Generation of Bi-monotone Patches from Quadrilateral Mesh for Reverse Engineering

Erkan Gunpinar a Hiromasa Suzuki a Yutaka Ohtake a Masaki Moriguchi b

aThe University of Tokyo
bChuo University

Abstract

Thanks to recent improvements, computational methods can now be used to convert triangular meshes into quadrilateral meshes so that the quadrilateral elements capture well the principal curvature directional fields of surfaces and intrinsically have surface parametric values. In this study, a quadrilateral mesh generated using the mixed integer quadrangulation technique of Bommes et al. is used for input. We first segment a quadrilateral mesh into four-sided patches. The feature curves inside these patches are then detected and are constrained to act as the patch boundaries. Finally, the patch configuration is improved to generate large patches. The proposed method produces bi-monotone patches, which are appropriate for use in reverse engineering to capture the surface details of an object. A shape control parameter that can be adjusted by the user during the patch generation process is also provided to support the creation of patches with good bi-monotone shapes. This study mainly targets shape models of mechanical parts consisting of major smooth surfaces with feature curves between them.

Key words: mesh segmentation, quadrilateral mesh, bi-monotone patches, b-spline surface fitting.

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 patches 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 patches 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 1 (a)), it may be necessary to allow patches with arbitrary shapes. However, this is problematic because it results in an increased number of trimmed curves.

We take the concept in between quad patches and patches with arbitrary shapes which is called as bi-
monotone patches. 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. 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 patch, 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 patch.

This study aims to generate large-size bi-monotone patches where the feature curves reside on the boundary. We first outline the partitioning a quadrilateral mesh into four-sided patches, then describe the detection of feature curves inside these patches. If a patch has feature curves, we split it into bi-monotone patches along these curves. Finally, patch configuration is improved to generate large patches. As it is undesirable to obtain patches 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 patch generation process. In this study, we mainly target shape models of mechanical parts which consisting of major smooth surfaces with feature curves between them.

1.1. Related work

Quad remeshing. 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 [3,4]. Some methods [5,6] propose an anisotropic remeshing algorithm which explicitly trace curves along the principal curvature directions. Such approaches generate quad-dominant meshes with many extraordinary vertices. Some other methods [7,8,1] have involved the initial construction of a globally smooth parameterization of the mesh and the subsequent extraction of a quadrilateral mesh as a collection of iso-parameter lines. [1] reduces the quadrangulation problem into mixed-integer problems and converts the given triangular mesh into a quadrilateral mesh by optimizing quality aspects such as element quality, orientation, alignment and global structure (i.e., the distribution of extraordinary vertices). Some other recent methods [9–12] can also generate quadrilateral meshes of good quality.

Mesh segmentation. Mesh segmentation techniques have a long tradition in the graphics community and a considerable body of literature surveys [13,14] on the topic exists. Some methods [15–17] involve utilizing surface curvature characteristics to segment a mesh into sub-meshes. [16] applies morphological watersheds (an image segmentation technique) to 3D surfaces, and [17] partitions the mesh while iterating between region growing and surface fitting. Other methods [18,19,21,20] involve using a geometric measure or filter for clustering or approximating mesh elements. [18] proposes a hierarchical clustering methodology utilizing edge contraction on a dual graph of the mesh, while [19] separates the mesh into sub-meshes, each approximated by geometrical primitives. [20] extends the optimization technique of geometric surface approximation [21] by allowing spheres, cylinders and rolling-ball blend patches in addition to planes.

Recently a number of mesh partitioning methods [2,24,23] involving decomposition of a given mesh into compact quad sub-meshes have been proposed. [2] suggests a method based on the motorcycle graph of Eppstein and Erickson [22] to partition semi-regular quadrilateral meshes into structured quadrilateral meshes. [23] optimizes the global structure of a given quadrilateral mesh by detecting and removing helices without changing the number and the position of singularities. The base complex (the union of all parametric lines starting and ending at singularities) of the optimized quadrilateral mesh is a coarse quadrilateral mesh, and is therefore good for fitting with surface patches. [24] proposes an algorithm to generate coarse quadrilateral patches which are appropriate to fit with T-splines. Here, a greedy optimization approach is applied to find a solution to several trade-offs in consideration of quality aspects such as patch quality, approximation, mesh complexity, orientation and alignment.

2. Method Overview

In this section, we describe the flow of the proposed method and define some fundamental elements used in the algorithm.

2.1. Algorithm Flow

Figure 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. [1] to generate large-size bi-monotone patches. Section 3 describes an algorithm which partitions the mesh into quadrilateral patches. Then a feature detection algorithm will be explained in Section 4. Section 5 and 6 details the generation of the bi-monotone patches.

2.2. Quadrilateral Mesh

A quadrilateral mesh is a polygonal mesh in which all faces are quadrilaterals, and can be defined by a set of vertices embedded in $\mathbb{R}^2$, a set of edges and a set of quadrilateral faces. A vertex in a quadrilateral mesh is called ordinary if its valence is four for interior vertices, or two or
three for boundary vertices. Vertices that are not ordinary are called extraordinary.

Parametric directions starting from singularities can be defined topologically at each ordinary vertex of a quadrilateral mesh as \( u, v, -u, -v \) by counter-clockwise labeling. Furthermore, tracing edges based on these parametric directions on ordinary vertices generates parametric lines. As a result, the quadrilateral mesh can be used for surface parameterization, and therefore for generating parametric surfaces.

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 [2]. The mesh is then partitioned into several sub-meshes with boundaries formed by the motorcycle graph edges. In this step, the extraordinary vertices (valence \( \neq 4 \)) 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 paper, 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 [2] is to decompose semi-regular quadrilateral meshes into structured quadrilateral sub-meshes based on the motorcycle graph concept of [22]. 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 [2], except the EMG algorithm checks the angle of two consecutive edges on the path during tracing. If it meets an angle \( \alpha \) whose value (in degrees) exceeds the user-defined threshold \( T_\psi \), it stops tracing and branches into right and left paths (see Figure 3). In this way, paths can go through highly curved features.
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 patch, and each patch has a regular quadrilateral mesh structure. As the final step, patches with annulus topology are partitioned to create two patches with disk-shaped topology to facilitate the operation of the next processes.

Figure 4 shows the results of motorcycle and extended motorcycle graph algorithms for 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 patches (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).

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 patches. In particular, the sharp features inside the patches cannot be represented by smooth B-spline surfaces.

This section details the extraction of surface features constrained to be boundaries of patches 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-patches. It must also be elongated through the surface.
- **Position:** They must go through highly curved patches to enable splitting into sub-patches 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.

4.1. Feature region detection

Let $Q$ be a patch 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, $< q_1, q_2, \ldots, q_m >$, 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 patches.

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 [25] 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 $T_F$. 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 patch $Q$, a face consists of four edges. In the inner part of the patch, 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 $T_F$.

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 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. 5 (b), only one non-flat region is detected in the patch. 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. 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. 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 $T_{\phi}$, the non-flat region is called a feature region. Otherwise, it is no longer considered a non-flat region (see Fig. 5 (d)). If a feature region encircles flat faces (holes) in the patch, 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. 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.

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 $\theta$ of the unsigned dihedral angles of all edges of $C_k$. If the average $\theta$ exceeds $\pi/2$, then $d(C_k)$ is 0. Otherwise, it is $1 - \theta/\pi$. 

Fig. 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 patch enclosed in the green rectangle. Red curves are patch boundaries. (b) Detection of non-flat regions (non-flat faces are shown in white, and non-flat edges are shown in green). There is one non-flat region in the patch; 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.
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.

Threshold $T_p$, 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.

5. The Quadrivial Region Growing (QRG) Process

In this step, we grow bi-monotone patches 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 patches with better shapes. Readers should refer to Figure 7 for the illustration of the QRG algorithm.

After the EMG and feature detection processes, the quadrilateral mesh $M$ is decomposed into several patches. Each patch $Q$ 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.

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 patch $R$ which is initialized as empty.

The second step of the QRG process involves growing the patch $R$ 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, $[a, b] \cap [c, d] \neq \emptyset$. 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 patch $R$. 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 patch of quadrilateral faces, and is added to the patch group $G$. If an unconquered face remains in $Q$, we repeat the above region-growing algorithm to obtain other regions.

A straightforward method of finding $ch[c, d; j + 1]$ is de-
scribed here. It is important to note that in order to make
the patch 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:
(i) Define a general poly-chord ch[a, b; j + 1] with the
same span as ch[a, b, j].
(ii) 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_j; j + 1], k = 1, 2, · · · , W , where a_k < b_k for k = 1, 2, · · · , W , and
b_k < a_k+1 for k = 1, 2, · · · , W − 1.
(iii) 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’ ≤ a. Then
if b_k = UB, extend ch[a_k, b_k; j + 1] to be ch[a_k, b’; j + 1]
so that b ≤ b’.
(iv) Select the longest poly-chord from ch[a’, b_0; j +
1], ch[a_1, b_1; j + 1], · · · , ch[a_k, b_k; j + 1] to be ch[c, d; j + 1].
(v) 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 patches.

Proof: The proposed algorithm prevents the growing of
a non-monotonic patch 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 if there are feature curve edges
between the poly-chord and its neighboring poly-chord so

![Fig. 7. Illustration of the QRG algorithm. (a) The quad patch here contains a feature curve (shown in green). (b) A random face is selected in the patch (shown in yellow). (c) A poly-chord (shown in red) is generated (downward/upward direction shown with green/blue arrow). (d) The patch 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 patches (shown in green and orange) are generated by repeating the process from (a) to (f).](image)

Algorithm 1 The QRG process

while all smooth quadrilateral elements q[i, j] ∈ F^c ⊂ M are not conquered do
region R = ∅
// 1. poly-chord generation
select an un conquered q[i, j] ∈ Q
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 curves to generate poly-chords ch[a_k, b_j; j+1], k = 1, 2, · · · , W if there are some edges of feature curves 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 = UB then
extend ch[a_k, b_k; 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 patch R to patch
break
end if
set faces of ch[c, d; j + 1] to conquered
add faces of ch[c, d; j + 1] to patch R
if LB < c then
LB = c
end if
if UB > d then
UB = d
end if
end while
end while

that only one of them is selected as a poly-chord.

Shape-control: A shape-control parameter P_β is intro-
duced to control the shape of the bi-monotone patches
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: ζ = ||(d−c)−(b−a)||. In
short, it is relative difference of the lengths of two successive
poly-chords. The value $\zeta$ is constrained to be smaller than the user-defined $P_\beta$. While assigning higher values to the parameter $P_\beta$ 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 $\zeta$ value is smaller than the user-defined value $P_\beta$. Additionally, during the poly-chord extension process, we check the $\zeta$ value after each extension and stop extending the poly-chord once $\zeta$ exceeds the user-defined value $P_\beta$.

6. The Patch Configuration Improvement (PCI) Process

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

6.1. Detection of planar regions

After the QRG process, there may be planar patches that need to be separated from other patches as they can be simply represented by plane primitives. To detect and merge such patches, we first calculate and assign a normal vector for each planar quadrilateral element. Randomly selecting a quadrilateral face in a patch $Q$, 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 $T_{Fr}$, whose value is set as 0.001, the patch $Q$ is defined as planar. Furthermore, if two neighboring patches are planar and the difference between their normal vectors is smaller than $T_{Fr}$, they are merged.

6.2. Screening of patch boundaries

The patch 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 $T_{Fr}$ (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. 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 patches, 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 patch boundary screening processes, and constrain them to act as the patch boundaries.

6.3. Patch Growing

The patch layout after the QRG algorithm contains unnecessary boundaries that need to be removed to create large patches. The proposed growing strategy is different from those in which process starts from a face and regions are grown from it. As the patches observed after the QRG algorithm have good shapes and all feature curves reside on patch boundaries, the PCI process improves patch configuration by growing these existing patches. The growing method of this process is similar to that of the QRG process in that it is also quadvritional. Starting from a patch, growing is performed in a one-by-one manner in four directions and stops when the patch meets a surface feature. Readers should refer to Figure 8 for the illustration of the PCI algorithm.

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

The energy $E$ of a patch is calculated as: $E = \frac{E_F A}{N}$ where $F$ is the patch non-flatness, $A$ is the shape aspect ratio of the patch, and $N$ is the number of the quadrilaterals in the patch. The patch non-flatness $F$ is computed by averaging the non-flatness values of the faces in the patch. The shape aspect ratio is $A = L/W$, where $L$ is the length and $W$ is the width of the patch in the rectangular grid structure. This formula encourages flat patches, patches with more faces and compact (i.e., square) patches to grow first. As a result, flat patches in the model are covered first. Despite the simplicity of the proposed energy formula, it works well because patches at the beginning of the PCI process have good shapes and all feature curves reside on patch 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 patch boundaries, the algorithm checks whether the region growing process generates non-monotone patches. If the bi-monotonicity of a patch itself is violated, it is made bi-monotone by splitting its last-added poly-chord. A grown patch may also make its neighboring patches non-monotone. In such
The pseudo code in Algorithm 2 summarizes the PCI process.

Algorithm 2 The PCI process

```plaintext
sort G by energy E
while any patch of G is unprocessed do
    get unprocessed patch R from G
    while all side sk of R are processed do
        get all unprocessed side poly-chords ch
        get neighbor poly-chords nch attached to side poly-chords ch
        check nch whether it has some processed faces or have extraordinary vertices and revise nch
        calculate un-flatness S_k of neighbor poly-chords nch_k
        get the flattest neighbor poly-chord ch[m, n; j] ∈ 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, ..., 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 ζ ≥ P_β then
        break
    end if
    if non-monotone neighbor patches N_r exist then
        split non-monotone patches N_r into several bi-monotone patches
    end if
    erase faces of ch[c, d; j + 1] from its previous belonging region
    add faces of ch[c, d; j + 1] to patch R
    set patch R and all its faces as processed
    sort G by energy E
end while
```

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 patches G obtained from the QRG process. For each one, the energy E is computed. First, the algorithm sorts the patches according to their energy values and grows patch R of G with the lowest energy value. Patch 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. 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. 9 (b) shows bottle model after the QRG process (left) and the PCI process (right). Note how the patch enclosed by the blue rectangle has grown after the process.

7. Results and Discussion

The method proposed in this paper 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_ϕ) and the feature detection process (T_Γ, T_ν, T_ρ) are set. At the same time, the
shapes of the generated patches can be controlled by adjusting the shape control parameter \( P \).

\( T_e \) (feature elongation size) and \( T_p \) (feature neighbor) are adjusted depending on the resolution of the mesh. \( T_e \), angle threshold used in the EMG process, is an angle which is set to a value between 30 and 70 degrees. \( T_F \) is used to detect unflat regions which we set between 0.15 and 0.4. The shape control parameter \( P_\beta \) 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 patch. From our experience, our parameters are easy to tune.

For the rockerarm model shown in Fig. 10, the proposed algorithm generates 103 bi-monotone patches. Large flat parts are captured successfully, because the PCI process gives flat patches priority to grow first. The cylindrical and fillet geometry of the model is also retained by the bi-monotone patches.

Figure 11 shows a beetle model whose whole roof is a single patch. The proposed algorithm captures almost all symmetries of the model such that the front part shown in dark blue is represented by two patches whose common boundary (shown in green) is found during the feature detection process. For different values of the shape control parameter \( P_\beta \), differently shaped bi-monotone patches are generated. When smaller values are assigned to \( P_\beta \), the shape of patches becomes more quad-like, but their number increases.

Figure 12 shows a bottle model whose geometry is captured with large bi-monotone patches. In the rightmost image in (a), there are two small quadrilateral faces representing a patch. These are not included in the large patch next to them during the QRG and the PCI processes because the patch with them contains irregular vertices, which is not desirable. Even though using a larger value for \( P_\beta \) reduces the number of bi-monotone patches, it may result in the generation of patches that are not appropriate for B-spline surface fitting. For instance, the the light-orange patch 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. 13 can be represented by large bi-monotone patches as shown in (a). As setting smaller values for \( P_\beta \) produces smaller fragments with quad shapes (b, c), the smooth surfaces cannot be captured by a single patch. The motorcycle graph of Eppstein et al. [2] (d), the method of Myles et al. [24] (e), and the method of Bommes et al. [23] utilize quad patches to recover the model, but these patches 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 patches from it. Therefore, the quality of the bi-monotone patches 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 patches which may not be desirable.

It is possible to improve the segmentation results. A few tiny patches 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 patches will automatically disappear. However, this process should be performed carefully to not generate patches 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 (patches). Similarly, our algorithm detects the surface features and constrains them to be the boundaries of the generated (bi-monotone) patches. Therefore, the patches generated by our algorithm can capture designer’s intent to some degree.

Fitting of B-splines. To fit bi-monotone patches with B-splines, we use the global approximating surfaces [26]. First, a four-sided surface that contains all patch points is chosen. The distances between the surface points and the patch 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 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 patches 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.13 (a), 13.666 seconds for the rockerarm model, 1.85 seconds for the beetle model of Fig.11 (a) and 1.934 seconds for the bottle model of Fig.12 (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 fewer patches or more feature curves on the patch boundaries (as seen with fandisk model) reduces the processing time of the PCI process.

8. Conclusions and Future Works

The algorithm presented in this paper 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 segmentation of a quadrilateral mesh into four-sided patches, feature curves residing in-
Fig. 10. A rockerarm model with 9,413 faces. A total of 103 bi-mono-
tone patches are generated by the proposed algorithm ($T_\psi = 40,
T_\Gamma = 0.2, T_\varphi = 8, T_\rho = 4, P_\beta = 0.2$).

Fig. 11. A beetle model with 3,779 faces and bi-monotone patches
generated by the proposed algorithm ($T_\psi = 70, T_\Gamma = 0.4, T_\varphi = 7,
T_\rho = 2$). Changing the shape-control parameter $P_\beta$ affects the shape
of the model. Numbers of bi-monotone patches: (a) 45 ($P_\beta = 0.2$) (b) 51
($P_\beta = 0.0$) (c) 34 ($P_\beta = 0.6$). When smaller values are assigned
to $P_\beta$, the shape of the patches becomes more quad-like, but their
number increases.

Fig. 12. A bottle model with 3,117 faces and bi-monotone patches
generated by the proposed algorithm ($T_\psi = 70, T_\Gamma = 0.4, T_\varphi = 6,
T_\rho = 1$). Changing the shape-control parameter $P_\beta$ affects the shape
of the model. Numbers of bi-monotone patches: (a) 38 ($P_\beta = 0.2$)
(b) 40 ($P_\beta = 0.0$) (c) 35 ($P_\beta = 0.6$)

Fig. 13. A fandisk model with 13,181 faces and bi-monotone patches
generated by the proposed algorithm ($T_\psi = 30, T_\Gamma = 0.15, T_\varphi = 8,
T_\rho = 4$) Changing the shape-control parameter $P_\beta$ affects the shape
of the model. Numbers of bi-monotone patches: (a) 46 ($P_\beta = 0.6$) (b) 51
($P_\beta = 0.4$) (c) 59 ($P_\beta = 0.2$). Quad patches generated by other
algorithms: (d) motorcycle graph (55 patches) [2] (e) feature-aligned
T-meshes [24] (images taken from [24]).
Fig. 14. B-spline surfaces generated for: (a) the bi-monotone patches of the beetle model in Fig. 11 (a); (b) the quad patches of the beetle model after the EMG process; (c) the bi-monotone patches of the fandisk model in Fig. 13 (a); (d) the quad patches of the fandisk model after the EMG process. Surface models in (a) and (c) have larger patches and capture the surface features better.

side 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.

Our work can be extended in many ways. While generating bi-monotone patches, a number of surface-related criteria can be taken into account to produce better quality B-spline surfaces.

Acknowledgements

The authors would like to thank the RWTH Computer Graphics Group for making the OpenFlipper geometry processing and rendering framework available, and Dr. Leif Kobbelt for providing the quadrilateral mesh models tested in this study.

References