Available online at www.sciencedirect.com
ScienceDirect
Computer-Aided Design 00 (2013) 1–16
Computer-Aided Design

Feature-aware Partitions from Motorcycle Graph

Erkan Gunpinar*,[1] Masaki Moriguchi[2] Hiromasa Suzuki[1] Yutaka Ohtake[1]
[1]The University of Tokyo
[2]Chuo University

Abstract

Today's quad-meshing techniques generate high-quality quadrilateral meshes whose extraordinary vertices (i.e., not four-valence vertices except on the boundary) are generally located in highly curved regions. The motorcycle graph (MCG) algorithm of Eppstein et al. can be used to generate structured partitions of such quadrilateral meshes. However, it is not always possible for it to capture feature curves in the highly-curved parts of the model on the partition boundaries because model geometry is not taken into account.

This study investigated feature-aware algorithms representing extensions of the MCG algorithm. Initial partitioning is first performed using a speed control algorithm identical to the MCG algorithm except that it assigns variable rather than constant speed to particles. Partition boundaries are then improved via local path flipping operations. The MCG algorithm and the speed control algorithm are intended to trace as many feature curves as possible, but do not necessarily trace all of them. For this reason, feature curves are extracted and integrated into the proposed framework by adding seeds located at ordinary vertices in addition to extraordinary seeds. The proposed algorithm generates partitions that are still structured, and has been tested with quadrilateral mesh models generated using the mixed integer quadrangulation technique of Bommes et al.

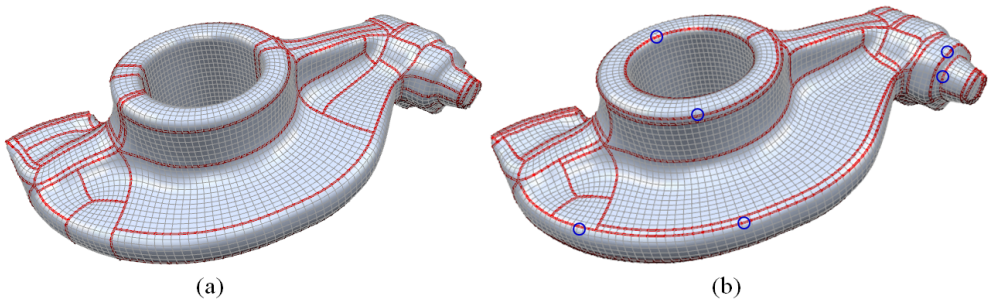
Keywords: Quadrilateral mesh, Motorcycle graph, Mesh segmentation

1. Introduction

State-of-the-art quad-meshing techniques have been proposed in recent years to support the generation of quadrilateral meshes from triangular meshes with optimized objective patch parameters (such as element alignment, orientation, sizing and aspect ratio). Besides applying to these parameters, many such techniques are intended to generate quadrilateral meshes with a simple underlying base complex of quad partitions in which each partition is *structured* of a regular grid involving no extraordinary vertices (i.e, not four-valence vertices except on the boundary). Such partitions are useful representations of objects, because they can be fitted with parametric surfaces with no parameterization step.

In terms of surface modeling applications (i.e. reverse engineering), it is crucial to locate highly curved parts of the model on partition boundaries as these regions residing in inner regions of partitions cannot be well represented by smooth surfaces. The motorcycle graph (MCG) algorithm proposed by Eppstein et al. [1] is a method that generates quad partitions from a given quadrilateral mesh. Partitions obtained using this algorithm do not always possess

*Email: erkan@den.rcast.u-tokyo.ac.jp **Adress:** Komaba 4-6-1, Meguro, Tokyo 153-8904, JAPAN **Tel:** +81(0)-3-5452-5185 **Fax:** +81(0)-3-5452-5186



(a) (b)

Figure 1. Quadrilateral mesh model partitioned by: (a) the motorcycle graph algorithm of Eppstein et al. (b) our algorithm. Our method locates the highly-curved regions on the partition boundaries, therefore better surface quality is expected.

highly-curved regions on their boundaries. Fig. 1 (a) 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. 1 (b)) on the partition boundaries, therefore better surface quality is expected when fitting parametric surfaces to the partitions generated using our method. Figure 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. 2 (b)) of highly-curved regions (as shown in blue in Fig. 2 (c)) are captured by the MCG algorithm.

This study examined the generation of feature-aware partitions with highly-curved regions (*feature curves*) located on partition boundaries. To achieve this, feature-aware algorithms representing extensions of the MCG algorithm are proposed. Figures 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. 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. 2 (d)). The partition layout is then improved using local path flipping operations (see Fig. 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. 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. 2. In this study, we have utilized quadrilateral mesh generated using mixed integer quadrangulation method [2].

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.

2. Related Work

Quad Meshing. Previous literature [3, 4] has outlined quad meshing comprehensively, as detailed selectively here. Some early methods [5, 6] made use of model anisotropy for quadrangulation. In these approaches, the principal curvature direction fields of the model are used to guide tracing iso-curves, 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 [2, 7, 8, 9, 10, 11] generate pure triangle-free quad meshes that fit the guiding fields as smoothly as possible. Unlike early methods [5, 6], the approach of Bommes et al. [2] utilizes only certain principal curvature directions, known as meaningful curvatures, to smoothly interpolate the cross fields. Huang et al. [12] 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 [14, 15] also involve using model anisotropy in the same way as some early techniques [5, 6]. Zhang et al. [14] aim to generate anisotropically sized quads, unlike the approach outlined in [2], using a wave-based quadrangulation method by which quad size can be controlled. The recently proposed method of Panozzo et al. [16] takes generalized symmetries of

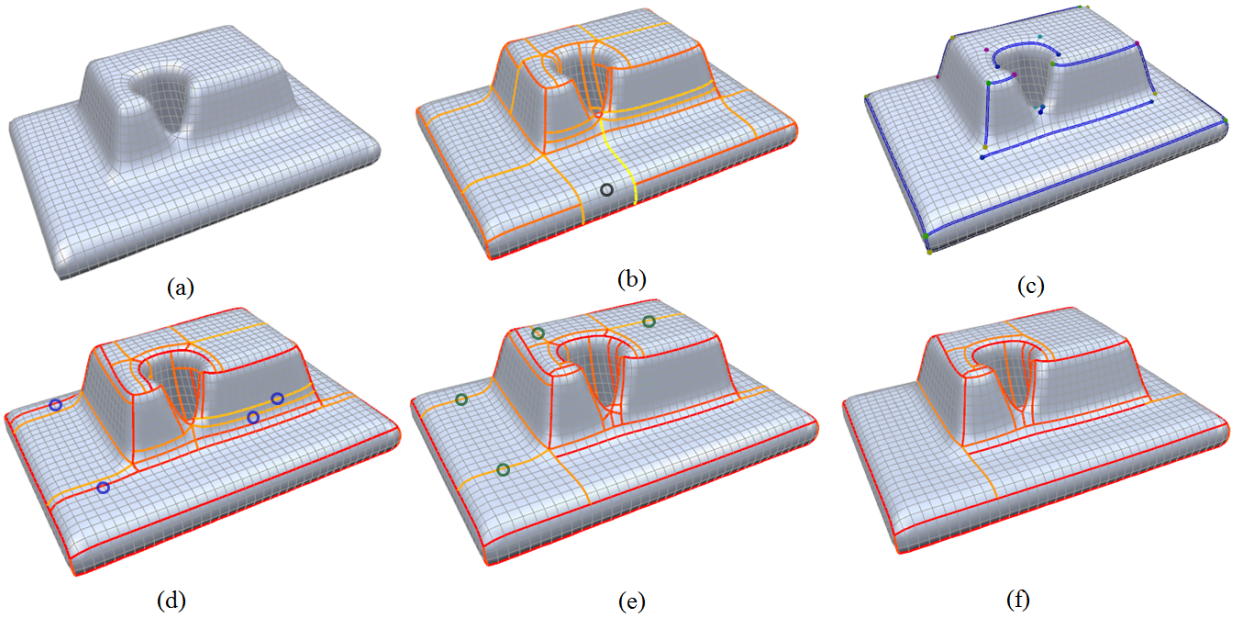


Figure 2. (a) Input quadrilateral mesh generated using the mixed integer quadrangulation [2] (b) Motorcycle graph partitioning using the technique of Eppstein et al. [1] The following images show the steps of feature-aware algorithms: (c) Particle (spheres in colors) placement for extraordinary vertices 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

the shape into account during quadrangulation. Zhang et al. [17] proposes a divide-and-conquer quadrangulation approach that uses the global structure information of the object such as symmetries, primitives, etc.

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 [18, 19, 20, 21, 22, 23, 24] have been proposed recently. Myles et al. [23] detail a singularity alignment problem for which misaligned pairs of extraordinary vertices are detected and then aligned on the parametric domain. [19, 21] disentangles the graph of separatrices in mesh by optimizing its global structure. Bommes et al. [19] propose an algorithm to detect helices and remove them using local connectivity operators by not changing the number or position of singularities. [21] aligns irregular vertices by changing the local connectivity with a sequence of steps. Campen et al. [24] 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.

Mesh Segmentation. A limited amount of work has been done on direct segmentation of quadrilateral meshes. Eppstein et al. [1] propose a motorcycle graph algorithm to segment a given quadrilateral mesh into regular sub-meshes (i.e., without extraordinary vertices). Myles et al. [23] 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. Gunpinar et al. [25] recently proposed a sequence of methods to segment quadrilateral meshes into bi-monotone patches that are appropriate for B-spline surface fitting and capture feature curves of input model well. There are also other works [26, 27, 28, 29] exploring methods for the generation of high-order surfaces.

The proposed method in this paper and the approaches [18, 1, 19, 20, 21, 22, 23, 24] all produce a quadrilateral layout. The methods [1, 23] and our method contains T-junctions, however all other methods are free from T-junctions. Once T-junctions are allowed, few number of quadrilateral patches are enough to represent the mesh which is important for reverse engineering applications.

3. Approach for Feature-aware Motorcycle Graph

Here, the motorcycle graph algorithm [1] is first summarized, then the proposed approach and basic terminology are described in relation to problem setting.

3.1. Motorcycle Graph (MCG) Algorithm

The MCG algorithm can be used for partitioning quadrilateral meshes into sub-meshes, each of which is structured so that no extraordinary vertices are contained. As recent quad-meshing techniques [4] generate high-quality quadrilateral meshes, using the MCG algorithm for quadrilateral mesh segmentation is much more useful because such high-quality meshes directly influence segmentation quality.

In the MCG 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. In this case, we say that the path is blocked by the other one. 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. In such cases, we say that the two paths collide. If three or four particles meet at a vertex simultaneously, they all stop. Eventually, the tracks (paths) of the particles form partition boundaries.

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. Such partitioning produces domains appropriate for reverse engineering, surface fitting and other purposes. From a topological aspect too, the MCG algorithm produces partitions (patches) having favorable shapes. As shown in [1], 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 segmentation. 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. [1] 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 [23, 22, 19, 24], 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.

3.2. 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 3 shows an example obtained using the MCG algorithm for a quadrilateral mesh generated via the mixed integer quadrangulation method [2]. 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. 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 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:

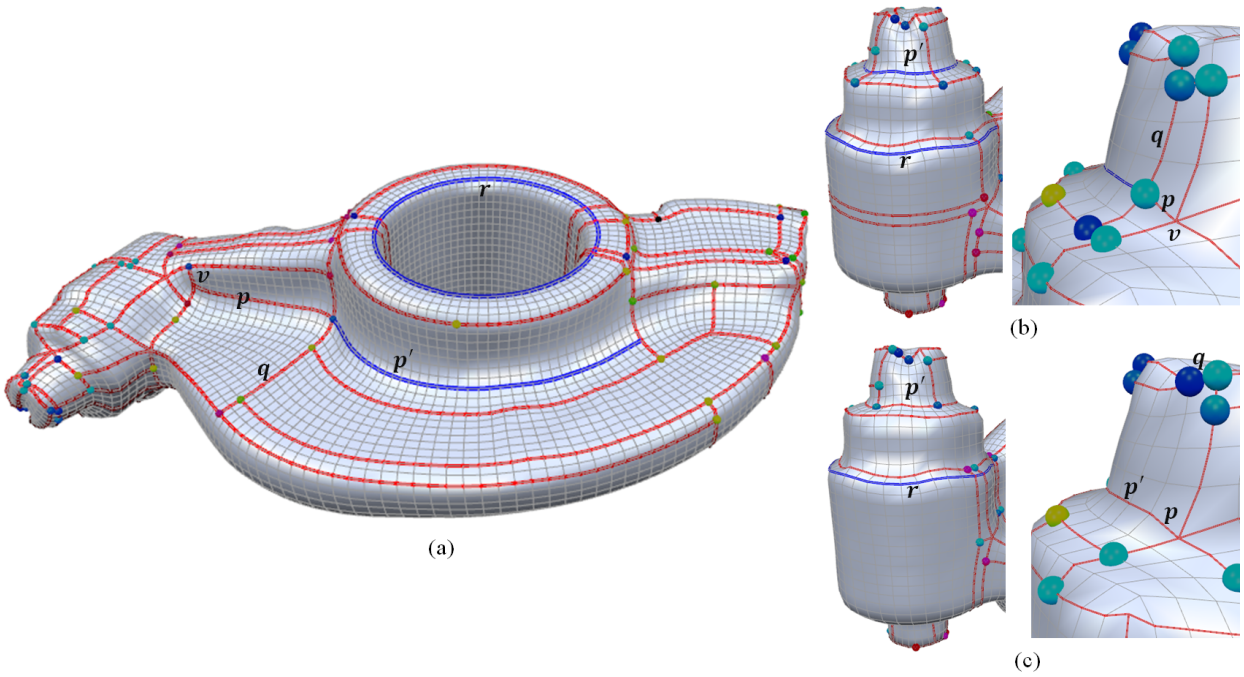


Figure 3. Red paths are traced by the MCG algorithm. It does not necessarily trace all feature curves (in blue). Only some portions of the feature curve p (not p') are traced. As there is no extraordinary vertex on the feature curve r, it is not traced either. After applying speed control algorithm, 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).

1. **Speed Control:** In the MCG algorithm, particles are given initial speeds and maintain them. If the particle starting from v in Figure 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 [1].
2. **Path Flipping:** A particle can reside on a non-feature edge whose adjacent edges are feature edges. Figure 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 3 (c)).
3. **Regular Seed Insertion:** The feature curves (depicted as r) in Figure 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.

3.3. 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.

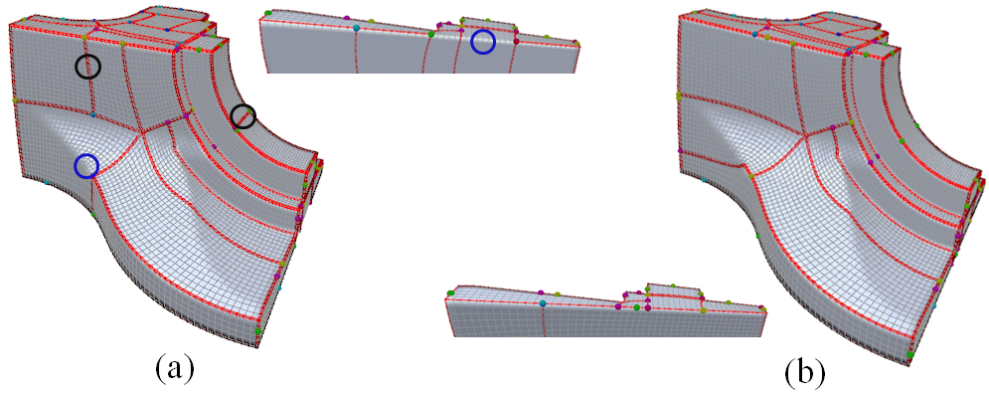


Figure 4. The SC algorithm ((b), cost $G_W = -1337.37$) generates partitioning different to that of the MCG algorithm ((a), $G_W = -1230.94$).

3.4. 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. Robust operators to identify features like field anisotropy such as in mixed-integer quadrangulation study [2] can also be utilized.

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 $D(W_k) = \sum_{e_j \in W_k} d(e_j)$. If $\theta(e_j) > \epsilon$, $d(e_j) = 0$. Otherwise it is set to 1. $D(W_k)$ is introduced because paths that are less curved and longer have a lower cost without it (see Fig. 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.

4. 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.

4.1. 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. For each particle i, time of arrival at next vertex (t_i) is assigned which is the ratio l_i/f_i. l_i is next edge length (in Eucledian length) and f_i is the speed. All particles are inserted in a heap (H) and sorted by shortest t_i, and current time t_c is set to 0. The top particle r in H is removed and advanced to its next vertex. t_c is then set to t_r. If particle r is arrived to already traced vertex or boundary vertex, or collided with another particle, it stops. Otherwise, t_c is set to t_r and t_r is updated as $t_r = t_c + l_r/f_r$ (l_r and f_r are also updated values). Particle r is then inserted in H. This procedure is performed until all particles stop. We define f_i based on the dihedral angle $\theta(e_i)$ of the next edge e_i ($f_i = \theta(e_i)$). To avoid zero values of f_i (otherwise t_i becomes undefined ($t_i = l_i/0$)), f_i is set to 0.01 if $\theta(e_i)$ is less than 0.01. The pseudo code in Algorithm 1 summarizes the SC algorithm.

Algorithm 1 The SC Algorithm

1: // Initialization
2: Compute t_i for each particle i ($t_i = l_i/f_i$)
3: Put all particles in a heap (H) and sort by shortest t_i
4: Set current time t_c to 0
5: // Loop for moving particles
6: Remove top particle r from H
7: Make particle r advance to next vertex
8: Update current time ($t_c = t_r$)
9: **if** particle R is arrived to already traced vertex or boundary vertex, OR collision occurs **then**
10: Goto line 16
11: **end if**
12: **else**
13: Update t_r of particle i ($t_r = t_c + l_r/f_r$)
14: Reinsert particle r in H
15: **endelse**
16: Repeat from line 6 until H is empty

Figure 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 it fully traces the feature curves shown (shown by blue circles), whereas the MCG algorithm does not. In contrast to the MCG algorithm, the SC algorithm does not produce undesirable segmentation 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.

4.2. 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. 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 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. 5 (c). In this case, the same PF operation can be applied (Fig. 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.

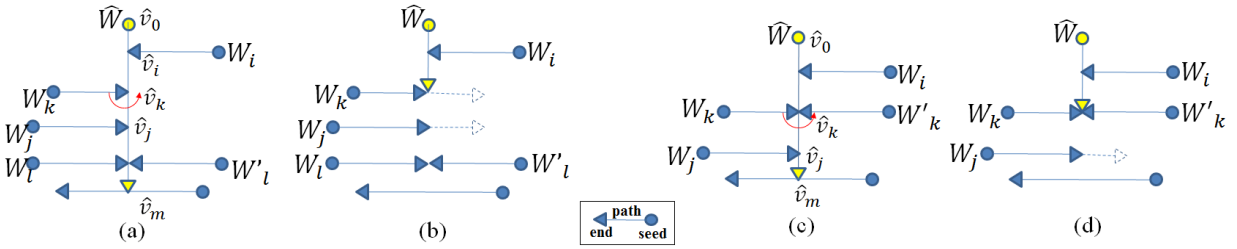


Figure 5. Paths (a) before and (b) after a PF operation at the vertex $\hat{v}_k$. Paths (c) before and (d) after PF operation at a + junction on the vertex $\hat{v}_k$.

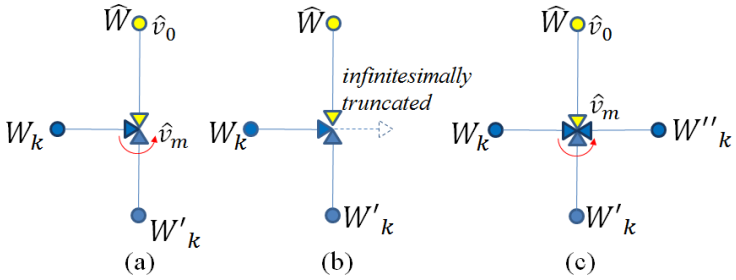


Figure 6. Paths (a) before and (b) after PF operation at a coincident T-junction on the vertex $\hat{v}_k$. The end vertex of $\hat{W}$ is infinitesimally truncated to prevent blocking at $\hat{v}_k$ by $\hat{W}$. (c) Four paths colliding at a vertex. PF operation can be applied, but no movement along these paths is possible because the blocking relationship cannot be resolved.

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 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. 6 (b)).

Figure 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 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 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.

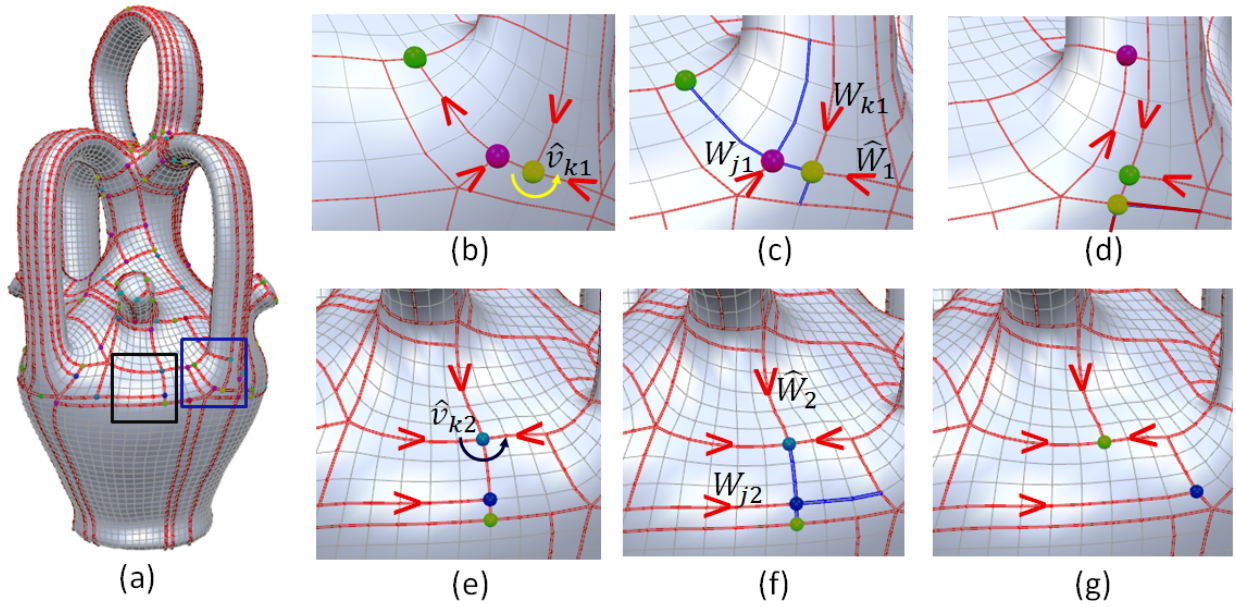


Figure 7. (a) Quadrilateral mesh model of a botijo container showing the traced paths (in red) and placed particles (colored dots). Images (b) to (d) show close-ups of the portion enclosed in the blue rectangle for PF operation at a T-junction. (b) Path configuration before PF operation at v_{k1} (c) 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.

4.3. 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. By discarding short feature curves, we can avoid the mesh being divided into too many partitions. And some discarded short feature curves can be traced by the SC and PF algorithms. Additionally, 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.

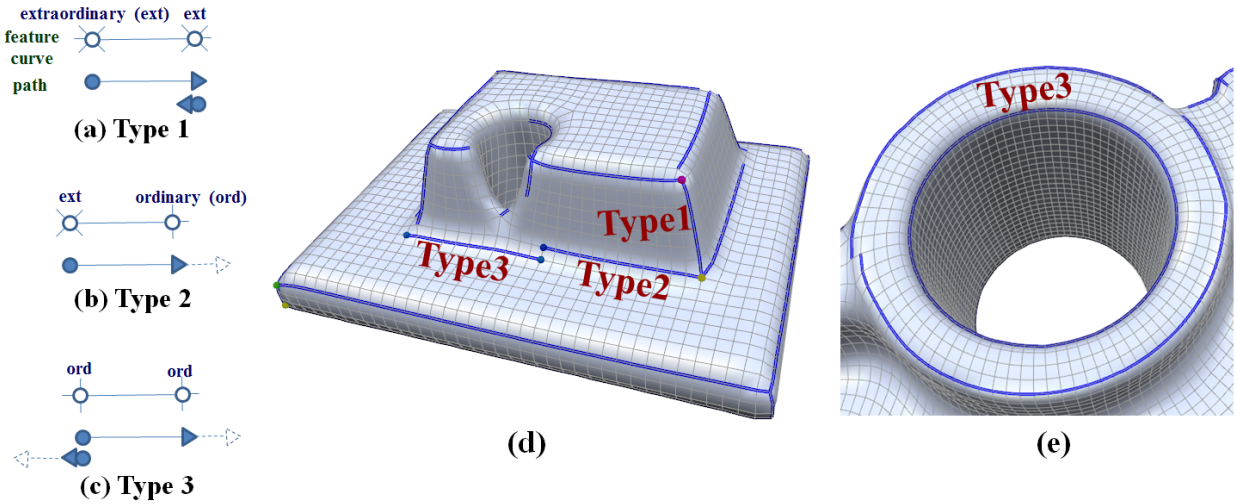


Figure 8. Feature curves. (a) **Type 1**: Starting from an extraordinary vertex and ending at an extraordinary vertex. (b) **Type 2**: Starting from an extraordinary vertex and ending at an ordinary vertex. (c) **Type 3**: Starting from an ordinary vertex and ending at an ordinary vertex. (d) The three feature curve types shown with added paths (in blue) and particles (colored dots) (e) The closed feature curve of Type 3

There exist several robust feature detection methods which can be used. For a triangle mesh with severe noise, mixed integer quadrangulation (MIQ) [2] 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. Note that feature curves detected using other sophisticated algorithms can easily be integrated to our framework if their integration do not violate motorcycle graph conditions.

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. 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 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 8 (d, e) shows examples of extracted feature curves.

5. Results and Discussion

The proposed algorithms are implemented, and experiments are conducted using four quadrilateral meshes generated based on the mixed integer quadrangulation method [2] for a drill hole (see Fig. 2), a rockerarm, a Beetle and a bottle. Figure 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

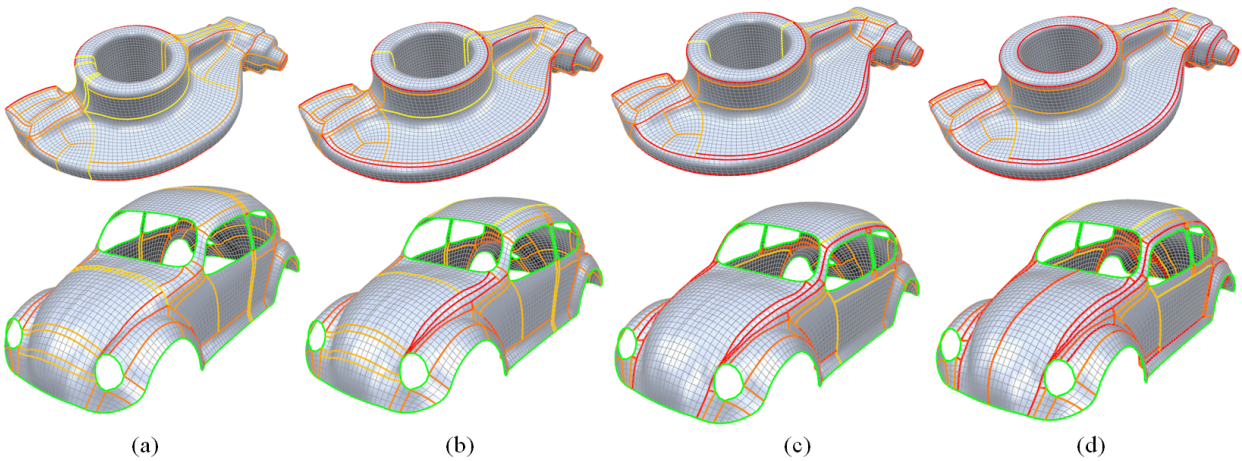


Figure 9. Partitions after applying (a) the MCG algorithm ($G_W = -41.2277$ (top), $G_W = 128.663$ (bottom)), (b) the SC algorithm ($G_W = -237.322$ (top), $G_W = 91.6073$ (bottom)), (c) the PF algorithm ($G_W = -382.521$ (top), $G_W = -25.8289$ (bottom)), and (d) the FC algorithm ($\tau = 15$ and $G_W = -525.127$ (top), $\tau = 8$ and $G_W = -121.639$ (bottom).

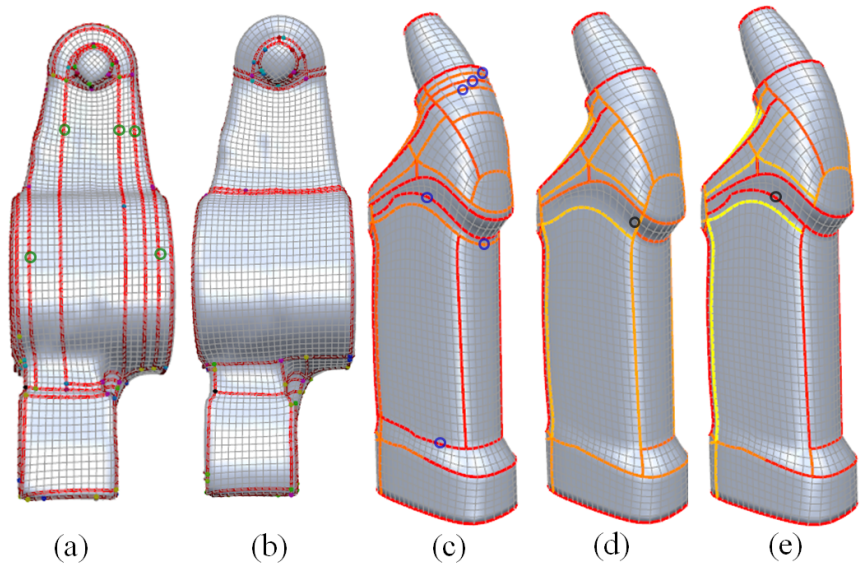


Figure 10. Paths of rockerarm model after path flipping operations when (a) $\alpha = 0.0$ (b) $\alpha = 0.4$, $\epsilon = 0.2$. To minimize the sum of edge curviness, undesirable low curved long paths are produced for $\alpha = 0.0$. Setting α to 0.4 reduces the number of such paths in (b); Partitioning of the bottle model ($\tau = 15$) with the parameters (c) $\alpha = 0.0$, (d) $\alpha = 0.4$, and (e) $\alpha = 0.8$

(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 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 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.

Path Coloring: A color is assigned for each path depending on the path cost in Figures 2, 9, 10 ((c) to (e)) and 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.

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 which is set to 0 for all models tested in this study. 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. Figures 2 (f) and 11 (a) show partitions seen after the removal process.

By removing a path, the shape of the boundary may become worse in terms of the reverse engineering. An additional criterion called as *boundary smoothness criterion* is considered during the removal process. If a sharp corner (like a kink) occurs on the boundary after this process, the removal is discarded. We simply check the angle between two neighboring edges (in red and orange on the right image) of the first and last edge (in black) in the paths. If this angle is larger than the parameter P (set to 25 degree), removal is discarded. Otherwise, it is performed. Figure 11 (b) shows partitions when the boundary smoothness criterion is used. As a part of the future work, we will investigate algorithms considering several criteria during removal process.

B-spline Surface Fitting: Quadrilateral mesh of [2] 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 $0 <= i <= m$ and $0 <= j <= n$ representing indices of columns and rows respectively. *Two-stage curve blending* is performed 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:

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

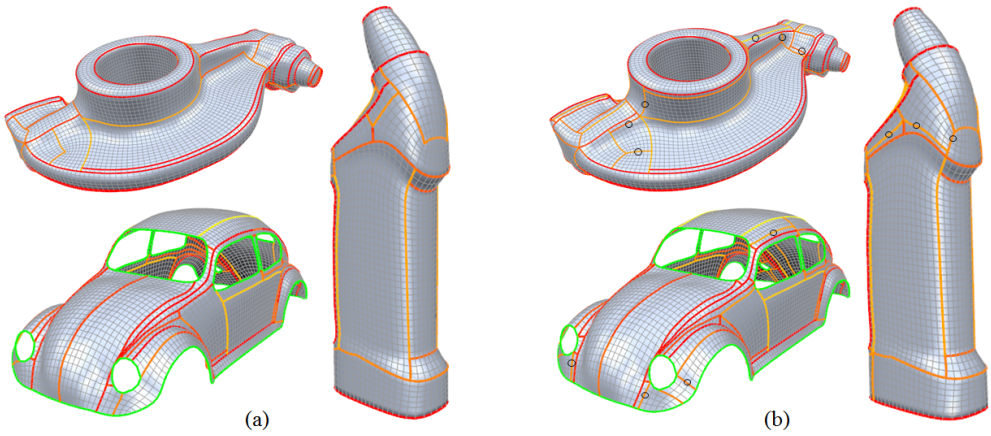


Figure 11. (a) Partitions after removal of flat (less curved) paths without consideration of the boundary smoothness criterion and (b) with consideration. Boundaries, marked with black circles in (b), are the boundaries that should not be removed in order to avoid generation of unsmooth boundaries.

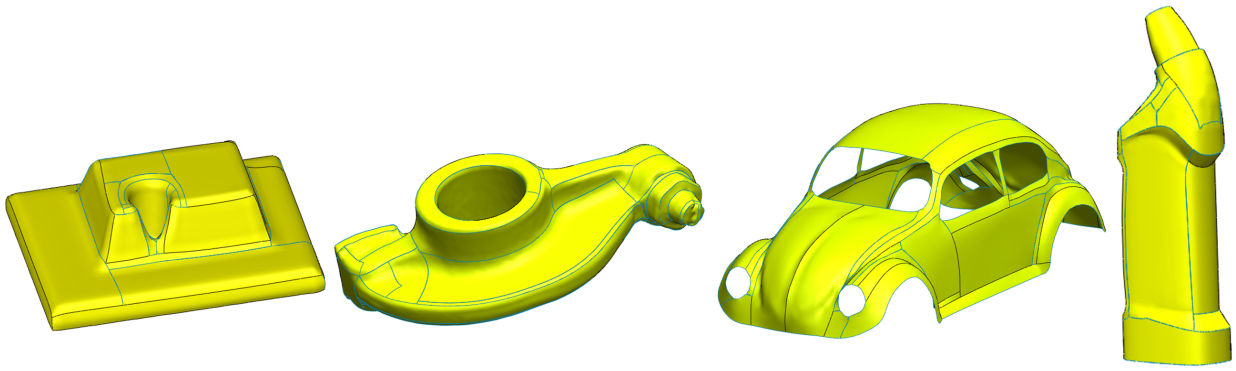


Figure 12. B-spline surfaces for the partitions (obtained after path removal step without the consideration of the boundary smoothness criterion) generated using our method.

Figure 12 shows B-spline surfaces for the partitions after flat path removal (without the consideration of the boundary smoothness criterion), and better surface quality is expected when our partitions are fitted with T-splines.

Comparison: There are several existing methods [1, 23, 22, 19, 24, 25] whose partitions can be suitable for B-spline surface fitting. Compared to other methods, the generated partitions of [25] 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 [25] and of our method with B-spline surfaces. Then these surface models are compared with the input quadrilateral mesh. Fig. 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 [25] (for its particular choice of thresholds), since our method improves the initial partitioning by local path flipping operations.

Fig. 14 shows the surface deviations for the partitions of [1] (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. 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. 14 (b)). The blue circle in Fig. 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).

Fig. 15 shows the partitions (for their particular choice of thresholds) generated using [23, 22, 19, 24] for which our algorithm generates a partition shown in Fig. 11 (a). As far as this rockerarm model is concerned with its results

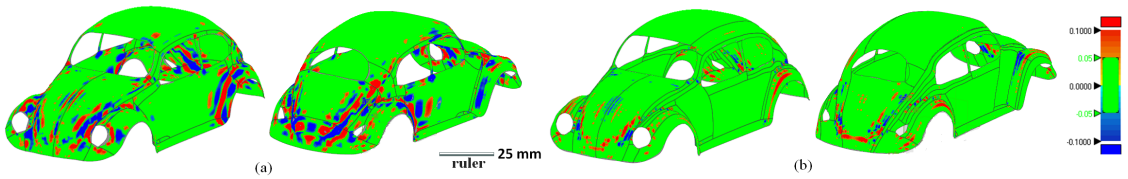


Figure 13. Deviations from the input quadrilateral mesh for the surfaces of partitions generated using: (a) the method of [25] (b) our method. Since our method captures the highly curved parts well, there is less surface fitting error compared to [25].

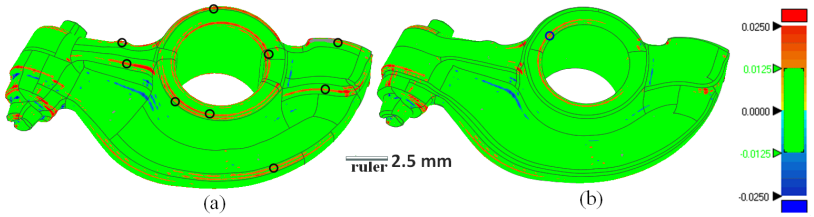


Figure 14. Deviations from the input quadrilateral mesh for the surfaces of partitions generated using: (a) the method of [1] (b) our method. Regions marked with black circles (top image in (b)) have large deviations since highly curved regions reside inside the B-spline surfaces and are not represented well with smooth surfaces.

shown in the papers [23, 22, 19, 24], the result of our algorithm looks less problematic for the B-spline surface fitting because the highly curved regions (marked with black circles) are located inside the partition boundaries. Recall that T-junctions do not exist in the partitions of [22, 19, 24], whereas they exist in that of [23] and the proposed method. [22, 19, 24] can support the features (that we marked in this paper) with the cost of increasing the number of extraordinary vertices, thereby the number of the quadrilateral patches in the base complex would be very high. Therefore, our work can more effectively place highly curved regions on the partition boundaries compared to the recent works [23, 22, 19, 24].

Further Evaluations: Right table lists up the number of T-junctions and quadrilateral partitions generated, and processing time (t) of the proposed method. The parameters written on the top column of the right table N_{t1}-N_{s1}, N_{t2}-N_{s2} and N_{t3}-N_{s3} refer to the number of *T-junctions-quadrilateral partitions* before the flat boundary removal process, after this process without and with the consideration of boundary smoothness criterion respectively. A 2.93 GHz PC was used for the experiments in this study. t_{FC}, t_{SC}, t_{PF} and t_R refer to the time taken for the FC, SC, PF and flat path removal algorithms respectively, and t_T is the total processing time. The PF, FC and flat path removal algorithms take negligible amounts of time in all cases. The processing times for the SC algorithm are less than 4 seconds for the rockerarm model, and less than 1 second for all other models. The time taken for the SC algorithm depends on the resolution of the model (i.e., number of edges in the model).

Models	N_{t1}	N_{s1}	N_{t2}	N_{s2}	N_{t3}	N_{s3}	t_{FC}	t_{SC}	t_{PF}	t_R	t_T
drill hole	56	52	35	31	51	47	0.078	0.577	0.078	0.062	0.795
rockerarm	119	98	81	62	107	87	0.156	3.806	0.156	0.125	4.243
bottle	56	58	33	33	36	47	0.079	0.671	0.076	0.093	0.91
Beetle	102	92	59	50	82	72	0.078	0.842	0.187	0.109	1.216

Limitations: The quality of the segmentation is heavily dependent on the quality of the quadrilateral mesh input. In other words, if there are too many extraordinary vertices in the mesh, our method will generate many quadrilateral partitions which may not be desirable.

6. Conclusions and Future Works

The method presented in this paper can be used to 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,

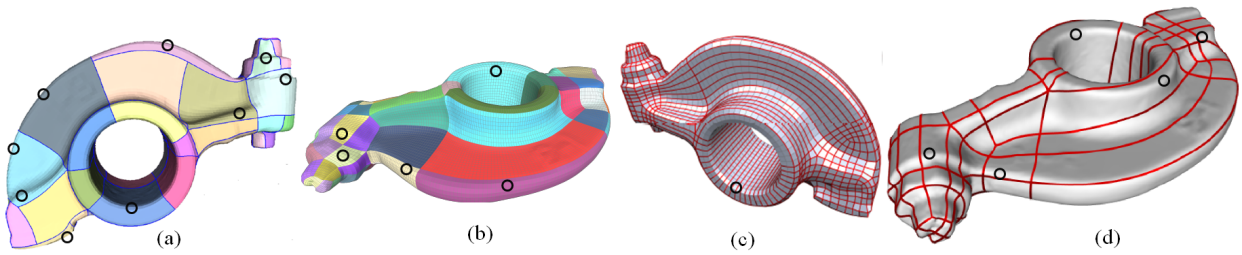


Figure 15. Partitions generated using the method of: (a) [23] (b) [22], (c) [19], (d) [24] (images taken from [23, 22, 19, 24]). Highly curved regions (marked with black circles) can be seen inside the partition boundaries which will be problematic for the B-spline surface fitting.

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.

In future work, the authors plan to find ways to expand the space of the motorcycle graph as it is rather narrower than that of all the structured partitions. The optimization procedure based on local PF operations also needs to be extended to reach nearer to the global optimal solution. As an application of partitioning, the proposed method is expected to work well with a new T-spline surface modeling approach. It must be interesting to feed our resulting partitions to the T-spline framework [30]. In addition to straight feature curves, we would like to extract and integrate the feature curves with arbitrary shapes into the proposed algorithm. Finally the use of other cost function(s) and better flat path removal algorithms will be investigated.

Acknowledgements

The authors would like to thank the anonymous reviewer for his suggestion that makes the speed control algorithm more efficient, the other anonymous reviewers for their comments that improves the paper, the RWTH Computer Graphics Group for making the OpenFlipper framework available, and Dr. Leif Kobbelt for providing the quadrilateral mesh models tested in this study.

References

[1] D. Eppstein, M. T. Goodrich, E. Kim, R. Tamstorf, Motorcycle graphs: canonical quad mesh partitioning, Eurographics Symposium on Geometry Processing 27 (5) (2008) 1477–1486.
[2] D. Bommes, H. Zimmer, L. Kobbelt, Mixed-integer quadrangulation, ACM Trans. Graph. 28 (3) (2009) 1–10.
[3] P. Alliez, G. Ucelli, C. Gotsman, M. Attene, Recent advances in remeshing of surfaces, Research report, AIM@SHAPE Network of Excellence (2005).
[4] D. Bommes, B. Levy, N. Pietroni, E. Puppo, C. Silva, M. Tarini, D. Zorin, Quad-mesh generation and processing: a survey, Computer Graphics Forum.
[5] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Levy, M. Desbrun, Anisotropic polygonal remeshing, ACM Trans. Graph. 22 (3) (2003) 485–493.
[6] M. Marinov, L. Kobbelt, Direct anisotropic quad-dominant remeshing, Proceedings of the Computer Graphics and Applications 24 (3) (2004) 207–216.
[7] F. Kalberer, M. Nieser, K. Polthier, Quadcover - surface parameterization using branched coverings, Computer Graphics Forum 26 (3) (2007) 375–384.
[8] N. Ray, W. C. Li, B. Levy, A. Sheffer, P. Alliez, Periodic global parameterization, ACM Trans. Graph. 25 (4) (2006) 1460–1485.
[9] N. Pietroni, M. Tarini, O. Sorkine, D. Zorin, Global parametrization of range image sets, ACM Transactions on Graphics, Proceedings of SIGGRAPH Asia 2011 30 (6).
[10] A. Myles, D. Zorin, Global parameterization by incremental flattening, ACM Trans. Graph. 31 (4) (2012) 1–11.
[11] N. Kowalski, F. Ledoux, P. Frey, A pde based approach to multidomain partitioning and quadrilateral meshing, Proceedings of the 21st International Meshing Roundtable (2013) 137–154.

[12] J. Huang, M. Zhang, J. Ma, X. Liu, L. Kobbelt, H. Bao, Spectral quadrangulation with orientation and alignment control, ACM Trans. Graph. 27 (5) (2008) 147.
[13] J. Daniels, C. T. Silva, E. Cohen, Semi-regular quadrilateral-only remeshing from simplified base domains, Computer Graphics Forum (Proceedings of the Symposium on Geometry Processing) 28 (5) (2009) 1427–1435.
[14] M. Zhang, J. Huang, X. Liu, H. Bao, A wave-based anisotropic quadrangulation method, ACM Trans. Graph. 29 (4) (2010) 1–8.
[15] D. Kovacs, A. Myles, D. Zorin, Anisotropic quadrangulation, Computer Aided Geometric Design 28 (8) (2011) 449–462.
[16] D. Panozzo, Y. Lipman, E. Puppo, D. Zorin, Fields on symmetric surfaces, ACM Trans. Graph. 31 (4).
[17] M. Zhang, J. Huang, X. Liu, H. Bao, A divide-and-conquer approach to quad remeshing, IEEE Transactions on Visualization and Computer Graphics 99.
[18] J. Daniels, C. T. Silva, J. Shepherd, E. Cohen, Quadrilateral mesh simplification, ACM Trans. Graph. 27 (5).
[19] D. Bommes, T. Lempfer, L. Kobbelt, Global structure optimization of quadrilateral meshes, Computer Graphics Forum 30 (2) (2011) 375–384.
[20] M. Tarini, N. Pietroni, P. Cignoni, D. Panozzo, E. Puppo, Practical quad mesh simplification, Computer Graphics Forum 29 (2) (2010) 407–418.
[21] C. Peng, E. Zhang, Y. Kobayashi, P. Wonka, Connectivity editing for quadrilateral meshes, ACM Trans. Graph. 30 (6) (2011) 141.
[22] M. Tarini, E. Puppo, D. Panozzo, N. Pietroni, P. Cignoni, Simple quad domains for field aligned mesh parametrization, ACM Trans. Graph. 30 (6) (2011) 142.
[23] A. Myles, N. Pietroni, D. Kovacs, D. Zorin, Feature-aligned t-meshes, ACM Trans. Graph. 29 (4) (2010) 1–11.
[24] M. Campen, D. Bommes, L. Kobbelt, Dual loops meshing: quality quad layouts on manifolds, ACM Trans. Graph. 31 (4) (2012) 1–11.
[25] E. Gunpinar, H. Suzuki, Y. Ohtake, M. Moriguchi, Generation of bi-monotone patches from quadrilateral mesh for reverse engineering, Computer-Aided Design 45 (2) (2013) 440–450.
[26] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, W. Stuetzle, Piecewise smooth surface reconstruction, Proceedings of the 21st annual conference on Computer graphics and interactive techniques (1994) 295–302.
[27] H. Pottmann, S. Leopoldseder, M. Hofer, Approximation with active b-spline curves and surfaces, Proceedings of the 10th Pacific Conference on Computer Graphics and Applications (2002) 8–25.
[28] D. Cohen-Steiner, P. Alliez, M. Desbrun, Variational shape approximation, ACM Transactions on Graphics 23 (3) (2004) 905–914.
[29] W. Ma, X. Ma, S. Tso, Z. Pan, A direct approach for subdivision surface fitting from a dense triangle mesh, Computer-Aided Design 36 (6) (2004) 525 – 536.
[30] T. Sederberg, J. Zheng, A. Bakenov, A. Nasri, T-splines and t-nurccs, ACM Trans. Graph. 22 (3) (2003) 477–484.