Computer Aided Geometric Design 48 (2016) 17–35

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

Converting a CAD model into a non-uniform subdivision surface

Jingjing Shen[a,*], Jiří Kosinka[a,b], Malcolm Sabin[c], Neil Dodgson[a,d]

[a] Computer Laboratory, University of Cambridge, 15 JJ Thomson Avenue, Cambridge CB3 0FD, United Kingdom
[b] Johann Bernoulli Institute, University of Groningen, Nijenborgh 9, 9747 AG, Groningen, The Netherlands
[c] Numerical Geometry Ltd., 19 John Amner Close, Ely, Cambridge CB6 1DT, United Kingdom
[d] Faculty of Engineering, Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand

ARTICLE INFO

Article history:
Received 12 April 2016
Received in revised form 14 July 2016
Accepted 19 July 2016
Available online 29 July 2016

Keywords:
CAD model
Trimmed NURBS
Non-uniform subdivision

ABSTRACT

CAD models generally consist of multiple NURBS patches, both trimmed and untrimmed. There is a long-standing challenge that trimmed NURBS patches cause unavoidable gaps in the model. We address this by converting multiple NURBS patches to a single untrimmed NURBS-compatible subdivision surface in a three stage process. First, for each patch, we generate in domain space a quadrangulation that follows boundary edges of the patch and respects the knot spacings along edges. Second, the control points of the corresponding subdivision patch are computed in model space. Third, we merge the subdivision patches across their common boundaries to create a single subdivision surface. The converted model is gap-free and can maintain inter-patch continuity up to C^2.

1. Introduction

NURBS are the standard representation for Computer-Aided Design (CAD). Because of their restriction to the regular grid structure, a CAD model typically consists of a collection of NURBS patches, some of which are trimmed. These surfaces are then stitched together with certain boundary management to represent a smooth shape with arbitrary topology. The trimming and stitching operations introduce unavoidable gaps (Fig. 1a) between adjacent patches (Sederberg et al., 2008). By contrast, subdivision, the de facto standard for computer animation (DeRose et al., 1998), is able to model a smooth shape of arbitrary topology using just a single control mesh, thus providing a gap-free representation. A key challenge of using subdivision in the CAD industry is that traditional subdivision methods (Catmull–Clark, Loop) are incompatible with NURBS, but recent advances (Cashman et al., 2009) have produced NURBS-compatible subdivision methods that do have the required expressive power.

Our method converts a CAD model into a single control mesh for a gap-free subdivision surface. We assume the model is given in a B-rep topology. Each B-rep face corresponds to a (trimmed) bi-cubic NURBS patch, and each B-rep edge corresponds to a (trimming) loop represented as a set of cubic NURBS curves.

Our target representation is the NURBS-compatible subdivision of Cashman et al. (2009), because it admits non-uniform knot spacings, which allows us to match exactly any untrimmed bi-cubic NURBS patch and patch edges represented by general cubic B-splines. Catmull–Clark subdivision is a special uniform case of this scheme.

[*] Corresponding author.
E-mail address: js2036@cam.ac.uk (J. Shen).

http://dx.doi.org/10.1016/j.cagd.2016.07.003
0167-8396/© 2016 Elsevier B.V. All rights reserved.

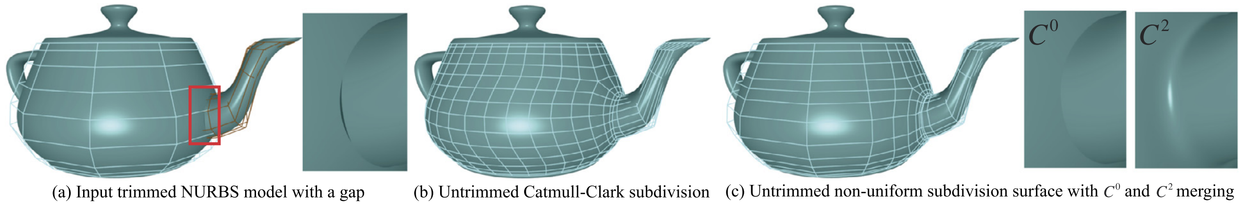
C^0
C^2
(a) Input trimmed NURBS model with a gap
(b) Untrimmed Catmull-Clark subdivision
(c) Untrimmed non-uniform subdivision surface with C^0 and C^2 merging

Fig. 1. (a) A model with trimmed NURBS patches. There is a gap between two of the patches owing to the unavoidable approximation in trimming. (b) Converting the NURBS patches to a single Catmull–Clark subdivision surface produces a dense control mesh. (c) Converting to a single non-uniform subdivision surface produces a less dense control mesh and a better approximation to the original NURBS surface than is possible with Catmull–Clark. In this case, the join between patches can be either a hard C^0 crease or a smooth C^2 join. The left half of the teapot body and all of the spout (other than the C^2 blend to the body) are preserved exactly.

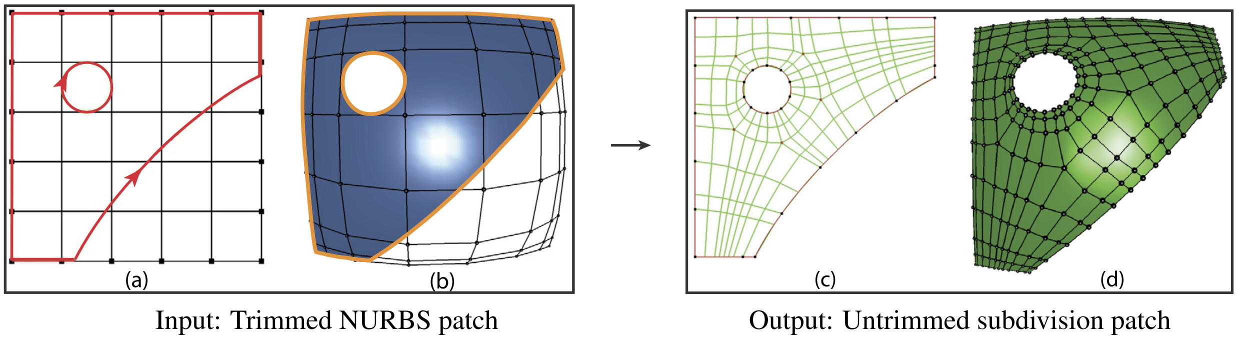
(a)
(b)
(c)
(d)
Input: Trimmed NURBS patch
Output: Untrimmed subdivision patch

Fig. 2. Converting a single trimmed NURBS patch to an untrimmed non-uniform subdivision patch. (a) *Domain space* showing knot lines (black) and oriented trimming curves (red). (b) *Model space* showing the rectangular control mesh and the trimming curves (orange) which form the boundary of the target subdivision surface. (c) The partitioning in domain space, giving a non-uniformly spaced quadrilateral mesh. (d) The final control mesh and non-uniform subdivision surface in model space. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)

The three stages and contributions of our method are:

1. Starting from a coarse quad layout in the trimmed domain of the NURBS patch, we construct a quadrilateral mesh topology with knot intervals associated with mesh edges (required by the non-uniform subdivision scheme) while keeping the input boundary curves unchanged (Section 3).
2. Using the mesh topology from Stage 1, we compute control point positions in that mesh (Section 4). We investigate how to compute limit stencils for the cubic, non-uniform Cashman scheme. In addition, to make smooth joins between patches easy, we apply Bézier edge conditions across the boundary (with convex corners only) and extend to the non-uniform case the constraints on control points for tangential continuity across the boundary (Shen et al., 2014). We extend Pixar sharp rules (DeRose et al., 1998) to non-uniform subdivision to deal with concave corners.
3. Given the meshes (created in Stage 2) for each patch in the model, we automatically merge them into a single subdivision mesh (Section 5). To ensure that this is possible, in Stage 1, for each shared edge we ensure that the same boundary curve is used and maintained on both sides and, in Stage 2, we use the preserved model-space boundary curves as the boundary of the target subdivision surface (in the limit case). This gives a gap-free join because the adjacent patches always share the same curve.

Before describing these, we cover background and related work.

2. Background and related work

Our input is a surface model comprising several trimmed bi-cubic NURBS patches with a B-rep topology. Fig. 3 illustrates our notation. NURBS patches $\mathcal{N}_1$ and $\mathcal{N}_2$ share the B-rep edge γ, whose geometry is given by the model-space curve γ^b, called the *b-curve*, represented as a cubic NURBS curve.

The trimming loops for the patches are specified in parameter space and consist of several individual cubic NURBS curves called *p-curves*, denoted by γ^p. In Fig. 3, γ^b corresponds to γ^p_- on $\mathcal{N}_1$ and to γ^p_+ on $\mathcal{N}_2$. If the trimming is introduced by intersecting two NURBS patches, this intersection is not rational, in general (Sederberg et al., 2008), and thus $\mathcal{N}_1(\gamma^p_-) \neq \gamma^b \neq \mathcal{N}_2(\gamma^p_+)$ and none of these curves is identical to the true intersection curve (shown in grey in Fig. 3). This is the source of the *unavoidable gaps*.

The whole set of edge curves associated with a shared γ are denoted as $\langle \gamma^b, \gamma^p_-, \gamma^p_+ \rangle$. We assume that all the three curves associated with any γ are given in the same parametrisation (Shen et al., 2014).

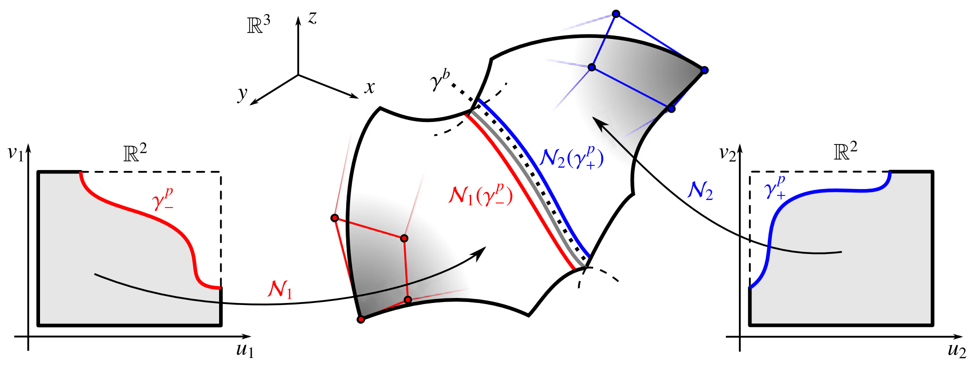

Fig. 3. The situation at the shared B-rep edge of two NURBS patches $\mathcal{N}_1$ and $\mathcal{N}_2$. Their true intersection curve is shown in grey. The embedding into model space of the B-rep edge γ is denoted γ^b (dotted) and called the b-curve. The trimming edges in parameter spaces, the p-curves, of the patches are denoted γ_-^p and γ_+^p.

As illustrated in Fig. 2a–b, patch trimming is done using oriented p-curves (red), which identify the region that are skipped during evaluation. The images (orange) of these p-curves on the surface either produce or approximate the desired boundary there.

Gaps are not acceptable in many applications (e.g., Computer-Aided Engineering, Finite Element Analysis) and so additional boundary management has to be enforced to give the desired inter-patch continuity.

Our aim is to replace the NURBS geometry with a subdivision surface that exactly contains the b-curves, thus rendering the entire configuration gap-free. The benefit of using Cashman's non-uniform subdivision (Cashman et al., 2009) is that untrimmed patches are kept exactly the same and so the approximation error is confined to trimmed patches and, in some cases, to smaller regions around the trimming curves. In addition, our subdivision mesh can be much coarser (Fig. 1) than is possible with a uniform subdivision mesh (Shen et al., 2014), which is an advantage for later processing.

The potential benefits of solving this problem are that it (1) produces gap-free models for the CAD/CAE industry, (2) enables the subdivision community to directly use all existing NURBS models, and (3) provides a conversion in which the user can edit the trimmed model directly without re-trimming. Because Catmull–Clark is a special case of our method, we are also able to provide Catmull–Clark models for those wanting to work with that uniform mechanism.

2.1. Methods to make trimmed NURBS gap-free

The seminal work in this area is undoubtedly that of Sederberg et al. (2008). Unfortunately their paper can leave the impression that the key to what they do is the T-junction mechanism. In fact T-junctions alone only address mesh density. To get the orientation right so that mesh lines run parallel near boundaries, it needs *extraordinary vertices* (EVs), which are present in their meshes. The major difference between their result meshes and ours is that their meshes have many EVs near the boundary. This is a limitation because the mesh quality near the boundary is crucial for numerical analyses. In contrast, our framework generates smooth alignment near the boundary, creates a relatively smaller number of EVs for the trimmed shape, and ensures that the EVs are not 'unnecessarily' close to the boundary.

Our work can be viewed as a significant generalisation of the work by Shen et al. (2014). They present a method to convert a *single* trimmed NURBS patch to a Catmull–Clark (uniform) subdivision surface. However, they do not deal with the practical case of multiple trimmed patches. Even though Shen et al. (2014) mention that a gap-free join can be achieved provided that the same boundary curve is used during the conversion of the adjacent patches, it is non-trivial to ensure that the same number of control points are used to uniformly re-approximate the shared boundary when the conversion of each patch is done independently. Also, owing to their use of *uniform* subdivision, their method re-approximates any non-uniform patches/edge curves, and creates meshes that can be unnecessarily dense. Furthermore, their method is restricted to trims that have only convex corners, while concave corners exist in common models.

In contrast, our new approach converts a group of bi-cubic NURBS patches (either trimmed or not) with a B-rep topology to a single untrimmed non-uniform subdivision surface and handles general trims with concave corners.

2.2. NURBS-compatible subdivision

Subdivision can handle arbitrary topology, while NURBS implementations have the flexibility of higher degrees and non-uniformity. To obtain the advantages of both representations, various non-uniform subdivision schemes have been presented for degree 2 and degree 3 (Müller et al., 2006, 2010; Sederberg et al., 1998, 2003). They extend the refinement of NURBS to arbitrary topology. NURBS-like knot intervals are assigned to the corresponding vertices (even degrees) or edges (odd degrees) of the subdivision mesh. Going beyond degree 3, Cashman et al. (2009) proposed a non-uniform subdivision scheme with arbitrarily high degree.

We use the cubic case of Cashman's subdivision scheme. It provides bounded curvature solutions at EVs. It requires that the control mesh contains only quadrilateral faces and each quadrilateral face has equal knot interval on opposite edges (as is the case with NURBS). Therefore, knot intervals are defined for whole strips of quadrilateral faces rather than for a single edge.

2.3. Quad layout

We construct the topology of the target non-uniform subdivision mesh and assign knot intervals to mesh edges in the domain space of a trimmed patch. This first requires a coarse boundary-aligned quad partition of the 2D domain (bounded by p-curves).

As shown in Shen et al. (2014), an intuitive way to create this quad layout is first to compute a smooth cross field with directional constraints along the boundary (Bommes et al., 2009) and then trace streamlines from the singularities of this field (Alliez et al., 2003; Palacios and Zhang, 2010). Recently, approaches based on the general frame field, an anisotropic and non-orthogonal extension of cross field, have been proposed (Diamanti et al., 2014; Jiang et al., 2015) and approaches for robust tracing have been introduced (Myles et al., 2014; Ray and Sokolov, 2014). In our work, we follow a similar procedure to Shen et al. (2014), but replace the cross field with the frame field proposed by Diamanti (2014) for better alignment along boundary with non-90° corners. See Appendix E for details.

We emphasise that our framework treats the process for a coarse quad layout as a black box. Any method that produces a reasonable coarse quad mesh would be acceptable here. Alternatives include pattern-based quad meshing methods (Peng et al., 2014; Takayama et al., 2014) and the state of the art in reliable quad meshing by Bommes et al. (2013) and Campen et al. (2012, 2014). Note that the applications of these methods focus on producing uniformly spaced quad meshes, and it is not straightforward to use them in the non-uniform setting. We leave to future work the challenge of extending such algorithms to the non-uniform case.

2.4. Surface fitting

Our Stage 2 requires a form of surface fitting. Methods for fitting B-spline surfaces and Catmull–Clark subdivision surfaces have both been well explored. Weiss et al. (2002) provide a detailed review for least squares B-spline surface fitting techniques. Algorithms include optimisation of parameterisation, efficient fitting with tight tolerance and smoothness functions (Floater, 2000; Deng and Lin, 2014). Halstead et al. (1993) introduced an efficient interpolation method for Catmull–Clark subdivision surface based on limit stencils. To improve the surface quality, they used a fairness term based on membrane and thin plate energy.

However, the interpolation method for non-uniform subdivision surfaces remains unexplored. The main challenge is to obtain the limit stencils of Cashman's scheme. Our new method is described in Section 4.

3. Topology construction of a non-uniform subdivision mesh

This section describes Stage 1 of our method: topology construction in the parameter space. This is done across patches with the constraint that adjacent patches have their shared edge curves preserved. The preserved b-curve will become the boundary of the target subdivision surface. For clarity, we explain this procedure in one trimmed patch.

The input is a single trimmed bi-cubic NURBS patch, with its bounding edges included in the B-rep structure. If γ is an open edge, its edge curves are $\langle \gamma^b, \gamma^p \rangle$, and if shared, then $\langle \gamma^b, \gamma^p_-, \gamma^p_+ \rangle$. These edge curves share the same parametrisation (Shen et al., 2014), meaning that each γ is associated with a unique knot vector.

The output from this procedure is a quad mesh $\mathcal{Q}$ in the trimmed domain, with knot intervals I_i assigned to mesh edges e_i. The boundary edges of $\mathcal{Q}$ constitute control polygon edges for γ_p. The image of the mesh point in $\mathcal{Q}$ on the NURBS patch will be used as the limit position of the corresponding control point later in our subdivision surface fitting procedure.

The process has four sub-stages (Fig. 4):

1. Create a coarse quad layout in the trimmed domain using the process outlined in Section 2.3 and detailed in Appendix E. The method that we have implemented produces good results, but alternative methods could be used, without changing the subsequent steps.
2. Allocate knot intervals to the mesh edges in the quad layout (Section 3.1).
3. Refine the layout in order to maintain the boundary curves (Section 3.2).
4. Smooth the partition to improve the quality of the result, giving the mesh $\mathcal{Q}$ (Section 3.3).

3.1. Quad layout with knot intervals

Every edge e in the quad layout must be associated with a *knot interval*. There are two conditions on knot interval values.

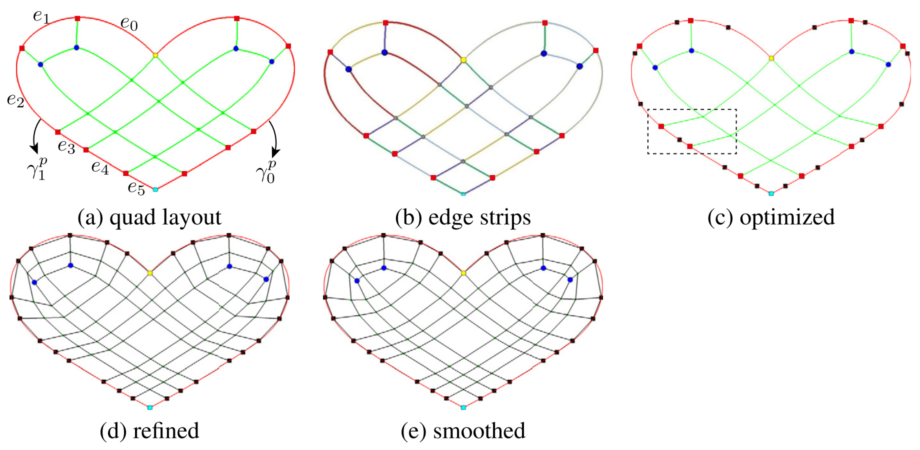

Fig. 4. *The topology construction.* (a) Initial quad layout with EV3 (an extraordinary vertex of valence 3) shown in blue dots, tracing rays shown in green segments and boundary partitioning nodes in red dots. (b) Edges across a strip of quads are shown in the same colour and they have the same knot interval. (c) After fixing the edge knot intervals (Section 3.1), the boundary partitioning nodes are moved to satisfy the ratios of the knot intervals. The black squares on the boundary in (c–e) indicate the knot spacings of the input boundary curves. (d) In order to preserve the curves, the quad layout is refined to include these spacings. Narrow intervals are snapped. (e) After 5 smoothing steps. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)

- The chosen non-uniform subdivision scheme (Cashman et al., 2009) requires that all the edges, e, that lie across any given strip of quads must have the same knot interval. For example, in Fig. 4b, edges in the same colour share the same knot interval.
- Even though we are restricted to preserve the B-rep edge curve, we still have the freedom of scaling its knot vector. The condition is that for a shared γ, its associated γ_-^p and γ_+^p are scaled with the same magnitude, so that $\langle \gamma^b, \gamma_-^p, \gamma_+^p \rangle$ still have the same parametrisation.
- The knot interval values must be positive.

Each boundary piece (separated by corners) is a p-curve γ^p that corresponds to a B-rep edge γ, see Fig. 4a. We begin the process by calculating appropriate knot intervals I_i for boundary partitioning edges e_i. The knot intervals of these edges (e.g. e_0 to e_5) indicate the scaled parametrisation of the curve (e.g., γ_1^p).

Note that the above conditions alone provide an under-determined system (if a feasible solution exists) because the total knot interval on each B-rep edge is free to scale. Thus, we choose to solve this problem as an optimisation problem, and the objective is to respect the initial partition as much as possible, which means to minimise the sliding of the boundary partitioning nodes along their p-curves, see red squares in Fig. 4a–c.

The knot intervals I_i, collected into $\mathbf{I}$, of boundary edges e_i are computed by solving a constrained linear programming problem. The inputs are the lengths, l_i, of the edges e_i in the layout. We use Π to denote the collection of adjacent edge pairs (e_i, e_j) on each boundary piece, and h_{ij} the length of the interior edge that meets the shared vertex of e_i and e_j. The objective function is formulated as

$$F(\mathbf{I}) = \sum_{(e_i, e_j) \in \Pi} w_{ij} \left| \frac{I_i}{l_i} - \frac{I_j}{l_j} \right|, \tag{1}$$

subject to three conditions:

1. for boundary partitioning edges, e_i and e_j lying across the same quad strip: $I_i = I_j$;
2. for edges e_k on a shared γ with fixed total knot interval $\hat{I}$: $\sum_k I_k = \hat{I}$;
3. $I_i > 0$.

The weight $w_{ij} = \sqrt{l_i^2 + l_j^2}/h_{ij}^2$ is ad hoc for two purposes: scaling the length will not affect the result and to prevent excessive pulling near boundary.

We solve the problem using the open source library lpsolve (Berkelaar et al., 2004). The size of the problem depends on the number of strips and shared B-rep edges in the coarse quad layouts. In our experiments, the number of variables and the number of constraints vary in the range of 10–80.

When grouping the conditions for multiple adjacent patches together for a single system in Condition 2, $\hat{I}$ is replaced with $\sum_m I_m$, where I_m denotes the knot intervals in the partition on the other side of its γ. Across multiple patches, the above system could be, in principle, over-constrained; see the discussion in Section 7. However, we have not encountered the over-constrained problem in our experiments on CAD models.

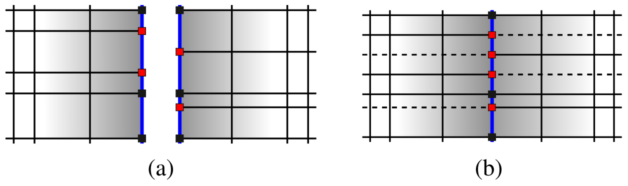

Fig. 5. (a) After partition, the adjacent patches have different set of knots along the shared boundary curve (blue). The original and the newly added knots are marked in black and red squares, respectively. (b) Our approach matches the knot spacings by propagating knots from patch to patch and across patches, see the dashed knot lines. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)

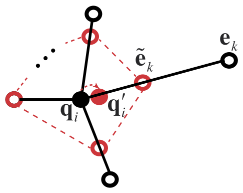

Fig. 6. Smoothing method illustration: $\mathbf{q}_i$ is updated to $\mathbf{q}_i'$.

After fixing the knot interval for boundary edges, all edges across the strip it belongs to share the same interval. The knot interval for a looping strip (i.e., a strip that contains no boundary edges) is then set proportionally to its average edge length, based on the fixed neighbouring edges along this strip.

3.2. *Refinement*

For each γ, the total knot interval allocated from the above process might be different from its original value. Therefore, we scale its original knot vector to this new total interval. Now we need to refine the quad layout so that:

- The edge curves of γ are preserved. Therefore, a new knot lines should be introduced for any original knot (black squares in Fig. 4c), associated with γ, that is not already a knot in the quad mesh.
- For a shared boundary γ, the refined parameterisations of its two p-curves are identical. When forming the quad layout, new knots may be introduced (red squares in Fig. 4a and Fig. 5a). Since different sets of new knots might be inserted in the p-curves on the two sides, we need to map these knots from patch to patch and across the patches. Here, patch means the domain-space partition of the patch. See the dashed lines in Fig. 5b.

The refinement is implemented as an iterative process: propagating across the boundary into the partition on the other side (Fig. 5), and within each partition, matching the knots (the spacings) on the start and end edge of each strip (Fig. 4d). On an all-quad layout with valid knot intervals, the matching and propagation will terminate (either by hitting an open boundary or by coming back to the start point). This process will take as many iterations as there are patches in the longest loop. Note that since the number of knots after the refinement phrase could grow significantly, a snapping phase (explained in the next paragraph) is important in order to produce a relatively sparser control mesh. After all knot intervals are mapped across the partitions, each refining knot line is added by interpolation along the edge strip it belongs to (black lines in Fig. 4d).

Snapping knots. Inserting new knots in the refinement step can lead to narrow intervals, which are undesirable because they introduce many unnecessary control points in the converted subdivision mesh. To avoid narrow intervals, we snap the knots when their intervals are smaller than a given threshold (set to 5% of the maximum knot interval in partitions in our examples). The input boundary curve is slightly modified if its original knot is moved. However, this small movement has only a small, bounded, effect on the shape of the curve (Imre and Hoffmann, 2001) and does not affect our ability to make a gap-free surface (Section 5). Fig. 4d shows the refined partition after snapping.

3.3. *Smoothing the quad partition*

The images of $\mathbf{q}_i$ (i.e., the points in the refined domain-space partition $\mathcal{Q}$) on the NURBS patch will be used as the limit positions in the later subdivision surface fitting procedure. In order to get a smooth distribution of the limit points, the partition is further smoothed based on the allocated knot spacings (Fig. 4d–e). We apply a simple smoothing step iteratively, which extends the uniform Laplacian smoothing algorithm (Fig. 6). For each point $\mathbf{q}_i$, find its edge-connected neighbours $\{\mathbf{e}_k\}, k = 1, ..., n$, with (connecting) edge knot intervals $\{I_k\}, k = 1, ..., n$, and then set

$$\mathbf{q'}_i = (1-\alpha)\,\mathbf{q}_i + \alpha\mathbf{C}\left(\tilde{\mathbf{e}}_1, .., \tilde{\mathbf{e}}_n\right), \tag{2}$$

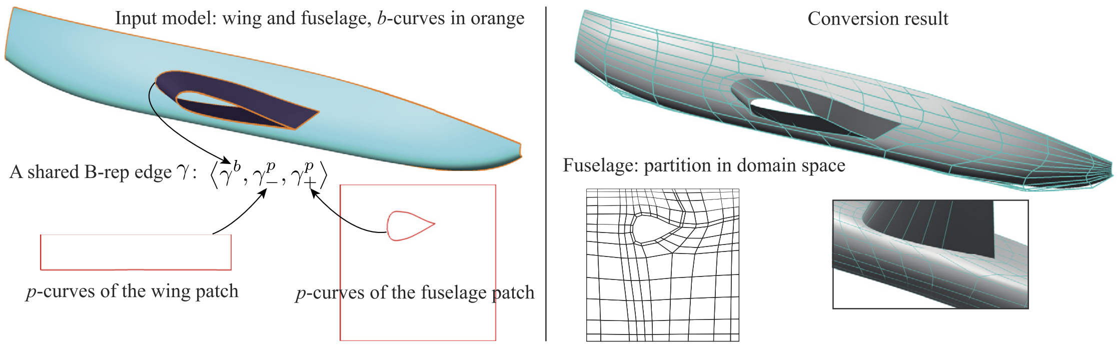
Input model: wing and fuselage, b-curves in orange
Conversion result
A shared B-rep edge γ: $\langle \gamma^b, \gamma^p_-, \gamma^p_+ \rangle$
Fuselage: partition in domain space
p-curves of the wing patch
p-curves of the fuselage patch

Fig. 7. Aircraft wing model. The inner trimming loop of the fuselage has a concave corner. Pixar sharp rules are applied on that shared edge during subdivision. The converted subdivision mesh and surface are shown on the right.

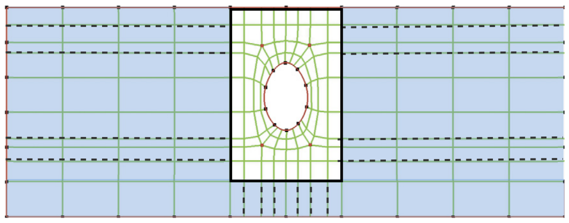

Fig. 8. Limiting the approximated area to the local neighbourhood of the trimming curve. This example is for the teapot body shown in Fig. 1c. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)

where

$$\tilde{\mathbf{e}}_k = \frac{I_{min}}{I_k}(\mathbf{e}_k - \mathbf{q}_i) + \mathbf{q}_i,$$

$\mathbf{C}\left(\tilde{\mathbf{e}}_1, .., \tilde{\mathbf{e}}_n\right)$ is the centroid of $\tilde{\mathbf{e}}_1, .., \tilde{\mathbf{e}}_n$, α is the average weight (set to 0.25 in our examples), and I_{min} is the minimum knot interval around $\mathbf{q}_i$. The partition becomes smooth quickly, typically after 3–5 steps (Fig. 4e).

3.4. *Limiting the area approximated (optional)*

By reconstructing the knot lines in the whole bounded domain, the approximation error of the conversion is on the entire trimmed surface. For patches where trimming is performed only in a small region of the patch, we can modify the method to limit the approximation process to a certain neighbourhood of the trimmed part. This is done by performing the quad partition in a local rectangular region. When a new knot is inserted into its local boundary (black box in Fig. 8, corresponding to the hole in the teapot body), it is mapped to the original boundary. The NURBS patch is then refined accordingly by knot insertion (dashed lines in Fig. 8). This leaves the blue shaded region unchanged and the limit surface of these parts matches the input NURBS exactly. The resulting subdivision mesh is shown in Fig. 1c with the error plot shown in Fig. 13b.

4. Computing control point locations

Halstead et al. (1993) and Shen et al. (2014) addressed the problem of computing control points for the Catmull–Clark case by using the subdivision *limit stencils* with certain boundary constraints (Shen et al., 2014). The challenge for the non-uniform case is that the limit stencils required are not available for control points inside the support regions of EVs. We first show how we get limit stencils for these points using nodal functions (Peters and Wu, 2006), and then explain the surface fitting with the boundary constraints.

Our quad partition algorithm generates a quad mesh, $\mathcal{Q}$, in domain space, which maps to a subdivision control mesh, $\mathcal{M}$, that defines a subdivision surface $\mathcal{S}$, in model space. $\mathcal{M}$ has similar mesh topology to $\mathcal{Q}$, depending on the end-conditions applied at the boundary, see Fig. 9 and Section 4.2. In domain space, we know the parametric locations of the vertices, $\mathbf{q}_i$, of $\mathcal{Q}$ and the knot intervals, I_j, of the edges e_j in $\mathcal{Q}$. Evaluating $\mathbf{q}_i$ on the input NURBS patch gives the corresponding points in model space, $\mathbf{l}_i$. We need to use these to calculate the locations of the vertices of the control points, $\mathbf{p}_i$, in $\mathcal{M}$.

4.1. Limit stencils

The limit stencil formulates the surface point $\mathbf{l}_i$ as a linear combination of the corresponding control point $\mathbf{p}_i$ and its one-ring neighbours $N^1(\mathbf{p}_i) = \{\mathbf{p}_j\}$, $j = 1, \ldots, J$. For non-uniform subdivision, the weights are decided by the knot intervals around $\mathbf{q}_i$ in $\mathcal{Q}$. Note that this is for our cubic case; higher degrees require larger neighbourhoods.

Points in regular regions: For points that are out of the support regions of EVs, since their local limit surfaces are just the tensor products of cubic B-splines, the limit stencil is obtained by applying the Cox–de Boor algorithm in each direction.

Points in one-ring neighbourhood of EVs: Because a selective knot insertion process is performed in the first few steps of Cashman's scheme (Cashman et al., 2009), e.g., large knot intervals are subdivided first, it is difficult to directly build a subdivision matrix for these steps.

One option would be to further refine the domain partition so that no selective knot insertions are required during subdivision. This, however, increases the density of the mesh. Instead, we use the approach of nodal functions (Peters and Wu, 2006), which are the limit, under subdivision, of associating the value one with a single control point and zero with all the others.

We use the following notations.

- The collection of EVs: $X = \{\mathbf{x}_i\}$, $i = 1, \ldots, K$, where K is the number of EVs.
- The union of the one-ring neighbours of all EVs: $R = \left\{\mathbf{p}^r_j \in N^1(X)\right\}$, $j = 1, \ldots, K_1$, referred to as *receivers*.
- The union of the two-ring neighbours of EVs and EV themselves: $C = \left\{\mathbf{p}^c_k \in \left\{N^2(X), X\right\}\right\}$, $j = 1, \ldots, K_2$, referred to as *contributors*.

The limit stencils for points in R are obtained after repeating the following process for each contributor $\mathbf{p}^c_k \in C$ (total number K_2):

(1) set $\mathbf{p}^c_k$ with value one and all others zero;
(2) perform the initial selective subdivision steps (plus one step of uniform subdivision if necessary) to isolate all receivers R from EVs, which updates $\mathbf{p}^r_j$ to $\bar{\mathbf{p}}^r_j$;
(3) because $\bar{\mathbf{p}}^r_j$ is in regular regions now, apply the Cox–de Boor algorithm to get its limit value, denoted as $w_{jk} = \left\langle \mathbf{p}^r_j, \mathbf{p}^c_k \right\rangle$, which is *the contribution of* $\mathbf{p}^c_k$ *to* $\mathbf{p}^r_j$ *in the limit case*.

The limit stencil of $\mathbf{p}^r_j$ is the collection of the contributions w_{jk} it received from the contributors in its one-ring neighbourhood, i.e., $\mathbf{p}^c_k \in \left\{N^1(\mathbf{p}^r_j), \mathbf{p}^r_j\right\}$.

Note that because the selective subdivision steps depend only on the knot interval configurations, the above process can be sped up by using a local identity matrix for the contributors as in Stam's method (Stam, 1998), instead of using a single unit value.

EVs: Because Cashman's scheme keeps a uniform knot configuration around EVs during subdivision, the limit stencils for the EVs can be obtained by first following the above steps (1) and (2), and then applying standard approaches for stationary schemes (Peters and Reif, 2008). Due to the bounded curvature solutions offered by Cashman's scheme, the limit stencils for the cubic case are different from those of Catmull–Clark subdivision. In our case, we need limit stencils only for degree 3 and valences 3 and 5, which are detailed in Appendix D.

4.2. Boundary

The limit stencils described above apply only on interior points. The boundary control points and the points next to the boundary are constrained differently depending on the end-conditions applied at the boundary.

End-conditions at the boundary. At the boundary, both Sederberg et al. (2008) and Shen et al. (2014) used *Bézier edge conditions*, i.e., multiple knot lines (with full multiplicity) running along the boundary. It has the advantages that (a) the boundary curves of the surface are defined by the boundary control points, and (b) the merging of adjacent patches with higher continuity can be easily achieved by knot removal, i.e., removing these multiple knot lines along the shared boundary.

However, while it works well for situations where the boundary has convex corners only, it causes problem when the boundary of a patch has a concave corner (the examples in Fig. 7 and Fig. 16). Applying Bézier edge conditions here would cause multiple knot lines running into the interior of the patch, potentially causing loss of derivative continuity due to numerical rounding or subsequent editing. We therefore apply a version of Pixar sharp edge rules (DeRose et al., 1998) generalised to NURBS-compatible subdivision at edges ending in concave corners, see Appendices A and B for details.

As illustrated in Fig. 9, $\mathcal{M}$ has the same mesh connectivity as $\mathcal{Q}$, with the exception that along any boundary where Bézier edge conditions are applied, there is one extra layer of control points just inside the boundary in $\mathcal{M}$ because of the existence of multiple knot lines. This extra layer does not change the overall topology of the output manifold.

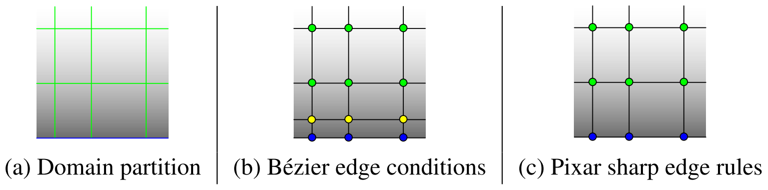

Fig. 9. Control mesh topology with two types of edge conditions. (a) The knot lines (green) in domain partition with boundary marked in blue. (b) Control mesh with Bézier edge conditions across the boundary. The boundary control points are shown in blue dots. One extra layer of control points, i.e., the yellow ones, are required because of the multiple knot lines along the boundary implied by Bézier edge conditions. (c) Control mesh with Pixar sharp edge rules across the boundary. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

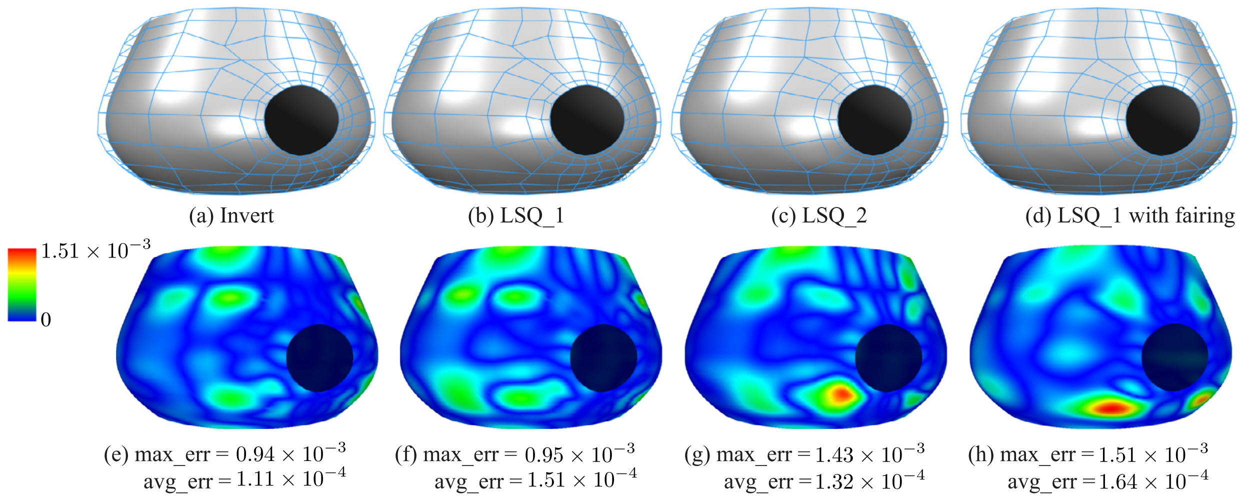
(a) Invert
(b) LSQ_1
(c) LSQ_2
(d) LSQ_1 with fairing
1.51 × 10−3
0
(e) max_err = 0.94 × 10−3 avg_err = 1.11 × 10−4
(f) max_err = 0.95 × 10−3 avg_err = 1.51 × 10−4
(g) max_err = 1.43 × 10−3 avg_err = 1.32 × 10−4
(h) max_err = 1.51 × 10−3 avg_err = 1.64 × 10−4

Fig. 10. The subdivision meshes calculated by (a) Invert: directly inverting the square linear system of Eq. (3), (b) LSQ_1: least squares fitting with 2×2 samples per face (of $\mathcal{Q}_{reg}$) and (c) LSQ_2: 20×20 samples per face, and (d) least squares fitting with 2×2 samples per face and with fairing. Their error plots are shown in (e–f) with the same scale, measured by sampling the distance of the output subdivision limit surface to the input NURBS surface (with number of samples over $30K$ and distance values scaled by the bounding box diagonal). This trimmed patch is remeshed and approximated within the entire domain. These result control meshes have the same topology: #V = 235, #F = 200.

Boundary constraints. The control points on the boundary are fixed as the points of the control polygons of the *b*-curves (after inserting new knots introduced during partitioning). *The control points in the first row inside the boundary* are fixed according to the first derivative information. We extended the boundary conditions, derived by Shen et al. (2014, Section 5.3), to support non-uniform knot intervals and generalised Pixar creases, see Appendix B for details.

4.3. Interior control points

We first tried to compute the rest of the interior control points from a square linear system built from the limit stencils of the subdivision scheme:

$$\mathbf{L} = \mathbf{M}\mathbf{P}, \tag{3}$$

where $\mathbf{L}$ and $\mathbf{P}$ are the vectors of surface samples $\mathbf{l}_i$ and control points $\mathbf{p}_i$, respectively. The ith row in $\mathbf{M}$ is the limit stencil associated with $\mathbf{p}_i$.

However, the control net computed by directly inverting the square linear system in Eq. (3) tends to be less fair than we would like (see Fig. 10a).

As an option to improve the smoothness of the final control mesh, we provide a least squares method with more samples and a fairing term (Floater, 2000; Halstead et al., 1993). We use $\mathcal{Q}_{reg}$ to denote the regular region that contains all faces of $\mathcal{Q}$ that are not incident with an EV. In our case, the local target subdivision surface corresponding to a quadrilateral face f_{reg} in $\mathcal{Q}_{reg}$ can be viewed as a small NURBS patch. Therefore, we can write the thin-plate energy on these local subdivision surfaces in terms of their surrounding control points.

Our increased set of samples $\mathbf{S}$ consists of all vertices $\mathbf{q}_i$ of the quad mesh, $\mathcal{Q}$, and additional $n \times n$ interior points, uniformly spread via bilinear interpolation, in each (quadrilateral) face f_{reg}. Our objective function is

$$E(\mathbf{P}) = \sum_{\hat{\mathbf{q}}_k \in \mathbf{S}} \left\| \mathcal{S}(\hat{\mathbf{q}}_k) - \mathcal{N}(\hat{\mathbf{q}}_k) \right\|^2 + \lambda \sum_{\mathcal{X} \in \mathcal{Q}_{reg}} \varepsilon(\mathcal{X}) \tag{4}$$

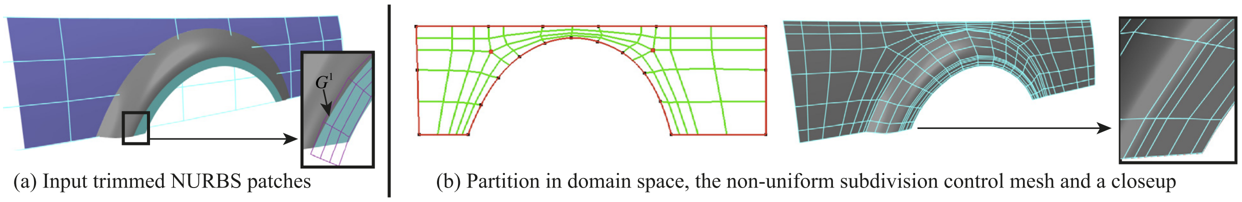
G^1
(a) Input trimmed NURBS patches
(b) Partition in domain space, the non-uniform subdivision control mesh and a closeup

Fig. 11. Wheel arch. The input model has two trimmed patches and an untrimmed blend surface in-between. The converted subdivision surface achieves (approximate) G^1 merging in the lower part.

$$\text{with}\quad \varepsilon(\mathcal{X}) = \int_0^{I_h}\int_0^{I_v} \|\mathcal{X}_{uu}\|^2 + 2\|\mathcal{X}_{uv}\|^2 + \|\mathcal{X}_{vv}\|^2 \, \mathrm{d}u\mathrm{d}v, \tag{5}$$

where $\mathcal{N}$ is the input NURBS patch being approximated, $\varepsilon(\cdot)$ is the standard thin-plate energy of a NURBS surface, $\mathcal{X}$ is a local part of the target subdivision surface $\mathcal{S}$, whose domain corresponds to a face f_{reg} in the regular region $\mathcal{Q}_{reg}$; I_h, I_v are the knot intervals associated with f_{reg} in each direction. All knot intervals are kept fixed, only control points $\mathbf{p}_i$ are solved for by minimising $E(\mathbf{P})$. The λ value is chosen as suggested in Floater (2000). The values of $\mathcal{S}(\hat{\mathbf{q}}_k)$ are computed using either limit stencils (if it corresponds to a control point) or the Cox–de Boor algorithm (if it corresponds to a sample point in f_{reg}), as explained in Section 4.1.

Our fairing term sums up the integration over the regular region of the subdivision surface, $\mathcal{S}|_{\mathcal{Q}_{reg}}$. The derivatives of the local subdivision surfaces are measured exactly (de Boor, 1972) and the integrations are done via *Gaussian Quadrature* (Vermeulen et al., 1992; Floater, 2000).

Fig. 10 compares the fitting results of a trimmed teapot body patch using four options. The resulting control mesh using least squares with 2×2 samples per face and fairing is fairer. The second row shows the corresponding error plots evaluated as described in Section 6.2. There is a tradeoff between the approximation error and the smoothness of the control mesh. As shown in (d) and (h), least squares fitting with fairing (Eq. 4) results in smoother control mesh but increases the approximation error.

5. Merging subdivision patches

We now merge the subdivision meshes computed in the surface fitting step. We consider joins along common edges and what further needs to be done at corners where more than two patches meet.

5.1. Two patches meeting at a common edge γ

There are three ways to merge two patches at a common edge.

C^0 **join (gap-free):** This is straightforward and is sufficient when the edge γ is a sharp one. Since the same model space curve is seen from both sides, the refined parameterisations are identical on both sides (Section 3.1), and both subdivision patches use this curve as boundary (Section 4), our process automatically gives a C^0 join.

G^1 **join with an untrimmed blend:** The tangent planes along the adjoining boundary on the blend side are used to compute the control points in the first row next to the boundary on the trimmed patch (see Appendix C for details). The result is G^1 at those shared control points. This is also the typical way that NURBS modelling systems (e.g. Rhinoceros, 2014) approximate G^1. The lower part of the wheel arch (Fig. 11) achieves this type of G^1 merging between the trimmed patch and the blend.

C^1 **and** C^2 **join:** If the adjoining boundary curve is a closed curve or the end corners of this curve in the two patches add to 180°, C^1 can be achieved by setting the control points in the first row either side of the boundary to be collinear with the respective points on the boundary, with ratio decided by the first non-zero knot intervals away from the boundary on the two sides. Furthermore, C^2 can be achieved by knot removal. Here, knot removal refers to reduction of the multiplicity of the knot lines introduced by Bézier edge conditions at the boundary.

We used the knot removal method presented in Tiller (1992) to reduce the multiplicity of the knots, and in cases that the required knot removal fails, i.e., the knot is not 'exactly' removable, the approximation using Lyche's L^2 norm (Lyche and Mørken, 1987a, 1987b) is applied. Alternative approximation methods can be found in Eck and Hadenfeld (1995).

This process for C^1 and C^2 might change the model space edge curve. However, here we regard the maintenance of smoothness as more important than exact fitting of the edge curve (which does not match the desired intersection curve in the first place). The C^1 merging in the sandal sole (Fig. 12) and C^2 merging in the teapot (Fig. 1c) were obtained in this way.

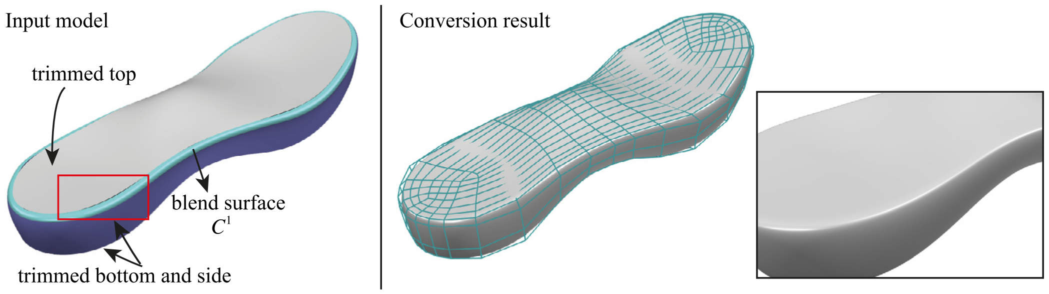

Fig. 12. Sandal sole with 4 patches. The result subdivision surface achieves the desired C^1 continuity between the top trimmed patch and the blend patch. Note the gaps between the input patches (highlighted in the red box). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

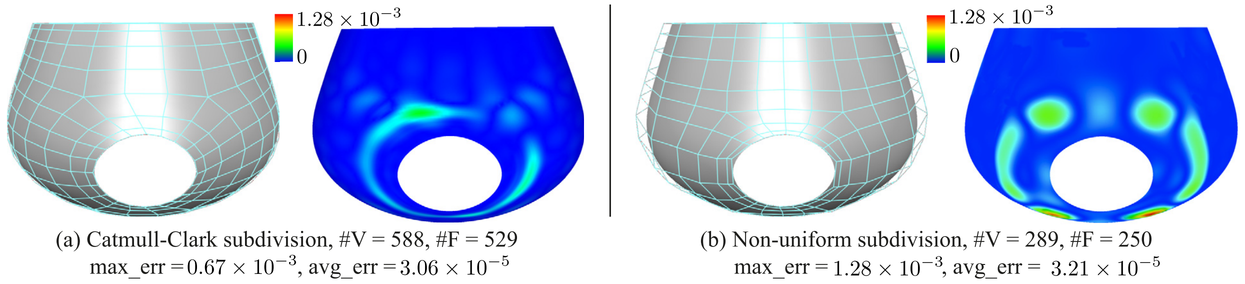
1.28×10^{-3}
0
(a) Catmull-Clark subdivision, #V = 588, #F = 529
max_err = 0.67×10^{-3}, avg_err = 3.06×10^{-5}
1.28×10^{-3}
0
(b) Non-uniform subdivision, #V = 289, #F = 250
max_err = 1.28×10^{-3}, avg_err = 3.21×10^{-5}

Fig. 13. Error plots of the conversion results using uniform and non-uniform subdivision schemes (the teapot body parts shown in Fig. 1b and c). In the non-uniform case, the input boundary curves (including the trimming curves) are preserved. In the Catmull–Clark subdivision case, these boundary curves have to be re-approximated using uniform cubic B-splines. The result in (a) is generated with the approximation error of these curves less than 1×10^{-4}.

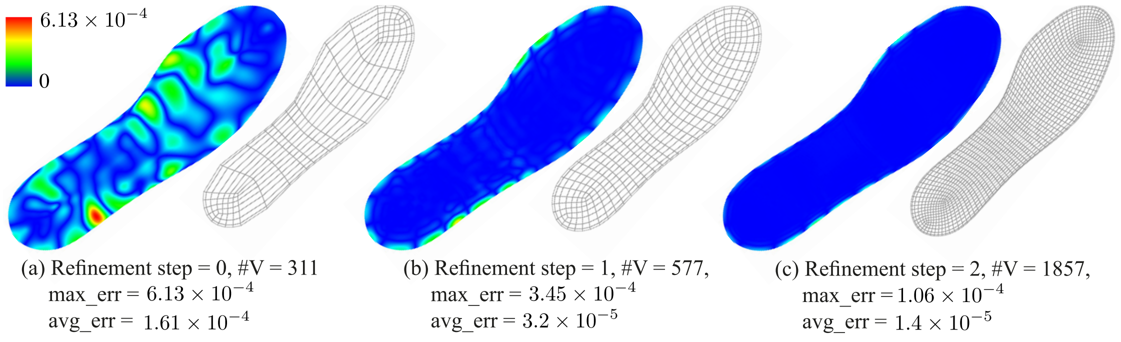
6.13×10^{-4}
0
(a) Refinement step = 0, #V = 311
max_err = 6.13×10^{-4}
avg_err = 1.61×10^{-4}
(b) Refinement step = 1, #V = 577,
max_err = 3.45×10^{-4}
avg_err = 3.2×10^{-5}
(c) Refinement step = 2, #V = 1857,
max_err = 1.06×10^{-4}
avg_err = 1.4×10^{-5}

Fig. 14. Error plots of the conversion results of sandal sole (the top patch). #V is the number of control points used in the result subdivision mesh, whose topology corresponds to the quad partition shown on the right. The approximation error drops when increasing #V by extra refinement steps in the topology construction step.

5.2. More than two patches at a common corner

If all edges are sharp then there are no issues, even if a concave corner is present in one (or more) of the faces (see the highlights on the left in Fig. 16). If all edges are smooth, then the corner may become an EV when the multiplicity of the knots is removed, and can be handled as such when computing control point locations (Section 4). If one or more non-smooth edges come into a corner which is essentially a flat one, then methods such as those of Kosinka et al. (2014b) can be used.

6. Results and discussion

We have processed a number of examples to illustrate features of our method.

6.1. Smooth and sharp joins

We show examples with smooth joins in Figs. 12 and 17. In Fig. 12, in the input sandal sole model, the trimmed top patch and the side patch have a C^1 blend surface in-between. The converted non-uniform subdivision surface has C^1 joins

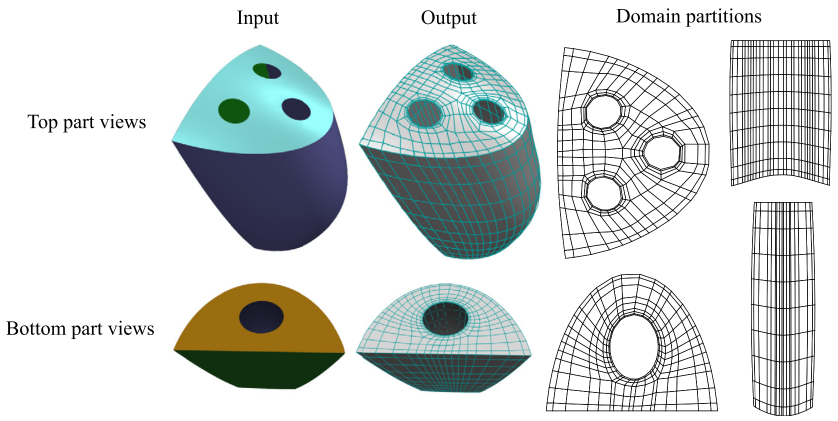

Fig. 15. Pepper shaker. Input model and converted subdivision mesh and surface. Bézier edge conditions are used along all sharp edges.

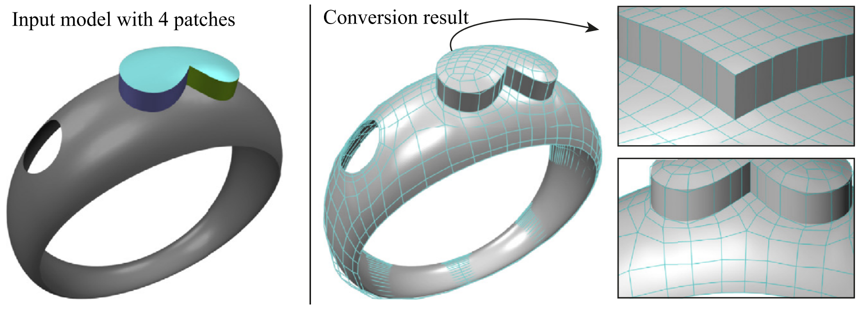

Fig. 16. Ring. Pixar sharp rules are applied on the sharp edges of the extrusion during subdivision (close-ups on the right). The partition of the top part is shown in Fig. 4.

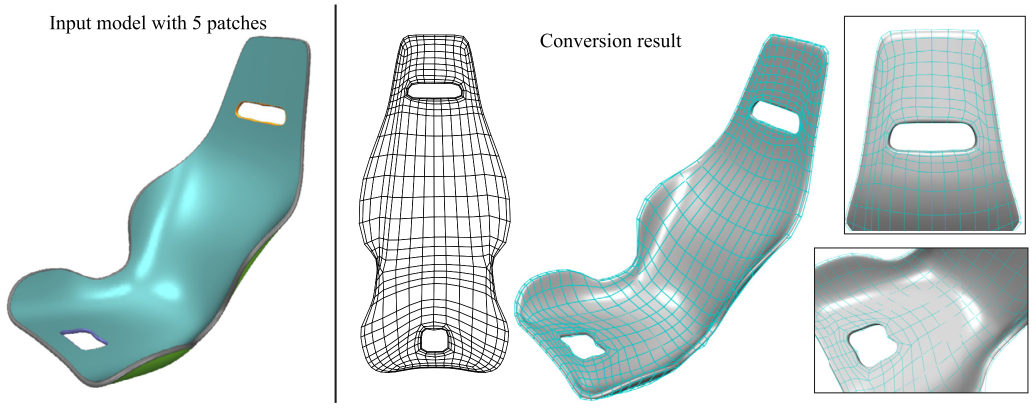

Fig. 17. Car seat. The converted subdivision mesh and surface after C^2 merging. The domain partition of the patch on the top is also shown.

along the two sides of the blend surface. In Fig. 17, the car seat, we show smooth fitting of five separate patches. This model used knot removal to achieve exact C^2 continuity.

The pepper shaker example shown in Fig. 15 illustrates gap-free sharp join of 4 patches using Bézier edge conditions. In Fig. 7, the aircraft wing/fuselage junction, and in Fig. 16, the ring with sharp edges along the extrusion, we show the use of Pixar sharp rules to handle the concave corner.

6.2. Approximation error

We evaluate the approximation error by sampling the distance of the output subdivision limit surface to the NURBS surfaces of the input model, with distance values scaled by the bounding box diagonal of the input model. More specifically, we subdivide the output subdivision control mesh sufficient times to generate over $30K$ vertices in the final mesh, compute

Table 1
Timings. The number of NURBS patches in the input model, the number of control points (#V) and faces (#F) of the result subdivision control meshes, and the computation time involved in the surface fitting stage. The teapot example is the result shown in Figs. 1c, where only local region around the trimming curve is re-approximated, which involves 88 control points and 96 faces.

Model	#Patch	#EV	#V	#F	Time (s)
Teapot (C^0 merge)	2	4	451	412	0.319
Wheel arch	3	2	265	200	0.271
Aircraft Wing	2	3	207	174	0.523
Sandal sole	4	8	550	548	2.614
Pepper shaker	4	12	1019	987	4.223
Ring	4	16	762	718	2.874
Car seat	5	32	1122	1124	6.345

the limit positions of these subdivided vertices, and then compute the minimum distance of these limit points to the input NURBS surfaces.

As shown in Fig. 14, the approximation error can be reduced by further refinement during the topology construction step. Following Cashman's method, we start with the quad partition generated in Section 3.1. Our first step of refinement inserts knot lines until no knot interval is larger than twice the minimum knot interval (Fig. 14a–b). The following steps insert knot lines in the middle of each knot interval (Fig. 14b–c).

6.3. Comparison with Catmull–Clark subdivision

In Fig. 1, we compare our teapot result for non-uniform subdivision against the result achieved using Catmull–Clark subdivision. The non-uniform subdivision mesh is much sparser than the Catmull–Clark subdivision mesh. In Fig. 13, we plot the approximation error of the teapot body using Catmull–Clark subdivision surface (Fig. 1b) and the non-uniform subdivision surface (Fig. 1c). This Catmull–Clark subdivision result is produced using the conversion method by Shen et al. (2014). A refined and smoothed uniform partition in the domain space is obtained with enough number of boundary control points so that the edge curves are re-approximated within tolerance 10^{-4} (using uniform knot intervals). Note that the conversion to Catmull–Clark subdivision leads to re-approximation of the untrimmed patches if its original boundary curves are changed. Although a coarser uniform partition could be made by adding appropriate constraints (Peng et al., 2014), it is still non-trivial to avoid re-approximation of the untrimmed patches.

In contrast, the conversion to Cashman's non-uniform subdivision approximates only the trimmed patches and keeps the untrimmed patches unchanged. Note that a patch trimmed along an iso-parameter line can be viewed as an untrimmed patch after knot insertion and splitting of the control mesh. The teapot spout (Fig. 1), the blend surface in the wheel arch (Fig. 11), and the wing in the aircraft (Fig. 7) belong to this case and are all matched exactly.

6.4. Timing

The control point computation is a least square fitting problem with terms built from the limit stencils (for the control points), bi-cubic B-splines evaluation (for extra samples), and thin-plate energy evaluation.

In Table 1, we list the number of control points and faces of the result subdivision control meshes, and the computation time involved in the surface fitting stage (including the time spent on building the matrix) for our examples. The timings were measured on a desktop PC equipped with Intel® Core™ i5-4570 CPU @ 3.20 GHz, 15.6 GB memory.

7. Limitations and future work

There are two cases that our method cannot directly handle. (1) Models that have a degenerate patch with a row of control points collapsed to one point, e.g., one quarter of a hemisphere. Pre-processing is required to convert the triangular patch to a non-degenerate rectangle patch which is half trimmed. (2) Models that require a mix of accurate continuity at the joined corner. In such cases, methods such as those of Kosinka et al. (2014a) need to be used.

The constrained linear programming problem in Section 3.1 can be unsolvable. The second spiralling configuration in Appendix E (Fig. 24, right) is an example of such case. One possible treatment is to introduce more EV pairs in the quad strips with incompatible knot intervals (Takayama et al., 2014; Myles et al., 2014).

In our method, most computations are done in the domains of the patches, which are, unlike in typical quad-meshing scenarios, readily available in CAD model definitions. This has advantages such as flexible control over the parametrisation of the non-uniform subdivision surface, the mesh density and inter-patch connectivity. On the other hand, this means that we are ignoring the Jacobian of the mapping to the 3D surface, which can have a detailed effect on the smoothing processes. It will be interesting to explore methods that are based on the combination of the information from both the domain and model space.

Finally, the output NURBS-compatible subdivision surface has the possibility to be converted to a collection of *untrimmed* NURBS patches (Cashman, 2010, Section 6.2), with extraordinary regions represented as a nested series of spline rings and finite fillings near the EVs. This means that our method can be generalised to support the conversion of a collection of

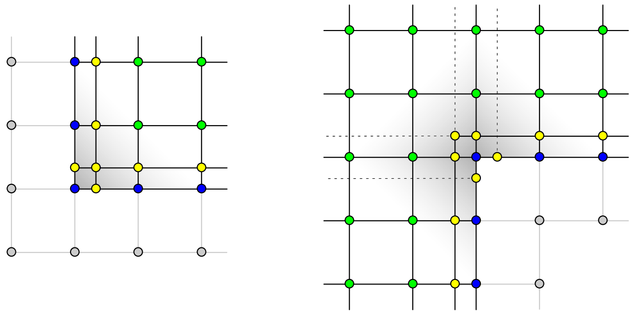

Fig. 18. Left: Convex corner. Right: Concave corner. Yellow points are required for enforcing Bézier edge conditions; grey 'ghost' points are required for the alternative edge conditions. Blue and green points are common to both ways of defining edge conditions. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)

trimmed NURBS patches to a (necessarily larger) collection of untrimmed NURBS patches, with the principal advantage that the collection of untrimmed patches will be watertight.

8. Conclusion

We present a novel framework to convert a trimmed NURBS model, which consists of several patches with inter-patch gaps, to a single gap-free non-uniform subdivision surface. The connectivity of the target subdivision base mesh is computed via quad partition in domain space. The control point positions are then obtained by solving a fitting problem based on the limit stencils of the subdivision scheme and the boundary conditions. We further introduce the Pixar sharp edge rules to the non-uniform subdivision to handle concave corners in trimming. The inter-patch merging (C^0 or G^1) is automatic. For some cases, C^1 and C^2 can be achieved. The approximation error is controllable via further refinements in the connectivity construction. Although this paper deals with cubic degree, the framework can be extended to support higher degrees, using Cashman's subdivision scheme.

Appendix A. Concave corners

To better reveal the problem with concave corners, first consider the simpler case of a *convex* corner (Fig. 18, left). When using Bézier edge conditions (i.e., knots of multiplicity 4 along the two edges), the surface is controlled by the blue, yellow and green control points. Alternatively, we can use only single knots along the edges, in which case the yellow points are not in the control mesh and are 'replaced' by the points in grey, the so-called 'ghost points' (Schweitzer, 1996; Lacewell and Burley, 2007). Blue points change position, but remain logically at the same place in the control mesh. The change from Bézier edge conditions to ghost points is facilitated by a set of linear conditions on the control points (we do not list them here as they are not needed in our system).

The important observation for the convex corner case is that the number of yellow points and the number of grey points is locally the same and thus both alternatives are able to represent the same underlying surface.

The situation at *concave* corners is more complicated (Fig. 18, right). A simple count reveals that there are locally 6 more yellow points than grey (ghost) points. This immediately reveals that it is impossible to use ghost points at concave corners and achieve full modelling freedom available at convex corners.

An alternative is to use multiple knots along boundary edges but, in the case of concave corners, these will propagate into the surface (indicated by the dashed lines in Fig. 18, right). This effectively splits the surface into several patches and causes potential further propagation into neighbouring patches. This is undesirable. A T-junction mechanism could be an alternative mechanism here. However, T-junctions are not currently part of Cashman's NURBS-compatible subdivision (Cashman et al., 2009) and adding them is a significant research project in its own right.

Instead, we chose to follow a different approach, that of sharp edges based on Pixar creases (DeRose et al., 1998), described in the next section, which solves the problem of concave corners.

Appendix B. Sharp edges

We need to be able to handle sharp edges in configurations such as at concave corners described above. We considered several alternatives including those proposed by Sederberg et al. (1998), Müller et al. (2006, 2010) and Kosinka et al. (2014b). We opted for a modification of Cashman's (2010) framework, which we based on a generalisation of both Pixar rules (DeRose et al., 1998) and ghost points (Kosinka et al., 2014a) to the non-uniform setting required in our method.

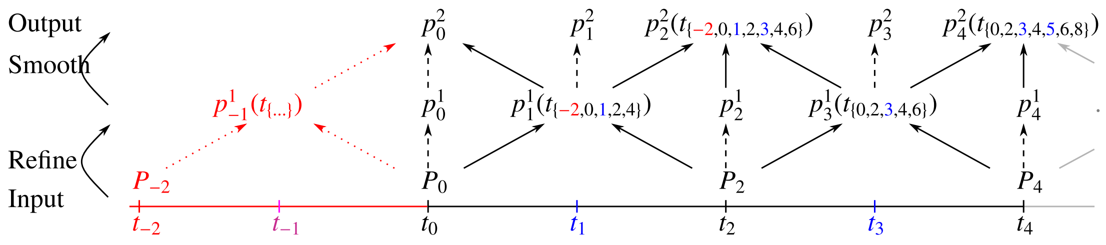

Fig. 19. An example in the cubic case. P_0 is the endpoint of the curve. Red ink indicates information that would be available in the smooth case but is missing in the sharp case. The input knot vector is given by the even knots. The odd knots are inserted, which corresponds to a subdivision step consisting of a refine stage and one smoothing stage. Note that P_0 keeps its position through all stages and that the rules for computing p_1^1 need to be modified as t_{-2} and P_{-2} are not available. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

B.1. The univariate case

Consider curves first; the surface case is then obtained by extending the ideas to the bivariate setting. Cashman's (2010) framework is based on a non-uniform refine-and-smooth algorithm (Cashman et al., 2009), which we need to extend to cover sharp edges. In the cubic case, there is one refine stage and one smoothing stage.

Fig. 19 shows an example. For clarity, instead of using notation based on blossoms (as in Cashman, 2010), we show *all* dependencies of points on knots, e.g. $p_3(t_{\{0,2,3,4,6\}})$ means that p_3 depends on knots t_0, t_2, t_3, t_4, t_6. In the example in Fig. 19, we see that p_3 is computed as a weighted average of P_2 and P_4. The weights are given by standard blossoming (Cashman, 2010, Section 3). However, both P_{-2} and t_{-2} are missing at a sharp corner (edge) represented by P_0 and we need to find appropriate replacements.

We generalise to the non-uniform setting an approach based on ghost points from the uniform setting (Kosinka et al., 2014c). Because P_{-2} and t_{-2} are not available (they are 'beyond' the endpoint P_0), we treat them as a ghost point and a ghost knot, respectively, which we are free to assign to achieve desirable end-conditions.

First, we set P_{-2} so that the resulting spline interpolates P_0. Using the limit stencil of P_0, denoted $\mathcal{L}(P_0)$, or equivalently the point on the spline curve $c(t)$ corresponding to t_0, we compute the position of P_{-2} from $P_0 = \mathcal{L}(P_0) = c(t_0)$. Omitting the lengthy but straightforward details (which can be obtained using blossoming), this leads to the (one-sided) end-derivatives:

$$\begin{aligned} c'(t_0) &= 3\frac{t_0-t_{-2}}{(t_2-t_0)(t_4-t_{-2})}(P_2-P_0), \\ c''(t_0) &= 6\frac{2t_0-t_{-2}-t_2}{(t_2-t_0)^2(t_4-t_{-2})}(P_2-P_0). \end{aligned} \tag{6}$$

The ghost knot t_{-2} remains a free parameter and can be used to control (the magnitude of) $c'(t_0)$. In principle, t_{-2} could be used as a shape parameter, but we fix it instead to achieve compatibility with the uniform case and to ensure that $c'(t_0)$ depends only on t_0 and t_2, but not t_4.

To this end, we have to ensure that $t_0 - t_{-2} = \alpha(t_4 - t_{-2})$ for some real α. Solving for t_{-2} gives $t_{-2} = (t_0 - \alpha t_4)/(1-\alpha)$; $\alpha \neq 1$. Choosing $\alpha = 1/3$ maintains consistency with the uniform setting and corresponds to $t_{-2} = (3t_0 - t_4)/2$, which yields

$$\begin{aligned} c'(t_0) &= \frac{1}{t_2-t_0}(P_2-P_0), \\ c''(t_0) &= 2\frac{2t_2-t_0-t_4}{(t_2-t_0)^2(t_4-t_0)}(P_2-P_0). \end{aligned} \tag{7}$$

Finally, observe that the position of the ghost point P_{-2} is not needed as P_0 simply keeps its position during the two stages. Consequently, only $t_{-2} = (3t_0 - t_4)/2$ needs to be used to compute p_1^1 and p_2^2 via blossoming. This is simple to implement.

We now extend this to the bivariate setting.

B.2. The bivariate case

The bivariate case (the edge conditions) uses the tensor-product of the end-conditions in the univariate case (across the boundary) with the normal ones (along the boundary). It is beyond the scope of this document to cover all the *theory* needed in the bivariate setting. We thus restrict the exposition to the changes required to the *implementation* of NURBS-compatible subdivision surfaces by T.J. Cashman available from http://www.cl.cam.ac.uk/research/rainbow/projects/subdnurbs/nurbswep.html.

Our modified implementation that supports creases was used to convert the aircraft model in Fig. 7 and the ring model in Fig. 16.

The implementation is based on the 'push' paradigm. This 'push' way of thinking about and implementing subdivision first topologically refines the control mesh, then collects and normalises all contributions from old points to new points

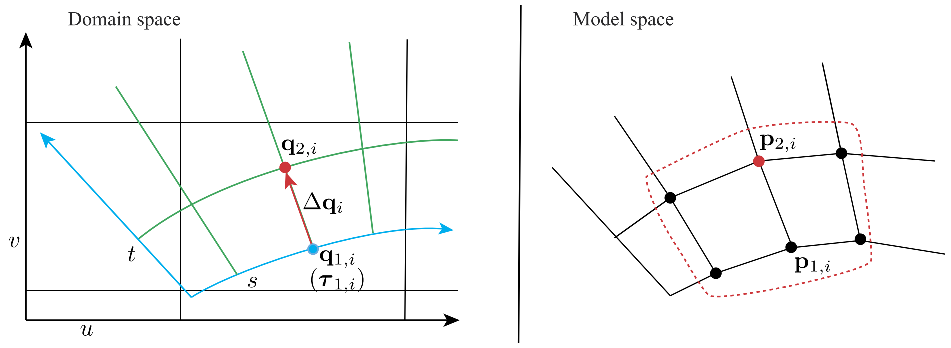

Fig. 20. Illustration on how to constrain the control points next to the boundary. Left: We show the domain space of the input NURBS (parameters u and v) in black and the local subdivision surface near the new boundary constructed from the trimmings (parameters s and t) in green. Right: The corresponding subdivision control mesh in model space. $\mathbf{q}_*$ is in (u, v) coordinates and $\boldsymbol{\tau}_*$ is in (s, t) coordinates. The subdivision control point $\mathbf{p}_{2,i}$ (red bullet) needs to lie on the tangent plane at $\mathbf{q}_{1,i}$ of the input NURBS surface (cyan dot in domain space), so that the tangent plane at $\mathbf{q}_{1,i}$ on the NURBS surface is the same as the tangent plane at $\boldsymbol{\tau}_{1,i}$ on the subdivision surface. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)

after a refine or smoothing step. This is achieved by looping over all faces (quadrilaterals) of the control mesh and 'pushing' contributions of their four vertices onto the new control points.

Our modification is to ensure that no information is passed 'across' a sharp edge. As explained above, ghost points are never explicitly needed, only one ghost knot is needed at each sharp (boundary) edge. With this ghost knot, the refine stage and the smoothing stage then proceed as in the original implementation via blossoming.

Our modified implementation will be made available.

Appendix C. Boundary tangential constraints

We have Bézier edge conditions or Pixar sharp edge rules applied along the subdivision surface boundary, and so the limit stencils for points in the first layer next to the boundary do not exist. These points actually control the tangent planes at the boundary. Therefore, in the subdivision surface fitting stage, we constrain them to lie on the tangent planes of the input NURBS surface at the boundary. Using the notation of Fig. 20, let $\Delta\mathbf{q}_i = \mathbf{q}_{2,i} - \mathbf{q}_{1,i}$, in (u, v) coordinates, and let $\boldsymbol{\tau}_{1,i} = (s_i, 0)$ in (s, t) coordinates. Then

$$\left(\frac{\partial\mathcal{N}}{\partial u}, \frac{\partial\mathcal{N}}{\partial v}\right)\bigg|_{\mathbf{q}_{1,i}} \cdot \Delta\mathbf{q}_i = \frac{\partial\mathcal{S}}{\partial t}\bigg|_{\boldsymbol{\tau}_{1,i}}. \tag{8}$$

The left side of Eq. (8) is known from the input NURBS surface $\mathcal{N}$. The right side is the first partial derivative of the target subdivision surface $\mathcal{S}$, derived from the chosen edge conditions. It involves the control points of $\mathcal{S}$ within the red dotted region in Fig. 20, right: the unknowns $\mathbf{p}_{2,*}$ and boundary control points $\mathbf{p}_{1,*}$ (which are already set up from the input boundary curves). Note that the boundary tangential constraint used by Shen et al. (2014) is the uniform case of this equation.

Appendix D. Limit stencils for EVs

As explained in Section 4.1, we only need limit stencils of EVs of valences 3 and 5 for degree 3. Using standard tools for obtaining limit stencils of stationary uniform subdivision schemes (Peters and Reif, 2008) and the bounded curvature coefficients from Section 5 of Cashman (2010), we obtained the limit stencils as the row eigenvector corresponding to the dominant eigenvalue (equal to 1) of the subdivision matrix for an EV of valence 3 and 5, respectively.

The limit stencils are shown in Fig. 21. The limit stencil comprises the 1-ring neighbourhood of the EV for valence 5, and an extra layer, due to an extra smoothing stage in Cashman's scheme, for valence 3.

Note that these limit stencils apply only after a uniform region has been created around the EVs. These stencils correspond to the bounded curvature solution. In the case of Catmull–Clark, the standard limit stencils apply.

Appendix E. Coarse quad layout

Here we describe the way we generate a coarse quad layout from a 2D domain bounded by curves for the examples in the paper. Alternative ways to produce a coarse quad layout include pattern-based quad meshing methods (Takayama et al., 2014; Peng et al., 2014) and Campen et al.'s quad layout design methods (Campen et al., 2012; Campen and Kobbelt, 2014).

Following a procedure similar to that of Shen et al. (2014), we triangulate the 2D domain, compute a smooth frame field with constraints along boundaries and at corners (Section E.1), locate singularities in the field, trace the field for a valid

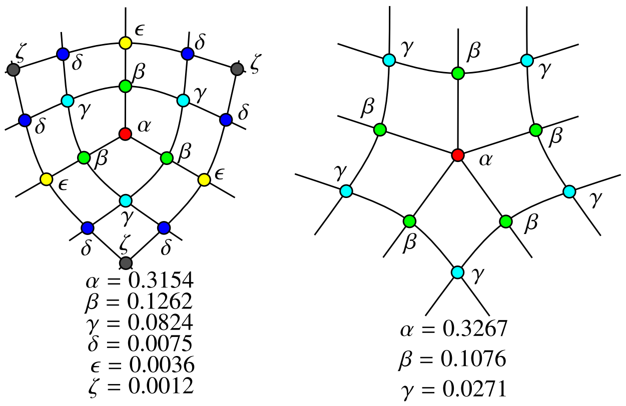

Fig. 21. Left: Limit stencil for the limit position of an EV of valence 3. Right: Limit stencil for an EV of valence 5.

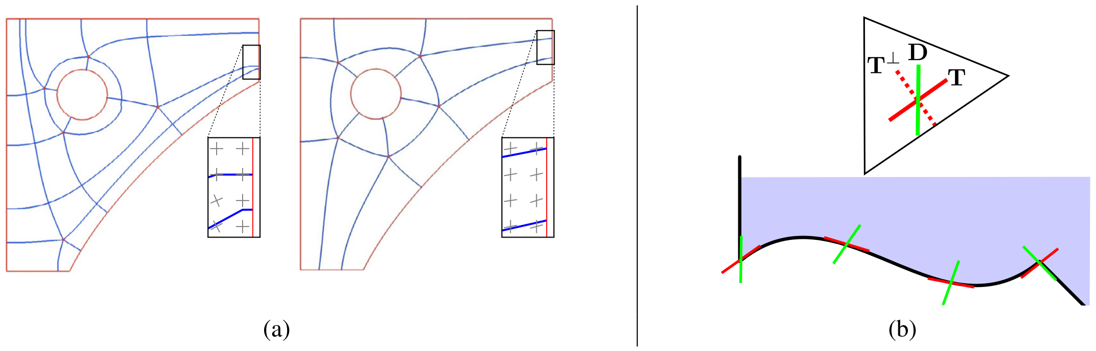

Fig. 22. (a) The frame field we adopted (right) generates a quad layout with better alignment along boundaries than the cross field based method (left) (Shen et al., 2014). (b) Frames at corners and along the boundary. The angle of the frame varies smoothly along the edge. (For interpretation of the colours in this figure, the reader is referred to the web version of this article.)

quad layout and finally simplify the layout if necessary (Section E.2, E.3). A simple comparison with the cross field is shown in Fig. 22a.

E.1. Frame field

To get good alignment at corners and along boundaries, we adopt the frame field of Diamanti et al. (2014). The field is defined on a 2D triangular mesh of the input trimmed domain. Each triangle face has a 2D frame which is constant inside the face.

Input directional constraints are set up along the oriented boundary curves. As illustrated in Fig. 22b, for each corner, we put a frame that matches both edge directions at this corner; for each boundary piece between two corners, we set up the frames along it: one direction, **T** (shown in red), follows the tangent of the boundary, and the other direction, **D** (shown in green), interpolates the corner frames in the following way.

$$\mathbf{D}(t) = \alpha(t)\mathbf{T}(t) + \beta(t)\mathbf{T}^{\perp}(t)$$
$$\alpha(t) = (1-t)\alpha_0 + t\alpha_1$$
$$\beta(t) = (1-t)\beta_0 + t\beta_1$$

where t is the parameter value (rescaled to $[0, 1]$) along the boundary curve, $\mathbf{T}(t)$ is the tangent of the curve, $\mathbf{T}^{\perp}(t)$ is the 90° rotation of the tangent, $\alpha_0, \beta_0, \alpha_1, \beta_1$ are fixed from the frames at the two corners, respectively.

A smooth frame field is then computed via the IGL library (Diamanti et al., 2014). We use unit-length frames and our frame angle is bounded with minimum 30°. The singularities of the frame field are the locations of the EVs in the desired quad mesh. They are identified by using the cross field defined by the frame bisectors (Diamanti et al., 2014).

E.2. Tracing for a quad layout

We trace streamlines from each EV and each concave corner (all defined on a vertex). The initial tracing directions from an EV (or corner) are selected by clustering the vector directions in the ring of faces round it. For each streamline, the

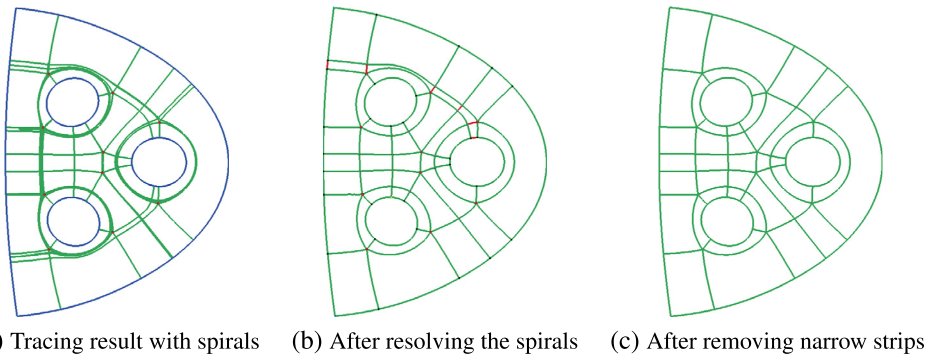
(a) Tracing result with spirals
(b) After resolving the spirals
(c) After removing narrow strips

Fig. 23. Tracing for a coarse quad layout. The final coarse quad layout is shown in (c) after removing the narrow quadrilateral face strip, i.e., eliminating the edges marked in red in (b). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

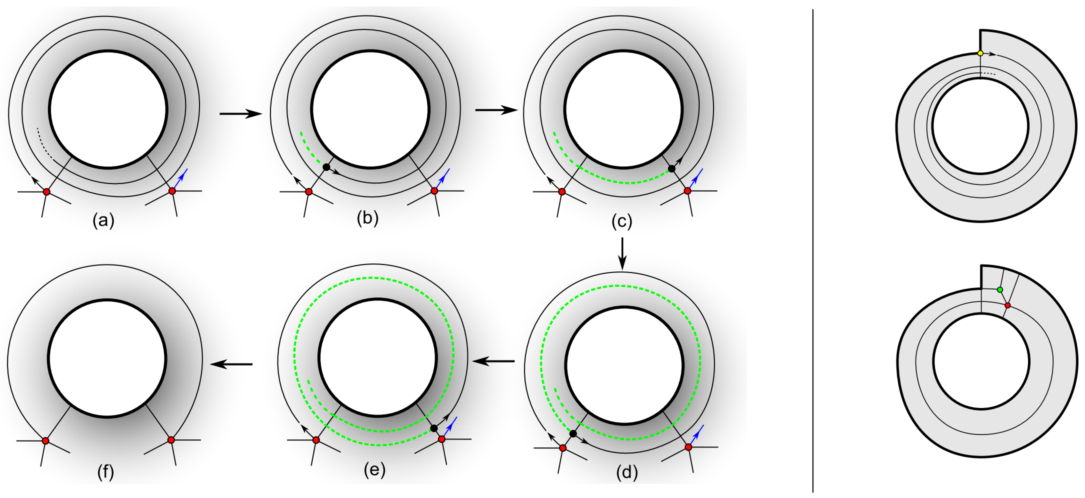

Fig. 24. Left: a spiral trace resolved by back-tracing and snapping. (a) Identify candidate points (singularity or concave corner) for snapping. The separatrix of the candidate point is referred to as 'slot' and is highlighted with an outgoing blue arrow if it has not been booked by other streamlines. (b)–(e) Trace backwards. Each black point shows a successive 'closest point' on the trace to one or other of the candidate points; the black point in (e) is identified as the optimum match with an available slot. (f) Snap the trace to the appropriate candidate point. Right: spiralling resolved by adding an EV3–EV5 pair.

tracing continues until it hits an EV, a corner or a boundary. Tracing also terminates if the length of the trace exceeds a certain threshold. This latter condition indicates a spiral has occurred. Other robustness issues in field tracing can be dealt with as explained by Myles et al. (2014) and Ray et al. (2014). We have not encountered any problems, other than spiralling, in any of our examples.

We resolve spirals by walking back along the spiral and seeking neighbouring EVs or concave corners to which they can be snapped (Fig. 24, see Section E.3 for details). An alternative way is to adopt the method proposed by Myles et al. (2014). They prevent spirals by forming a partition that allows T-joins and then resolving the topology by either removing zero quad chains or introducing new singularity pairs.

E.3. Spiralling

There are two manifestations of spirals. The first is owing to inaccuracies in the (discrete) skew field and tracing (Fig. 24 left); this is remedied by back-tracing and snapping to a nearby available slot. The second manifestation is not an implementation issue but inherent in the given topology (Fig. 24 right); it can be remedied by an appropriate topological refinement comprising the introduction of an EV3–EV5 pair.

For our examples, the spiralling occurs when tracing fails to join up with an existing vertex and circles forever round a hole in the mesh (Fig. 23a). We identify that a spiral has occurred when the length of the trace becomes larger than 5 times the bounding box diagonal. As illustrated in Fig. 24 left, our remedial algorithm traces back along the spiral, seeking neighbouring EVs or concave corners to which the trace could be snapped. The *snap condition* is that there is no incoming trace in the candidate slot of that neighbouring EV (or concave corner). It snaps the trace to the matching neighbour with an available slot and the maximum eliminating length, or introduces two new EVs if there is no match (Fig. 24 right).

Once all the streamlines are normal, they form a coarse quad layout together with the domain boundary. The quad layout from the above process might have narrow strips and so we further simplify the topology to a coarse quad layout (Fig. 23b–c) by removing these strips (Tarini et al., 2011).

References

Alliez, Pierre, Cohen-Steiner, David, Devillers, Olivier, Lévy, Bruno, Desbrun, Mathieu, 2003. Anisotropic polygonal remeshing. ACM Trans. Graph. 22 (3), 485–493.

Berkelaar, Michel, Eikland, Kjell, Notebaert, Peter, 2004. lpsolve: open source (Mixed-Integer) Linear Programming system.

Bommes, David, Campen, Marcel, Ebke, Hans-Christian, Alliez, Pierre, Kobbelt, Leif, 2013. Integer-grid maps for reliable quad meshing. ACM Trans. Graph. 32 (4).

Bommes, David, Zimmer, Henrik, Kobbelt, Leif, 2009. Mixed-integer quadrangulation. ACM Trans. Graph. 28 (3), 1.

Campen, Marcel, Bommes, David, Kobbelt, Leif, 2012. Dual loops meshing. ACM Trans. Graph. 31 (4), 1–11.

Campen, Marcel, Kobbelt, Leif, 2014. Dual strip weaving: interactive design of quad layouts using elastica strips. ACM Trans. Graph. 33 (6), 183:1–183:10.

Cashman, Thomas J., 2010. NURBS-compatible subdivision surfaces. Technical report UCAM-CL-TR-773. University of Cambridge, Computer Laboratory (Doctoral thesis).

Cashman, Thomas J., Augsdörfer, Ursula H., Dodgson, Neil A., Sabin, Malcolm A., 2009. NURBS with extraordinary points: high-degree, non-uniform, rational subdivision schemes. ACM Trans. Graph. 28 (3), 1.

de Boor, Carl, 1972. On calculating with b-splines. J. Approx. Theory 6 (1), 50–62.

Deng, Chongyang, Lin, Hongwei, 2014. Progressive and iterative approximation for least squares B-spline curve and surface fitting. Comput. Aided Des. 47, 32–44.

DeRose, Tony, Kass, Michael, Truong, Tien, 1998. Subdivision surfaces in character animation. In: Proc. SIGGRAPH '98. ACM Press, pp. 85–94.

Diamanti, Olga, Vaxman, Amir, Panozzo, Daniele, Sorkine-Hornung, Olga, 2014. Designing N-polyVector fields with complex polynomials. Comput. Graph. Forum 33 (5), 1–11.

Eck, Matthias, Hadenfeld, Jan, 1995. Knot removal for b-spline curves. Comput. Aided Geom. Des. 12 (3), 259–282.

Floater, Michael S., 2000. Meshless parameterization and B-spline surface approximation. In: Proc. 9th IMA Conference on the Mathematics of Surfaces. Springer-Verlag, pp. 1–18.

Halstead, Mark, Kass, Michael, DeRose, Tony, 1993. Efficient, fair interpolation using Catmull–Clark surfaces. In: Proc. SIGGRAPH '93. ACM Press, pp. 35–44.

Jiang, Tengfei, Fang, Xianzhong, Huang, Jin, Bao, Hujun, Tong, Yiying, Desbrun, Mathieu, 2015. Frame field generation through metric customization. ACM Trans. Graph. 34 (4), 40:1–40:11.

Imre, Juhász, Hoffmann, Miklós, 2001. The effect of knot modifications on the shape of B-spline curves. J. Geom. Graph. 5, 111–119.

Kosinka, J., Sabin, M.A., Dodgson, N.A., 2014a. Semi-sharp creases on subdivision curves and surfaces. Comput. Graph. Forum 33 (5), 217–226.

Kosinka, J., Sabin, M.A., Dodgson, N.A., 2014b. Subdivision surfaces with creases and truncated multiple knot lines. Comput. Graph. Forum 33 (1), 118–128.

Kosinka, Jiří, Sabin, Malcolm, Dodgson, Neil, 2014c. Creases and boundary conditions for subdivision curves. Graph. Models 76 (5), 240–251.

Lacewell, Dylan, Burley, Brent, 2007. Exact evaluation of Catmull–Clark subdivision surfaces near B-Spline boundaries. J. Graph. GPU Game Tools 12 (3), 7–15.

Lyche, Tom, Mørken, Knut, 1987a. Algorithms for Approximation. Clarendon Press.

Lyche, Tom, Mørken, Knut, 1987b. Knot removal for parametric b-spline curves and surfaces. Comput. Aided Geom. Des. 4 (3), 217–230.

Müller, Kerstin, Fünfzig, Christoph, Reusche, Lars, Hansford, Dianne, Farin, Gerald, Hagen, Hans, 2010. Dinus: double insertion, nonuniform, stationary subdivision surfaces. ACM Trans. Graph. 29 (3), 1–21.

Müller, Kerstin, Reusche, Lars, Fellner, Dieter, 2006. Extended subdivision surfaces: building a bridge between NURBS and Catmull–Clark surfaces. ACM Trans. Graph. 25 (2), 268–292.

Myles, Ashish, Pietroni, Nico, Zorin, Denis, 2014. Robust field-aligned global parametrization. ACM Trans. Graph. 33 (4), 135:1–135:14.

Palacios, Jonathan, Zhang, Eugene, 2010. Interactive visualization of rotational symmetry fields on surfaces. IEEE Trans. Vis. Comput. Graph. 17 (7), 947–955.

Peng, Chi-Han, Barton, Michael, Jiang, Caigui, Wonka, Peter, 2014. Exploring quadrangulations. ACM Trans. Graph. 33 (1), 12:1–12:13.

Peters, Jörg, Reif, Ulrich, 2008. Subdivision Surfaces. Springer Publishing Company, Incorporated, ISBN 978-3-540-76405-2.

Peters, Jörg, Wu, Xiaobin, 2006. On the local linear independence of generalized subdivision functions. SIAM J. Numer. Anal. 44 (6), 2389–2407.

Ray, Nicolas, Sokolov, Dmitry, 2014. Robust polylines tracing for n-symmetry direction field on triangulated surfaces. ACM Trans. Graph. 33 (3), 30:1–30:11.

Rhinoceros, 2014. http://www.rhino3d.com/tutorials.

Schweitzer, Jean E., 1996. Analysis and application of subdivision surfaces. Doctoral thesis. University of Washington.

Sederberg, Thomas W., Finnigan, G. Thomas, Li, Xin, Lin, Hongwei, Ipson, Heather, 2008. Watertight trimmed NURBS. ACM Trans. Graph. 27 (3), 1.

Sederberg, Thomas W., Zheng, Jianmin, Bakenov, Almaz, Nasri, Ahmad, 2003. T-splines and T-NURCCs. ACM Trans. Graph. 22 (3), 477.

Sederberg, Thomas W., Zheng, Jianmin, Sewell, David, Sabin, Malcolm, 1998. Non-uniform recursive subdivision surfaces. In: Proc. SIGGRAPH '98. ACM Press, pp. 387–394.

Shen, Jingjing, Kosinka, Jiří, Sabin, Malcolm, Dodgson, Neil, 2014. Conversion of trimmed NURBS surfaces to Catmull–Clark subdivision surfaces. Comput. Aided Geom. Des. 31 (7–8), 486–498.

Stam, Jos, 1998. Exact evaluation of Catmull–Clark subdivision surfaces at arbitrary parameter values. In: Proc. SIGGRAPH '98. ACM, pp. 395–404.

Takayama, Kenshi, Panozzo, Daniele, Sorkine-Hornung, Olga, 2014. Pattern-based quadrangulation for n-sided patches. Comput. Graph. Forum 33 (5), 177–184.

Tarini, Marco, Puppo, Enrico, Panozzo, Daniele, Pietroni, Nico, Cignoni, Paolo, 2011. Simple quad domains for field aligned mesh parametrization. ACM Trans. Graph. 30 (6), 142:1–142:12.

Tiller, Wayne, 1992. Knot-removal algorithms for nurbs curves and surfaces. Comput. Aided Des. 24 (8), 445–453.

Vermeulen, A.H., Bartels, R.H., Heppler, G.R., 1992. Integrating products of b-splines. SIAM J. Sci. Stat. Comput. 13 (4), 1025–1038.

Weiss, V., Andor, L., Renner, G., Várady, T., 2002. Advanced surface fitting techniques. Comput. Aided Geom. Des. 19 (1), 19–42.