,j)}$) goes beyond the crack, the crack is called *penetrated*; otherwise it is called *non-penetrated*. Fig. 4.21 shows penetrated and non-penetrated cracks.

Let $T_1, \ldots, T_n$ be valid tracks. Suppose there is a non-penetrated crack between tracks T_i and $T_{\mathrm{opp}(i)}$. By changing the position of the common crashing point of motorcycles m_i and $m_{\mathrm{opp}(i)}$ to an endpoint of S_i (i.e. setting $x_i = 0$ and $x_{\mathrm{opp}(i)} = \mathrm{length}(S_i)$, or $x_i = \mathrm{length}(S_i)$ and $x_{\mathrm{opp}(i)} = 0$), the crack between T_i and $T_{\mathrm{opp}(i)}$ can be removed. The validity of the tracks and the motorcycle graph formed by them are not changed by this operation. All non-penetrated cracks can be removed by repeated application of the operation, and the motorcycle graph does not change during this process. This indicates that, for any valid tracks that have non-penetrated cracks, there exist valid tracks that have the same motorcycle graph and that do not have non-penetrated cracks.

Thus, for motorcycle graph enumeration, it is not necessary to enumerate tracks that have non-penetrated cracks. To skip enumeration of these tracks, the update rule (B4) is modified as follows:

B4′. (Preventing dangling tracks and L-junctions). Suppose the coordinate x_i is set to j ($0 < j < \mathrm{length}(S_i)$). Let $v = \mathrm{vertex}(i, j)$, $l = \mathrm{left}(i, j)$ and $r = \mathrm{right}(i, j)$. There are two cases.

(1) $\underline{x}_l$ is updated to $\varphi_l(v) + 1$.

(2) $\underline{x}_r$ is updated to $\varphi_r(v) + 1$.

Our improved algorithm uses (B1), (B2), (B3) and (B4′).

4.3.2 Cut-and-Merge Based Enumeration

As separatrices in the base complex originate from extraordinary vertices, many partitions are produced when the number of extraordinary vertices is large. As this is undesirable, the number of extraordinary vertices can be used as a measure for the complexity of the base complex.

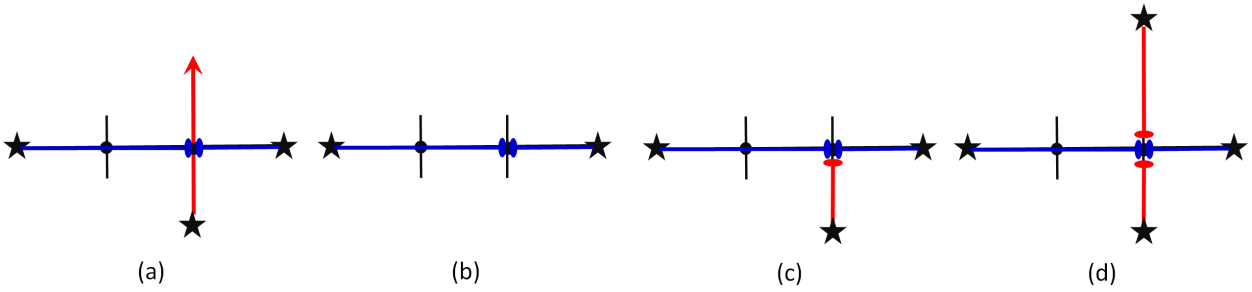


Figure 4.21: (a) Penetrated crack (b-d) Non-penetrated cracks.

A lower number of vertices in a base complex correspond to a simpler quadrilateral mesh base complex. Recent works [42, 49, 11, 14, 48, 39, 7] have proposed ways to simplify the base complex of a given quadrilateral mesh. The main approach to such simplification involves locating as few extraordinary vertices as possible and choosing appropriate positions for them, which in turn reduces number of vertices in the base complex. Locating extraordinary vertices in appropriate positions with individual and global consideration is also crucial in reducing the complexity of the base complex. As described in [39, 7], the misalignment of these vertices can cause helical structures (shown with dark blue circles in Fig. 4.28 (a)) that turn around the model several times, and thereby drastically increasing the complexity of the base complex. Although recent works [42, 49, 11] have outlined promising ways to eliminate these structures, complete elimination may not always be possible, particularly for complicated models, because requirements (such as geometric fidelity and base complex simplicity) to be satisfied during quadrangulation generally conflict with each other. Quadrilateral meshes generated using the methods proposed by these recent works also still have a high number of extraordinary vertices, meaning that the number of motorcycle graphs to be enumerated is very high.

Suppose there are n particles each with c candidate positions, and naive method described previously enumerates c^n number of particle positions. For the motorcycle graph enumeration, this number will be smaller because motorcycle graphs are restricted. However, as the number is still exponential, the enumeration has high computational cost, particularly with meshes for which the base complex is not simple. To simplify the motorcycle graph enumeration problem and speed up its resolution, our main approach involves dividing a given mesh into sub-meshes to allow separate enumeration of their motorcycle graphs, and then selecting optimum graphs for each sub-mesh. These motorcycle graphs are then merged to generate one for the whole mesh. Figure 4.28 shows base complexes (a, c) of a drill hole model. After the mesh is cut into sub-meshes, 8 sub-meshes are generated from the initial model. The total number of ordinary junctions is reduced to 45 from 1344.

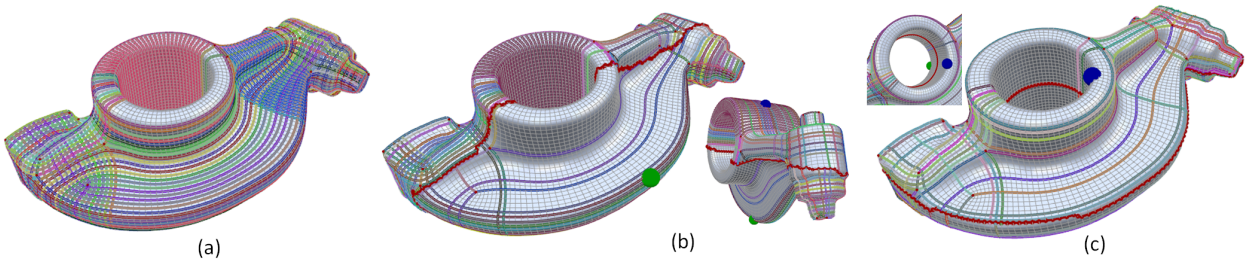


Figure 4.22: Total number of ordinary junctions in base complexes (separatrices in various colors): (a) without cut: 4574. (b) with cut (in red): 1304. (c) with cut: 426. The positions of source (green sphere) and sink (blue sphere) terminals therefore affect the base complex complexity of sub-meshes obtained using the graph-cut method.

With a mesh cut into sub-meshes, it is obvious that the optimum motorcycle graph of the mesh may not be found. If the absolute optimum solution is needed, the enumeration algorithm should be directly applied without applying Cut-and-Merge based enumeration, which has high computational cost. However, based on the outcomes of this study's experiments, the motorcycle graphs generated with the Cut-and-Merge based enumeration are suitable for reverse engineering and they are considered to approximate a global solution.

Mesh-Cut Algorithm with Graph-Cut: In this step, sub-meshes with simpler underlying base complexes are generated from a given quadrilateral mesh using the minimum cut algorithm of Boykov et al. [10], and any separatrices with helical structures are also eliminated. A weighted graph $G < V, E >$ is constructed using vertex and edge elements of the quadrilateral mesh $Q < V, E, F >$, where undirected edges in E connect the set of nodes in V. The terminals *source* and *sink*, which are two different vertices in V, are first found, and the *minimum cut* separating these terminals is then computed. The cut divides the mesh Q into two sub-meshes. In order to avoid generation of short cuts where one sub-mesh is large and the other is very small, the energy function is designed based on Golovinskiy et al.'s work [21], which favors the generation of sub-meshes with largely similar areas. This algorithm is applied repeatedly until sub-meshes with the desired level of base complex simplicity are obtained.

Selection of Terminals. To generate sub-meshes with well-simplified base complexes, *source* and *sink* vertices of a cut should be appropriately selected. The basic idea behind such selection is to enable the separation of extraordinary vertices in a given mesh (or sub-mesh) as equally as possible by finding appropriate positions for the source and sink because the complexity of a base complex is related to the locations and number of these extraordinary vertices, at which the separatrices of a base complex start and end. Accordingly, cutting is desirable if the resulting sub-

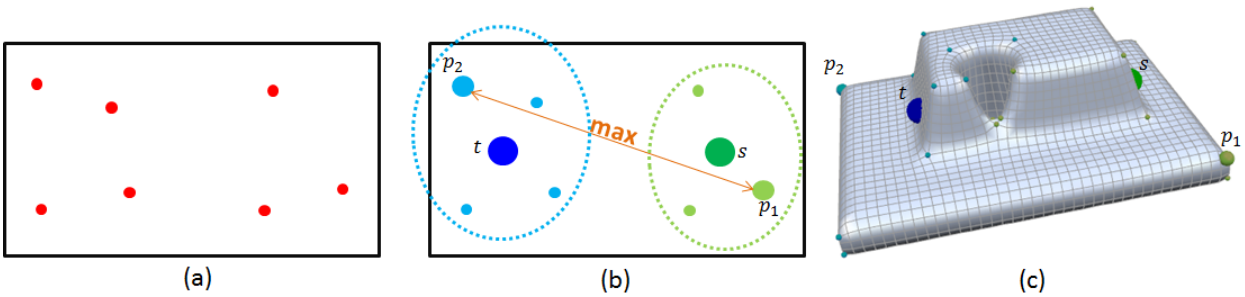


Figure 4.23: (a) Distribution of irregular vertices (in red) in the mesh boundaries in black (mesh edges, faces and ordinary vertices are not shown) is taken into account for the computation of source s and sink t terminals. (b) After the irregular vertex pair $p_1 - p_2$ with the maximum distance apart is found, all irregular vertices are grouped into two by checking proximity to p_1 and p_2. The vertices closest to the group centers are selected as source and sink. (c) Irregular vertex distribution and computed source-sink vertices (green-blue spheres) are shown for the drill hole model.

meshes contain almost equal numbers of extraordinary vertices. Figure 4.22 shows base complexes with separatrices depicted in various colors. Two different cuts (in red) are obtained for different settings of the source (green sphere) and sink (blue sphere) terminals. The base complexes of sub-meshes in the latter (c) are much simplifier than those in the former (b) such that the total number of ordinary junctions in the base complexes is further reduced.

Irregular vertices in the mesh are used to compute the source and sink terminals. The irregular vertex pair $p_1 - p_2$ (see Fig. 4.23) furthest apart is first found by checking distances between all irregular vertices (in red) in the mesh (boundaries in black). All irregular vertices are then grouped into two according to their distances to p_1 and p_2. Geodesic distance is used for these computations. Finally, the center of the two groups is calculated and the two mesh vertices closest to it are assigned as the source and sink terminals. Figure 4.23 shows source (green sphere) and sink (blue sphere) vertices computed using this approach, which enables almost-equal separation of irregular vertices and produces a simpler base complex after cutting.

Energy Function. The two-terminal graph constructed is partitioned into two sub-graphs by finding a cut that minimizes the following energy function:

$$E(L) = \sum_{p \in V} G_p(L_p) + \lambda \sum_{(p,q) \in N} f(L_p, L_q),$$

where λ is set to 1 and $L = \left\{ L_p | p \in V \right\}$ is a labeling of the mesh vertices V where $L_p \in \{S, T\}$ denotes a label given to a vertex. If the vertex p is assigned to the source terminal side, $L_p = S$.

Otherwise, $L_p = T$. $G_p(L_p)$ is a data penalty function and $f(L_p, L_q)$ reflects the interaction potential between all pairs of 1-ring neighboring vertices where N represents a set of all of these pairs. $f(L_p, L_q)$ is set to 1 if $L_p \neq L_q$; otherwise, it is 0.

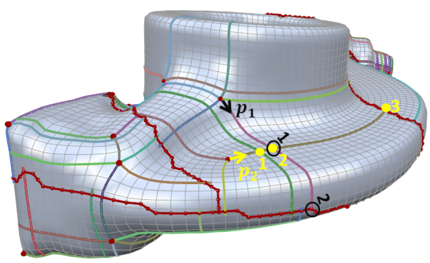
From each vertex in V to both the source and sink terminals, a penalty cost $G_p(L_p)$ is assigned as follows:

$$G_p(L_p) = \begin{cases} W_h & \text{if } D(L_p, p) \leq m_1; \\ W_l + v(m_2 - D(L_p, p)) & \text{if } m_1 < D(L_P, p) \leq m_2; \\ W_l & \text{if } D(L_p, p) > m_2. \end{cases}$$

where $D(L_P, p) = 1 - (dst(L_P, p)/(dst(S, p) + dst(T, p)))$ varying between 0 and 1, and $v = (W_h - W_l)/(m_2 - m_1)$. $dst(L_P, p)$ is the geodesic distance between vertex p and the source terminal when $L_P = S$ or the sink terminal when $L_P = T$. This formula favors cutting that is not very close to either of the source or sink terminals. In this way, short cuts are avoided and cuts that are equi-distant from terminals are promoted. In this study, the coefficients are set to $W_h = 80$, $W_l = 1$, $m_1 = 0.4$ and $m_2 = 0.5$. The setting $m_2 = 0.5$ favors cuts that are equi-distant from terminals, and if $D(L_p, p) \leq 0.4$, it is penalized with a high-cost W_h.

To favor equal separation of extraordinary vertices, penalties are imposed on such vertices when they are not assigned to their previously determined terminal group (see ”Selection of Terminals”). Recall that extraordinary vertices are classified into two groups shown with encircled ellipses in Fig. 4.23 (b) (the source group in light green and the sink group in light blue). For an extraordinary vertex previously determined to be the source (or sink) group, the value $D(v_{L_P}, p)$ is updated as 1 so that separation of this vertex from the source (or sink) group is penalized.

Number of Cuts. The previously explained graph cut algorithm partitions a given mesh into two sub-meshes. The base complexes of these sub-meshes are computed separately to be much simpler than that of the initial mesh, and the number junctions where separatrices meet is thus drastically reduced. However, cutting the mesh only once is not generally enough to obtain the desired level of base complex simplicity. In such cases, it is necessary to apply the graph cut algorithm several times until all sub-meshes obtained have the desired level of base complex simplicity.



As it is not easy to accurately find the number of motorcycle graphs that will be enumerated without enumeration, a naive approach to computation is chosen. In the technique reported here, particle candidate positions (junctions) where two separatrices meet are used to determine the number of cuts. Particles can reside only on these positions during the enumeration step. The image on the right shows candidate

positions for the particles p_1 and p_2 in a sub-mesh (boundaries are shown in red) such that p_1 has two candidate positions while p_2 has three. Let there be a sub-mesh S with m particles, and c_i is the number of candidate positions for the particle i. The quasi-complexity U of S is computed as $U = c_1 \cdot c_2 \cdot \ldots \cdot c_m$. As the value of U can become very large, it is simplified using a logarithm such that $U = \log_2 c_1 + \log_2 c_2 + \ldots + \log_2 c_m$. The U value of each sub-mesh should be smaller than the parameter χ, otherwise the sub-meshes will be partitioned into two using the graph-cut algorithm until sub-meshes with U values smaller than χ are obtained. Here, the parameter χ is set to 20 and 40.

Results. Figure 4.24 shows cuts generated (boundaries shown in various colors) for different test models with placed particles in red. The total number of ordinary junctions (n_j) in the base complexes and the number of sub-meshes (n_G) before (*Original*) and after ($\chi = 40, 20$) applying the Mesh Cut algorithm are detailed in the table on the right. The number of ordinary junctions drastically decreases after applying the proposed cut algorithm, and this number falls even further for smaller χ values so that base complexes become less complex. It should be noted that the drill hole and rockerarm models have a high number of ordinary junctions in their initial base complexes due to the presence of helical structures as shown in Fig. 4.28. Cutting the mesh into two sub-meshes eliminates the complexity resulting from these structures.

	Original	$\chi = 40$		$\chi = 20$	
Models	n_j	n_G	n_j	n_G	n_j
Drillhole	1344	5	73	8	45
Beetle	199	5	94	8	52
Bottle	623	6	71	8	52
Rockerarm	4574	8	109	12	66

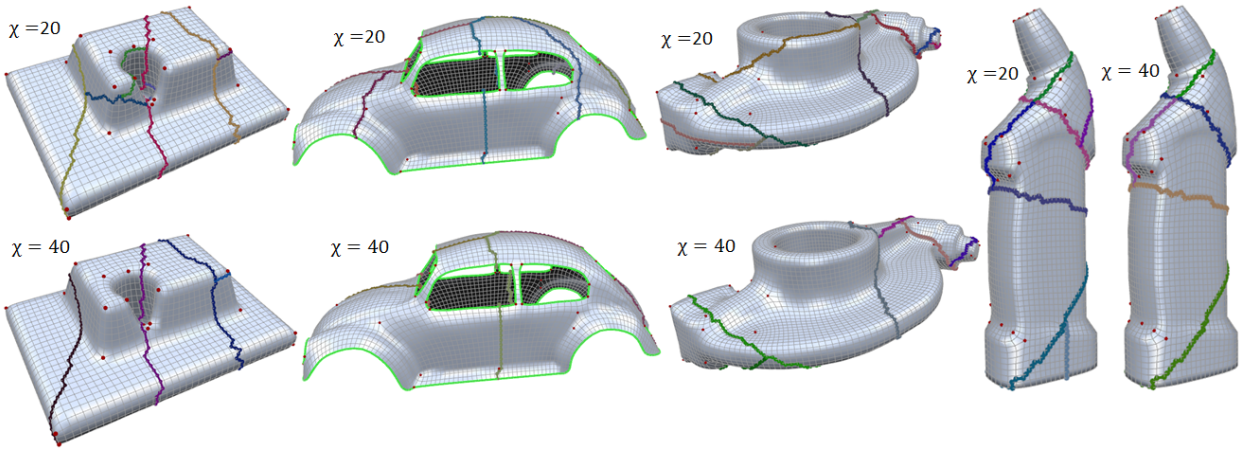


Figure 4.24: Cuts generated (boundaries in various colors) with different test models for $\chi = 20$ and $\chi = 40$. Particles placed are shown in red.

Extended Enumeration Algorithm: The enumeration algorithm detailed previously works for a quadrilateral mesh homeomorphic to a two-manifold without boundary. A sub-mesh may

have two types of boundary after applying the Mesh-Cut algorithm: Mesh boundary (like in the Beetle model) or *Cut boundary* (see cuts in Figure 4.24). The Mesh-Cut algorithm separates the vertices in the mesh into two groups and the generated cuts may pass through some faces which will not be assigned as the elements of two generated sub-meshes. Let o sub-meshes $B_1, B_2, ..., B_o$ are generated after cutting. A sub-mesh B_i has the vertices $V_i \subset V$, the edges $E_i \subset E$ and the faces $F_i \subset F$ where $V_1 \cup V_2 \cup \ldots \cup V_o = V$, $E_1 \cup E_2 \cup \ldots \cup E_o \neq E$ and $F_1 \cup F_2 \cup \ldots \cup F_o \neq F$. *Neighboring vertices* of B_i are the vertices in 1-ring neighborhood that are elements of another sub-mesh B_j ($j \neq i$). Note that a vertex on the boundary is an extraordinary vertex whose valence is at least four.

While computing base complex of a sub-mesh, separatrices are formed that start from extraordinary vertices and can end at extraordinary vertices, Mesh boundary vertices or neighboring vertices. Motorcycle graph enumeration for a sub-mesh is performed to obtain particle tracks using its base complex in a similar way explained previously and the particle candidate positions can be intersection vertices of separatrices, extraordinary vertices, Mesh boundary vertices or neighboring vertices. Motorcycle graph of a sub-mesh is the union of the obtained particle tracks after the enumeration. However, some separatrices in a sub-mesh may not have their opposite, left or right separatrices. Accordingly, the update rules (B1), (B2), (B3) and (B4′) should be customized for these separatrices (not for others).

B1. (Preventing overlapping tracks.) Suppose the coordinate x_i is setting to j. No need to update $\underline{x}_{\text{opp}(i)}$ if the separatrix S_i does not have opposite separatrix (opp(i)).

B2. (Preventing crossing tracks). Suppose the coordinate x_i is setting to j. For $k = 1, \ldots, j-1$, no need to update $\overline{x}_l$ if the separatrix S_i does not have left(i,k), where $v = \text{vertex}(i,k)$, $l = \text{left}(i,k)$ and $r = \text{right}(i,k)$. Similarly, no need to update $\overline{x}_r$ if the separatrix S_i does not have right(i,k).

B3. (Preventing inactive particles.) No recursive call needed if the separatrix S_i does not have opposite separatrix.

B4′. (Preventing dangling tracks and L-junctions). Suppose the coordinate x_i is set to j ($0 < j < \text{length}(S_i)$). Let $v = \text{vertex}(i,j)$, $l = \text{left}(i,j)$ and $r = \text{right}(i,j)$.

(1) No recursive call needed for the separatrix S_i with no left(i,k).

(2) No recursive call needed for the separatrix S_i with no right(i,k).

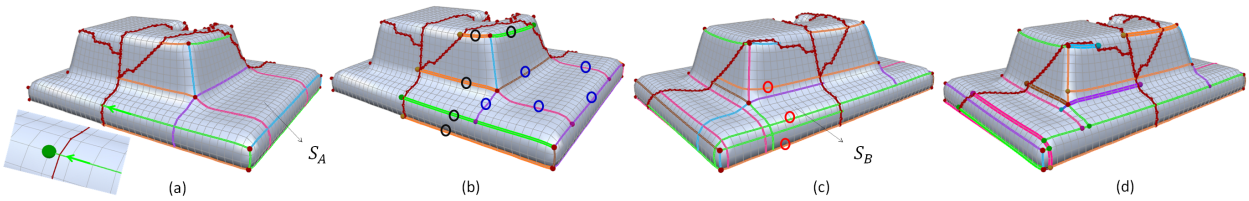


Figure 4.25: (a) The base complex of the sub-mesh S_A is computed. Particles are stopped after passing cuts (in red). (b) Motorcycle graph enumeration is performed. Some particles (called as *free particles*) go beyond the cut and some stop (tracks marked with black and blue circles, respectively) (c) Free particles (tracks marked with red circles) in S_A are taken into account during base complex computation of the sub-mesh S_B. (d) Particle positions after the graph enumeration in S_B (tracks of free particles thickened

Merging Motorcycle Graphs: Motorcycle graphs of the obtained sub-meshes are enumerated using the Partial MC Generator and the Complete MC Generator and finally a graph representing a motorcycle graph for the whole mesh is obtained. Selection of the optimum graph for each sub-mesh is based on a cost function which will be also described here.

Partial MC Generator (PMG). The main aim here is to find the optimum motorcycle graph for each sub-mesh by applying the enumeration algorithm in Section 3. After the enumeration, the optimum graph is selected, and some particles in this graph go beyond the cut which are called as *free particles* while others stop inside the sub-mesh. Figure 4.25 shows the cuts (in red) and enumeration for the sub-meshes S_A and S_B. The tracks of free particles are marked with black circles (b) and are thickened, and the tracks of others are marked with blue circles (b). For the next enumeration performed in another sub-mesh, free particles are also used in base complex computation. Namely, free particles of S_A (b) are also taken into account when the base complex of S_B is computed (marked with red circles (c)). In this way, particles with dangling tracks go further and stop (d).

Complete MC Generator (CMG). The union of optimum graphs after applying PMG may not represent a motorcycle graph when the whole mesh is considered. Therefore, CMG is implemented to perform successive enumeration steps in sub-meshes separately until a motorcycle graph for the whole mesh is obtained. The same methodology as that for the PMG is used for the CMG, but only certain particles are considered during the enumeration step. These are free particles and *linked particles*, which are those to be included in the enumeration with free particles to avoid the generation of invalid motorcycle tracks. In Figure 4.26 (b), particles with tracks shown in orange, pink, purple, gray and blue colors are linked particles.

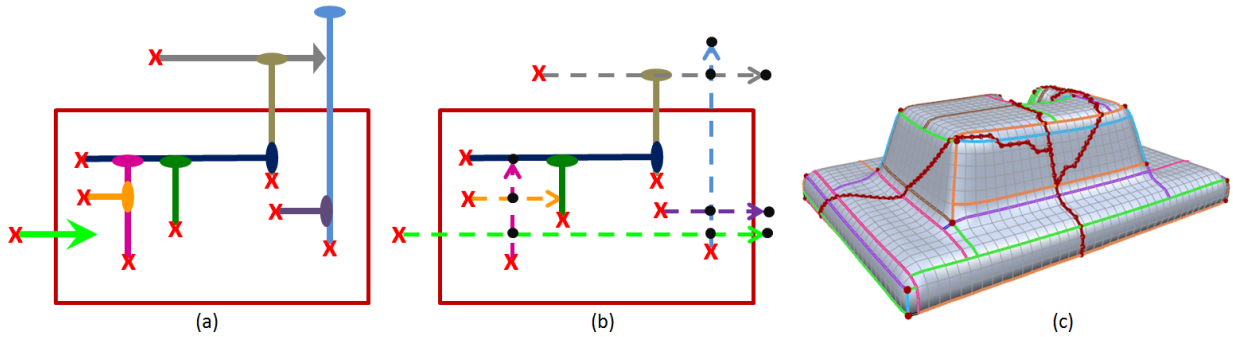


Figure 4.26: A motorcycle graph in a sub-mesh (boundaries shown in red) is shown (a) with a free particle (track shown in light green). To assign a position for this particle, enumeration in the sub-mesh is required. The base complex is computed (b) using only certain particles (tracks shown with dashed lines), and other particles (tracks shown with solid lines) do not need to be considered during graph enumeration. Candidate positions are depicted with black dots. (c) After applying the PMG, a motorcycle graph for the whole mesh is generated where no free particles are left.

Figure 4.26 (a) shows an invalid motorcycle graph consisting of particle tracks. The particle track shown in light green is that of a free particle which should be positioned after performing enumeration in the sub-mesh (boundaries shown in red). The base complex is computed using this free particle and related particles, whose separatrices traced are shown with dashed lines (b). The particle candidate positions that are used for the enumeration are marked with black dots. The tracks of other particles (tracks marked with solid lines) do not need to be considered during the enumeration process. Starting with such a base complex, enumeration is performed and the optimum motorcycle graph is selected from among those enumerated. Figure 4.26 (c) shows the motorcycle graph generated after several successive enumeration steps. As mentioned in the previous paragraph, such successive steps are performed until a motorcycle graph for the whole mesh is obtained. It should be noted that the CMG may not always converge to a valid motorcycle graph for the whole mesh; that is free particles may always be present after enumerations. However, no such cases were observed in the experiments of this study. To overcome this convergence problem, various naive approaches can be proposed.

Selection of Optimal partitioning with Cost Function. A structured partition can be obtained from a set of particle tracks ($\{T_i\}$) which can also be denoted as a set of paths $\{W_i\}$. Same cost function defined in Section 4.1 is utilized.

4.3.3 Integration of Feature Curves

Extraordinary vertices are not always connected to or near the highly curved regions. As a result, these regions inevitably reside inside the generated quadrilateral partitions of motorcycle graph. For reverse engineering applications such as surface modeling, it is crucial to locate as many such regions as possible on partition boundaries. After the highly-curved regions that are very difficult to capture with the particles placed on extraordinary vertices have been detected and extracted, and additional particles are placed on these curves to enable them to be traced. These particles are then integrated to the motorcycle graph enumeration algorithm in the same way as particles located on extraordinary vertices.

The highly-curved regions mentioned above are called as *feature curves*, and have three main properties:

- They are sets of highly curved connected edges in the same parametric direction. In other words, all these connected edges have dihedral angles greater than the flatness threshold ρ.
- They have an appropriate length (number of quadrilateral edges) to avoid extraction of short curves. Feature curves with length less than the threshold τ is discarded.
- They are not close to extraordinary vertices. To control proximity, the parameter ζ is introduced so that the number of edges in the same parametric direction should exceed ζ from end points of an extracted feature curve to any extraordinary vertex.

Figure 4.27 shows extracted feature curves. The drill hole model has only one feature curve, the Beetle model has five, the bottle model has twelve and the rockerarm model has three. After this extraction, two particles moving in opposite directions (b) are inserted next to the middle of the feature curve. The parameter values used here are shown in the table on the right. Based on our experience, the parameters are easy to tune and fine-tuning does not generate very different results. In this step, it is also possible to utilize sophisticated algorithms for feature curve extraction and the feature curves extracted using other methods can easily be integrated into the proposed framework once their edges are aligned in the same parametric direction. Note that the mixed-integer quadrangulation technique [9] of Bommes et al. smoothens the given mesh during quadrangulation so that mesh noise is reduced. It can therefore be assumed that there is no significant noise in the quadrilateral mesh taken as input in the proposed method.

Parameters / **Models**	ρ	τ	ζ
Drill hole	45	15	30
Rockerarm	45	30	30
Bottle	50	5	30
Beetle	40	8	60

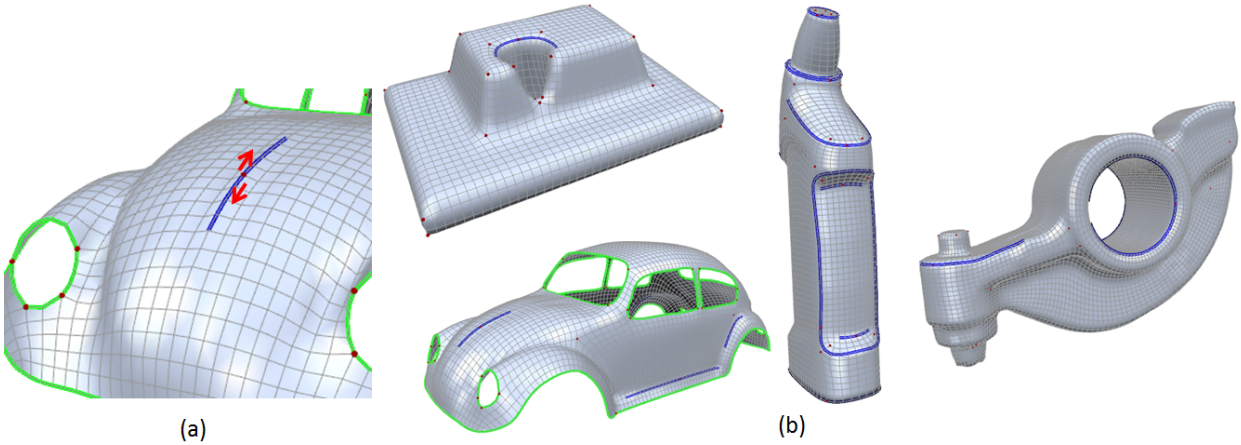


Figure 4.27: Feature curves (in blue) in the mesh that are not close to extraordinary vertices are extracted. Two particles moving in opposite directions (a) are placed next to the middle of the curve.

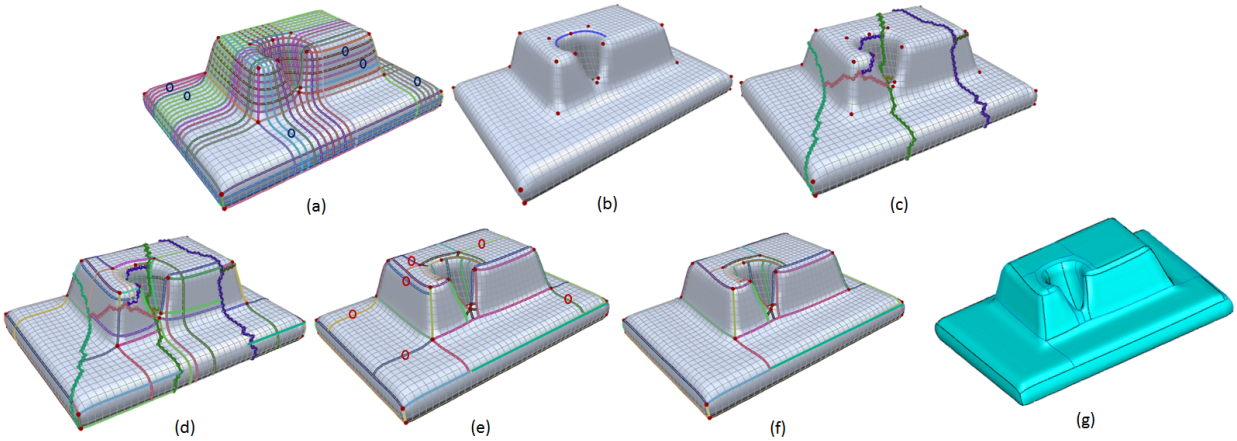


Figure 4.28: (a) The initial mesh with its base complex is given. The total number of ordinary junctions is 1344. (b) Feature curves far away from extraordinary vertices are detected. Particles moving in opposite directions are placed on the middle of these feature curves in opposite directions and on extraordinary vertices. (c) To obtain sub-meshes with simpler base complexes, the mesh is cut into several sub-meshes each with the desired base complex complexity. A total of 8 sub-meshes are obtained with 45 ordinary junctions. (d) The base complex for each sub-mesh is found. (e) Optimum motorcycle graphs are enumerated for each sub-mesh. After several enumerations, a motorcycle graph of the initial mesh is obtained. (f) Flat particle tracks are removed ($\eta = 1$) and much more compact partitioning is achieved. (g) Generated partitions are validated with B-spline surfaces.

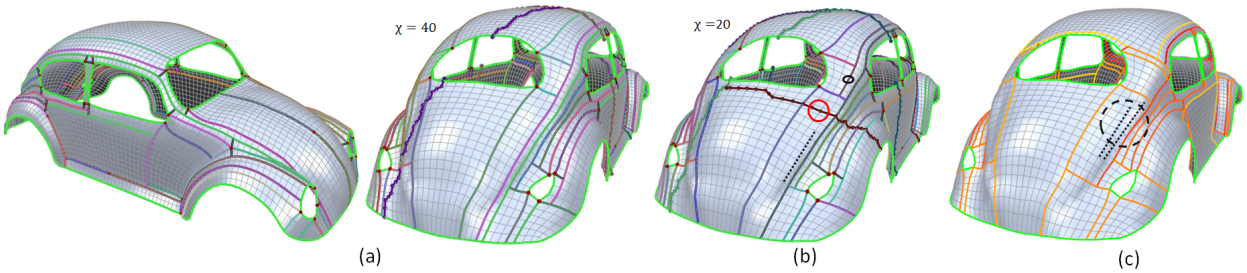


Figure 4.29: Setting small values for χ creates a smaller solution. For $\chi = 20$ (b), the cut marked with the red circle does not allow the particle (track marked with a black circle) to go further and trace the highly curved edges (marked with black round dots) traced when $\chi = 20$ (a). The proposed algorithm (a) generates better partitioning than the motorcycle graph algorithm [19]. Highly curved edges are present inside the partitions marked with a black circle.

4.3.4 Results and Discussion

Figure 4.28 illustrates the flow of the proposed algorithm. We apply this algorithm to the following four quadrilateral mesh models in the experiments reported here: a drill hole, a rockerarm, a Beetle and a bottle. This section covers the results for different cut parameter (χ) values, comparison with existing quadrilateral partitioning techniques, the number of motorcycle graphs generated during the enumeration step, the computational time taken for each model, a flat boundary removal technique, generated B-spline surface models and deviations of generated surfaces from the original quadrilateral meshes.

Result evaluation: As seen from Figures 4.28, 4.29, 4.30 and 4.31, a large number of highly curved edges in the models are placed on partition boundaries (shown in various colors). It can be seen that highly curved edges in Fig.4.29 (b) marked with black round dots cannot be traced by any particle when χ is set to 20, but can be traced when $\chi = 40$. Assigning smaller values for χ cuts the mesh into more sub-meshes, thereby reducing the solution space and making solutions farther from the global optimum solution. The cut (in brown) marked with red circle does not allow the particle (track marked with a black circle) to go further and trace highly curved edges (marked with black round dots). Another comparison is made for the proposed algorithm (b) and the motorcycle graph algorithm [19] (c). Highly curved edges (marked with a black circle in (c)) are not placed on partition boundaries.

As detailed in the feature curve integration algorithm, it is crucial to place particles on highly curved regions that cannot be traced by particles placed on extraordinary vertices. Figure 4.30 (a) shows such a case for the bottle model. A large number of highly curved edges (marked with black circles) can be seen inside the generated partitions, which is problematic for B-spline

surface fitting. The feature curve integration step improves the quality of partitioning (b) for reverse engineering applications.

Different quadrilateral partitioning is obtained when χ is set to 20, 40 and 60 as shown in Figure 4.30 (a), (b) and (c). However, highly curved edges in the model are captured for all these χ settings. The quadrilateral partitions generated using the motorcycle graph algorithm [19] (d) and the method of Campen et al. [11] ((e), for their particular choice of thresholds) have highly curved edges seen in the inner regions marked with black circles. Quadrilateral partitions generated using other methods ([39, 7, 49, 7]) also contain these highly curved edges as far as their images shown in the papers are concerned. Recall that T-junctions do not exist in the partitions of [49, 7, 11], whereas they exist in that of [39, 19] and the proposed method. The number of T-junctions and quadrilateral partitions generated are as follows: 54 and 51 for the drill hole (with $\chi = 20$), 116 and 95 for the rockerarm (with $\chi = 20$), 70 and 60 for the bottle (with $\chi = 40$), 100 and 90 for the Beetle (with $\chi = 20$) respectively.

Computational time: A 3.4 GHz PC was used for the experiments in this study. Table 4.1 shows the computational time taken for each model. The processing time for the Mesh Cut algorithm depends on the χ setting, and is much longer for smaller values because the mesh is cut into a large number of sub-meshes. Computation of the geodesic distance from each vertex in the mesh to the source and sink terminals dominates the processing time in this step. The motorcycle graph enumeration step also depends on the χ settings, and the computational cost is higher for greater values because a large number of graphs are enumerated. As an example, for rockerarm model with $\chi = 60, 340, 396, 807$ motorcycle graphs were enumerated at a time cost of more than eight hours. However, the test results indicate that, better solutions (with minimum cost) were obtained with large χ values (compare the costs of rockerarm with $\chi = 20, 40, 60$ and the Beetle with $\chi = 20, 40$). For the drill hole and bottle models, the results are very similar when χ is set to 20, 40 or 60. Finally, assigning large values such as 80 for χ results in such high computational cost that the graph enumeration step takes hours.

Flat boundary removal: To achieve partitioning with much greater compactness (i.e., a fewer partitions), flat (less-curved) boundaries are removed using the the explained method in Section 4.2.3. The images on the top of Fig. 4.32 show the models after this process.

B-spline surfaces: The generated partitions are fitted with B-spline surfaces using the the explained method in Section 4.2.3. The images on the bottom of Figure 4.32 and the right bottom of Figure 4.28 (g) show B-spline surfaces for partitions generated using the proposed algorithm. Deviation of the generated surfaces from the original quadrilateral mesh is shown in Fig. 4.33. Less deviation occurs with partitions generated using the proposed algorithm (PA) than with those

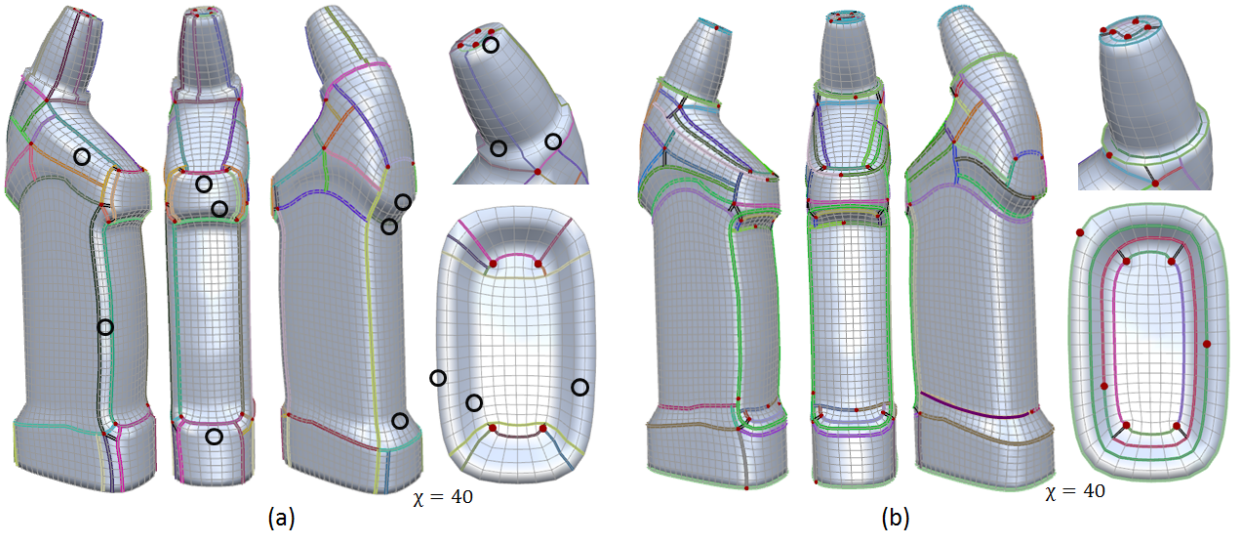


Figure 4.30: (a) Without the feature curve integration step, large numbers of highly curved edges cannot be placed on partition boundaries. (b) partitioning with feature curve integration.

generated using the motorcycle graph (MC) algorithm and the path flipping (PF) algorithm.

4.4 Conclusion and Future Works

The proposed algorithms in this section generates quadrilateral partitioning from given semi-regular quadrilateral meshes. The PF approach generate feature-aware partitions from motorcycle graph. As an extension of the MCG algorithm, the three approaches, speed control (SC algorithm), path flipping (PF algorithm) and feature curve integration (FC algorithm), are proposed to take surface geometry into account. In the SC algorithm, particle speeds are controlled to run faster on feature edges in order to move forward paths in feature regions. In the PF algorithm, paths are flipped to change blocking relationships and to enable further progress of blocked paths. To guarantee the capture of all feature curves in a model, the FC algorithm extracts feature curves and integrates them into the SC algorithm. Note that the insertion of the feature curves can be reduced by using more sophisticated surface fitting algorithm with non-uniform knots to capture highly curved regions. The proposed algorithm is not canonical like the original MCG algorithm, but is guaranteed to generate structured partitioning. The experiments show that the proposed approach can be used to generate partitioning (by capturing feature curves) that are useful in reverse engineering.

The MCG enumeration approach involves enumerating all the motorcycle graphs of a given mesh so that the optimum one (based on a cost function) can be selected. As the computational

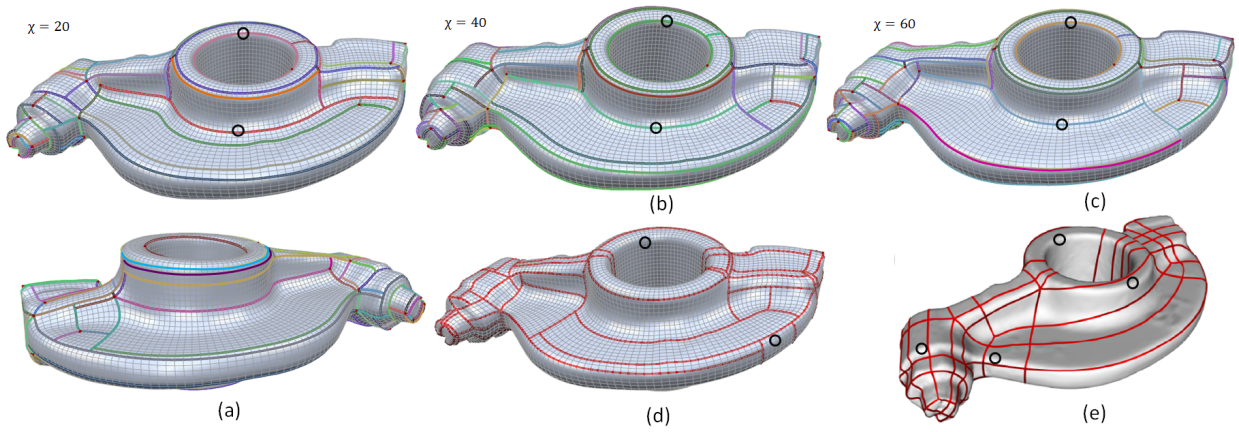


Figure 4.31: Different partitioning (a, b, c) is obtained with different χ settings $(20, 40, 60)$. partitions generated using [19] (d) and [11] (e) may contain highly curved edges (marked with black circles), which is problematic for B-spline surface fitting.

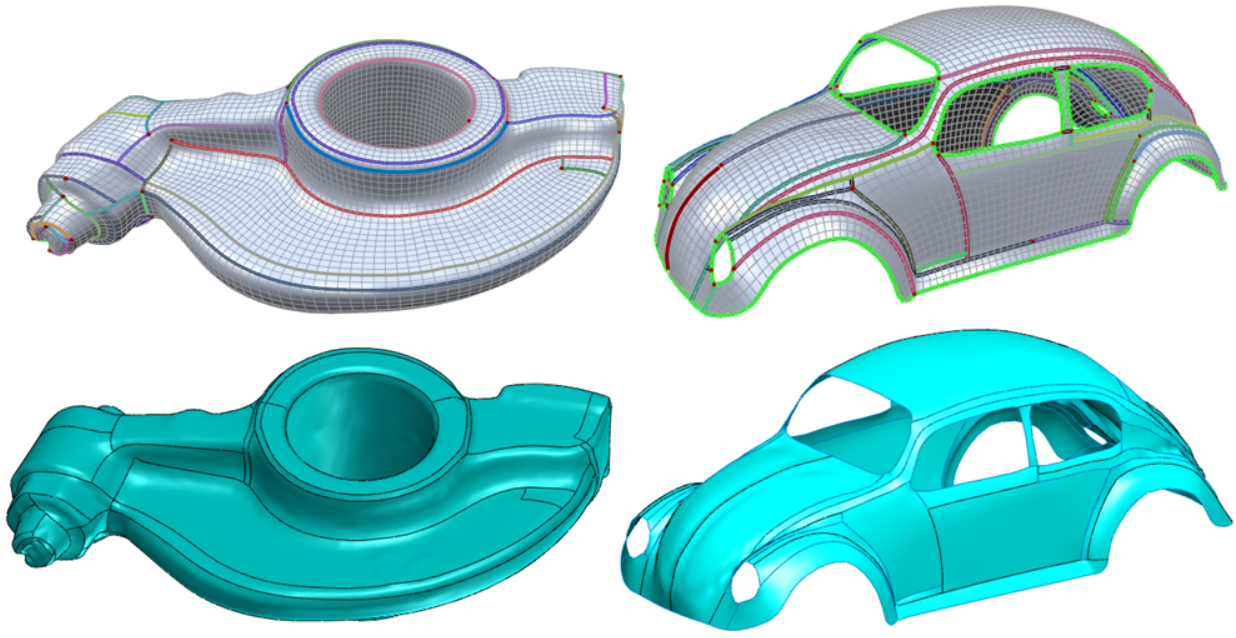


Figure 4.32: Top: Partitions after applying the flat boundary removal algorithm ($\eta = -3$). Bottom: B-spline surfaces (in blue) fitted to these partitions.

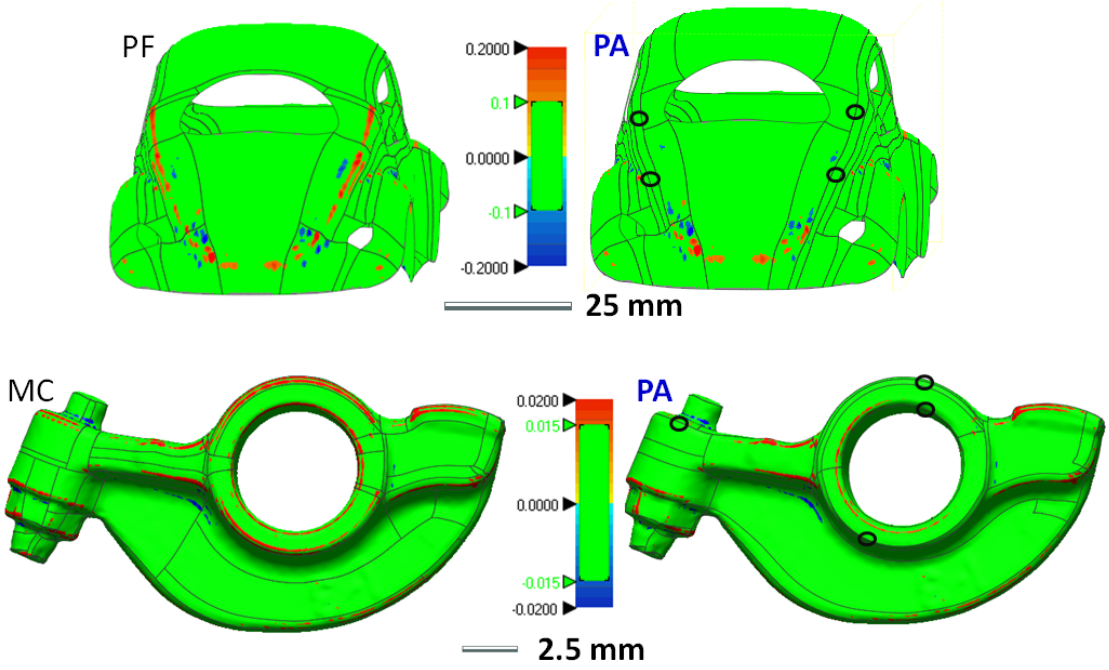


Figure 4.33: Less deviation is observed for partitions generated using the proposed algorithm (PA) than for those generated using the motorcycle graph (MC) algorithm (see locations marked with black circles).

cost of enumerating all graphs is high, the proposal includes the use of the Mesh-Cut algorithm to cut the mesh into several sub-meshes, and graphs are then enumerated separately for each sub-mesh. Using a feature curve integration algorithm, highly curved edges that are very difficult to trace with the particles placed on the extraordinary vertices can be traced. The experiments showed that the proposed approach generates partitions with highly curved edges on their boundaries.

The proposed algorithms can be enhanced in several ways. To achieve optimum cuts for a given mesh, curvature information and other different criteria will be considered in the Cut-and-Merge algorithm in addition to that of equal separation of extraordinary vertices. The use of mobile cuts whose positions change after each enumeration step may also be worthwhile which may generate better partitioning results. Other cost function(s) (that can be used in the selection of enumerated graphs) directly related to B-spline surface quality and better flat boundary removal algorithms will also be investigated. Finally, partitions generated using the proposed algorithm will be validated using T-spline surfaces [45].

Table 4.1: Process data are shown for different χ settings (e.g., drill20 represents the drill hole model with $\chi = 20$). G_T = global cost for the motorcycle graph produced, *t_Cut* = processing time for the Mesh Cut algorithm, *t_Enum* = processing time for the motorcycle graph enumeration step, *it#* = iteration number (*it#* = 1 representing applying PMG, and *it#* = 2, *it#* = 3, and others representing applying CMG). *mc#_i* represents the number of motorcycle graphs generated in the i^{th} sub-mesh where $i = 1..12$.

Model	G_T	t_Cut	t_Enum	it#	mc#_1	mc#_2	mc#_3	mc#_4	mc#_5	mc#_6	mc#_7	mc#_8	mc#_9	mc#_10	mc#_11	mc#_12
drill20	-185.1	25.5	1.1	1	10	3	788	33	1692	137	9	13	-			
				2	156	30	7035	31	866	8025	22	2				
				3	-	12	3	-	-	14	3	2				
rock20	-501.9	185.2	6.7	1	10	216	40	140	2	8	570	28	627	201	96	96
				2	1083	3700	954	114373	78	182	2604	79	10	12	1	48
				3	-	1079	24	966	4	3	-	-	4	-	2	24
				4	-	273	-	10	4	3	-	2	-	-	-	24
rock40	-531.0	181.6	70.9	1	422	216	40	393272	628	22926	2095	198	-			
				2	997965	908	1530	4600	7741	69	270	18				
				3	-	1188	-	10	6	-	270					
rock60	-540.7	163.9	29302.8	1	652124	216	93413596	4581692	434028	-						
				2	340396807	12384	252	-	6810							
				3	-	1362	4	580	2							
beat20	-130.6	110.2	0.45	1	63	15	327	426	46	74	19	75	-			
				2	860	1380	4	30	3	-	17	2				
				3	10	378	-	-	-	6	-	3				
beat40	-133.3	99.3	1.6	1	63	4956	9366	22605	46	-						
				2	120	63	125	9	3							
bott40	-164.0	38.9	2.8	1	117	80	1894	14076	183	1304	-					
				2	1047	1906	2728	20038	4	3897						
				3	7727	11	-	17884	-	734						

Chapter 5

Conclusion and Future Works

5.1 Conclusion

3D models are essential in many computer-based applications in order to analyze and improve these models. Reverse engineering aims to generate 3D models from a scanned data obtained by measurement systems. The research in this dissertation strives for the generation of B-spline surfaces from a given quadrilateral mesh since they are good fit for each other. Recent quad-meshing techniques can generate quadrilateral meshes where the vertex elements are placed regularly and therefore uniform B-spline surfaces can easily fit to them without parameterization step.

Partitioning of a given mesh can be done using various criteria for the generation of high-quality B-spline surfaces. The most direct criteria to check the B-spline surface quality is to fit the data to B-spline surfaces every time during partitioning which has computationally high cost. Our approach uses a criteria having lower computational cost such that highly-curved elements in the mesh are located on the surface boundaries because they cannot be represented by smooth B-spline surfaces. The bi-monotone approach in this dissertation proposes bi-monotone patch schema and its generation which has ability to capture feature curves in the model easily. The other two proposed approaches formulate the same problem using optimization and positions highly-curved regions on the partition boundaries as many as possible.

The bi-monotone approach demonstrates the advantages of bi-monotone patches when utilized on quadrilateral meshes. The surface features of an object can be captured more easily with bi-monotone patches than with quad patches. After partitioning of a quadrilateral mesh into four-sided patches, feature curves residing inside them are detected and they are constrained to be the patch boundaries. Finally, the configuration is improved to generate large patches. A user-adjustable *shape control* parameter is also provided to support the production of patches with good bi-monotone shapes.

The Path Flipping approach in this dissertation generates feature-aware partitions from motorcycle graph. As an extension of the original motorcycle graph algorithm, the three approaches, speed control (SC algorithm), path flipping (PF algorithm) and feature curve integration (FC algorithm), are proposed to take surface geometry into account. In the SC algorithm, particle speeds are controlled to run faster on feature edges in order to move forward paths in feature regions. In the PF algorithm, paths are flipped to change blocking relationships and to enable further progress of blocked paths. To guarantee the capture of all feature curves in a model, the FC algorithm extracts feature curves and integrates them into the SC algorithm. The proposed algorithm is not canonical like the original algorithm, but is guaranteed to generate structured partitioning.

The MCG enumeration approach aims to generate a motorcycle graph of a given quadrilateral

mesh which is nearer to the global optimum for B-spline surface fitting. As listing all motorcycle graphs have computationally high cost, the mesh is cut into several sub-meshes in each of which the optimum motorcycle graph is found by checking all producible motorcycle graphs. When the motorcycle graphs in each sub-meshes are combined, a valid quasi-optimum motorcycle graph may not be obtained. Therefore the solutions in each sub-meshes are combined using a methodology and a valid motorcycle graph is obtained.

Bi-monotone partitions can capture feature curves that are not aligned with $u-v$ directions with an appropriate number of partitions whereas the Path Flipping and MCG enumeration approaches cannot capture them. Based on a defined cost function, MCG enumeration approach generates better partitioning than the bi-monotone and Path Flipping approaches since many motorcycle graphs are enumerated and one of them is selected as the best one. The Path Flipping approach can generate better results (more highly-curved regions can be placed on partition boundaries) than the bi-monotone approach and it can generate similar results in a shorter time compared to the MCG enumeration approach for the models having smooth features. According to the test results of the surfaces generated, the MCG enumeration approach produces partitions having less deviation from the original data. The experiments show that the proposed approaches can be used to generate partitioning that are useful in reverse engineering.

5.2 Future Works

The research in this dissertation can be extended in many ways:

- **Surface-related criteria.** While generating partitions for the use of B-spline surface fitting, a number of surface-related criteria can be taken into account. Our current approach indirectly aims to have surfaces having less fitting error by locating highly-curved regions on the surface boundaries. For a better surface quality, $G0$ and $G1$ continuities are also crucial such that two neighboring surfaces are good to have smooth transitions. These criteria can be integrated into our cost formula that is used in the proposed approaches.

- **T-splines.** As an application of partitioning, the generated partitions of the path flipping and MCG enumeration approaches are expected to fit well with T-spline surface modeling approach. It must be interesting to feed our resulting partitions to the T-spline framework [45]. T-splines are much more advantageous than B-splines since T-joints are permissible such that $G0$ and $G1$ continuities are guaranteed.

- **Isogeometric analysis.** Possible use of the generated partitions (by the proposed approaches) can be isogeometric analysis which is a recently developed computational approach and employs complex NURBS geometry in the FEA application, and thereby allowing simultaneously designing and testing models.

- **Feature curves with arbitrary shapes.** The path flipping and MCG enumeration approaches produces quad partitioning by only considering the straight feature curves in the model. In order to improve the efficiency of capturing feature curves that are not aligned with parameter $u - v$ directions, we would like to extract and integrate the feature curves with arbitrary shapes into the proposed algorithms.

- **Optimum mesh cutting.** The MCG enumeration approach cuts the mesh into sub-meshes in order to reduce the computational time of the problem. However, this cut may affect the final solution such that different cuts may generate different solutions. Therefore it is required to do research on the generation of *good cuts*. To do this, "What is a good cut?" should first be defined and then algorithm should find good cuts.

- **Motorcycle graph-based partitioning.** Compared to using an triangular mesh as initial input, utilizing quadrilateral mesh enables better partitioning in which the partitions have good shapes such as quad or bi-monotone partitions. We expect that motorcycle graph-based algorithms can be used effectively in many areas like ship building. In ship hull manufacturing, the generated partitions of this algorithm can easily be manufactured since this algorithm only produce quad partitions.

- **Parameter setting.** Our algorithms are automatic once all required parameters are adjusted by the user. It can be good contribution if there is a way found to set this parameters automatically by an algorithm.

- **Dihedral angle.** To detect feature curve or to assign cost for a motorcycle path, we have utilized dihedral angle of an edge since it has computationally low cost. It is also possible to utilize different criteria such as max/min curvatures, developability conditions, etc. Analyzing the changes in the results after using different geometrical criteria can be interesting.

Acknowledgements

This dissertation hold far more than the culmination of years of study which reflects the relationships with many generous and inspiring people I have met since beginning my graduate work. Without the support, patience and guidance of the following people, this study would not have been possible to write. It is to them that I owe my deepest gratitude:

- I would like to gratefully and sincerely thank my supervisor, Professor Hiromasa Suzuki for his guidance, understanding, patience, and his friendship during my graduate studies at The University of Tokyo. His mentorship will definitely help my future career. I would like also to express my gratitude to Professor Yutaka Ohtake and Dr. Masaki Moriguchi for their collaboration and valuable technical advices.

- I would like to acknowledge to MEXT (Ministry of Education, Culture, Sports, Science and Technology) for my scholarship support.

- I would also like to thank to my committee members Professor Jun Ota, Professor Kiyoshi Takamasu and Professor Tomonori Yamada for their encouraging words and thoughtful criticism.

- Deep gratitude to my MSc supervisor from KAIST, Professor Soonhung Han for his encouragement and support all the time.

- Special thank to all members of Fine Digital Engineering Lab, particularly to Sagar, Vaibhav, Liu, Yamanaka-san, Haitham, Michikawa-san, Kamal, Nagai-san, Iwata-san, Nishihata-san, Jinnouchi-san, Hishida-san, Tsujiguchi-san, Yamauchi-san, Yuna Kang, Hyungki Kim, Xue and Yang for their generous and loving friendship.

- I would also like to thank to the Precision Engineering Department staff, Okino-san, Saitou-san and Arai-san for assisting me with the administrative tasks necessary for completing my doctoral program.

- Thank to my special friends Basmil and Ismail abi who were always with me for everything during my life in Japan. Deep gratitude to Keiko for enjoyable coffee breaks, to Yuri for introducing me Japanese culture, food, places in Japan, and to my Turkish students Mami and Kazuki who have excellent performance to speak Turkish language. Special thank to my Japanese Sensei (Nao-Sensei) for her encouragements to learn Japanese language and to Takahiko-Sensei for his warm support for my adaptation to Japanese life and culture. I am

also grateful to all other my beloved friends particularly; Hasan, Omer, Erman, Hyonkyong, Eri and Serhan abi.

- Finally, I would like to express my deep gratitude to my parents, my brother Serkan and all my relatives for their support all throughout my life.

Bibliography

[1] A. Agathos, I. Pratikakis, S. Perantonis, N. Sapidis, P. Azariadis. 3d mesh segmentation methodologies for CAD applications. *Computer Aided Design And Applications*, 4(6):827–841, 2007.

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

[3] P. Alliez, G. Ucelli, C. Gotsman, and M. Attene. Recent advances in remeshing of surfaces. *Research report, AIM@SHAPE Network of Excellence (2005)*, 2005.

[4] N. Amenta, S. Choi, and R. K. Kolluri. The power crust. pages 249–266, 2001.

[5] P. Benko and T. Várady. Direct segmentation of smooth, multiple point regions. *Geometric Modeling and Processing*, pages 169–178, November 2002.

[6] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999.

[7] D. Bommes, T. Lempfer, and L. Kobbelt. Global structure optimization of quadrilateral meshes. *Computer Graphics Forum*, 30(2):375–384, 2011.

[8] D. Bommes, B. Levy, N. Pietroni, E. Puppo, C. Silva, M. Tarini, and D. Zorin. State of the art in quad meshing. *Eurographics STARS (2012)*, 2012.

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

[10] M. Botsch and L. Kobbelt. A remeshing approach to multiresolution modeling. pages 189–196, 2004.

[11] M. Campen, D. Bommes, and L. Kobbelt. Dual loops meshing: quality quad layouts on manifolds. *ACM Trans. Graph.*, 31(4):1–11, 2012.

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

[13] J. Daniels, C. T. Silva, and E. Cohen. Semi-regular quadrilateral-only remeshing from simplified base domains. *Computer Graphics Forum (Proceedings of the Symposium on Geometry Processing)*, 28(5):1427–1435, 2009.

[14] J. Daniels, C. T. Silva, J. Shepherd, and E. Cohen. Quadrilateral mesh simplification. *ACM Trans. Graph.*, 27(5), 2008.

[15] B. Delaunay. Sur la sphere vide. pages 793–800, 1934.

[16] E. N. Dvorkin and K. J. Bathe. A continuum mechanics based four-node shell element for general non-linear analysis. *Eng. Comput.*, 1:77–88, 1984.

[17] M. Eck and H. Hoppe. Automatic reconstruction of b-spline surfaces of arbitrary topological type. In *SIGGRAPH'96*, pages 325–334, 1996.

[18] D. Eppstein and J. Erickson. Raising roofs, crashing cycles, and playing pool: applications of a data structure for finding pairwise interactions. *Discrete and Computational Geometry*, 22(4):569–592, 1999.

[19] D. Eppstein, M. T. Goodrich, E. Kim, and R. Tamstorf. Motorcycle graphs: canonical quad mesh partitioning. *Eurographics Symposium on Geometry Processing*, 27(5):1477–1486, 2008.

[20] M. Garland, A. Willmott, and P. S. Heckbert. Hierarchical face clustering on polygonal surfaces. *Symposium on Interactive 3D Graphics*, pages 49–58, 2001.

[21] A. Golovinskiy and T. Funkhouser. Randomized cuts for 3d mesh analysis. *ACM Transactions on Graphics (2008)*, 27(5), 2008.

[22] E. Gunpinar, H. Suzuki, Y. Ohtake, and M. Moriguchi. Generation of bi-monotone patches from quadrilateral mesh for reverse engineering. *Computer-Aided Design*, 45(2):440–450, 2013.

[23] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 295–302, 1994.

[24] H. C. Huang. A new nine node degenerated shell element with enhanced membrane and shear interpolation. *Int. J. Numer. Methods Eng.*, 22:73–92, 1986.

[25] J. Huang, M. Zhang, J. Ma, X. Liu, L. Kobbelt, and H. Bao. Spectral quadrangulation with orientation and alignment control. *ACM Trans. Graph.*, 27(5):147, 2008.

[26] D. Julius, V. Kraevoy, and A. Sheffer. Quasi-developable mesh segmentation. *Computer Graphics Forum*, 24(3):581–590, 2005.

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

[28] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. pages 61–70, 2006.

[29] D. Kovacs, A. Myles, and D. Zorin. Anisotropic quadrangulation. *Computer Aided Geometric Design*, 28(8):449–462, 2011.

[30] N. Kowalski, F. Ledoux, and P. Frey. A pde based approach to multidomain partitioning and quadrilateral meshing. *Proceedings of the 21st International Meshing Roundtable*, pages 137–154, 2013.

[31] D. A. Lavender and D. R. Hayhurst. An assessment of higher-order isoparametric elements for solving an elastic problem. *Comput. Methods Appt. Mech. Eng.*, 56:139–165, 1986.

[32] G. Lavoue, F. Dupomt, and A. Baskurt. Curvature tensor based triangle mesh segmentation with boundary rectification. *IEEE Computer Graphics International*, pages 10–17, 2004.

[33] H. Lin, W. Chen, and H. Bao. Adaptive patch-based mesh fitting for reverse engineering. *Computer-Aided Design*, 39(12):1134–1142, 2007.

[34] S. H. Lo and C. K. Lee. On using mesh of mixed element types in adaptive finite element analysis. *Finite Elements Anal. Des.*, 11:307–336, 1992.

[35] W. Ma, X. Ma, S. Tso, and Z. Pan. A direct approach for subdivision surface fitting from a dense triangle mesh. *Computer-Aided Design*, 36(6):525 – 536, 2004.

[36] A. P. Mangan and R. T. Whitaker. Partitioning 3d surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):308–321, 1999.

[37] M. Marinov and L. Kobbelt. Direct anisotropic quad-dominant remeshing. *Proceedings of the Computer Graphics and Applications*, 24(3):207–216, 2004.

[38] I. B. Martin, H. Rushmeier, and J. Jin. Parameterization of triangle meshes over quadrilateral domains. pages 193–203, 2004.

[39] A. Myles, N. Pietroni, D. Kovacs, and D. Zorin. Feature-aligned t-meshes. *ACM Trans. Graph.*, 29(4):1–11, 2010.

[40] A. Myles and D. Zorin. Global parameterization by incremental flattening. *ACM Trans. Graph.*, 31(4):1–11, 2012.

[41] D. Panozzo, Y. Lipman, E. Puppo, and D. Zorin. Fields on symmetric surfaces. *ACM Trans. Graph.*, 31(4), 2012.

[42] C. Peng, E. Zhang, Y. Kobayashi, and P. Wonka. Connectivity editing for quadrilateral meshes. *ACM Trans. Graph.*, 30(6):141, 2011.

[43] H. Pottmann, S. Leopoldseder, and M. Hofer. Approximation with active b-spline curves and surfaces. *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, pages 8–25, 2002.

[44] N. Ray, W. C. Li, B. Levy, A. Sheffer, and P. Alliez. Periodic global parameterization. *ACM Trans. Graph.*, 25(4):1460–1485, 2006.

[45] T. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and t-nurccs. *ACM Trans. Graph.*, 22(3):477–484, 2003.

[46] A. Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556, 2008.

[47] V. Surazhsky, P. Alliez, and C. Gotsman. Isotropic remeshing of surfaces: a local parameterization approach. pages 215–224, 2003.

[48] M. Tarini, N. Pietroni, P. Cignoni, D. Panozzo, and E. Puppo. Practical quad mesh simplification. *Computer Graphics Forum*, 29(2):407–418, 2010.

[49] M. Tarini, E. Puppo, D. Panozzo, N. Pietroni, and P. Cignoni. Simple quad domains for field aligned mesh parametrization. *ACM Trans. Graph.*, 30(6):142, 2011.

[50] T. Várady, M. A. Facello, and Z. Terék. Automatic extraction of surface structures in digital shape reconstruction. *Computer-Aided Design*, 39(5):379–388, 2007.

[51] T. Várady, R. R. Martin, and J. Cox. Reverse engineering of geometric models-an introduction. *Computer-Aided Design*, 29:255–268, 1997.

[52] M. Vieira and K. Shimada. Surface mesh segmentation and smooth surface extraction through region growing. *Computer-Aided Geometric Design*, 22(8):771–792, 2005.

[53] J. H. Wu and L. Kobbelt. Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum*, 24(3):277–284, 2005.

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

[55] M. Zhang, J. Huang, X. Liu, and H. Bao. A divide-and-conquer approach to quad remeshing. *IEEE Transactions on Visualization and Computer Graphics*, 99, 2012.

[56] J. Z. Zhu, E. Hintorc, and O. C. Zienkiewicz. Adaptive finite element analysis with quadrilaterals. *Comput. Struct.*, 40:1097–1104, 1991.

Publications

List of publications related to this thesis:

- **International Journal Papers.**

  1. Erkan Gunpinar, Hiromasa Suzuki, Yutaka Ohtake and Masaki Moriguchi, ”Generation of Bi-monotone Patches from Quadrilateral Mesh for Reverse Engineering”, Computer-Aided Design, 45(2): 440-450, February 2013 (Proc. of SPM2012).

- **International Journal Papers Under Review.**

  1. Erkan Gunpinar, Masaki Moriguchi, Hiromasa Suzuki and Yutaka Ohtake, ”Feature-aware Partitions from Motorcycle Graph”, Computer-Aided Design journal.
  2. Erkan Gunpinar, Masaki Moriguchi, Hiromasa Suzuki and Yutaka Ohtake, ”Motorcycle Graph Generation from a Semi-regular Quadrilateral Mesh for Reverse Engineering”, Graphical Models journal.

- **International Conference Papers with Review.**

  1. Erkan Gunpinar, Hiromasa Suzuki, Masaki Moriguchi, Yutaka Ohtake, ”Geometry-aware Motorcycle Graph Segmentation”, Proceedings of Asian Conference on Design and Digital Engineering (ACDDE) 2012, USB disk publication, Niseko, Japan, Dec 2012.

- **Domestic Conference Papers without Review.**

  1. Erkan Gunpinar, Hiromasa Suzuki, Yutaka Ohtake, Masaki Moriguchi, ”Extracting Smooth Surfaces from Quadrilateral Mesh Model”, Proceedings of the Spring Conference of The Japan Society Precision Engineering (JSPE) 2012, Tokyo, Japan, K15-3, May 2012.

List of publications and patents not related to this thesis:

- **International Journal Papers.**

  1. Erkan Gunpinar, and Soonhung Han, ”Interfacing heterogeneous PDM systems using the PLM services”, Advanced Engineering Informatics, 22(3):307-316, July 2008.

- **International Conference Papers with Review.**

  1. Erkan Gunpinar, Hanmin Lee, and Soonhung Han, ”Interfacing heterogeneous PDM systems by PLM services for design collaboration”, Proceedings of the 10th International Conference on CSCW in Design, Nanjing, China, 776 - 781, May 2006.

- **Patents.**

  1. E. Gunpinar, S. Rhee, K. Kwon, J. Park, ”Assembly Prediction System and Method of Curved Plates”, Samsung Heavy Industries, Korean Patent No. 10-1086389, Nov 17, 2011.
  2. E. Gunpinar, S. Rhee, K. Kwon, J. Park, ”Geometric Synchronization of Measured and Designed Curved Plate Models”, Samsung Heavy Industries, Korean Patent No. 10-2009-65376, July 17, 2009.
  3. K. Kwon, E. Gunpinar, S. Rhee, J. Park, Y. Park, ”Development of a New Method and Instrument for Plate Location Measurement”, Samsung Heavy Industries, Korean Patent No. 10-2009-67471, July 23, 2009.
  4. S. Rhee, E. Gunpinar, S. Kim, C. Cha, S. Hwan, ”A Modeling Method of 3D Closed Space Using 3D Measurement System”, Samsung Heavy Industries, Korean Patent No. 10-2007-119745, November 22, 2007.

Vita

Erkan Gunpinar graduated from The University of Tokyo on September 2013. His research interests are reverse engineering, geometric modeling, CAD/CAM, PDM and data exchange standards. From 2006 to 2010, Erkan worked for Samsung Heavy Industries as a research engineer in Mechatronics Center and had involved several projects to increase the speed and accuracy of manufacturing processes in shipbuilding. He earned his MSc in Mechanical Engineering from Korean Advanced Institute of Science and Technology (KAIST) with an e-manufacturing minor diploma in 2006 and his BSc in Mechanical Engineering from Middle East Technical University (METU) in 2003.