Optimized Refinable Surface Enclosures

Jörg Peters, Xiaobin Wu, University of Florida

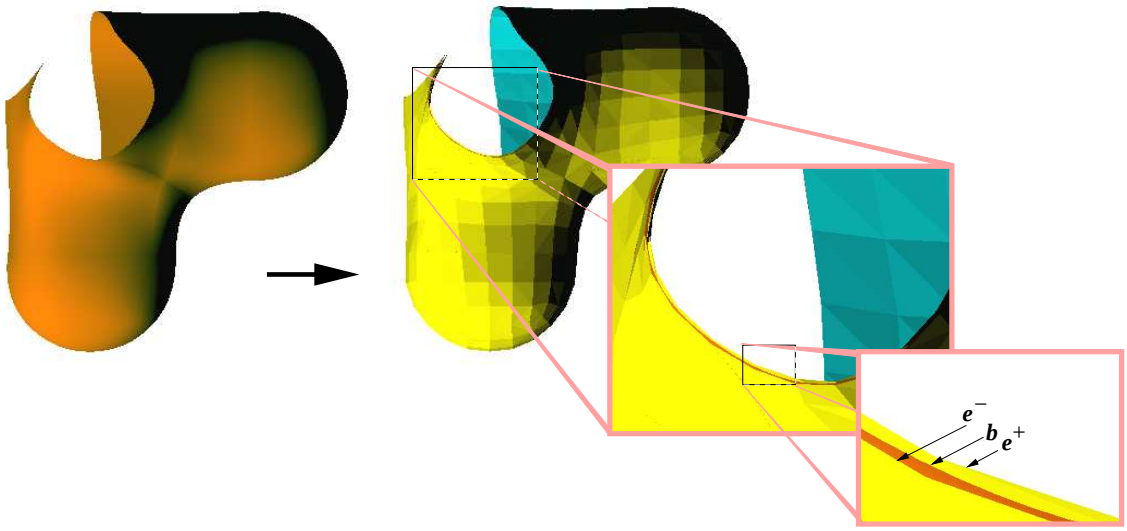

Figure 1: (*left*:) a piecewise bicubic spline surface $\mathbf{b}$. (*right of arrow*:) Zooming in on the optimized surface enclosure of $\mathbf{b}$. The enclosure consists of the inner triangulation $\mathbf{e}^+$ and outer, matching triangulation $\mathbf{e}^-$. The triangulation sandwiches $\mathbf{b}$ so tightly that the surface is all but invisible in a full view (*top right*) and appears as the dark curved strip in the cut through the surface after repeated magnification (*bottom right*).

Abstract

An enclosure of a composite spline surface is a pair of simpler approximations that sandwich the surface. In particular, we are interested in efficiently constructing two triangulations, so that matched triangle pairs enclose a piece of the curved surface. The width of the enclosure, i,e. the distance between inner and outer hull, can be easily measured, because it is taken on at a vertex. Enclosures are therefore approximate implicitizations with known error; such bounding constructs are useful to support, say, collision detection, re-approximation for format conversion, meshing with tolerance, or silhouette detection.

The surface enclosure developed in this paper is effective, because it is optimized and refinable: an optimization specific to a given geometry representation is done off-line and tabulated once and for all. Moreover, given an enclosure of a smooth surface, the number and location of refinement steps can be announced that guarantee that the distance to the object falls below a given tolerance, because the width generically shrinks to 1/4 under subdivision at midpoints.

CR Categories: I.3.5 [surface representation, splines]: I.3.6—graphics data structures

Keywords: curved spline surface enclosure, 2-sided bounds, triangulated surface enclosure, approximate implicitization

1 Motivation

Measuring closeness to the silhouette or determining the distance between objects are fundamental issues in computer graphics. While efficient algorithms exist for piecewise linear surfaces with not too many pieces, objects in B-spline, Bézier or generalized subdivision representation pose numerical and implementation challenges due to the curved geometry. Naive linearization of a curved surface, say triangulation by sampling, incurs an error that is difficult to quantify and is particularly noticeable when estimating the range of the normal. To extend existing techniques, we therefore developed a two-sided, piecewise linear approximation to curved surfaces that is cheap to compute, has a near optimal L^∞ error and is refinable for adaptive multiresolution.

The enclosures are based on recent advances in bounding con-

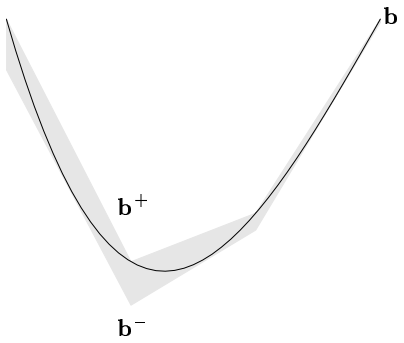

Figure 2: A cubic Bézier segment and its optimized $n = 3$-piece linear enclosure.

structs for *polynomial functions* [16]. For a function $\mathbf{b}$, an enclosure is an explicit two-sided approximation $\mathbf{b}^+, \mathbf{b}^-$ so that $\mathbf{b}^- \leq \mathbf{b} \leq \mathbf{b}^+$ over the domain of interest, where $\mathbf{b}^+$ and $\mathbf{b}^-$ are, for example, piecewise linear functions as shown in Figure 2. We make the new techniques practically useful for computer graphics by extending them to *surfaces*. This extension from function enclosures to surface enclosures is nontrivial and requires several improvements of the original approach, e.g. the separation of polynomial degree and the number of pieces by which the polynomial piece is approximated and the combined consideration of position and normal of the surface.

Since the basic ideas can be explained in the curve case, we carefully develop the construction first for functions in one variable (Section 2) and then for planar curves (Section 3) in Bézier form. The approach bootstraps, almost without further complication, to the surface case through tensoring (Sections 4 and 5). Generalizations to other surface representations and applications are briefly discussed in the final section. Before we start with the details, we contrast enclosures with other commonly used bounding constructs.

1.1 Bounding constructs

The enclosure of a geometric object is a *bounding construct*, consisting typically of two sheets, such that each sheet is guaranteed not to intersect the object, i.e. each sheet lies to 'one side' of the object. For example, a surface without boundary can be enclosed by an inner and an outer triangulation. In this section, we contrast enclosures with other bounding constructs and enclosing functions.

If we distinguish between elementary bounding constructs and hierarchical structures that employ these elementary bounding constructs as their oracles, enclosures fall into the category of elementary bounding constructs. A gallery of elementary bounding constructs is shown in Figure 3. That is, enclosures add to the arsenal of axis-aligned bounding boxes (AABB), oriented bounding boxes (OBB), quantized bounding boxes also called 'k-dops' or discrete orientation polytopes (convex polytopes whose facets are determined by halfspaces whose outward normals come from a small fixed set of k orientations) [5, 12, 13], fat arcs [22], convex hulls, bounding spheres and minimal enclosing ellipsoids [24]. Publications [8] and [14] give a good overview of how elementary bounding constructs are used in the context of hierarchical interference detection (for space partitioning methods see e.g. [1]): simpler constructs like AABBs and spheres provide fast rejection tests in sparse arrangements, while more expensive k-dops and OBBs perform better on complex objects in close proximity. Enclosures with adaptive resolution promise to outperform other bounding constructs for curved, non-polyhedral objects (cf. Figure 4) due to their two construction principles that will be explained in detail in Section 2:

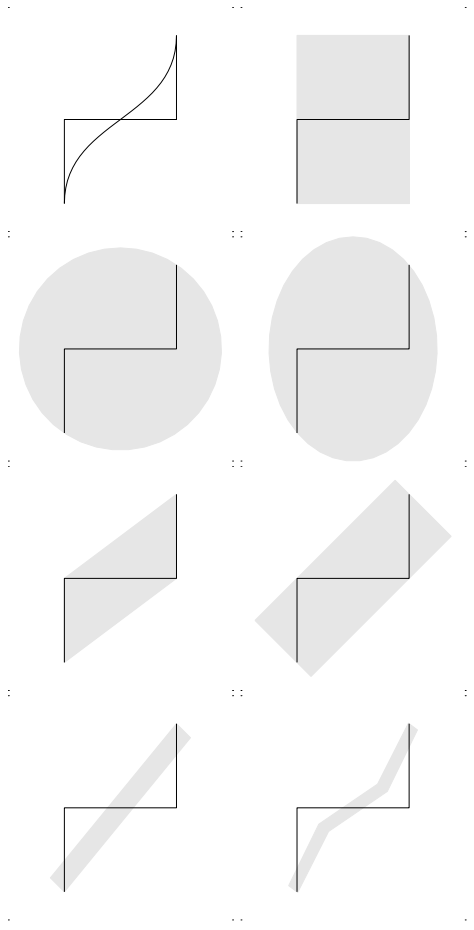

Figure 3: Gallery of bounding constructs and control polygon of a cubic curve: (*left*) curve and control polygon, bounding circle, convex hull, here equal to an 8-dop, generalized fat arc, (*right*) axis-aligned box, bounding ellipse, oriented bounding box, 3-linear enclosure.

- Representation-specific pre-optimization: expensive optimization specific to a given geometry representation is done off-line and tabulated once and for all,
- Predictive refinability: given an enclosure, the number and location of refinement steps can be announced that guarantee that the distance to the object falls below a given tolerance.

The theory of function enclosures has its roots in bounds on the distance of piecewise polynomials to their Bézier or B-spline control net [17, 19]. Compared to these constructions, enclosures yield dramatically tighter bounds for the underlying functions since they do not enclose the control polygon. To emphasize the advantage of sandwiching the object, compared to sampling at a finite number of points and connecting by straight line segments (or flat triangles for surfaces), Figure 7 compares a curve approximation consisting of three linear segments with a certain average of the upper and lower enclosure bounds. — Farin [7] shows that for rational Bézier–curves, the convex hull property can be tightened to the convex hull of the first and the last control point and the *weight* points, which are points on the legs of the control polygon associated with the weights of the rational Bézier representation. A similar idea can

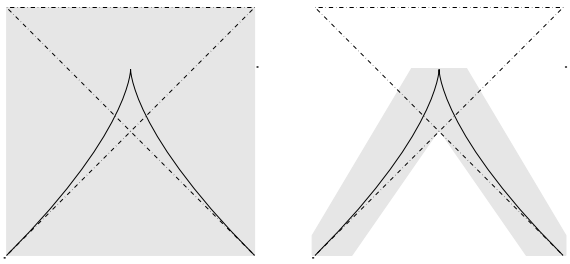

Figure 4: Bounding constructs (grey regions) based on the control polygon (dashed) of a cubic Bézier curve (solid). (*left*) The oriented bounding box, axis-aligned bounding box, quantized hull and convex hull all have identical extent. (*right*) A 3-piece linear enclosure with special treatment of the endpoints.

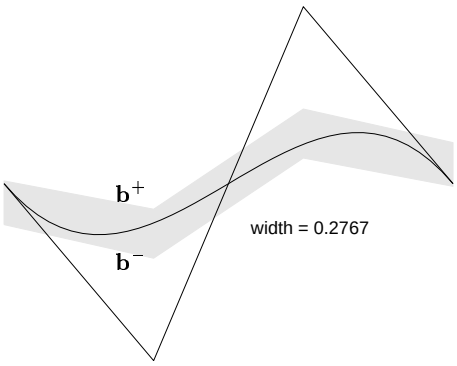

Figure 5: A cubic Bézier segment with coefficients $0, -1, 1, 0$. The control polygon exaggerates the curve far more than the *grey* 3-piece linear enclosure bounded by $\mathbf{b}^+, \mathbf{b}^-$.

also be leveraged for enclosures. — Approximation theory has long considered the problems of *one-sided approximation* and *two-sided approximation*. Based on Buck's seminal "Applications of Duality in Approximation Theory" [2], DeVore [6] established the close relation of one-sided approximation to quadrature formulas with nonnegative coefficients and gave a Remez-type algorithm for determining a unique solution. The rich body of literature and the best algorithms for one-sided approximation have been collected in the monograph [18]. The algorithms have in common that the only termination guarantee is that a subsequence must exist that converges. Already in one variable 'convergence is generally very slow' ([18], page 181). Enclosures do not attempt solve the hard problems of one-sided approximation as formulated by approximation theorists: to establish optimality and uniqueness over all sufficiently smooth functions. Rather we are satisfied with an efficiently computable approximation with a small, quantifiable error. The underlying techniques, however, come in handy for determining bounds when pre-optimizing certain functions $\lfloor A_i^d \rfloor_n$ and $\lceil A_i^d \rceil_n$ crucial to the construction of enclosures (see 2).

Surface simplification for triangulated surfaces has been modified to generate (locally) inner and outer hulls [4, 20]. This requires solving a sequence of linear programs at runtime and applies to already triangulated surfaces. – Kobbelt [15] uses a secant based approach to construct OBBs for subdivision curves or surfaces using a min–max criterion. This requires the evaluation of several points on the curve or surface. – Hu et al.[9, 10, 11, 23] promote the use of *interval spline representation* (see Farouki and Sederberg [21]) for tolerancing, error maintenance and data fitting. The key ingredient of this use of interval arithmetic are AABBs based on the positivity and partition of unity property of the B-splines. Enclosures complement this work by offering tighter two-sided bounds.

2 An example in one variable

To exhibit the fundamental ideas, we consider the polynomial $\mathbf{b}$ of degree $d = 3$, in Bézier representation with coefficients $b_0 = 0$, $b_1 = -1, b_2 = 1, b_3 = 0$:

$$\mathbf{b}(u) := \sum_{i=0}^{d} b_i B_i^d(u), \qquad B_i^d(u) := \frac{d!}{(d-i)!i!}(1-u)^{d-i}u^i.$$

The goal is to build an optimized refinable enclosure for $\mathbf{b}$ on the interval $U = [0..1]$ with n-piece linear upper and lower bounds as shown in Figure 5 for $n = 3$. Specifically, we want to enclose $\mathbf{b}$ from above and from below by piecewise linear functions $\mathtt{h} \in \mathcal{H}^n$ that are linear combinations of the hat functions with break points at i/n:

$$H_i^n(t) := \begin{cases} nt - (i-1) & \text{if } \frac{i-1}{n} \le t \le \frac{i}{n} \\ (i+1) - nt & \text{if } \frac{i}{n} \le t \le \frac{i+1}{n} \\ 0 & \text{else.} \end{cases}$$

Let $\lfloor B_i^d \rfloor_n$ be a function in $\mathcal{H}^n$ that bounds B_i^d from below on the interval U and $\lceil B_i^d \rceil_n$ a function that bounds B_i^d from above. Then on U one easily checks that

$$\begin{aligned} &\sum_{i=0}^{d} \lfloor B_i^d \rfloor_n \max\{b_i, 0\} + \lceil B_i^d \rceil_n \min\{b_i, 0\} \\ &\qquad \le \mathbf{b} = \sum_{i=0}^{d} B_i^d b_i \le \qquad (1) \\ &\sum_{i=0}^{d} \lfloor B_i^d \rfloor_n \min\{b_i, 0\} + \lceil B_i^d \rceil_n \max\{b_i, 0\}. \end{aligned}$$

However, this bounding construct is not useful since $\mathrm{diff}(\mathbf{b}) := \sum_i (\lceil B_i^d \rceil_n - \lfloor B_i^d \rfloor_n)\,|b_i|$ is not invariant under addition of constant functions: if $\mathtt{k} >> |b_i|$ for all i, then $\mathrm{diff}(\mathbf{b} + \mathtt{k}) \approx \mathtt{k} \sum_i \lceil B_i^d \rceil_n - \lfloor B_i^d \rfloor_n$ increases linearly with $\mathtt{k}$.
A better approach is to restate

$$\mathbf{b} - \ell(\mathbf{b}) = A_1^d \Delta_1^2 \mathbf{b} + A_2^d \Delta_2^2 \mathbf{b},$$

where

$$\ell(\mathbf{b})(u) := b_0(1-u) + b_3 u, \quad \Delta^2\mathbf{b} := \begin{bmatrix} \Delta_1^2\mathbf{b} \\ \Delta_2^2\mathbf{b} \end{bmatrix} := \begin{bmatrix} b_0 - 2b_1 + b_2 \\ b_1 - 2b_2 + b_3 \end{bmatrix},$$
$$A_1^d := -(2B_1^d + B_2^d)/3, \quad A_2^d := -(B_1^d + 2B_2^d)/3.$$

The polynomials A_i^d of degree d are called *antidifference functions* and the restatement is easily checked by observing that $\Delta_i^2 A_j^d = 1$ for $i = j$ and 0 else. This orthogonality and $A_i^d(0) = 0 = A_i^d(1)$ uniquely define A_i^d. Applying the idea underlying (1) to $\mathbf{b} - \ell(\mathbf{b})$ and then adding $\ell(\mathbf{b})$ to all three terms of the result yields

$$\begin{aligned} \mathbf{b}^+ &:= \ell(\mathbf{b}) + \sum_{i=1}^{d-1} \lfloor A_i^d \rfloor_n \ \min\{\Delta_i^2\mathbf{b}, 0\} + \lceil A_i^d \rceil_n \ \max\{\Delta_i^2\mathbf{b}, 0\} \\ &\qquad \ge \mathbf{b} \ge \qquad (2) \\ \mathbf{b}^- &:= \ell(\mathbf{b}) + \sum_{i=1}^{d-1} \lfloor A_i^d \rfloor_n \ \max\{\Delta_i^2\mathbf{b}, 0\} + \lceil A_i^d \rceil_n \ \min\{\Delta_i^2\mathbf{b}, 0\}, \end{aligned}$$

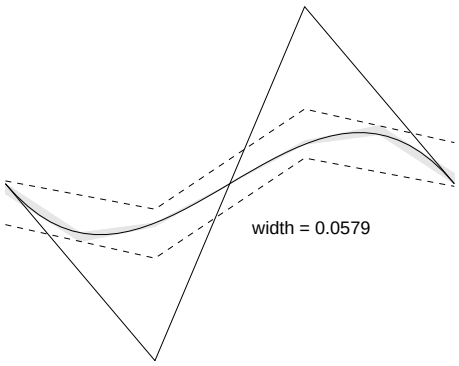

Figure 6: Enclosure of the cubic Bézier curve segment with coefficients $0, -1, 1, 0$ after one subdivision at the midpoint. The width of the enclosure (*grey*) is guaranteed to be at most 1/4th of the unsubdivided enclosure (*dashed*).

where $\lfloor A_i^d \rfloor_n$ is an n-piece linear function in $\mathcal{H}^n$ that bounds A_i^d from below and $\lceil A_i^d \rceil_n$ bounds from above; $\mathbf{b}^-$ and $\mathbf{b}^+$ form the lower and upper boundary of the grey $n = 3$-linear enclosure in Figure 5. Now, for any norm $\|\cdot\|$,

$$\text{width}_U(\mathbf{b}) := \|\mathbf{b}^+ - \mathbf{b}^-\| = \|\sum_{i=1}^{d-1} (\lceil A_i^d \rceil_n - \lfloor A_i^d \rfloor_n)\, |\Delta_i^2 \mathbf{b}|\|$$

is invariant under addition of constant and linear terms to $\mathbf{b}$, a prerequisite for affine invariance.

What is the cost of computing such an enclosure? Since $\mathbf{b}$ is of order four, we need to compute two second differences of its coefficients. Since either $\Delta_i^2 \mathbf{b} < 0$ or $\Delta_i^2 \mathbf{b} \geq 0$, each second difference contributes once to each bound by scaling $\lfloor A_i^d \rfloor_n$, respectively $\lceil A_i^d \rceil_n$. $\lfloor A_i^d \rfloor_n$ and $\lceil A_i^d \rceil_n$ are represented by their values at i/n, $i = 0, \ldots, n$. Since A_1^d and A_2^d are symmetric with respect to $1/2$, we need to store only $2n+2$ numbers. In particular, for $d = n = 3$,

$$\begin{bmatrix} \lfloor A_1^3 \rfloor_3 \\ \lceil A_1^3 \rceil_3 \end{bmatrix} \sim= \begin{bmatrix} -0.07723457 & \alpha & \beta & -0.00513580 \\ 0.0 & \gamma & \delta & 0.0 \end{bmatrix}$$

where $\alpha := 0.22992593$, $\beta := 0.01116049$, $\gamma := 0.29629630$, $\delta := 0.03703704$. The decisive point is that these numbers can be precomputed and tabulated once and for all to optimize the enclosure of the function A_1^d with respect to, say the L^∞ norm, as shown in Figure 2. In fact, we can do better by not just minimizing the largest of the differences $\lceil A_1^d \rceil_n(i/n) - \lfloor A_1^d \rfloor_n(i/n)$ but subsequently each of the remaining differences, keeping the higher differences fixed. For low degrees, values at the break points resulting in tight bounds can be obtained by inspection or by a simple, iterative, divide-and-conquer approach [16]. This is the *representation-specific pre-optimization* mentioned earlier. The enclosure of $\mathbf{b}$ (Figure 5) then requires 16 multiplications and 16 additions, comparable to computing a 4-direction quantized hull or 8-dop (see 1.1).

If $\mathbf{b}^1$ and $\mathbf{b}^2$ are the left and the right pieces of $\mathbf{b}$ resulting from de Casteljau evaluation at $1/2$ and if $\|\Delta^2 \mathbf{b}\|_\infty := \max\{|\Delta_1^2 \mathbf{b}|, |\Delta_2^2 \mathbf{b}|\}$, then it is well-know (see e.g. Lemma 6.1 of [17]) that

$$\max\{\|\Delta^2 \mathbf{b}^1\|_\infty, \|\Delta^2 \mathbf{b}^2\|_\infty\} \leq \frac{1}{4} \|\Delta^2 \mathbf{b}\|_\infty .$$

That is, one subdivision cuts the width to *a quarter or less*. The refined, optimized enclosure is shown in Figure 6. Conversely, we

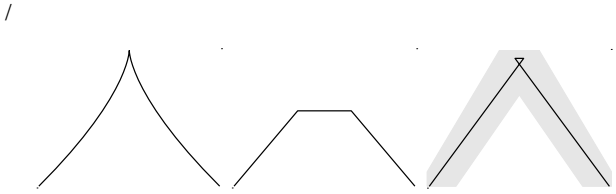

Figure 7: A cubic curve $\mathbf{b}$, a 3-piece sampled approximation of the curve, the midpath $\overline{\mathbf{b}}$ of an $n = 3$-piece enclosure of the curve.

can predict the number of subdivision steps σ needed to reduce the width below a given tolerance ϵ. Since $\text{width}_U(\mathbf{b}) \leq W := \|\sum_{i=1}^{d-1} (\lceil A_i^d \rceil_n - \lfloor A_i^d \rfloor_n)\| \|\Delta^2 \mathbf{b}^2\|_\infty$ we have $\sigma = \lceil \log_4 \frac{W}{\epsilon} \rceil$.

We define the *midpath*, $\overline{\mathbf{b}}$, of $\mathbf{b}$ as the n-piece linear function in $\mathcal{H}^n$ with values

$$\overline{\mathbf{b}}(\frac{i}{n}) := \begin{cases} \frac{1}{2}(\mathbf{b}^+ + \mathbf{b}^-)(\frac{i}{n}) & \text{if } 0 < i < n, \\ b_i & \text{if } i = 0 \text{ or } i = n. \end{cases}$$

The choice for $i = 0$ and $i = n$ guarantees that midpaths of continuously joined Bézier pieces match up at their endpoints. The distance between the polynomial $\mathbf{b}$ and $\overline{\mathbf{b}}$ on the interval $[\frac{i}{n} .. \frac{i+1}{n}]$ is bounded by the linear average of the distances at the endpoints; and these distances are evidently bounded by

$$|\mathbf{b} - \overline{\mathbf{b}}|(\frac{i}{n}) \leq \frac{\epsilon_i}{2} (\mathbf{b}^+ - \mathbf{b}^-)(\frac{i}{n})$$

where $\epsilon_i = 2$ for $i = 0$ or $i = n$ and $\epsilon_i = 1$ otherwise.

Finally, we note that we can bound the derivative of $\mathbf{b}$ in the same manner and with the same number of $n = 3$ linear pieces:

$$(\mathbf{b}')^+ := \ell(\mathbf{b}') + \sum_{i=1}^{d-1} \lfloor A_i^{d-1} \rfloor_n \ \min\{\Delta_i^2 \mathbf{b}', 0\} + \lceil A_i^{d-1} \rceil_n \ \max\{\Delta_i^2 \mathbf{b}', 0\}.$$

Since the derivative $\mathbf{b}'$ of $\mathbf{b}$ is of degree $d = 2$ there is only one function A_1^2 to bound and all we need are the numbers

$$\begin{bmatrix} \lfloor A_1^2 \rfloor_3 \\ \lceil A_1^2 \rceil_3 \end{bmatrix} \sim= \begin{bmatrix} -0.02777780 & -0.25 & -0.25 & -0.02777780 \\ 0.0 & -0.2\overline{2} & -0.2\overline{2} & 0.0 \end{bmatrix}$$

to assemble $\overline{\mathbf{b}'}$. This will allow us to associate a midnormal with every breakpoint of the midpath.

3 Constructing planar curve enclosures

Since both the x and the y component of a planar curve $\mathbf{x}$ provide an upper and a lower bound, we obtain four segments

$$\begin{bmatrix} x^+ \\ y^+ \end{bmatrix}, \begin{bmatrix} x^+ \\ y^- \end{bmatrix}, \begin{bmatrix} x^- \\ y^+ \end{bmatrix}, \begin{bmatrix} x^- \\ y^- \end{bmatrix}$$

for each interval between breakpoints (see Figure 8, *top*). A certain 'union' of these combinations of function bounds encloses the curve. A simple way to give some structure to this mess of line segments, is to observe that, due to linearity, each piece of the enclosure is a convex combination of consecutive point enclosures $\square_i$, $\square_{i+1}$, where $\square_j$ has the four vertices

$$\square_j \sim \begin{bmatrix} x^+ \\ y^+ \end{bmatrix}(\frac{j}{n}), \begin{bmatrix} x^+ \\ y^- \end{bmatrix}(\frac{j}{n}), \begin{bmatrix} x^- \\ y^+ \end{bmatrix}(\frac{j}{n}), \begin{bmatrix} x^- \\ y^- \end{bmatrix}(\frac{j}{n}).$$

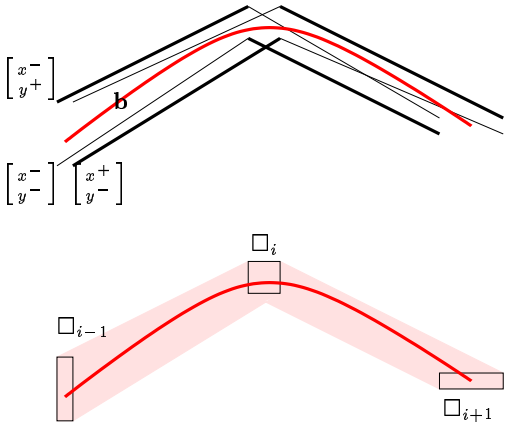

Figure 8: *top:* The curve $\mathbf{b}$ is bounded by a 'union' of four component enclosures. The extreme, outermost components that stay to one side of the curve, are emphasized as fat line segments. Note the gap and the intersection between consecutive extreme segments. *bottom:* The bounding region is equivalently generated as the piecewise linear combination of point enclosures (axis-aligned rectangles) $\square_i$.

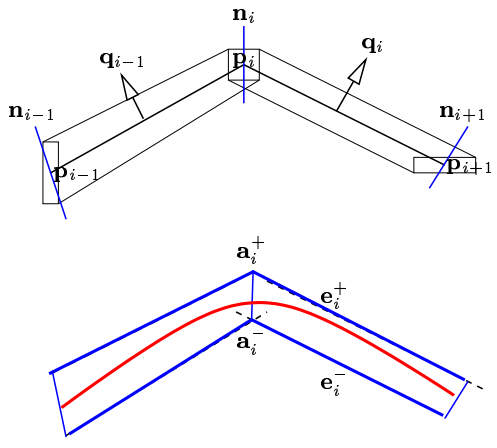

Figure 9: *top:* Anchor points $\mathbf{p}_j$, normals $\mathbf{n}_j$ and directions $\mathbf{q}_j$ for identifying extreme segments of the enclosure. *bottom:* Antipodal points $\mathbf{a}_i$ and enclosure pieces $\mathbf{e}_i$.

That is, a *point enclosure* $\square_j$ is an axis-parallel rectangle or box. (Figure 8 bottom). Differently put, the function enclosures directly yield an enclosure *in interval Bézier representation* (see Farouki and Sederberg [21]).

These interval enclosures, however, have two shortcomings: multiplicity, and intersections or gaps. By keeping information on all four components, interference checking between two interval objects would require 16 intersection tests. Moreover, the piecewise linear outer bounds have more pieces or need to be trimmed due to the intersections and gaps between adjacent pieces (fat lines in Figure 8, *top*).

3.1 Gaps and Intersections

To address the problem of gaps and intersections, we associate, with each local parameter i/n, a point $\mathbf{p}_i$ that lies in *all* point enclosures associated with that location (there may be two point enclosures of differing size if $i \in \{0, n\}$, i.e. the point enclosure corresponds to the end of one curve segment and the start of another and the two do not meet C^2). We call the point, *anchor point*, because we have in mind to attach a line segment to it with direction $\mathbf{n}_i$, roughly normal to the curve (c.f. Figure 9 top). The two endpoints $\mathbf{a}_i^+$ and $\mathbf{a}_i^-$ of the line segment will be called *antipodal points* and will serve as the vertices of the two sheets $\mathbf{e}_i^+$ and $\mathbf{e}_i^-$ of the curve enclosure (c.f. Figure 9 bottom). If the Bézier pieces join with tangent continuity,

$$\mathbf{p}_i := \left[\begin{smallmatrix} \overline{x} \\ \overline{y} \end{smallmatrix}\right](\frac{i}{n}), \text{ and } \mathbf{n}_i := \left[\begin{smallmatrix} \overline{y'} \\ -\overline{x'} \end{smallmatrix}\right](\frac{i}{n}) \qquad (3)$$

fit the bill and we can process each piece independent of its neighbor. If the curves meet just with continuity of position then (the normalized) $\mathbf{n}_n$ of the first and (the normalized) $\mathbf{n}_0$ of the second segment need to be averaged.

3.2 Multiplicity

To address the problem of multiplicity, we observe that there is always a pair of linear function enclosures, say $\left[\begin{smallmatrix} x^+ \\ y^- \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} x^- \\ y^+ \end{smallmatrix}\right]$, whose linear extensions or trims enclose the other two enclosures over the region of interest. To select this *extreme pair* of line segments from the four possible choices, we compute the *direction* $\mathbf{q}_i$ *of the line segment* with endpoints $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ as

$$\mathbf{q}_i := \mathbf{n}_i + \mathbf{n}_{i+1}.$$

Then, for each vector dimension α, we choose, if $\mathbf{q}_i^{[\alpha]} \geq 0$, the αth component of the upper function enclosure $\mathbf{b}_i^{+,[\alpha]}$ and for the other sheet $\mathbf{b}_i^{-,[\alpha]}$. If $\mathbf{q}_i^{[\alpha]} < 0$, we reverse the choice.

3.3 Planar Curve Enclosure Algorithm

We now make the construction precise. The input are Bézier curve pieces of degree d and the enclosure of each piece is to have n linear segments.

`Precondition:` The Bézier pieces are consistently oriented and regularly parametrized.

1. (`read basic enclosures`) Read the coefficients of the upper and lower enclosures $\lfloor A_\kappa^d \rfloor_n$ and $\lceil A_\kappa^d \rceil_n$ for degree d and n pieces, and for degree $d-1$ and n pieces. For a piecewise cubic curve and $n = 3$, these are the two times eight numbers displayed in Section 2.

2. For each Bézier segment

a (`create component enclosures`) Compute the second differences $\Delta_\kappa^2 \mathbf{b}$ of the Bézier coefficients. Depending on their sign use them to scale the upper enclosure $\lceil A_\kappa^d \rceil_n$ or the lower enclosure $\lfloor A_\kappa^d \rfloor_n$ in the linear combination:

$$\mathbf{b}^+ = \ell(\mathbf{b}) + \sum_{k=1}^{d-1} \lfloor A_\kappa^d \rfloor_n (\Delta_k^2 b)_- + \lceil A_\kappa^d \rceil_n (\Delta_k^2 b)_+,$$

$$\mathbf{b}^- = \ell(\mathbf{b}) + \sum_{k=1}^{d-1} \lfloor A_\kappa^d \rfloor_n (\Delta_k^2 b)_+ + \lceil A_\kappa^d \rceil_n (\Delta_k^2 b)_-.$$

Repeat this for $\mathbf{b}'$.

b (`compute` $\mathbf{p}_i$ `and` $\mathbf{n}_i$ by (3) of Section 3.1). Normalize $\|\mathbf{n}_i\| = 1$.

c (`select extreme line segments`) For each line segment of the control polygon compute and store the direction $\mathbf{q}_i := \mathbf{n}_i + \mathbf{n}_{i+1}$. For each vector dimension α set

$$\text{if } \mathbf{q}_i^{[\alpha]} \geq 0 \text{ then } \mathbf{h}_i^{+[\alpha]} := \mathbf{b}_i^{+[\alpha]}, \mathbf{h}_i^{-[\alpha]} := \mathbf{b}_i^{-[\alpha]},$$
$$\text{else } \mathbf{h}_i^{+[\alpha]} := \mathbf{b}_i^{-[\alpha]}, \mathbf{h}_i^{-[\alpha]} := \mathbf{b}_i^{+[\alpha]}.$$

Here $\mathbf{h}_i^+$ is a linear bounding segment in the direction $\mathbf{q}_i$ and $\mathbf{h}_i^-$ is the enclosure in the opposite direction ($\mathbf{h}_i^+$ and $\mathbf{h}_i^-$ are precursors of the curve enclosure).

d (`find the furthest intersections`) For each line segment for each anchor point $\mathbf{p}_i$, compute the intersection of the line through $\mathbf{p}_i$ in the direction $\mathbf{n}_i$ with the extreme enclosures: $\mathbf{p}_i + \lambda_{i,j}^+ \mathbf{n}_i = (1-u)\mathbf{h}_i^+ + u\mathbf{h}_j^+, j \in \{i-1, i+1\}$. By Cramer's rule,

$$\lambda_{i,j}^+ = \frac{\det(\mathbf{h}_i - \mathbf{p}_i, \mathbf{h}_i - \mathbf{h}_j)}{\det(\mathbf{n}_i, \mathbf{h}_i - \mathbf{h}_j)}.$$

3. (`create antipodal pairs`) For each anchor point $\mathbf{p}_i$ compute the antipodal pair

$$\mathbf{a}_i^+ := \mathbf{p}_i + \max\{\lambda_{i,i-1}^+, \lambda_{i,i+1}^+\}\mathbf{n}_i,$$
$$\mathbf{a}_i^- := \mathbf{p}_i + \min\{\lambda_{i,i-1}^-, \lambda_{i,i+1}^-\}\mathbf{n}_i.$$

4. (`form the enclosure`) For each line segment connect $\mathbf{a}_i^+, \mathbf{a}_{i+1}^+$ and connect $\mathbf{a}_i^-, \mathbf{a}_{i+1}^-$ to obtain the enclosure segments $\mathbf{e}_i^+$ and $\mathbf{e}_i^-$.

The algorithm is stable, unless the approximate normal $\mathbf{n}_i$ is parallel to the approximate secant $\mathbf{h}_i - \mathbf{h}_j$. In such a pathological case, subdivision is triggered, since the midpath is a poor approximation to the curve.

4 Enclosures of tensor-product polynomial pieces

The point of developing the material in one variable in such detail is not just to explain the fundamental ideas with less machinery but that we can bootstrap the techniques by tensoring. A tensor-product polynomial $\mathbf{b}(u,v)$ of degree d_1, d_2 is in Bézier form if

$$\mathbf{b}(u,v) = \sum_{i=0}^{d_1}\sum_{j=0}^{d_2} b_{ij} B_j^{d_2}(v) B_i^{d_1}(u),$$

$$\text{where } B_k^d(t) = \frac{d!}{(d-k)!k!}(1-t)^{d-k}t^k.$$

For example, a bi-cubic patch in $\mathbb{R}^3$ has 16 coefficients $b_{ij} \in \mathbb{R}^3$. We enclose the function for $(u,v) \in U := [0..1]^2$ by $n \cdot m$-piecewise bilinear functions

$$\sum_{i=0}^{n}\sum_{j=0}^{m} l_{ij} H_j^m(v) H_i^n(u) \quad \in \mathcal{H}^n \times \mathcal{H}^m.$$

The hat functions $H_i^n(t)$ were already defined in Section 2. We now simply apply the inequality (2) twice, once in v and once in u.

To make this simple idea precise, let A_j^d be the antidifference functions of degree d defined for $k, j \in \{1, \ldots, d-1\}$ by

$$\Delta_k^2 A_j^d = 1 \text{ if } k = j, \Delta_k^2 A_j^d = 0 \text{ if } k \neq j, \quad A_j^d(0) = A_j^d(1) = 0.$$

Let $\mathbf{b}_i(t)$ be the univariate Bézier polynomial with coefficients $b_{i0}, b_{i1}, \ldots, b_{id_2}$ and $\mathbf{b}_j(t)$ the polynomial with coefficients $b_{0j}, b_{1j}, \ldots, b_{d_1 j}$. Just as in Section 2, we compute for the parameter v

$$\begin{aligned}\mathbf{b}_i^+ &:= \sum_{j=0}^{m} \mathtt{u}_{ij}^v H_j^m(v) := b_{i0}(1-v) + b_{id_2} v \\ &\quad + \sum_{j=1}^{d_2-1} \lfloor A_j^{d_2} \rfloor_m \ \min\{\Delta_j^2 \mathbf{b}_i, 0\} + \lceil A_j^{d_2} \rceil_m \ \max\{\Delta_j^2 \mathbf{b}_i, 0\}, \\ \mathbf{b}_i^- &:= \sum_{j=0}^{m} \mathtt{l}_{ij}^v H_j^m(v) := b_{i0}(1-v) + b_{id_2} v \\ &\quad + \sum_{j=1}^{d_2-1} \lfloor A_j^{d_2} \rfloor_m \ \max\{\Delta_j^2 \mathbf{b}_i, 0\} + \lceil A_j^{d_2} \rceil_m \ \min\{\Delta_j^2 \mathbf{b}_i, 0\},\end{aligned}$$

and, again, now in u, $\mathbf{l}_j := \sum_{i=0}^{d_1} \mathtt{l}_{ij}^v B_i^{d_1}$, $\mathbf{u}_j := \sum_{i=0}^{d_1} \mathtt{u}_{ij}^v B_i^{d_1}$,

$$\begin{aligned}\sum_{i=0}^{n} \mathtt{u}_{ij} H_i^n(u) &:= \mathtt{u}_{0j}(1-u) + \mathtt{u}_{d_1 j} u \\ &\quad + \sum_{i=1}^{d_1-1} \lfloor A_i^{d_1} \rfloor_n \ \min\{\Delta_i^2 \mathbf{u}_j, 0\} + \lceil A_i^{d_1} \rceil_n \ \max\{\Delta_i^2 \mathbf{u}_j, 0\}, \\ \sum_{i=0}^{n} \mathtt{l}_{ij} H_i^n(u) &:= \mathtt{l}_{0j}(1-u) + \mathtt{l}_{d_1 j} u \\ &\quad + \sum_{i=1}^{d_1-1} \lfloor A_i^{d_1} \rfloor_n \ \max\{\Delta_i^2 \mathbf{l}_j, 0\} + \lceil A_i^{d_1} \rceil_n \ \min\{\Delta_i^2 \mathbf{l}_j, 0\}.\end{aligned}$$

Then

$$\begin{aligned}\mathbf{b}^-(u,v) &:= \sum_{j=0}^{m}\sum_{i=0}^{n} \mathtt{l}_{ij} H_i^n(u) H_j^m(v) \\ &\leq \sum_{j=0}^{m}\Big(\sum_{i=0}^{d_1} \mathtt{l}_{ij}^v B_i^{d_1}(u)\Big) H_j^m(v) = \sum_{i=0}^{d_1}\sum_{j=0}^{m} \mathtt{l}_{ij}^v H_j^m(v) B_i^{d_1}(u) \\ &\leq \mathbf{b}(u,v) = \sum_{i=0}^{d_1}\sum_{j=0}^{d_2} b_{ij} B_j^{d_2}(v) B_i^{d_1}(u) \qquad (4) \\ &\leq \sum_{i=0}^{d_1}\sum_{j=0}^{m} \mathtt{u}_{ij}^v H_j^m(v) B_i^{d_1}(u) = \sum_{j=0}^{m}\Big(\sum_{i=0}^{d_1} \mathtt{u}_{ij}^v B_i^{d_1}(u)\Big) H_j^m(v) \\ &\leq \mathbf{b}^+(u,v) := \sum_{j=0}^{m}\sum_{i=0}^{n} \mathtt{u}_{ij} H_i^n(u) H_j^m(v).\end{aligned}$$

Analogous to midpaths in one variable, we can now define the *midpatch* $\overline{\mathbf{b}}$ of $\mathbf{b}$ as the $n \cdot m$-piece bilinear function in $\mathcal{H}^n \times \mathcal{H}^m$ with

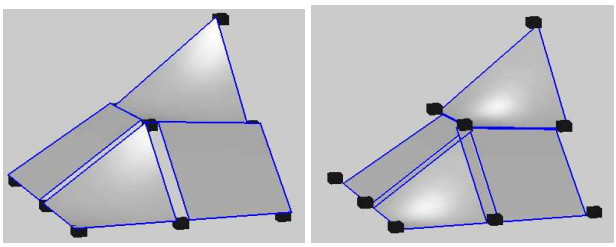

Figure 10: Four pieces of an upper piecewise bilinear enclosure $\mathbf{b}^+$ and four pieces of a lower bilinear facets $\mathbf{b}^-$. The nine cubes represent point enclosures. Note the gaps and overlaps.

values corresponding to interior points $(\frac{i}{n}, \frac{j}{m})$ of U, the $u = 0$ or $u = 1$-boundary, the $v = 0$ or $v = 1$-boundary, respectively a corner of U:

$$\overline{\mathbf{b}}(\frac{i}{n}, \frac{j}{m}) := \begin{cases} \frac{1}{2}(\mathbf{b}^+ + \mathbf{b}^-)(\frac{i}{n}, \frac{j}{m}) & \text{if } i \notin \{0, n\} \text{ and } j \notin \{0, m\}, \\ \frac{1}{2}(\mathbf{b}_i^+ + \mathbf{b}_i^-)(\frac{j}{m}) & \text{if } i \in \{0, n\} \text{ and } j \notin \{0, m\}, \\ \frac{1}{2}(\mathbf{b}_j^+ + \mathbf{b}_j^-)(\frac{i}{n}) & \text{if } i \notin \{0, n\} \text{ and } j \in \{0, m\}, \\ b_{ij} & \text{if } i \in \{0, n\} \text{ and } j \in \{0, m\}. \end{cases}$$

That is, we associate the average of a bivariate enclosure with the interior, the average of a univariate enclosure with the boundaries, and a constant with the vertices of U, so that adjacent midpatches match up continuously along boundaries.

4.1 Cost and approximation under refinement

At run time, we need only scale the univariate table entries with the second differences and add the result either to the upper or to the lower bound depending on the sign of the second difference. Since the maximal second difference reduces to at most $1/4$th of its original size under subdivision at the midpoint, the width $\mathbf{b}^+ - \mathbf{b}^-$ *shrinks to 1/4.*

4.2 Interval patch enclosures

For x, y, z we each have an upper and a lower bound yielding eight candidates for enclosures (Figure 12) for u, v in the domain square $[i..i+1]/m_1 \times [j..j+1]/m_2$:

$$\begin{bmatrix} x^+ \\ y^+ \\ z^+ \end{bmatrix}, \begin{bmatrix} x^+ \\ y^+ \\ z^- \end{bmatrix}, \begin{bmatrix} x^+ \\ y^- \\ z^+ \end{bmatrix}, \begin{bmatrix} x^+ \\ y^- \\ z^- \end{bmatrix}, \begin{bmatrix} x^- \\ y^+ \\ z^+ \end{bmatrix}, \begin{bmatrix} x^- \\ y^+ \\ z^- \end{bmatrix}, \begin{bmatrix} x^- \\ y^- \\ z^+ \end{bmatrix}, \begin{bmatrix} x^- \\ y^- \\ z^- \end{bmatrix}.$$

All combinations with positive weights summing to 1 of the eight enclosures form a shell that is a 3D enclosure of the surface piece (Figure 12, surfaces with parameter grid). The union of the shells of all patches form an enclosure of the surface.

Since the pieces are bilinear, we can also view the shell as a bilinear combination of the four point enclosures $\square_{i+\alpha, j+\beta}, \alpha, \beta \in \{0, 1\}$ of the corner points $[x(i+\alpha, j+\beta), y(i+\alpha, j+\beta), z(i+\alpha, j+\beta)]^T$. A *point enclosure* $\square_{i,j}$ is an axis-parallel box whose vertices are the eight combinations of the corner points of the component enclosures (the boxes displayed in Figures 10, 11 and 12):

$$\begin{bmatrix} x^+ \\ y^+ \\ z^+ \end{bmatrix}(\frac{i}{n}, \frac{j}{m}), \begin{bmatrix} x^+ \\ y^+ \\ z^- \end{bmatrix}(\frac{i}{n}, \frac{j}{m}), \begin{bmatrix} x^+ \\ y^- \\ z^+ \end{bmatrix}(\frac{i}{n}, \frac{j}{m}), \begin{bmatrix} x^- \\ y^+ \\ z^+ \end{bmatrix}(\frac{i}{n}, \frac{j}{m}), \text{etc.}$$

That is, the function enclosures directly yield a bilinear enclosure in *interval Bézier representation.*

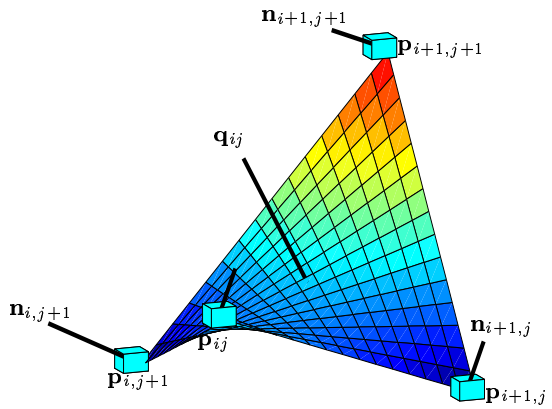

Figure 11: A single bilinear facet of the midpatch with direction $\mathbf{q}_{ij}$, anchor points $\mathbf{p}_{ij}$ and normal direction $\mathbf{n}_{ij}$.

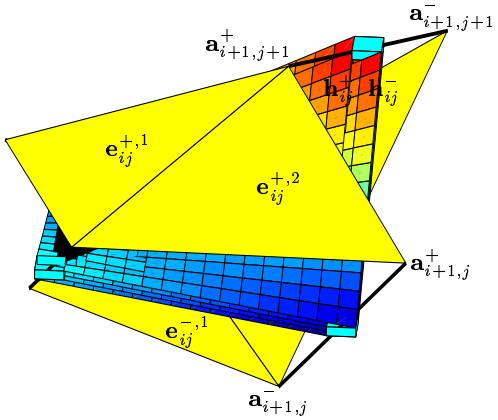

Figure 12: Antipodal pairs $\mathbf{a}^+_{ij}$ and $\mathbf{a}^-_{ij}$ as interpolation points of the two sheets $\mathbf{e}^{+,1}_{ij}$, $\mathbf{e}^{+,2}_{ij}$ and $\mathbf{e}^{-,1}_{ij}$, $\mathbf{e}^{-,2}_{ij}$ of the surface enclosure of eight bilinear facets (with parameter grid) whose extreme pair is $\mathbf{h}^+_{ij}$ and $\mathbf{h}^-_{ij}$. (The black spot is due to Matlab's depth sorting algorithm in the presence of many overlapping surfaces).

5 Constructing surface enclosures

The bilinear interval enclosures just defined have three shortcomings for efficient use: nonlinearity, multiplicity and gaps or intersections. The bilinearity of the facets implies that intersections between enclosures result in algebraic curves of degree 4 and force iterative techniques for intersections with rays as opposed to short explicit formulas for triangles. Slivers arise when computing the exact union of the shells which entails intersection of bilinear facets and trimming bilinear patches. Multiplicity, i.e. the choice from eight possible bilinear function enclosures implies up to 64 nonlinear intersection tests when intersecting two patch enclosures.

5.1 Gaps, Intersections and Slivers

Just as in the case of one variable, anchor points $\mathbf{p}_{ij}$ and normal $\mathbf{n}_{ij}$ (Figure 11) allow the construction of antipodal points $\mathbf{a}^+_{ij}$ and $\mathbf{a}^-_{ij}$ that serve as the vertices of the two triangle pairs $\mathbf{e}^{+,1}_{ij}, \mathbf{e}^{+,2}_{ij}$ and $\mathbf{e}^{-,1}_{ij}, \mathbf{e}^{-,2}_{ij}$. The surface enclosure is thus a tent-like construction

with support beams in the direction $\mathbf{n}_{ij}$ as shown in Figure 12.

5.2 Multiplicity

Just as in the case of one variable, we select an extreme pair of bilinear facets from the eight possible choices with the help of the direction

$$\mathbf{q}_{ij} := (\mathbf{p}_{i+1,j+1} - \mathbf{p}_{i,j}) \times (\mathbf{p}_{i,j+1} - \mathbf{p}_{i+1,j}).$$

normal to the facet of the midpatch at its central point. For each spatial vector dimension α, we choose, if $\mathbf{q}_{ij}^{[\alpha]} \geq 0$, the αth component of the upper function enclosure $\mathbf{b}_{ij}^{+,[\alpha]}$ and for the other sheet $\mathbf{b}_{ij}^{-,[\alpha]}$. If $\mathbf{q}_{ij}^{[\alpha]} < 0$, we reverse the choice.

5.3 Bilinearity

The new challenge is to replace the extreme bilinear facets by two pairs, say $\mathbf{e}_{ij}^{+,1}$ and $\mathbf{e}_{ij}^{+,2}$ of triangles (Figure 12). We have to be careful in two respects: we have to choose the diagonal correctly so as not to intersect the bilinear facet, and we have to trim or extrapolate the triangles to avoid overlaps and gaps that would result in many sliver pieces.

A subtle point is that the extrapolation of a triangle interpolating three vertices of a bilinear facet may intersect the extrapolated bilinear facet and ceases to be a one-sided approximation. Fortunately, we need not actually bound the extension of the extreme bilinear facets but, at vertices, only their extension by faces of the point enclosures and, along the boundaries, linear combinations of the edges of point enclosures. Therefore the surface is still enclosed by the extensions of the extreme triangle pairs.

To pick the correct diagonal crease when defining the triangle pairs $\mathbf{h}^+$ and $\mathbf{h}^-$ as interpolants to the four vertices of the bilinear facet, we compute the *crease direction* of the bilinear facet. With $\langle , \rangle$ denoting the inner product of two vectors, the boolean variable

$$\text{crease02}_{ij} := \langle \mathbf{p}_{i,j} + \mathbf{p}_{i+1,j+1} - \mathbf{p}_{i+1,j} - \mathbf{p}_{i,j+1}, \mathbf{q}_{ij} \rangle \geq 0$$

records the direction in which the ij bilinear facet turns with respect to its direction $\mathbf{q}_{ij}$. If $\text{crease02}_{ij} = 1$ then $\mathbf{h}^+$ with coefficients $[\mathbf{h}_{ij}^+, \mathbf{h}_{i+1,j}^+, \mathbf{h}_{i+1,j+1}^+, \mathbf{h}_{i,j+1}^+]$ is enclosed from the $(-\mathbf{q}_{ij})$ direction by the two triangles $[\mathbf{h}_{ij}^+, \mathbf{h}_{i+1,j}^+, \mathbf{h}_{i+1,j+1}^+]$ and $[\mathbf{h}_{ij}^+, \mathbf{h}_{i+1,j+1}^+, \mathbf{h}_{i,j+1}^+]$, i.e. is creased along the diagonal $[\mathbf{h}_{ij}^+, \mathbf{h}_{i+1,j+1}^+]$, otherwise by the two triangles $[\mathbf{h}_{ij}^+, \mathbf{h}_{i+1,j}^+, \mathbf{h}_{i,j+1}^+]$ and $[\mathbf{h}_{i,j+1}^+, \mathbf{h}_{i+1,j}^+, \mathbf{h}_{i+1,j+1}^+]$. The diagonal of $\mathbf{h}^-$ is $[\mathbf{h}_{i+1,j}^-, \mathbf{h}_{i,j+1}^-]$ if crease02_{ij} and $[\mathbf{h}_{i,j}^-, \mathbf{h}_{i+1,j+1}^-]$ otherwise.

5.4 Putting it all together

All three points, nonlinearity, multiplicity and slivers, have to be addressed simultaneously. For, given $\mathbf{p}$ and $\mathbf{n}$ of a anchor point, we can, in principle, pick out one upper and one lower bilinear facet from the eight candidates by extrapolating the bilinear facets beyond their domain unit square and look for the furthest intercept. However, the resulting collection of bilinear facets would have cracks, gaps and intersections. The cracks and gaps can be filled by planar sections. But that would increase the overall number of facets of the enclosure considerably compared to the facets in the control polyhedron of the surface.

The formal algorithm for generating surface enclosures is given in the Appendix. It is easy to check that the cost of the algorithm is linear in the number of anchor points and therefore in the number of bilinear facets and antipodal points. and that the calculations are stable, unless $\det(\mathbf{n}, \mathbf{h}_i - \mathbf{h}_j, \mathbf{h}_i - \mathbf{h}_k) \approx 0$, i.e. unless the approximate normal to the surface is coplanar with the tangent plane of the midpatch. Just as in the case of one variable, this anomaly triggers a refinement of the surface representation since the midpatch does not represent the surface well.

6 Extensions and Applications

So far, we focused on the fundamental construction and the (tensor-product) Bézier representation. Since NURBS surfaces and the regular, tensor-product part of Catmull-Clark subdivision surfaces [3] can be represented in terms of Bézier patches, this gives us some mileage (near extraordinary nodes, a combination of refinement and the generic bounding box construction can be used to cap the enclosure of a Catmull-Clark surface).

We challenge the general construction therefore by applying the approach to *total degree, three-sided Bézier patches* and use the information on position and normal inherent in the enclosure construction to drive adaptive evaluation.

6.1 Adaptive refinement

For complex objects, optimized enclosures are so tight that they appear indistinguishable from a finely triangulated version of the true surface. Only zooming in and slicing as in Figure 1 shows how the surface is 'sandwiched'. In Figure 14, we therefore illustrate adaptive refinement showing just a single three-sided Bézier patch (the required tables of numbers $\lceil A_i^d \rceil_n$ and $\lfloor A_i^d \rfloor_n$ for three-sided, total degree patches, as well as the tables for tensor-product surfaces will be available in the electronic, CD version of the paper.) We normalize by projecting the point enclosures of the normal direction to the unit sphere and take the dot product with the viewing direction to identify regions near the silhouette. To obtain high resolution near the silhouette, we enforce that the width of the surface enclosure is proportional to this dot product. The resulting partition of the domain is shown in Figure 13. To avoid gaps, the boundary points of the finer enclosure are placed on the boundary of the coarser enclosure.

7 Summary

The construction for enclosing curved surfaces by two matching triangulations presented in this paper is both simple and supported by a theory that provides guaranteed error bounds as well as reduction of the width under uniform refinement of the surface representation.

Optimized, refinable surface enclosures are therefore building blocks for extending efficient triangle-based or mesh-based, discrete algorithms to curved surfaces.

Acknowledgments

References

[1] Julien Basch. *Kinetic Data Structures*. PhD thesis, Stanford University, 1999.

[2] R. C. Buck. Applications of duality in approximation theory. In *Approximation of Functions (Proc. Sympos. General Motors Res. Lab., 1964)*, pages 27–42. Elsevier Publ. Co., Amsterdam, 1965.

[3] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10:350–355, Oct 1978.

[4] Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Greg Turk, Hans Weber, Pankaj Agarwal, Frederick P. Brooks, Jr., and William Wright. Simplification envelopes. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 119–128. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[5] A. Crosnier and J. R. Rossignac. Technical section — tribox bounds for three-dimensional objects. *Computers and Graphics*, 23(3):429–437, June 1999.

[6] R. DeVore. One-sided approximation of functions. *J of Approximation Theory*, 1:11–25, 1968.

[7] Gerald Farin. Tighter convex hulls for rational Bézier curves. *Comput. Aided Geom. Design*, 10:123–125, 1993.

[8] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A hierarchical structure for rapid interference detection. *Computer Graphics*, 30(Annual Conference Series):171–180, 1996.

[9] Chun-Yi Hu, Takashi Maekawa, Evan C. Sherbrooke, and Nicholas M. Patrikalakis. Robust interval algorithm for curve intersections. *Computer-aided Design*, 28(6-7):495–506, 1996.

[10] Chun-Yi Hu, Nicholas M. Patrikalakis, and Xiuzi Ye. Robust interval solid modelling part I: representations. *Computer-aided Design*, 28(10):807–817, 1996.

[11] Chun-Yi Hu, Nicholas M. Patrikalakis, and Xuizi Ye. Robust interval solid modelling part II: boundary evaluation. *Computer-aided Design*, 28(10):819–830, 1996.

[12] Timothy L. Kay and James T. Kajiya. Ray tracing complex scenes. In David C. Evans and Russell J. Athay, editors, *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 269–278, August 1986.

[13] James T. Klosowski, Joseph S. B. Mitchell, Henry Sowizral, and Karel Zikan. Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, January 1998.

[14] James Thomas Klosowski. *Efficient collision detection for interactive 3d graphics and virtual environments*. PhD thesis, State Univ. of New York at Stony Brook, May 1998.

[15] Leif P. Kobbelt, Katja Daubert, and Hans-Peter Seidel. Ray tracing of subdivision surfaces. In *Rendering Techniques '98 (Proceedings of the Eurographics Workshop)*, pages 69–80, New York, June 1998. Springer-Verlag.

[16] David Lutterkort. *Envelopes for Nonlinear Geometry*. PhD thesis, Purdue University, May 2000.

[17] D. Nairn, J. Peters, and D. Lutterkort. Sharp, quantitative bounds on the distance between a polynomial piece and its Bézier control polygon. *Computer Aided Geometric Design*, 16(7):613–633, Aug 1999.

[18] Allan M. Pinkus. *On L^1-approximation*. Cambridge University Press, Cambridge, 1989.

[19] U. Reif. Best bounds on the approximation of polynomials and splines by their control structure. *Comput. Aided Geom. Design*, 17(6):579–589, 2000.

[20] P.V. Sander, Xianfeng Gu, S.J. Gortler, H. Hoppe, and J. Snyder. Silhouette clipping. *Computer Graphics*, 34(Annual Conference Series):327–334, 2000.

[21] T. W. Sederberg and R. T. Farouki. Approximation by interval bezier curves. *IEEE Computer Graphics and Applications*, 12(5):87–95, September 1992.

[22] Thomas W. Sederberg, Scott C. White, and Alan K. Zundel. Fat arcs: A bounding region with cubic convergence. *Comput. Aided Geom. Design*, 6:205–218, 1989.

[23] S. T. Tuohy, T. Maekawa, G. Shen, and N. M. Patrikalakis. Approximation of measured data with interval B-splines. *Computer-aided Design*, 29(11):791–799, 1997.

[24] E. Welzl. Smallest enclosing disks (balls and ellipsoids). In Hermann Maurer, editor, *Proceedings of New Results and New Trends in Computer Science*, volume 555 of *LNCS*, pages 359–370, Berlin, Germany, June 1991. Springer.

8 Appendix: Tensor-Product Surface Enclosure Algorithm

We state the surface enclosure algorithm for a surface defined by Bézier patches $\mathbf{b}$ of degree (d_1, d_2) with normal direction $N := \frac{\partial \mathbf{b}}{\partial u} \times \frac{\partial \mathbf{b}}{\partial v}$ of degree $(2d_1 - 1, 2d_2 - 1)$. The enclosure patches are to be partitioned into $2mn$ triangles.

`Precondition:` The Bézier patches are consistently oriented and regularly parametrized.

1. (`read basic enclosures`) Read the coefficients of the piecewise bilinear upper and lower enclosures $\lfloor A_i^d \rfloor_k$, $\lceil A_i^d \rceil_k$ with for the degree and partition levels $(d, k) \in \{(d_1, n), (d_2, m), (2d_1 - 1, n), 2d_2 - 1, m\}$. For a bicubic surface and $m = n = 3$, these are two times eight numbers.

2. For each Bézier patch

   a (`create spatial component enclosures`) Compute the vector-valued $\mathbf{b}_{ij}^+$ and $\mathbf{b}_{ij}^-$, and N_{ij}^+ and N_{ij}^- according to (4).

   b (`compute anchor points and normals`) Set $\mathbf{p}_{ij} := \overline{\mathbf{b}}(\frac{i}{n}, \frac{j}{m})$ and $\mathbf{n}_{ij}$ to the vector $\overline{N}(\frac{i}{n}, \frac{j}{m})$, normalized. Where the surface is not tangent continuous, the $\mathbf{n}_{ij}$ have to be averaged and normalized.

   c (`select the extreme bilinear facets`) For each bilinear facet compute and store the direction

   $$\mathbf{q}_{ij} := (\mathbf{p}_{i+1,j+1} - \mathbf{p}_{i,j}) \times (\mathbf{p}_{i,j+1} - \mathbf{p}_{i+1,j}).$$

   For each spatial dimension α set

   $$\text{if } \mathbf{q}_{ij}^{[\alpha]} \geq 0 \text{ then } \mathbf{h}_{ij}^{+[\alpha]} := \mathbf{b}_{ij}^{+[\alpha]}, \mathbf{h}_{ij}^{-[\alpha]} = \mathbf{b}_{ij}^{-[\alpha]},$$
   $$\text{else } \mathbf{h}_{ij}^{+[\alpha]} := \mathbf{b}_{ij}^{-[\alpha]}, \mathbf{h}_{ij}^{-[\alpha]} = \mathbf{b}_{ij}^{+[\alpha]}.$$

   Here $\mathbf{h}_{ij}^+$ is a bilinear bounding facet in the direction $\mathbf{q}_{ij}$ and $\mathbf{h}_{ij}^-$ is the enclosure in the opposite direction. (The enclosures $\mathbf{h}_{ij}^+$ and $\mathbf{h}_{ij}^-$ are precursors to the surface enclosure).

   d (`establish the crease direction`) For each bilinear facet set the boolean

   $$\text{crease02}_{ij} := \langle \mathbf{p}_{ij} + \mathbf{p}_{i+1,j+1} - \mathbf{p}_{i+1,j} - \mathbf{p}_{i,j+1}, \mathbf{q}_{ij} \rangle \geq 0.$$

   Here $\langle , \rangle$ denotes the inner product of two vectors.

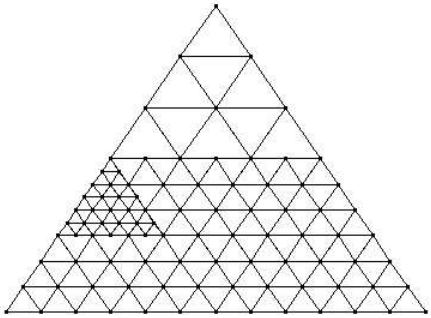

Figure 13: Adaptive refinement of the domain of the patch in Figure 14.

e (`find the furthest intersections`) For each bilinear facet, for each of the four anchor points $\mathbf{p}_\alpha$ where $\mathbf{p}_0 := \mathbf{p}_{i,j}$, $\mathbf{p}_1 := \mathbf{p}_{i+1,j}$, $\mathbf{p}_2 := \mathbf{p}_{i+1,j+1}$, $\mathbf{p}_3 := \mathbf{p}_{i,j+1}$, intersect the line through $\mathbf{p}_\alpha$ in the direction $\mathbf{n}_\alpha$ with the three triangles that include a point with index α. That is, if crease02$_{ij}$, intersect with

$$[\mathbf{h}_0^+, \mathbf{h}_1^+, \mathbf{h}_2^+], [\mathbf{h}_0^+, \mathbf{h}_2^+, \mathbf{h}_3^+] \text{ and } [\mathbf{h}_0^-, \mathbf{h}_1^-, \mathbf{h}_3^-]$$

else intersect with

$$[\mathbf{h}_0^+, \mathbf{h}_1^+, \mathbf{h}_3^+], \text{ and } [\mathbf{h}_0^-, \mathbf{h}_1^-, \mathbf{h}_2^-], [\mathbf{h}_0^-, \mathbf{h}_2^-, \mathbf{h}_3^-].$$

The equation in λ, u, v, corresponding to the triangle $\mathbf{h}_i, \mathbf{h}_j, \mathbf{h}_k$ is

$$\mathbf{p} + \lambda\mathbf{n} = \mathbf{h}_i + u(\mathbf{h}_j - \mathbf{h}_i) + v(\mathbf{h}_k - \mathbf{h}_i)$$

and the solution for λ is

$$\lambda = \frac{\det(\mathbf{h}_i - \mathbf{p}, \mathbf{h}_i - \mathbf{h}_j, \mathbf{h}_i - \mathbf{h}_k)}{\det(\mathbf{n}, \mathbf{h}_i - \mathbf{h}_j, \mathbf{h}_i - \mathbf{h}_k)}.$$

Record the maximal λ-value, λ_{ij}^+, when intersecting with the $\mathbf{h}^+$ triangles, and the minimal λ-value, λ_{ij}^-, when intersecting with the $\mathbf{h}^-$ triangles over all bilinear facets attached to an anchor point $\mathbf{p}_{ij}$.

3. (`create antipodal pairs`) For each anchor point $\mathbf{p}_{ij}$ compute the antipodal pair

$$\mathbf{p}_{ij} + \lambda_{ij}^+\mathbf{n}_{ij}, \quad \mathbf{p}_{ij} + \lambda_{ij}^-\mathbf{n}_{ij}.$$

4. (`form the enclosure triangles`) For each bilinear facet connect the $\mathbf{p}_{ij} + \lambda_{ij}^+\mathbf{n}_{ij}$ according to crease02$_{ij}$ and (2e) to form the two triangles $\mathbf{e}_{ij}^{+,1}$, $\mathbf{e}_{ij}^{+,2}$ and connect the $\mathbf{p}_{ij} + \lambda_{ij}^-\mathbf{n}_{ij}$, with opposite crease, to form the pair $\mathbf{e}_{ij}^{-,1}$, $\mathbf{e}_{ij}^{-,2}$.

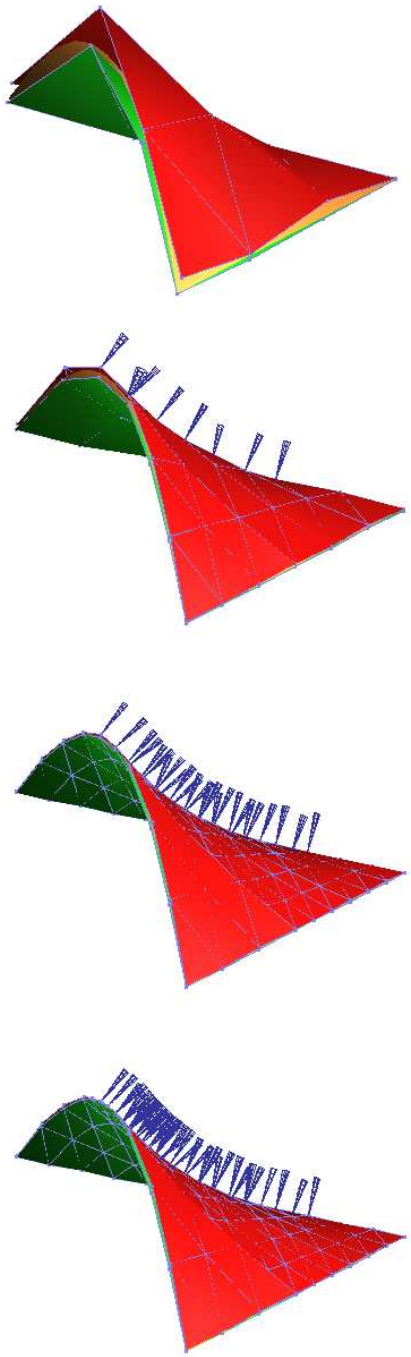

Figure 14: Adaptive refinement near the silhouette. (The right corner is not part of the silhouette.) The cones correspond to projected point enclosures of the normal direction.