Boundaries and Creases for Spline Surfaces with Extraordinary Vertices

Param Gupta[1], Kęstutis Karčiauskas[b], Jörg Peters[1,*]

[a] *University of Florida. Gainesville. USA*
[b] *Vilnius University, Lithuania*

Abstract

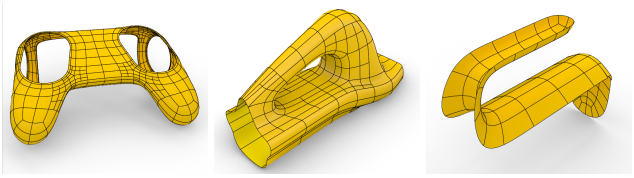
Global boundaries of spline surfaces require special treatment due to the lack of a complete neighborhood, in particular in the presence of extraordinary vertices (EVs). Spline surfaces with EVs and/or n-sided facets in their polyhedral control net are akin to generalized subdivision surfaces but cannot leverage infinite refinability to satisfy boundary or semi-sharp crease constraints.

The new boundary and crease treatment for high-end, free-form spline surface algorithms aims to replicate subdivision options by using as much as possible the algorithm developed for closed spline surfaces, i.e. eschewing special rules at the level of Bézier coefficients or trying to define knot spacing for EVs. In particular, by virtually augmenting the control net, the new rules offer interpolation and approximation of the curve defined by the global boundary of the original control net, corner treatment and semi-sharp creases.

Keywords: high-end free-form spline, boundary treatment, extraordinary vertices (EVs), mirroring, partial loop insertion, semi-sharp creases

1. Introduction

The treatment of global boundaries and semi-sharp creases by high-end, free-form spline surface (HFS) algorithms is often an afterthought in the development of their intricate rules. Yet, in practice, boundaries and creases are important for compatibility of piecemeal designs, see Fig. 1. Ideally, one would avoid developing special new rules for these complex algorithms for treatment of global boundaries and localized creases.



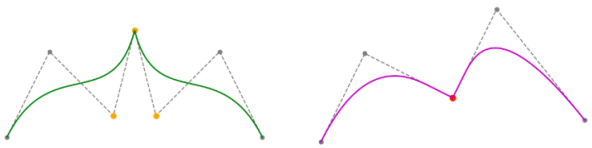
(a) game controller (b) hedge trimmer casing (c) mixer handle

Figure 1: Snap-in plastic surfaces where the boundary matters. The three curvature-continuous surfaces were generated in a CAD pipeline and modeling system, and match prescribed boundaries. The partition delineates G^2-joined Bézier patches.

Subdivision-based modeling is increasingly popular for early design stages. 'Subd' environments provide canonical boundary and localized crease rules. However, they are a dead-end without conversion to the existing CAD pipeline and modeling systemthat are based on NURBS and the step-format. Existing re-approximation conversion tend to be unsatisfactory due to one of: loss of smoothness, deterioration of shape, proliferation of b-rep pieces, high degree and number of spans or omissions of boundary features or local creases. Conversely, high-end NURBS HFS excel at handling smooth EVs internally but typically fail to offer the convenience and features of current 'subd' packages. The goal of this paper is to add these features without touching the complex HFS code and so allow modeling directly with the existing HFS as if it were a 'subd' environment.

1.1. Curves

In one variable, already the classic texts on splines [1] and many numerical ode solver codes, address the treatment of the global mesh boundary, namely the interpretation of the B-spline control net at the end of the definition interval. Where the neighbors are missing to one side, there are too few functions defined over the knot intervals associated with the end control points to span the full spline space. A natural solution for a spline curve is to simply restrict its evaluation to where it is well-defined, resulting in an apparent shortening of the curve vis-a-vis its control polygon. Adopting this approach would eliminate the need for boundary rules both for curves and surfaces. In practice, however – to better control end behavior by information in the control net *visible* to a designer – it is customary to additionally match the end-points in one variable, and to interpolate or approximate an end curve in two variables. If the designer expects to interpolate the last control point, there are fundamentally three options, see Fig. 2: (i) repeating knots, i.e. singular namely zero length, knot intervals in the domain, often called Bézier end conditions; (ii) repeating control points (CPs), i.e. zero length differences between control points in the range; or (iii) extrapolating the control net so that subsequent shortening to well-defined curve segments interpolates the end point.



(a) multiple knots (b) multiple CPs

Figure 2: Multiple knots vs multiple control points in same place

*Corresponding author
Email addresses: p.gupta@ufl.edu (Param Gupta), kestutis.karciauskas@mif.vu.lt (Kęstutis Karčiauskas), jorg.peters@gmail.com (Jörg Peters)

Repeating knots (i) mimics the knot spacing of Bézier segments in B-spline form and so gives explicit local control of position and derivative. In numerical computations, such Dirichlet and von Neumann boundary conditions are typically available to define the function space at the ends. However, repeating knots will not be a well-defined option for surfaces with extraordinary vertices (EVs) on the boundary and would require adding special rules to an already complex high-end algorithm. The simple rule (ii), namely repeating control points, is not an option for *functions*, but it is for free-form curves. The resulting zero derivative vectors of several orders at the end-point that can cause problems if not judiciously treated with L'Hôpital's Rule as parametric, but not geometric, singularity. The third option (iii), extrapolation, is the focus of this paper. However, extrapolation gives the last control point(s) two meanings: first as a provider of a B-spline to define the full space over an interval in the interior, and second as an interpolation target. Different strategies can be employed: (1) to extend, (2) to pull the surface into position or (3) to add a spline segment to interpolate.

The introduction of semi-sharp creases – i.e. and intentional reduction of internal smoothness of free-form curves or surfaces – has treatment options analogous to the repeating knots or control points (apart from inventing special rules at the granularity of manipulating Bézier coefficients). We can create singular intervals between knots in the domain – by increasing knot multiplicity, see Fig. 2a – or work in the range – by stacking up control-points one on top of the other Fig. 2b. Again, stacking is not an option for graphs of functions but it is for free-form curves and surfaces. In the first case, the complication of zero knot intervals is taken care of by an 'if-then' branching statement in de Boor's algorithm. Analogously, in the case of repeated control points, a branching statement has to react to zero derivatives, e.g. by reaching out further and computing differential geometric quantities from the first non-zero control point differences via L'Hôpital's rule. That is, the anticipated singularity is predictable and controllable. In both cases a tolerance is required to trigger the branching statement. Moreover, the treatment of zero differences in the range is not universally implemented in commercial and other geometry packages, although typically available to deal zero first derivatives of simple singularities in the parameterization. For double singularities in the range, commercial solutions like the ubiquitous ParaSolid kernel, return failure messages.

1.2. Surfaces

High-end, free-form surfaces (HFS), the focus of this paper, can use the univariate curve end-point and crease rules, but only if, locally along the global mesh boundary, the surface is a tensor-product spline. The global mesh boundary of a tensor-product spline consist of corner points of valence 2 and regular boundary points, namely 3-valent vertices. For non-regular, extraordinary points (EVs) on the boundary of the control nets of HFS, the notion of multiple knots in the direction orthogonal to the global mesh boundary is not well-defined – despite valiant attempts in the context of subdivision, see NURCCS [2, 3]. Boundary treatment therefore requires the addition of extrapolating strips or loops, if the control net (without special rules at the level of Bézier coefficients) is to define the behavior. Such extrapolation can use rules such as mirroring the inner control net across the global mesh boundary or by extending with quad strips of zero area, i.e. replicating the outermost control loop. Mirroring is trivial when the control net neighborhood of the global mesh boundary is a tensor-product but requires careful considerations when an EV is located within the support of the global mesh boundary, already because the naively-mirrored control net may self-intersect for concave boundary EVs.

A common practice, apparently introduced by Pixar's implementation of the Catmull-Clark algorithm, is for subdivision surfaces with rim to interpolate a *global boundary curve*, i.e. the space curve obtained by interpreting the global boundary polygon as the control polygon of a uniform cubic spline curve. Pixar's OpenSubdiv [4] states as the goal for interpolating global boundary curves 'to create a smooth, continuous limit surface that seamlessly connects to adjacent geometry, preventing gaps and sharp, undesirable "pull-away" artifacts common with standard subdivision rules that otherwise try to smooth towards the boundary'. It should be noted that, while aimed at convenience, this motivation is dubious for at least two reasons. First and foremost, a geometric rule is introduced to achieve the *topological* aim of joining two surface designs. A more principled way to join the two designs is by merging the control nets since this is resilient to differeing accuracies when exporting a model. If a sharp transition is wanted, this can be indicated by tagging the edge as a sharp crease. By contrast, relying on geometric proximity seems flawed: should either geometry change thereafter, the connectivity and apparent design intent may be lost. Second, choosing the agreed-upon curve to be the curve defined by the global boundary polygon interpreted as a uniform degree 3 (B-)spline curve is *arbitrary* and makes the global boundary polygon do double duty: defining the curve as well as contributing to the surface near the boundary. Alternatively an explicit curve could separately be prescribed, or, more principled, both objects would model the same overlapping global mesh boundary from both sides so that the retraction by well-defined B-spline rules has both splines meet up at the same boundary. This unsophisticated approach also works for non-manifold designs, where more than two spline designs meet at an interface. However, interpolating the global boundary curve is a standard, and, if universally followed, has the advantage of convenience for quick work. This standard will therefore be a focus of the boundary rules of this paper.

The second topic of this paper, internal semi-sharp creases, can be implemented by manipulating Bézier coefficients. However, this increases the complexity of an underlying high-end polyhedral-net algorithm. Consequently, here too, we aim to only manipulate the control net and avoid innovations like curved knot lines [5, 6]. The straightforward insertion of mesh loops and their placement is made more complex when the creasing only local. Terminating partial mesh loops and EVs require innovation.

In summary, for existing high-end free-form spline algorithms with EVs, this paper contributes new control-net based rules for treating global mesh boundarys and modeling semi-sharp creases:

- Enforcement of global boundary and crease properties across EVs – by extending the B-spline level polyhedral control net, but keeping the base algorithm unchanged, i.e.
- eschewing special rules at the Bernstein-Bézier coefficient level or trying to define knots at EVs;
- interpolation and approximation of the global boundary polygon curve;
- semi-sharp corner interpolation rules;
- localizable semi-sharp creases.

Overview After prior work review Section 2, Section 3 develops the interpolation or approximation of the global boundary polygon curve, Section 4 gives algorithms for rounded vs. sharp corners and Section 5 introduces an approach to create crease and semi-sharp creases, also across EVs. Section 6 concludes with a discussion of choices and limitations, and points to possible future work.

2. Prior work

Our emphasis is on spline surfaces controlled by a polyhedral control-net that includes extraordinary vertices (EVs) and polygons. In particular, [7, 8, 9, 10, 11, 12], i.e. smooth piecewise polynomial surfaces classified as G-splines in the survey[13]. Multi-sided transfinite constructions such as [14, 15, 16] typically start by prescribing boundary data as a 'ribbon' and so need no special global boundary rules. Subdivision surfaces encounter very similar challenges at boundary EVs and internal crease edges whose rules have been discussed over the years. This literature is therefore separately reviewed in Section 2.2. We remind already here that solutions leveraging the infinite refinement of subdivision surfaces to address boundary EVs and semi-sharp creases cannot be leveraged for splines. We start with a look at implementations in current use.

2.1. Commercial and open source implementations

Major CAD-software houses offer both a mainstream and a high-end surface modeling product. Autodesk offers Alias, Inventor and Fusion 360/T-spline, Dassault Systémes offers 3D Experience/CATIA (xDesign), IcemSurf and Solidworks, Siemens SolidEdge and NX (Realize Shape), PTC Creo (FreeStyle), Rhino3D (SubD, [17]), Blender (Subdiv or Subsurf). Often both mainstream and a high-end products have an option for calling a variant of Catmull-Clark subdivision [18] akin, identical to the open source product OpenSubdiv [4], also available via Blender [19]. Above some of these subdivision options were listed in parentheses. While the subdivision surface option is convenient for fast ideation, manufacturing-grade surfaces are primarily modeled by NURBS. Since subdivision has in principle infinitely many pieces (of degree bi-3) such surfaces must be converted for most downstream processing. The transition from the subdivision model to the NURBS model may not preserve smoothness. For example, Rhino3D offers resurfacing, with an option of continuity set to G^1 or G^2, but these options can fail to deliver even G^1 at EVs.

A standard for subdivision surfaces is set by the Pixar/Microsoft OpenSubdiv implementation and the Renderman rules [20] (see Section 2.2). Notably, this standard includes (i) interpolation of the *global boundary curve*, defined by the global boundary polygon, (ii) rounded or sharp corners for $n = 2$ boundary vertices, and (iii) semi-sharp creases. NURBS modeling tools often achieve such effects by least squares fit, fillets and more general blends, i.e. many additional custom tools. High-end NURBS packages in Alias, NX and Creo are of high degree (bi-6 or bi-7) and have multiple spans, i.e. B-rep surfaces (patches) that consist of 13×13 or even $16\times$ polynomial pieces. That makes it challenging to develop special rules for EVs on boundaries or semi-sharp creases.

2.2. Boundary and crease treatment for subdivision surfaces

Internal EVs of subdivision algorithms have been formally and thoroughly analyzed [4] and Renderman rules and Nasri and Sabin [21] provide a rich 'taxonomy of interpolation constraints on recursive subdivision surfaces'. Specifically, [21] enumerates five cases: (A) truncation, i.e. evaluating only up to where the underlying spline surface is well-defined, (B) polygon modification, (C) subdivision modification, see [22], akin to rules at the level of Bézier coefficients, (D) coalesced vertices, i.e. zero width extension and (E) modification after one or more steps of subdivision. For Catmull-Clark-like subdivision, (E) converts a boundary n-gon to an n-valent EV in the interior, but does not remove an EV on the global mesh boundary. See also [23] for curve subdivision. One additional option (F), that truly leverages the recursive nature and infinite number of polynomial pieces – but therefore also does not apply to our spline scenario – is the combined subdivision approach [24]. This approach has a separately-defined boundary participate unchanged in every subdivision step, pulling the surface to the separately-defined boundary limit curve, or additonal data. [21] is expository and leaves the more complicated cases to be worked out. For example, concave global boundary polygon corners are left as future work. The present paper can be viewed as providing such work in the context of HFS. Beyond the regular mirroring rules

$$\mathbf{a}_i := 2\mathbf{e}_i - \mathbf{i}_i \tag{1}$$

that create extensions across a boundary vertex $\mathbf{e}_i$ by reflecting its interior neighbor $\mathbf{i}_i$ across $\mathbf{e}_i$, one concrete suggestion of [21] applies to Doo-Sabin subdivision [25]: extending by reflecting the vertices of n-gon with two or more vertices on the global mesh boundary about Chebychev points on the edges emanating from $\mathbf{e}_i$, i.e. assuming a uniform symmetric distribution in the domain. We generalize this idea to the corner case and apply a variant in Section 3.3.

3. Interpolation or approximation of the global boundary polygon curve for EVs ($n > 3$)

Joining control nets is a topological operation that should not rely on geometric proximity. Yet, joining meshes is the stated reason for the OpenSubdiv [4] option that sets the standard of interpolating the *global boundary curve* defined as a uniform cubic spline whose control polygon is the global boundary polygon. [21] calls this interpolation 'natural'. At closer view, this is a convenient but arbitrary choice since the global boundary polygon now serves two purposes: as the control polygon of a curve to be interpolated by the surface rim, and as part of the surface's control net defining near-boundary patches. Interpolating any prescribed curve is also not necessary to geometrically join pieces: replicating outer mesh layers and generating surface only where well-defined (called 'truncation' in [21]) would achieve the same outcome based on overlapping control strips along the two objects' boundaries.

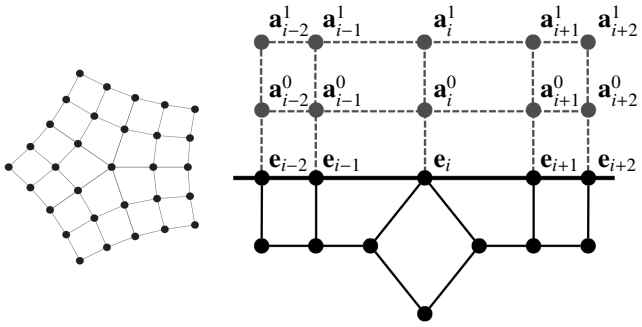
Interpolating the global boundary curve requires an extension of the surface beyond the domain of definition. *Regular global mesh boundary* with corner vertices, i.e. those of valence 2 and $n = 3$-valent vertices otherwise, can be treated with the existing univariate rules, foremost the regular mirroring rules (1). We focus on the case of EVs with $n > 3$ on the global boundary polygon using virtual control-net extension, to avoid special rules at the level of Bézier coefficients. (Special rules must also depend on existing data, i.e. an extension of the control net based on the existing control net.)

Based on existing algorithms, we posit that two additional virtual layers suffice for HFS constructions and that EVs separated, if only after a local step of refinement. To apply a HFS when an

EV lies on the global mesh boundary, we need to add two quad-strips, see Fig. 3a. In all cases, we *virtually* complete the required neighborhood. For illustrations and specific verification, we use as a prototypical, exemplar HFS algorithm the G^2 bi-6 2×2 span surface construction of [26]. We offer three approaches, that can be seen as adding respective 0,1 or many virtual neighbors to the EV. Algorithm $\text{HFS}_{3.1}$ is the most principled in that no new control-net geometry is created; however, $\text{HFS}_{3.1}$ creates new b-rep surfaces that result in a 'not implemented' fault for kernels like ParaSolid that do not implement repeated Hôpital's rule. Algorithm $\text{HFS}_{3.3}$ preserves the b-rep structure of the input control net by not adding b-rep surfaces, but does not interpolate the global boundary curve. Algorithm $\text{HFS}_{3.2}$ is a good compromise in practice. Table 1 compares the features and trade-offs.

Feature	$\text{HFS}_{3.1}$	$\text{HFS}_{3.2}$	$\text{HFS}_{3.3}$
Preserve control-net B-Rep	×	×	✓
Interpolate global boundary curve	✓	✓	×
Reliable Kernel Support	×	✓	✓

Table 1: Comparison of algorithm features and practical trade-offs.



(a) Internal EV (b) Virtual control-net extension for a boundary EV

Figure 3: (a) Internal EV and 2-ring neighborhood. (b) Structure of singular virtual extension (dotted layers).

3.1. $\text{HFS}_{3.1}$: Double zero-width extension

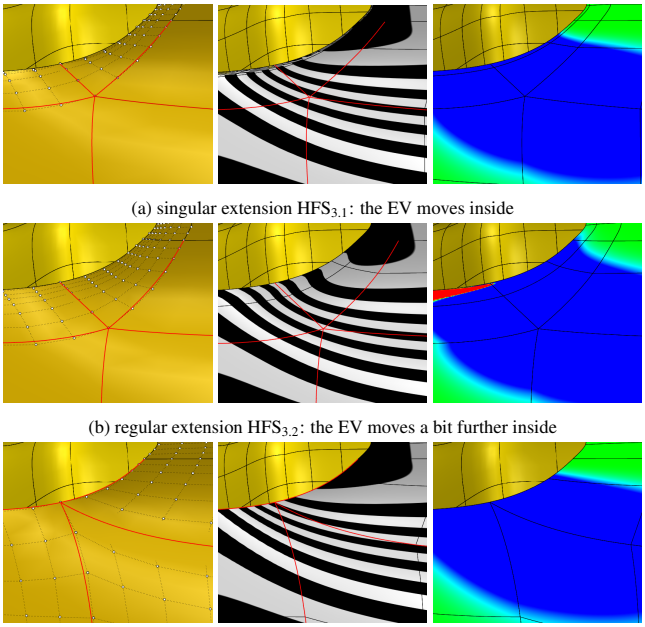
A straightforward solution to the boundary problem is the zero width extension without new geometry. Figure 3b shows as solid lines the global mesh boundary with vertices $\mathbf{e}_j$. The EV, of valence $n = 4$, is labeled $\mathbf{e}_i$. The two virtual extension layers have vertices $\mathbf{a}^0_i$ and $\mathbf{a}^1_i$ that are joined by dotted lines. In $\text{HFS}_{3.1}$, the additional layers coincide with the global boundary polygon: $\mathbf{a}^0_i = \mathbf{a}^1_i = \mathbf{e}_i$. Since the zero area HFS becomes a spline curve, the extended $\text{HFS}_{3.1}$ surface interpolates the global boundary curve spline defined by the global boundary polygon. Note that the extension layers introduce a strip of four-sided patches along the boundary, see Fig. 4a. These patches have three layers of Bézier coefficients coincide along the interpolated global boundary curve. Hôpital's Rule allows calculating tangent and normal vectors, curvature, etc., but some geometric modeling kernels such as ParaSolid do not support zero width singularities beyond the first.

3.2. $\text{HFS}_{3.2}$: Boundary extension placing the EV into the interior

Using again the control-net structure of Fig. 3b, but without zero-width, we place the first extension layer $\mathbf{a}^0$ to coincide with the global boundary polygon, but pull the original global boundary polygon layer $\mathbf{e}$ back in a direction $\mathbf{d}$ orthogonal to the global boundary curve by an amount $\varepsilon > 0$ (default $\varepsilon = 0.25$) to a position $\mathbf{a}^{-1}$ (The global boundary curve was originally defined by $\mathbf{e}_i$, now the same curve is defined by using $\mathbf{a}^0_i$ as the control polygon). Then the second extension layer is placed symmetrically at the opposite side of $\mathbf{a}^0$ resulting in $\mathbf{a}^{-1}$ and $\mathbf{a}^1$ being reflections of each other across the global boundary polygon layer $\mathbf{a}^0$:

$$\mathbf{a}^{-1}_i := \mathbf{e}_i - \varepsilon\mathbf{d}_i. \quad \mathbf{a}^0_i := \mathbf{e}_i, \quad \mathbf{a}^1_i := \mathbf{e}_i + \varepsilon\mathbf{d}_i. \tag{2}$$

A good default choice for the direction $\mathbf{d}_i$ is the direction from the average of the centroids of the quad facets meeting at $\mathbf{e}_i$ to the point $(\mathbf{e}_{i-1} + 4\mathbf{e}_i + \mathbf{e}_{i+1})/6$ on the global boundary curve. The singular construction corresponds to $\varepsilon = 0$. Due to symmetry preservation, the spline curve defined by $\mathbf{a}^0$ becomes the global boundary curve. Essentially, this approach adds a regular boundary strip and applies the univariate mirroring approach of Eq (1): $\text{HFS}_{3.2}$ introduces an additional extension strip of quadrilateral patches and pulls the extraordinary vertex towards the surface interior, see Fig. 4b and Fig. 5b.



(a) singular extension $\text{HFS}_{3.1}$: the EV moves inside

(b) regular extension $\text{HFS}_{3.2}$: the EV moves a bit further inside

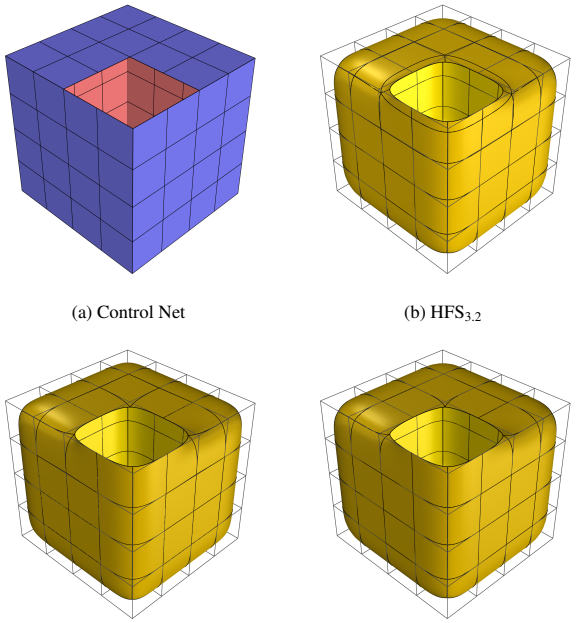
(c) mirroring $\text{HFS}_{3.3}$: the EV remains on the boundary

Figure 4: Boundary treatment (zoom of Fig. 5) according to (a) Section 3.1, (b) Section 3.2, (c) Section 3.3. (a,b) interpolate the global boundary polygon curve by adding a layer of patches. The singular extension in (a) results in coinciding BB-net layers at the curve so that only 5 of the 7 layers are visible. (c) Mirroring allows the EV to stay on the boundary but does not reproduce the global boundary curve. (*left* column) BB control net near boundary EV. (*middle*) highlight lines (*right*) Mean curvature.

3.3. $\text{HFS}_{3.3}$: Virtual Extension by Mirroring across EVs

The previous two options created an additional quad strip buffering the EV from the global boundary curve; and to interpolate the curve using a regular global mesh boundary. $\text{HFS}_{3.3}$ does not interpolate the global boundary curve, but generates a smooth boundary that is consistent in the sense that, if applied to two surfaces, joins them without gap. The advantage of $\text{HFS}_{3.3}$ is that the EV remains on the boundary and no additional B-rep (at least C^1-, not G^1-joined) surfaces are generated.

Here, the extension layer completing the required neighborhood is created by mirroring across the EV as illustrated in Fig. 6.



(a) Control Net (b) $\mathrm{HFS}_{3,2}$ (c) SubD (d) $\mathrm{HFS}_{3,3}$

Figure 5: Boundary EV with $n = 4$. See Fig. 4 for zoom. Note the similarity of the HFS surfaces to Rhino3D's SubD.

Any surfaces generated outside the global mesh boundary are discarded. We define a first extension layer with vertices $\mathbf{a}^0_{i,j}$, mirroring vertices $\mathbf{i}_{i,j}$, $j = 0, \ldots, N$, $N := 2n-6$. Specifically, the 'inner ones' with index $j < 0$ and $j < 2n-6$, reflected across a mirror point $\mathbf{q}$ that is to be determined and the 'outer ones' mirror across each of the two edges connected to $\mathbf{e}_i$, in order to preserve the ordering for the boundary pieces to the left and right of the EV patch construction:

$$\begin{aligned} \mathbf{a}^0_{i,j} &:= 2\mathbf{q} - \mathbf{i}_{i,j} \quad 1 \le j < 2n-6, \qquad c := \cos(\frac{\pi}{n-1}), \\ \mathbf{a}^0_{i,0} &:= 2\mathbf{e}^+_i - \mathbf{i}_{i,2n-6}, \quad \mathbf{e}^+_i := (1-c)\mathbf{e}_i + c\mathbf{e}_{i+1}, \\ \mathbf{a}^0_{i,2n-6} &:= 2\mathbf{e}^-_i - \mathbf{i}_{i,0}, \qquad \mathbf{e}^-_i := (1-c)\mathbf{e}_i + c\mathbf{e}_{i-1}. \end{aligned} \tag{3}$$

This choice of mirroring is more subtle than naive mirroring commonly used for subdivision algorithms since that can yield shape deficiencies. For typical HFS constructions, the surface point $\mathbf{p}$ surrounded by n patches is a weighted average of the EV and its surrounding 1-ring of vertices with weight w^e vertices connected to the EV by an edge, w^o vertices not connected by an edge, and $\bar{w}$ for the EV. For example, for [26], $\mathbf{p}$ is the limit point of Catmull-Clark subdivision. Using the limit point weights (see Appendix) we can solve for the reflection center $\mathbf{q}$ to that the resulting center surface point $\mathbf{p}$ lies on the global boundary curve: $6\mathbf{p} = \mathbf{e}_{i-1} + 4\mathbf{e}_i + \mathbf{e}_{i+1}$. Simplification yields

$$\mathbf{m}_i := (\mathbf{e}_{i-1} + \mathbf{e}_{i+1})/2, \;\; \mathbf{g}_i := \mathbf{m}_i - \mathbf{e}_i, \tag{4}$$
$$\mathbf{q} := \mathbf{m}_i + \alpha_n \mathbf{g}_i, \quad \alpha_n := \begin{cases} 0 & n = 4, \\ 8/9 - 2\sqrt{2}/3 & n = 5, \\ 6/11 - 2\sqrt{5}/11 & n = 6. \end{cases}$$

The practically relevant values of n are $n = 4, 5, 6$, see also [27] since mirroring implies that an interior EV valence of $n = 10$ corresponds to a boundary EV valence of $n = 6$.

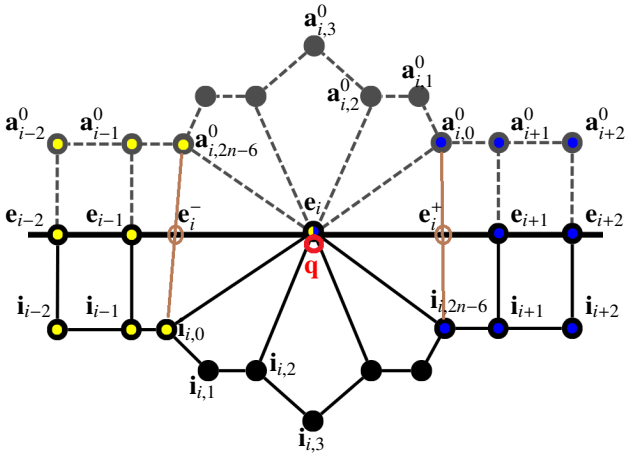


Figure 6: Boundary extension by first-order reflection: the quad structure of virtual boundary extension is marked by dotted lines; edge reflection centers are marked brown; the central mirror point $\mathbf{q}$ near $\mathbf{e}_i$ is marked ∘.

Since the influence on shape is minimal, we can choose $\mathbf{a}^1_i = \mathbf{a}^0_i$ for the second layer. The resulting singularity is not part of the retained surface. Moreover, for $n > 4$, we can simplify the formulas by setting $c := 1/2$, i.e. the value for $n = 4$. The changes to the shape are barely visible and $\alpha_4 = 0$, $\alpha_5 = 2/9$, $\alpha_6 = 4/11$.

The middle and the right column of Fig. 4c show the highlight line and mean curvature for $\mathrm{HFS}_{3,3}$ using (4).

4. Valence $n = 2$: rounded vs. sharp corners

We present a formula that allows for a gamut of corner behaviors from sharp interpolation of the corner point $\mathbf{c}$ to a rounded version that uses $\mathbf{c}$ as a B-spline control point. In the previous section, where the boundary EV valence was $n > 3$, we can generate sharp corners, simply by removing the outer quad strip and exposing the sector separating curves of the EV surrounding surface as boundary curves.

For valence $n = 2$, we consider the *regular corner* case where the corner $\mathbf{c}$ of the global mesh boundary has only two edge neighbors $\mathbf{e}_1$ and $\mathbf{e}_2$ and no interior direct neighbor. We assume that an extension layer of outer mesh vertices $\mathbf{a}_j$ has been defined, for example by mirroring. This regular setup can be displayed in the 3×3 grid of points Fig. 8

Let the valence of $\mathbf{e}_1$ be n_1 and the valence of $\mathbf{e}_2$ be n_2 and define

$$c_1 := \cos(\frac{\pi}{n_1 - 1}) \quad c_2 := \cos(\frac{\pi}{n_2 - 1}).$$

For a surface of degree is bi-3, the surface corner point $\mathbf{p}$ is defined by

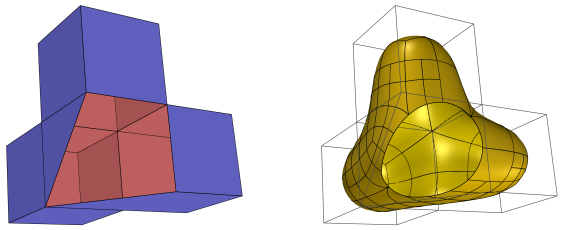
$$36\mathbf{p} = 16\mathbf{c} + 4(\mathbf{e}_1 + \mathbf{a}_2 + \mathbf{a}_4 + \mathbf{e}_2) + \mathbf{a}_1 + \mathbf{a}_3 + \mathbf{a}_5 + \mathbf{i}_1. \tag{5}$$

If we apply mirroring in the form

$$\begin{aligned} \mathbf{a}_1 &:= 2((1-c_1)\mathbf{e}_1 + c_1\mathbf{c}) - \mathbf{i}_1, \quad \mathbf{a}_2 := 2\mathbf{c} - \mathbf{e}_2, \\ \mathbf{a}_4 &:= 2\mathbf{c} - \mathbf{e}_1, \quad \mathbf{a}_5 := 2((1-c_1)\mathbf{e}_2 + c_1\mathbf{c}) - \mathbf{i}_1 \end{aligned} \tag{6}$$

then (5) simplifies to

$$36\mathbf{p} = \mathbf{a}_3 + \mathbf{e}_1(2 - 2c_1) + \mathbf{e}_2(1 - c_2) - \mathbf{i}_1 + \mathbf{c}(2c_1 + c_2 + 32)$$



(a) Control net with boundary (b) $\mathrm{HFS}_{3.3}$

(c) SubD (d) $\mathrm{HFS}_{3.2}$

Figure 7: Triangle with EV of valence $n = 4$ on boundary.

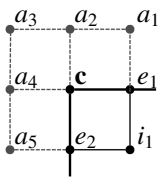


Figure 8: Virtual corner boundary extension by dotted layers.

To interpolate $\mathbf{p} = \mathbf{c}$

$$\begin{aligned}\mathbf{a}_3(0) &:= \mathbf{a}_3 \\ &= \mathbf{c}(-2c_1 - c_2 + 4) + \mathbf{e}_1(2c_1 - 2) + \mathbf{e}_2(c_2 - 1) + \mathbf{i}_1. \quad (5_{\mathbf{p}})\end{aligned}$$

If, instead we want to match the global boundary curve, i.e. interpret the global boundary polygon as C^2 degree 3 spline polygon, then $\mathbf{p} = (\mathbf{e}_1 + \mathbf{e}_2 + 4\mathbf{c})/6$ and

$$\begin{aligned}4(\mathbf{e}_1 + \mathbf{e}_2) &= 8\mathbf{c} + \mathbf{a}_3 - \mathbf{i}_1 \\ \Leftrightarrow \quad \mathbf{a}_3(1) &:= \mathbf{a}_3 = -8\mathbf{c} + 4\mathbf{e}_1 + 4\mathbf{e}_2 + \mathbf{i}_1. \quad (5_0)\end{aligned}$$

Together, $\mathbf{a}_3(w) := \mathbf{a}_3(0) + w\,6(\mathbf{e}_1 + \mathbf{e}_2 - 2\mathbf{c})$. Then $w = 0$ interpolates, $w = 1$ rounds to global mesh boundary interpreted as cubic spline. For the resulting rounded or sharp corners, see Fig. 9.

5. Sharp and semi-sharp creases

This section describes the construction of semi-sharp creases [7, 28] in the HFS setting. Subdivision, say [28, 27], relies on an infinite sequence of refinements that applies some steps of linear refinement in one direction to decrease the roundedness. In the HFS setting, a crease and the family of semi-smooth roundings of a crease must be modeled with a fixed number of patches. A simple approach is to insert mesh loops and space them to either side and have them coincide– and so make the crease sharp – or to spread apart for more rounding and standard smoothness, see Fig. 10.

5.1. Inserting loops

Inserting additional loops and coalescing them creates a parametric singularity that some CAD systems struggle with. Conversely, adding crease capabilities at the control-net level has the advantage of being able to apply the original algorithm for semi-sharp creases of various curvedness. Provided the HFS algorithm depends only on averaging and does not divide say by area – this holds for our exemplar algorithm [26] – the HFS algorithm can be applied unchanged to the limit case (cf. Fig. 11b vs. Fig. 11c). Additionally, adding control net features avoids having to break the object topologically into two as in Fig. 11a.

As a concrete example, for a degree bi 3 tensor-product surface, it suffices to add, at a distance δ, mesh lines on either side of the edge designated as a semi-sharp crease. As δ goes to zero, the transition across the three mesh lines becomes sharper and in the limit, for $\delta = 0$ the three mesh lines coincide and the crease is sharp. That is, a user can control sharpness while maintaining full local smoothness up to the limit, see Fig. 11, making the representation differentiable and without case distinction. Varying δ along the loop allows shaping, e.g. widening to feather out sharp creases. The placement of the inserted loops controls the normal of a semi-sharp crease. For EVs, as for regular vertices, we split each vertex in the loop, Fig. 14d.

5.2. Localizing a semi-sharp crease curve

Adding complete loops can be wasteful. To terminate a crease curve defined by three mesh lines, we need to shed the outer two. This is achieved by a Δ_3^1 configuration, Fig. 12a, of the control-net. The red edges in the bottom row represent the three mesh lines that define the semi-sharp crease curve, in the middle row the magenta edges represent the transition; and in the top row the blue edge represents a regular (non-crease) curve.

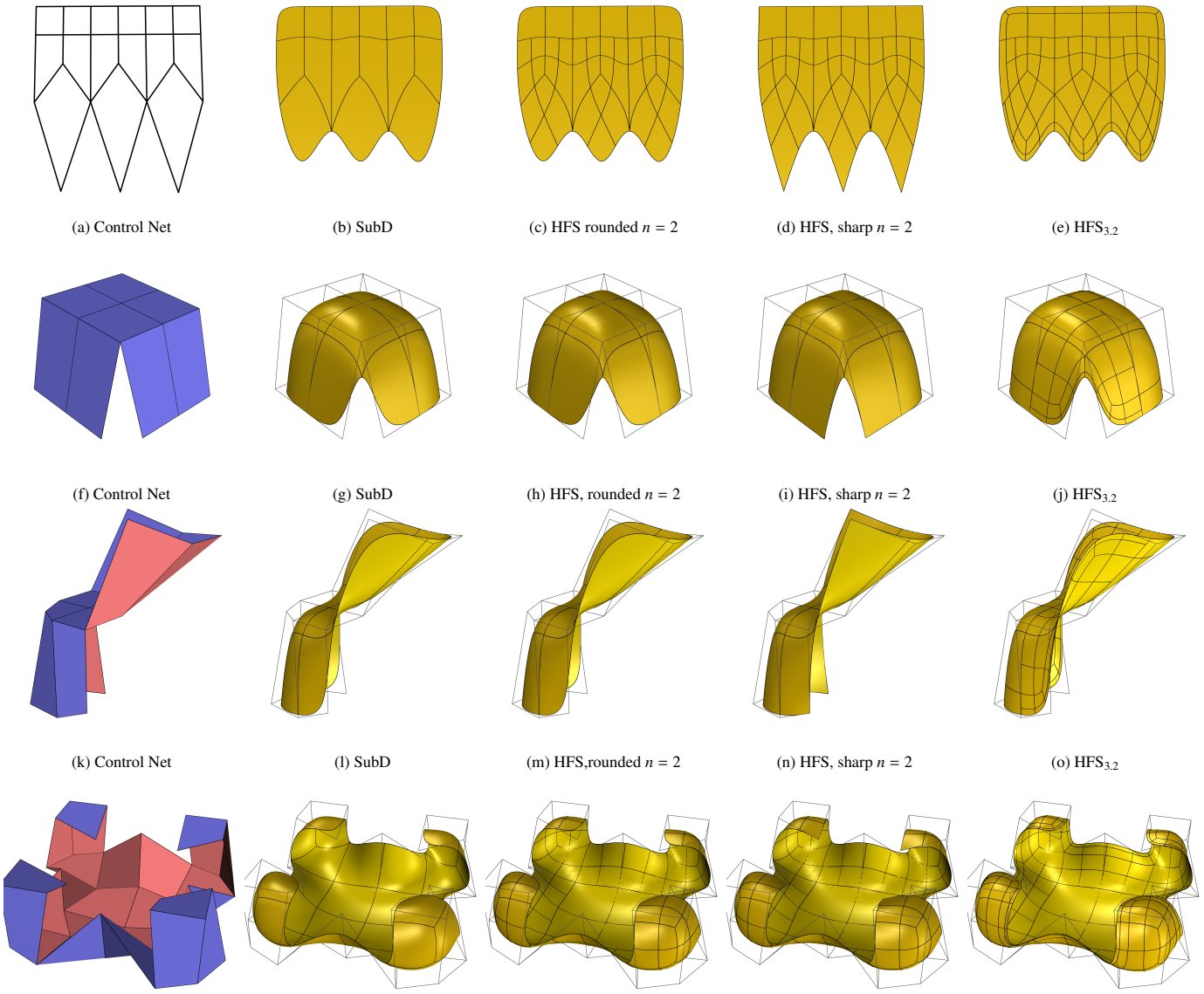
To avoid creating new algorithms or patch types we (virtually) apply Catmull-Clark subdivision [18] to the Δ_3^1 configuration, Fig. 12a. Connecting the resulting points as shown in Fig. 12c yields twin control nets of type Fig. 12b, i.e. suitable for applying [29], a G^1 smooth degree bi-5 surface. The resulting Δ_3^1 surface can be used to *locally* end or add two partial loops. The localized semi-sharp crease generated by SubD, see Fig. 13b, can so be replicated by the HFS, see Fig. 13c with good highlight lines Fig. 13d.

6. Discussion, Limitations, and Future Work

The simplest treatment of boundaries is also the most natural: to not generate surface pieces attached to the boundary, but only use rules that consistently interpret the spline setup of B-spline and HFS rules. Such 'open' HFS surfaces can be joined at a common boundary by generating overlapping global mesh boundary from both sides such that the retraction has the surfaces share only the boundary curve(s).

However, many practitioners think it more convenient to control of the global boundary curve by the global boundary polygon. The virtual control-net augmentations of $\mathrm{HFS}_{3.1}$, $\mathrm{HFS}_{3.2}$, and $\mathrm{HFS}_{3.3}$, can be seen as adding respective 0,1 or many virtual neighbors to the EV. The generalization of classical mirroring, $\mathrm{HFS}_{3.3}$, is both the most complicated construction and the weakest in the sense of not interpolating the global boundary curve. It is motivated by the desire of some modeling frameworks to exactly match one B-rep surface with one control net facet – whereas $\mathrm{HFS}_{3.1}$ and $\mathrm{HFS}_{3.2}$ add a quad strip.

When mirroring, one could consider rules that are specific to the angle formed by the EV and its neighbors $\mathbf{e}_{i-1}, \mathbf{e}_i, \mathbf{e}_{i+1}$ and



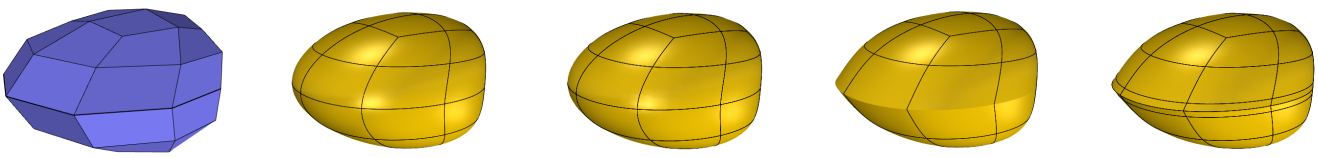
(a) Control Net (b) SubD (c) HFS rounded $n = 2$ (d) HFS, sharp $n = 2$ (e) $\text{HFS}_{3.2}$

(f) Control Net (g) SubD (h) HFS, rounded $n = 2$ (i) HFS, sharp $n = 2$ (j) $\text{HFS}_{3.2}$

(k) Control Net (l) SubD (m) HFS,rounded $n = 2$ (n) HFS, sharp $n = 2$ (o) $\text{HFS}_{3.2}$

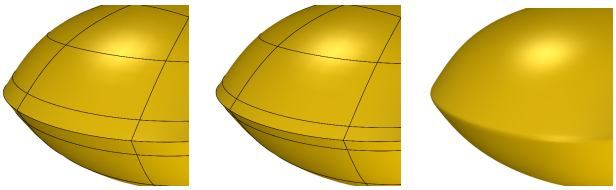
(p) Control net (q) SubD (r) HFS, rounded $n = 2$ (s) HFS, sharp $n = 2$ (t) $\text{HFS}_{3.2}$

Figure 9: Treatment of corners. The right columns show both the control net and the patch boundaries for HFS. All examples were generated in an existing CAD pipeline and modeling system.



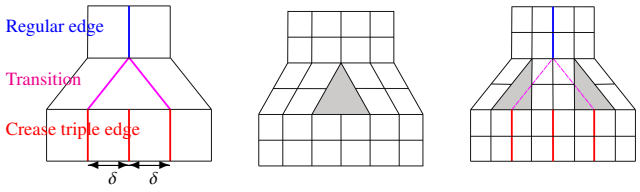
(a) control net (b) no crease SubD (c) no crease HFS (d) sharp crease SubD (e) sharp crease HFS

Figure 10: (b,c) Regular vs (d,e) creased surfaces



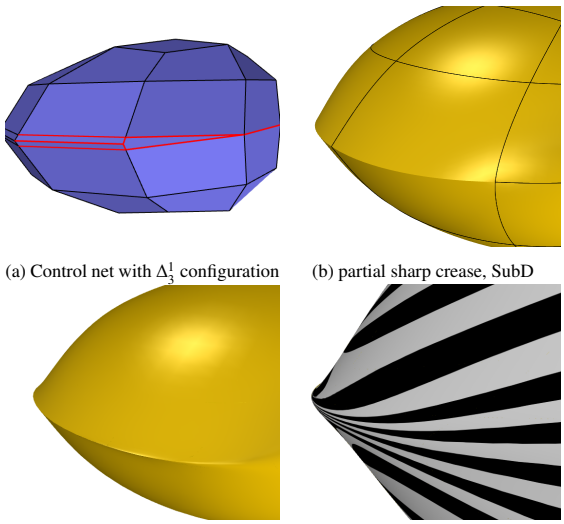
(a) 2 boundaries matched (b) $\delta = 0$ (c) $\delta = 0.1$

Figure 11: Sharp crease in C^2 surface: (a) by geometrically matching the global boundary curves g of two open objects or (b) by setting $\delta = 0$.



(a) Δ_3^1 configuration net (b) Control net for [29] (c) Refined net

Figure 12: Crease localization. (a) The Δ_3^1 surface control net layout adds two mesh lines. (b) Unified control net for T-junction surfaces according to [29]. (c) left and right-biased control-net refinement of (a) averaged to yield two control nets of type (b).



(a) Control net with Δ_3^1 configuration (b) partial sharp crease, SubD

(c) partial sharp crease $\delta = 0$ (d) Zoom on transition

Figure 13: Sharp crease ($\delta = 0$) for only a part using the transition Δ_3^1 configuration of Fig. 12a, shown as red mesh-lines in (a).

add mirrored edges by estimating the circle portion not covered, see [30, Fig 4]. It is not clear, and worth testing, what this more complex heuristic would gain.

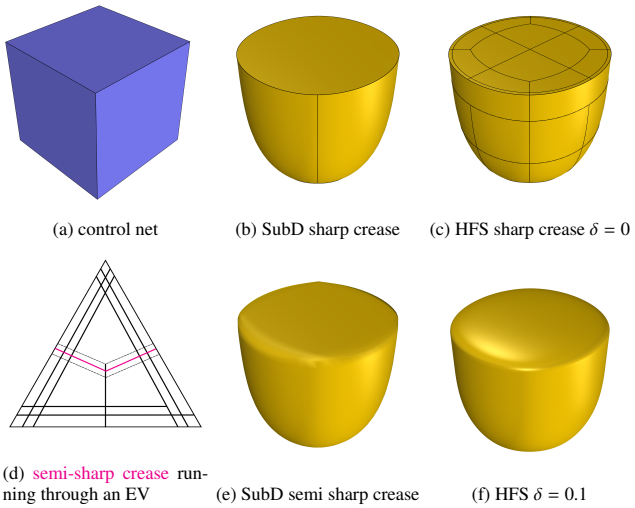
For semi-sharp creases, we did not discuss the larger framework of darts and localized creases crossing each other at EVs. We suspect that for high-end surface quality and a small footprint specialized patch constructions may be needed to avoid complexity in virtual control-net extensions.

7. Conclusion

We successfully added boundary and semi-sharp crease options – that are familiar from subdivision surfaces – to high-end, free-form spline surfaces (HFS) that, being CAD-compatible, use only a finite number of polynomial pieces. The rules and their derivations are not overly complicated, which is a practical advantage, given that they leverage the often complicated but implemented construction rules of HFS.

The practical impact and need for adding these features to an existing CAD compatible representations is easily underestimated – and so is the complexity in the finite setting. Localization of semi-sharp crease rules is both important in practice and non-trivial in the presence of an otherwise G^2 HFS. Similarly, offering sharpness-parameterized corner rounding is a practically useful addition. The trade-off for boundary rules is responsive to requests of a b-rep with a minimal layout. We argued that such minimality is inconsistent with boundary interpolation when dealing with a finite number of polynomial pieces and discussed the trade-off among three options.

Acknowledgements The work benefitted from the designer perspective of Toni Azzella. Jörg Peters acknowledges the support of the Fields Institute, Toronto, Canada.



(a) control net (b) SubD sharp crease (c) HFS sharp crease $\delta = 0$

(d) semi-sharp crease running through an EV (e) SubD semi sharp crease (f) HFS $\delta = 0.1$

Figure 14: Sharp and semi-sharp creases in SubD and HFS.

References

[1] C. de Boor, A Practical Guide to Splines, Springer, 1978.

[2] T. W. Sederberg, J. Zheng, D. Sewell, M. Sabin, Non-uniform recursive subdivision surfaces, in: Proceedings of the 25th annual conference on Computer Graphics and Interactive Techniques, 1998, pp. 387–394.

[3] T. W. Sederberg, J. Zheng, A. Bakenov, A. Nasri, T-splines and t-nurccs, ACM transactions on graphics (TOG) 22 (3) (2003) 477–484.

[4] Pixar, Opensubdiv, https://github.com/PixarAnimationStudios/OpenSubdiv, version 3.7.0, PIxar Animation Studios, Rendreman (2013).

[5] J. Kosinka, M. A. Sabin, N. A. Dodgson, Subdivision surfaces with creases and truncated multiple knot lines, in: Computer Graphics Forum, Vol. 33, Wiley Online Library, 2014, pp. 118–128.

[6] M. A. Sabin, C. Fellows, J. Kosinka, Cad model details via curved knot lines and truncated powers, Computer-Aided Design 143 (2022) 103137.

[7] J. Peters, C^1-surface splines, SIAM Journal on Numerical Analysis 32 (2) (1995) 645–666.

[8] T. Nguyen, K. Karčiauskas, J. Peters, C^1 finite elements on non-tensor-product 2d and 3d manifolds, Applied Mathematics and Computation 272 (1) (2016) 148–158.

[9] A. Collin, G. Sangalli, T. Takacs, Analysis-suitable G^1 multi-patch parametrizations for C^1 isogeometric spaces, Computer Aided Geometric Design 47 (2016) 93–113.

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

[11] A. Blidia, B. Mourrain, G. Xu, Geometrically smooth spline bases for data fitting and simulation, Computer Aided Geometric Design 78 (2020) 101814.

[12] J. Peters, K. Lo, K. Karčiauskas, Algorithm 1032: Bi-cubic splines for polyhedral control nets, ACM Transactions on Mathematical Software 49 (1) (2023) 1–12.

[13] J. Peters, Splines for meshes with irregularities, The SMAI journal of computational mathematics S5 (2019) 161–183.

[14] T. Várady, P. Salvi, G. Karikó, A multi-sided Bézier patch with a simple control structure, Comput. Graph. Forum 35 (2) (2016) 307–317.

[15] G. J. Hettinga, J. Kosinka, A multisided C^2 B-spline patch over extraordinary vertices in quadrilateral meshes, Computer-Aided Design 127 (2020) 102855.

[16] M. Vaitkus, P. Salvi, T. Várady, Interior control structure for generalized bézier patches over curved domains, Computers & Graphics 121 (2024) 103952.

[17] Rhino, Subd objects, https://docs.mcneel.com/rhino/8/help/en-us/seealso/sak_subd.htm (accessed January 2026).

[18] E. Catmull, J. Clark, Recursively generated B-spline surfaces on arbitrary topological meshes, Computer-Aided Design 10 (1978) 350–355.

[19] Blender Online Community, Blender - a 3D modelling and rendering package, Blender Foundation, Blender Institute, Amsterdam (2025). URL http://www.blender.org

[20] Pixar, Renderman docs, https://rmanwiki-26.pixar.com/space/REN26/19661421/Subdivision+Surfaces (accessed January 2026).

[21] A. H. Nasri, M. A. Sabin, Taxonomy of interpolation constraints on recursive subdivision surfaces, The Visual Computer 18 (5-6) (2002) 382–403.

[22] H. Biermann, A. Levin, D. Zorin, Piecewise smooth subdivision surfaces with normal control, in: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, 2000, pp. 113–120.

[23] J. Kosinka, M. Sabin, N. Dodgson, Creases and boundary conditions for subdivision curves, Graphical Models 76 (5) (2014) 240–251.

[24] A. Levin, Combined subdivision schemes for the design of surfaces satisfying boundary conditions, Computer Aided Geometric Design 16 (5) (1999) 345–354.

[25] D. Doo, M. Sabin, Behaviour of recursive division surfaces near extraordinary points, Computer-Aided Design 10 (1978) 356–360.

[26] K. Karčiauskas, J. Peters, Minimal bi-6 G^2 completion of bicubic spline surfaces, Computer Aided Geometric Design 41 (2016) 10–22.

[27] D. Lacewell, B. Burley, Exact evaluation of catmull-clark subdivision surfaces near b-spline boundaries, Journal of Graphics Tools 12 (3) (2007) 7–15.

[28] T. DeRose, M. Kass, T. Truong, Subdivision surfaces in character animation, in: Seminal Graphics Papers: Pushing the Boundaries, Volume 2, 2023, pp. 801–810.

[29] K. Karčiauskas, J. Peters, High quality refinable G-splines for locally quad-dominant meshes with T-gons, Computer Graphics Forum 38 (5) (2019) 151–161.

[30] D. Zorin, D. Kristjansson, Evaluation of piecewise smooth subdivision surfaces, The Visual Computer 18 (5-6) (2002) 299–315.

[31] M. Halstead, M. Kass, T. DeRose, Efficient, fair interpolation using catmull-clark surfaces, in: Proceedings of the 20th annual conference on Computer graphics and interactive techniques, 1993, pp. 35–44.

Appendix

For the Ansatz

$$\begin{aligned}\mathbf{p} &= \bar{w}\,\mathbf{e}_i + w^o(\mathbf{a}^0_{i-1} + \mathbf{a}^0_{i+1} + \mathbf{i}_{i-1} + \mathbf{i}_{i+1}) \\ &+ w^e(\mathbf{e}_{i-1} + \mathbf{e}_{i+1}) + w^e \sum_{\substack{j=0 \\ even}}^{N}(\mathbf{i}_{i,j} + \mathbf{a}^0_{i,j}) + w^o \sum_{\substack{j=1 \\ odd}}^{N}(\mathbf{i}_{i,j} + \mathbf{a}^0_{i,j}).\end{aligned} \tag{7}$$

substituting the reflection rules (3) to eliminate $\mathbf{a}^0_{ij}$ yields

$$\begin{aligned}\mathbf{p} &= \mathbf{e}_i\bar{w} + w^e(4\mathbf{e}_i(1-c) + 2\mathbf{e}_{i+1}c + \mathbf{e}_{i+1} + 2\mathbf{e}_{i-1}c + \mathbf{e}_{i-1} \\ &+ \mathbf{q}(2n-8)) + w^o(2\mathbf{e}_{i+1} + 2\mathbf{e}_{i-1} + \mathbf{q}(2n-6)).\end{aligned} \tag{8}$$

The weights for the Catmull-Clark subdivision limit point are [31]

$$w^o := \frac{1}{2\,(n-1)\,(2n+3)} \quad w^e := \frac{2}{(n-1)\,(2n+3)} \quad \bar{w} := \frac{2\,(n-1)}{2n+3}.$$

Placing $\mathbf{p}$ on the global boundary curve, $6\mathbf{p} = \mathbf{e}_{i-1} + 4\mathbf{e}_i + \mathbf{e}_{i+1}$, and solving for $\mathbf{q}$ yields

$$\begin{aligned}6(5n-19)\mathbf{q} = \;& \mathbf{e}_i(48c - 4n^2 + 28n - 72) \\ &+ \mathbf{e}_{i+1}(-24c + 2n^2 + n - 21) + \mathbf{e}_{i-1}(-24c + 2n^2 + n - 21)\end{aligned} \tag{9}$$

and hence Eq. (4).