Watertight Trimmed NURBS

Thomas W. Sederberg*
G. Thomas Finnigan‡
Brigham Young University

Xin Li†
University of Science and Technology of China

Hongwei Lin§
Zhejiang University

Heather Ipson
Brigham Young University

Abstract

This paper addresses the long-standing problem of the unavoidable gaps that arise when expressing the intersection of two NURBS surfaces using conventional trimmed-NURBS representation. The solution converts each trimmed NURBS into an untrimmed T-Spline, and then merges the untrimmed T-Splines into a single, watertight model. The solution enables watertight fillets of NURBS models, as well as arbitrary feature curves that do not have to follow iso-parameter curves. The resulting T-Spline representation can be exported without error as a collection of NURBS surfaces.

CR Categories: I.3.5 [Computer Graphics]: Computational geometry and object modeling—Curve, surface, solid, and object representations

Keywords: Surface intersection, Booleans, NURBS, T-Splines

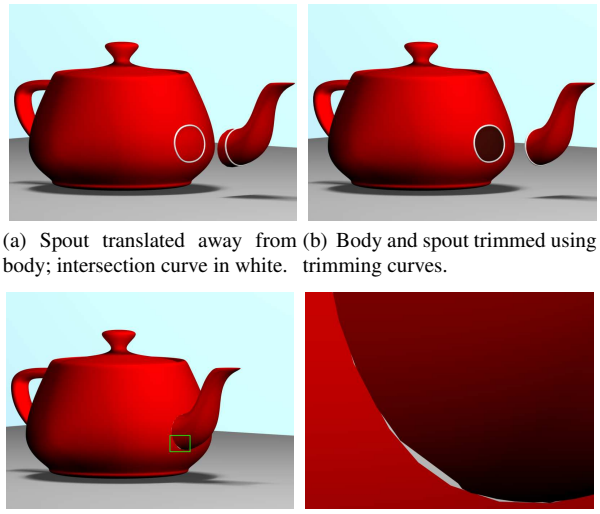
1 Introduction

The trimmed-NURBS modeling paradigm suffers from a serious fundamental flaw: parametric trimming curves are mathematically incapable of fulfilling their primary role, which is to represent the curve of intersection between two NURBS surfaces. Consequently, trimmed-NURBS models are not mathematically watertight, as illustrated in Figure 1 in which trimming curves are used to express the intersection between the body and spout of the Utah teapot model. We use the term "watertight" to connote no *unwanted* gaps or holes. That is, the surface is a gap-free 2-manifold in the neighborhood of intersection curves.

This paper presents a two-step algorithm for representing the Boolean combination of two NURBS objects as a single watertight T-Spline. In the first step, each trimmed NURBS is converted into a T-Spline without trimming curves, as illustrated in Figure 2.a. In the second step, each pair of untrimmed T-Splines is merged along their intersection curve into a gap-free T-Spline model, as illustrated in Figure 2.b. The resulting model, shown in Figure 2.c is C^2, except at the C^0 crease along the intersection curve. This T-Spline model facilitates the creation of watertight fillets. The model in Figure 2.d contains a C^2 gap-free fillet between the body and spout.

Section 2 reviews the history and significance of the problem this paper addresses, and reviews prior literature. Section 3 presents an algorithm for converting a trimmed-NURBS into an untrimmed T-Spline. Section 4 explains how to merge two NURBS or T-Spline surfaces with mis-matched parametrizations. Section 5 details how the algorithms presented in Sections 3 and 4 work together to create watertight trimmed-NURBS models, and examines the approximation error. This section also discusses the creation of gap-free, C^2 fillets and the placement of feature lines on a T-Spline surface that are not aligned with iso-parameter curves. Section 6 summarizes.

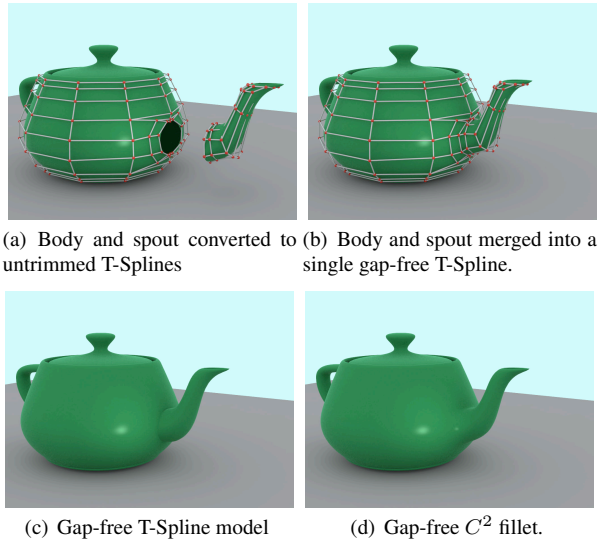
*tom@byu.edu ‡tomfinnigan@gmail.com
†xinliustc@gmail.com §hwlin@cad.zju.edu.cn



(a) Spout translated away from body; intersection curve in white. (b) Body and spout trimmed using trimming curves.

(c) Trimmed body and spout translated back into original orientation (d) Blowup of green rectangle in (c), showing gap

Figure 1: *Trimmed-NURBS Representation of the Utah Teapot.*



(a) Body and spout converted to untrimmed T-Splines (b) Body and spout merged into a single gap-free T-Spline.

(c) Gap-free T-Spline model (d) Gap-free C^2 fillet.

Figure 2: *Gap-free Teapot.*

2 Background

The fact that gaps are unavoidable in conventional trimmed-NURBS mathematical models can be shown as follows. A trimming curve is typically a degree-three NURBS curve defined in the parameter domain of a NURBS surface. The image of such a trimming curve on a bicubic patch (i.e., the curve on the bicubic patch in

R^3 that the trimming curve maps to) is degree ≤ 18 and algebraic genus zero. However, a generic intersection curve of two bicubic surfaces is degree 324 in R^3 [Sederberg et al. 1984] and algebraic genus 433 [Katz and Sederberg 1988]. Hence, intersection curves can only be approximated by parametric trimming curves.

The existence of these gaps in trimmed NURBS models seems innocuous and easy to address, but in fact it is one of the most serious impediments to interoperability between CAD, CAM and CAE systems [Kasik et al. 2005]. Software for analyzing physical properties such as volume, stress and strain, heat transfer, or lift-to-drag ratio will not work properly if the model contains unresolved gaps. Since 3D modeling, manufacturing and analysis software does not tolerate gaps, humans often need to intervene to close the gaps. This painstaking process has been reported to require several days for a large 3D model such as an airplane [Farouki 1999] and was once estimated to cost the US automotive industry over $600 million annually in lost productivity [NIS 1999]. At a workshop of academic researchers and CAD industry leaders [Farouki 1999], the existence of gaps in trimmed-NURBS models was singled out as the single most pressing unresolved problem in the field of CAD.

Prior Art

Several solutions to the gap problem have been put forward, but none address the problem adequately. The best solution from a theoretical standpoint is to use the precise representation for trimming curves, which is an *implicit* (not parametric) equation of the form $f(s,t) = 0$. In the case of two intersecting bicubic patches, $f(s,t)$ is a polynomial of bi-degree 54×54. [Krishnan and Manocha 1996] presents a solution to the surface intersection problem based on such a representation, and [Krishnan et al. 2001] describes a solid modeling system based on this approach, using exact arithmetic. Unfortunately, exact arithmetic can be very expensive and the method has not been adopted by the CAD industry. In applications for which a tessellation of the surfaces suffices, gaps can easily be filled with a triangle strip or avoided altogether by careful coordination while tessellating adjoining trimmed surfaces [Kumar 1996; Moreton 2001]. However, once a NURBS model has been reduced to a C^0 tessellation, it loses its character as a smooth surface and operations such as offsetting become impossible.

[Song et al. 2004] and [Farouki et al. 2004] describe methods for creating a non-tessellated, watertight approximation of two or more intersecting NURBS surfaces. Each method produces a set of piecewise C^0 (but approximately C^1) Bézier patches, although if patches adjacent to an intersection curve are edited, the surfaces become discontinuous. Our technique produces a watertight C^2 surface defined using a single T-Spline control grid, so the surface remains C^2 if the control points are moved. [Song et al. 2004] requires the solution of a system of linear equations that under some conditions can produce huge approximation errors. Our method is more amenable to creating fillets than [Song et al. 2004] and [Farouki et al. 2004].

[Kristjansson et al. 2001] takes as input Loop subdivision surfaces, although extension to other types of subdivision surfaces is possible. [Kristjansson et al. 2001] produces a G^2 watertight subdivision surface defined by a multi-resolution control grid; the surface remains G^2 if the control grid is edited. The goal of [Kristjansson et al. 2001] is an efficient algorithm suitable for animation, but not necessarily for CAD.

A pertinent prior art to Section 3 is [Litke et al. 2001], which describes a process of converting a trimmed subdivision surface into an untrimmed subdivision surface such that each trimming curve on the trimmed surface becomes a boundary curve on the untrimmed surface. [Litke et al. 2001] uses an enhanced Loop surface (triangle based), whereas our algorithm is based on tensor-product NURBS surfaces. Both algorithms must perturb the surface in the neighborhood of each trimming curve, but are capable of confining the perturbation to an arbitrarily small magnitude and narrow neighborhood. [Litke et al. 2001] uses trimming curves that lie in world space, thus permitting the true intersection curve to serve as the trimming curve, whereas we use conventional parametric trimming curves, which can only approximate true intersection curves.

A key disadvantage of subdivision surfaces for use in CAD is their incompatibility with NURBS. Billions of dollars have been invested in NURBS software and models, and there is tremendous economic pressure against abandoning the NURBS paradigm. Furthermore, NURBS do have some advantages over subdivision surfaces. For example, numerous versions of subdivision surfaces exist, with no current industry standard, and many capabilities of subdivision surfaces involve special refinement rules. Also, subdivision surfaces are limit surfaces, involving infinite sequences of patches. Although there are efficient ways to evaluate subdivision surfaces [Stam 1998], the infinite number of patches is more difficult to deal with than a finite number of NURBS patches, especially when doing data file exchange.

One approach used in commercial CAD software to manage the NURBS gap problem is to use a procedural definition of intersection curves, which keeps track of which surfaces intersect. Intersections can then be approximated, on demand, to any desired tolerance. This approach complicates subsequent tasks such as offsetting or filleting that require an explicit representation of the intersection curve. Furthermore, if a procedural definition of the same intersection is used by two different programs, it is possible to arrive at different results. This has resulted in incompatibilities between NURBS representations by different CAD, CAM, and CAE software applications, and the growth of an entire software industry around translating, fixing and healing 3D models and surfaces.

The surface intersection problem has been very thoroughly researched. A sampling of the vast literature can be found in [Patrikalakis and Maekawa 2002; Song et al. 2004]. The algorithms described in this paper assume the existence of a robust surface intersection algorithm that can represent an intersection curve using trimming curves to within a prescribed tolerance, such as in [Krishnan and Manocha 1997]. Such capability is now standard in most commercial geometric modeling programs.

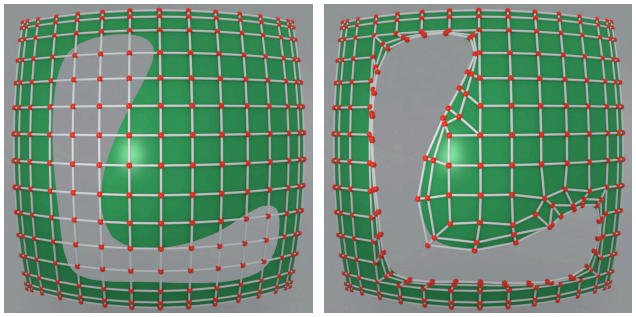
The topic of fillets has likewise been widely researched. See [Song and Wang 2007] for a list of references, and a solution to the fillet problem that provides G^n continuity. Most commercial software approximates fillets of free-form surfaces as NURBS surfaces that lie on the base surfaces, with approximate G^1 continuity. The advantage of the fillet solution presented in this paper is that it is part of a unified geometric framework. An entire geometric model, including fillets, can be represented as a single watertight T-Spline.

3 Trimmed-NURBS to Untrimmed T-Splines

This section presents a method for converting a bicubic NURBS surface with trimming curve into an approximately equivalent T-Spline with no trimming curve. The approximation error can be made arbitrarily small, and the perturbation can be confined to an arbitrarily narrow neighborhood of the trimming curve. We describe the algorithm using the example in Figure 3. Figure 4.a diagrams the trimming curve C in the parameter domain of the NURBS surface. The grid lines are knot lines for the NURBS surface. Points in the domain that correspond to NURBS control points are highlighted in red.

Trimmed-NURBS to Untrimmed T-Splines Conversion

Step 1. Form an axis-aligned polygon $\mathbf{A}$ (i.e., a polygon whose edges are parallel to one of the two parameter directions) that en-



(a) Trimmed NURBS. (b) Untrimmed T-Spline.

Figure 3: *Trimmed-NURBS to Untrimmed T-Splines Conversion.*

closes the trimming curve, as illustrated in Figure 4.b.

Step 2. At each vertex of $\mathbf{A}$ that does not lie on a red point, perform a T-Spline control point insertion as described in [Sederberg et al. 2004]. In this example, the control point insertions will occur at the five red points lying on the black line in Figure 4.c. (The insertion operation adds two additional control points to the left of the black corner, and two beneath it, as shown in Figure 4.d.)

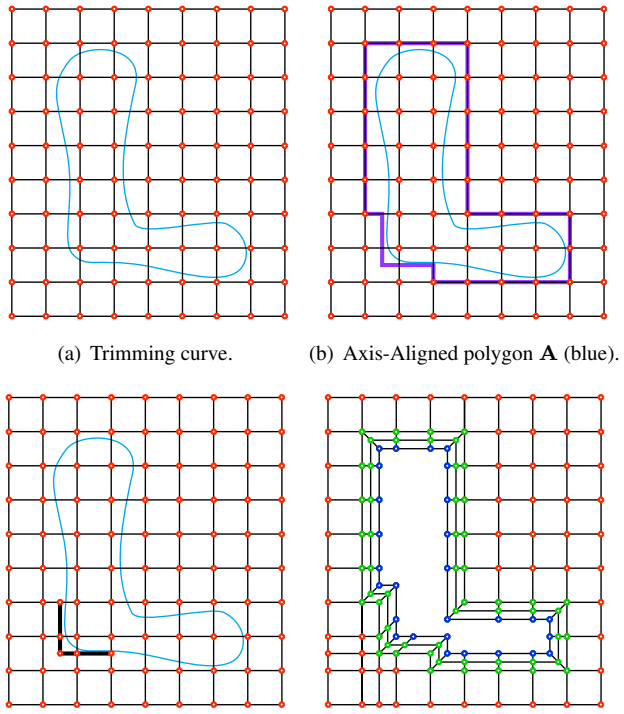
Step 3. Remove the portion of the control mesh that lies on the interior of $\mathbf{A}$ and replace it with the mesh topology illustrated in Figure 4.d. Note that each convex corner in $\mathbf{A}$ introduces a valence three control point in the modified control grid, and each concave corner in $\mathbf{A}$ creates a valence five control point. Assign a knot interval of zero to all edges of the control grid that connect a blue control point to a green control point, as shown in Figure 5.a. These zeros create a Bézier end condition for this boundary curve. Assign a small knot interval α to all edges connecting the outer layer of green control points to the inner layer of green control point. A good choice for α is the average of all parameter distances between the vertices on $\mathbf{A}$ and the trimming curve.

Step 4. Leave all red control points in Figure 4.d in their initial location. The blue control points in Figure 4.d define a NURBS curve that approximates the image of C. The positions of those control points are chosen to minimize the orthogonal distance between the NURBS curve and the image of C, using an algorithm such as in [Wang et al. 2006]. Likewise, the positions of the green control points are chosen to minimize the orthogonal distance between the T-Spline surface and the interior of the trimmed NURBS surface. The resulting T-Spline and its control grid are shown in Figure 3.b.

This procedure introduces some perturbation error, the magnitude of which in this example is 0.001 times the width of the model. The domain of the perturbed region lies within the support the green and blue control points. Figure 5.a illustrates the perturbation region in yellow. For a fixed axis-aligned polygon, we can make the perturbation region on the exterior of the polygon arbitrarily narrow by performing a local T-Spline refinement, as illustrated in Figure 5.b. Likewise, we can make the distance between $\mathbf{A}$ and C arbitrarily small by finding an axis-aligned polygon that approximates C to within a tolerance ϵ. Clearly, there are countless such axis-aligned polygons, and numerous possible algorithms for finding such polygons. We now present one such algorithm.

3.1 Finding an Axis-Aligned Polygon

The algorithm, illustrated by the example in Figure 6, has the flavor of a curve rasterization in which the pixels are cells of a quadtree. Related applications of quadtrees are reported in [Hunter and Steiglitz 1979; Samet 1984]. We begin by defining a color-based classification system for a rectangular domain $\mathbf{R}$ with respect to a trimming curve C and a tolerance ϵ as follows:



(a) Trimming curve. (b) Axis-Aligned polygon $\mathbf{A}$ (blue).

(c) Control points inserted at vertices of $\mathbf{A}$. (d) Topology modification.

Figure 4: *Algorithm for Converting a Trimmed-NURBS into an Untrimmed T-Spline.*

White C does not intersect R.

Blue C does intersect R and the width or height of C is $> \epsilon$.

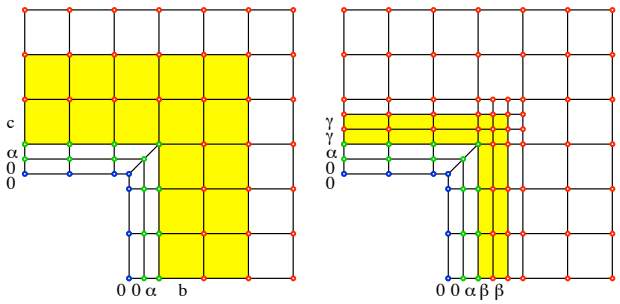
Red C does intersect R, the width and height of C are $< \epsilon$, but the one-neigborhood of cells adjacent to R intersect C in more than one connect component.

Gray C does intersect R, the width and height of C are $< \epsilon$, and the one-neigborhood of cells adjacent to R intersect C in exactly one connect component.

Begin by assigning a color to each rectangle bounded by knot lines in the parameter domain, using the above classification. Split each red or blue rectangle into four axis-aligned rectangles and reclassify each of those four new rectangles. Repeat these splitting and reclassification operations until all cells are either white or gray. At this point, each component of C will be covered by a contiguous set of gray cells, which we might call a rasterization of the component. For each component, the perimeter of its rasterization will serve as an acceptable axis-aligned polygon. In Figure 6.f, the axis-aligned polygon is highlighted in blue.

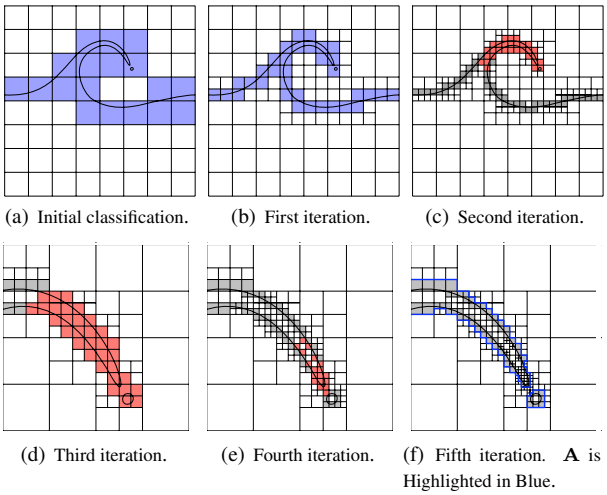
Extraordinary Points

[Sederberg et al. 2003] suggests dealing with extraordinary points in T-Spline surfaces using the method presented in [Sederberg et al. 1998], which is a generalization of Catmull-Clark refinement that takes into account knot intervals. For our purposes, we can modify Step 2 in Algorithm 1 to include doing T-Spline refinements to force all knot intervals to be identical in the 2-neighborhood of each extraordinary point created in Step 3. This converts the



(a) Perturbation domain (yellow). (b) Decreasing the perturbation domain through T-Spline local refinement. Choose $\beta < b$, $\gamma < c$.

Figure 5: *Limiting the Perturbation Domain.*



(a) Initial classification. (b) First iteration. (c) Second iteration.

(d) Third iteration. (e) Fourth iteration. (f) Fifth iteration. **A** is Highlighted in Blue.

Figure 6: *Algorithm for Finding Bounding Polygons.*

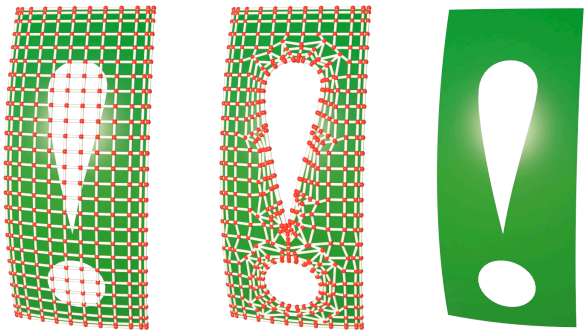
extraordinary points into conventional Catmull-Clark style with uniform knots, making possible the use of methods such as [Peters 2000] for patching valence n extraordinary points using G^1 bicubic patches, with one patch per face of the control grid. Alternatively, the extraordinary region can be filled using G^2 patches using a method such as in [Loop 2004].

Discussion

An important property of this procedure is that the resulting T-Spline is fully editable, meaning that its control points can be adjusted and all the properties of a C^2 spline are honored. The algorithm offers a tradeoff between accuracy and number of control points. If the goal is to trim away some holes but then to continue to modify the resulting T-Spline, an artist or designer can opt for fewer control points. The resulting larger approximation error should be acceptable since the surface will undergo additional modification.

Figure 7 shows an example involving two loops and a sharp corner. In this case, the perturbation error is 0.00025 relative to the width of the patch.

An improved algorithm for computing the green control points is a problem calling for future research. To create our example figures, we chose a traditional method that appeared simplest to implement. We begin by obtaining a set of sample points on the region of the trimmed surface that will be perturbed and then specify an initial position for the green control points. The following process is then repeated: Assign each sample point a parameter pair on the untrimmed T-Spline, then solve for the green control



(a) Trimmed NURBS. (b) Untrimmed T-Spline. (c) Untrimmed T-Spline.

Figure 7: *Trimmed NURBS to Untrimmed T-Spline Conversion.*

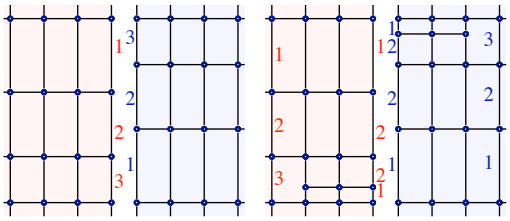
points that minimize the least squares error based on these parameter assignments. It is known that this algorithm converges only linearly [Bjorck 1996], and indeed it can take several tens of seconds to obtain good results if the initial positions of the green control points are not chosen wisely. The algorithm has also been observed to converge to a local min.

One possible solution is to extend to surfaces the curve-fitting algorithm in [Wang et al. 2006], something the authors of that paper are working on. Another line of research is to study whether the quasi-interpolation methods used in [Litke et al. 2001] can be adapted to T-Splines.

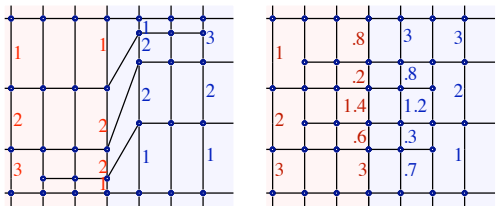
4 Merging using NU-NURBS

After two intersecting surfaces are converted into untrimmed T-Splines using the method in Section 3, the final step is to merge those two T-Splines into a single, gap-free T-Spline. A basic algorithm for merging two T-Splines is presented in [Sederberg et al. 2003]. However, that algorithm gives poor results if the two surfaces to be merged do not have consistent parametrizations, as illustrated in Figure 8. The first step in the merge algorithm in [Sederberg et al. 2003] is to insert knots such that the two surfaces have the same set of knot intervals, as shown in Figure 8.b. (This might also require that all knot intervals on one surface be scaled so that their sum matches the sum of the knot intervals on the other surface). The final step is to connect the two control grids, as shown in Figure 8.c. However, if the adjoining boundary curves are not parametrized similarly, the isoparameter curves will experience an abrupt bend, imparting a kink in the resulting surface. For example, Figure 9 shows a hand and arm modeled as separate NURBS, whose parametrizations do not align. Figure 9.b shows the result of merging them using the algorithm described in [Sederberg et al. 2003]. Unfortunately, most pairs of untrimmed T-Splines generated using the method in Section 3 have this problem.

The problem is related to the fact that the refinements shown in Figure 8.b must honor a restriction that is placed on the knot intervals in a T-Spline: the sum of knot intervals on one edge of a face on the control grid must equal the sum of knot intervals on the opposing edge in the face. Better results could be obtained by, instead of refining each surface as in Figure 8.b, we refine the two surfaces so that their knot lines align, as shown in Figure 8.d. However, the resulting knot interval configuration violates the definition of a T-Spline. Previous methods for dealing with such knot intervals [Sederberg et al. 1998; Müller et al. 2006] devise variations on Catmull-Clark refinement in which faces of the control grid map to an infinite sequence of bicubic patches which, like extraordinary points in a Catmull-Clark surface, are G^1. The infinite sequence of patches violates a key objective of this paper, which is to be ex-



(a) Initial knot intervals. (b) After refinement.

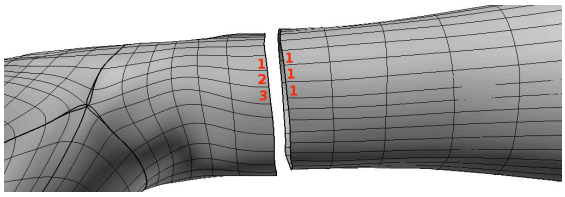


(c) Control grids with mismatched parametrization. (d) Refinement to align isoparameter curves.

Figure 8: *Merging Two NURBS Surfaces.*

portable using a finite number of tensor-product patches.

To address this problem, we introduce a generalization of tensor-product B-Spline surfaces that supports the knot interval configuration in Figure 10.b, that is C^2, and that yields one tensor-product patch per face of the control grid. Since the knot intervals change and hence are not "uniform," we will refer to this surface as a non-uniform NURBS surface, or NU-NURBS (spoken "new NURBS").



(a) Arm and hand showing mismatched knot intervals.

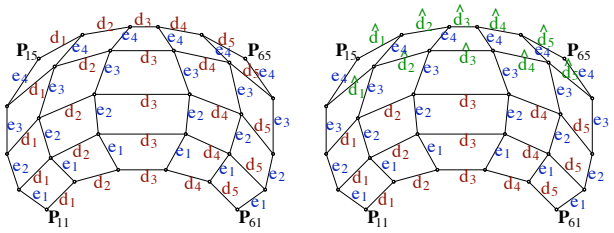
(b) Merge using the algorithm in [Sederberg et al. 2003].

(c) Merge using NU-NURBS.

Figure 9: *Merging NURBS Hand and Arm Models. (Model courtesy of Zygote Media Group)*

The idea is based on the fact that a tensor-product B-Spline surface can be viewed as a family of iso-parameter curves:

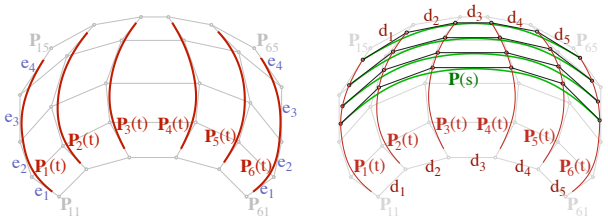
$$\mathbf{P}(s,t) = \sum_{i=1}^{m} \mathbf{P}_i(t) B_i(s) \quad \text{where} \quad \mathbf{P}_i(t) = \sum_{j=1}^{n} \mathbf{P}_{ij} B_j(t) \quad (1)$$



(a) Knot intervals in a bicubic NURBS control grid. (b) Knot intervals in a merge region.

Figure 10: *Knot Interval Configurations.*

The $\mathbf{P}_i(t)$ can be viewed as "moving control points" that slide along B-Spline curves, as illustrated in Figure 11.a.



(a) "Moving control points" $\mathbf{P}_i(t)$. (b) Iso-parameter curves, $\mathbf{P}(s)$.

Figure 11: *Constructing a Family of Isoparameter Curves on a NURBS Surface.*

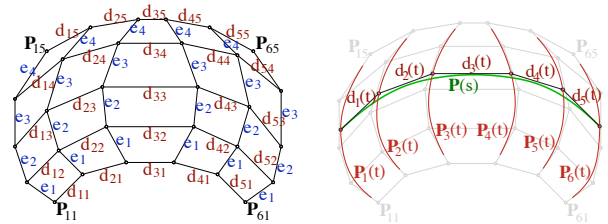
For a fixed value of $t = \tau$,

$$\mathbf{P}(s) = \sum_{i=1}^{m} \mathbf{P}_i(\tau) B_i(s) \quad (2)$$

defines a cubic B-Spline curve that lies on the bicubic B-Spline surface, and is the iso-parameter curve for $t = \tau$. If we let τ vary, the resulting family of iso-parameter curves sweeps out the B-Spline surface. Figure 11.b shows four such iso-parameter curves for various values of τ.

Since most readers will be more familiar with B-Splines defined using knot vectors rather than with knot intervals, we note that it is straightforward to convert between the two representations. Define $\tilde{e}_{-2} = 0$, $\tilde{e}_{i+1} = \tilde{e}_i + e_i$, $i = -2, \ldots, 5$ (e_{-1}, e_0, e_5, and e_6 are not shown in the figure). Then the knot vector for the B-spline curves $\mathbf{P}_i(t)$ is $\{\tilde{e}_{-2}, \tilde{e}_{-1}, \ldots, \tilde{e}_6\}$. Likewise, the knot vector for each of the isoparameter curves in Figure 11.b is $\{\tilde{d}_{-2}, \tilde{d}_{-1}, \ldots, \tilde{d}_7\}$ where $\tilde{d}_{-2} = 0$, $\tilde{d}_{i+1} = \tilde{d}_i + d_i$, $i = -2, \ldots, 6$

We now modify that description of a B-Spline surface to permit a knot interval arrangement as in Figure 12.a in which the d_{ij} can be any non-negative number. The basic idea is to treat the knot intervals themselves as cubic spline functions, which in turn control the basis functions $B_i(s)$ in (2).



(a) Knot intervals for NU-NURBS. (b) Iso-parameter curve, $\mathbf{P}(s)$.

Figure 12: *NU-NURBS.*

Figure 12.b shows an iso-parameter curve on a NU-NURBS surface. The "moving control points" $\mathbf{P}_i(t)$ in this figure are identical to those used in the description of NURBS surfaces in Figure 11.b. The only difference is that in the NURBS case in Figure 11.b, the knot intervals d_i are constants whereas for NU-NURBS, the $d_i(t)$ are spline functions. The coefficients of spline function $d_i(t)$ are d_{i0}, d_{i1}, d_{i2}, ..., d_{i5}, d_{i6} and the knot vector for the spline function is $\{\tilde{e}_{-2}, \tilde{e}_{-1}, \ldots, \tilde{e}_6\}$. The NU-NURBS is thus defined as a family of iso-parameter curves.

This NU-NURBS formulation has the following properties:

1. It specializes to NURBS in the case where the knot interval configuration is identical to that in Figure 10.a.
2. This NU-NURBS is C^2 in s, since each iso-parameter curve $\mathbf{P}(s)$ is a cubic spline curve whose knot intervals are constant for a fixed value of t.. Since the knot interval functions are C^2 splines in t, the NU-NURBS is also C^2 in t.
3. The cost of evaluating this NU-NURBS is comparable to the cost of evaluating a bicubic NURBS surface, the only difference lies in evaluating the knot interval spline functions $d_i(t)$.
4. Although this surface formulation is new and hence not directly supported in existing commercial software, it can be exactly represented—and exported—as a set of rational Bézier patches, with one patch per face of the control grid. Unfortunately, the degree of those patches can be rather high.

We have devised other, lower-degree versions of NU-NURBS, all that produce one patch per face, including a version that is C^2 in t and G^1 in s with patches that are degree 3×6, and one that is C^2 in t and G^2 in s with patches that are degree 4×9. We have also devised a version of NU-NURBS that permits arbitrary non-negative knot intervals in both parameter directions. These variations are being recorded in a separate paper [Sederberg et al.].

5 Examples

This section examines the behavior of the algorithms presented in Sections 3 and 4 when they combine to represent two intersecting trimmed-NURBS surfaces as a single watertight T-Spline. It also shows how fillets are supported in this representation, along with arbitrary feature lines.

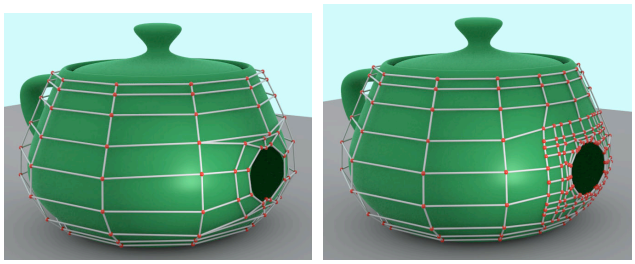
As reviewed in Section 2, the problem of computing intersection curves is very well-studied, and algorithms for computing the intersection of two NURBS surfaces P_1 and P_2 are standard in most geometric modeling programs. These algorithms can compute trimming curves C_1 and C_2 in the parameter domains of P_1 and P_2, along with an approximation C of the intersection curve in R^3, to within a prescribed tolerance. This paper assumes that C_1, C_2, and C have been computed using an existing algorithm, and that the geometric and topological accuracy of these curves is deemed acceptable.

Perturbation Error

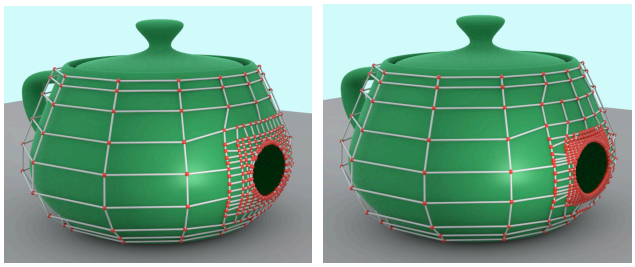
Figure 13 shows the trimmed teapot body being converted into untrimmed T-Splines with different degrees of precision. The NURBS model of the teapot used throughout this paper is actually a C^2 NURBS model based on the original C^1 Bézier model. In our NURBS model, the body is defined using 90 control points.

Figure 14 shows the error distribution in both the body and the spout. This figure shows how the perturbation domain decreases as ϵ decreases.

By adjusting ϵ, the perturbation magnitude and extant can be held below a specified tolerance.

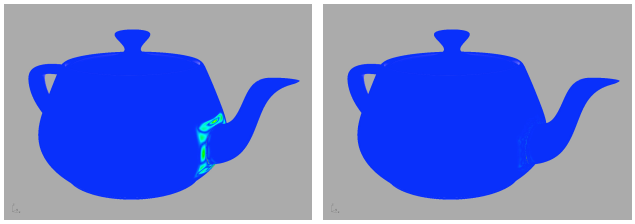


(a) 108 Control points. $\epsilon = 1$. Error = 1.5×10^{-3} (b) 224 Control points. $\epsilon = .5$. Error = 5.5×10^{-4}



(c) 338 Control points. $\epsilon = .25$. Error = 2.2×10^{-4} (d) 495 Control points. $\epsilon = .125$. Error = 7.5×10^{-5}

Figure 13: *Approximation Error for the Teapot Body.*



(a) Error for $\epsilon = 1$. Max error = $1.5 \times 10{-}3$ (b) Error for $\epsilon = .25$. Max error = $2.2 \times 10{-}4$.

Figure 14: *Error Plots. Dark Blue Denotes Zero Error.*

The perturbation errors reported in the captions in Figure 13 are relative to a teapot that is one unit wide. Hence, the error in Figure 13.a is about one tenth of one percent of the width of the teapot.

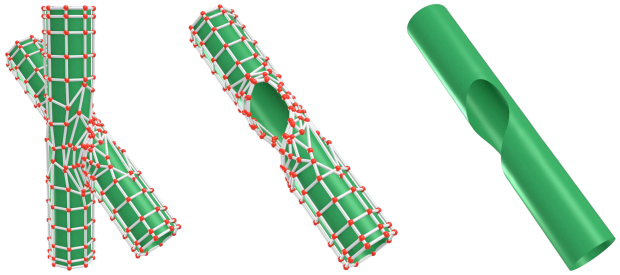


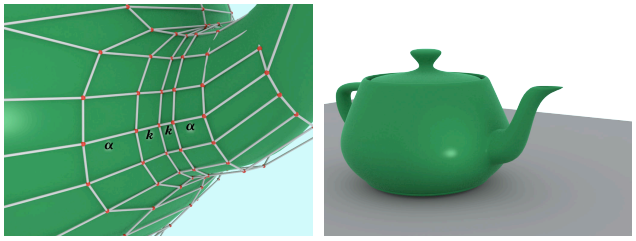
Figure 15: *Trimless T-Spline Cylinders. Relative Error* = 0.0009

Fillets

Figure 16.a shows the knot intervals adjacent to the intersection curve. Immediately after the merge is completed, $k = 0$. This creates a triple knot at the intersection curve and forces the T-Spline to be C^0 along the intersection curve. If the value of k is changed to a small positive value, the C^0 crease along the intersection curve is changed into a C^2 fillet whose radius increases with k.

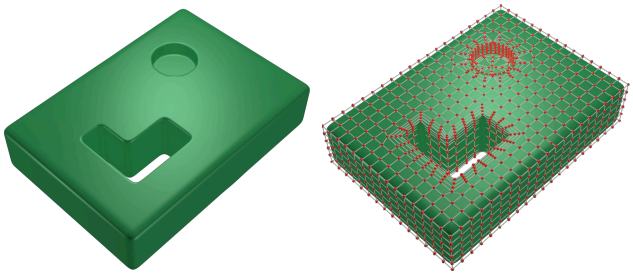
Feature Lines

Feature lines in NURBS models must follow iso-parameter curves. T-Splines allow for sharp features along portions of iso-parameter



(a) Knot intervals following merge. Initially, $k = 0$. (b) Setting $k = 0.1$ to create a small fillet.

Figure 16: *Fillet.*

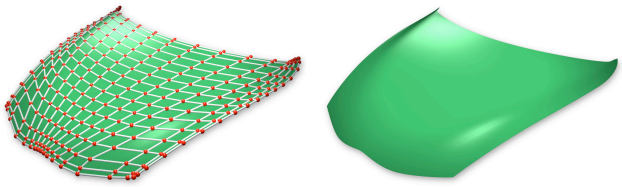


(a) Untrimmed T-Spline. (b) With control grid.

Figure 17: *Filleted CSG Object as an Untrimmed T-Spline.*

curves [Sederberg et al. 2003]. Subdivision surfaces don't have a strong notion of iso-parameter curves, and hence procedures have been devised for placing a fillet or feature curve in an arbitrary direction on a subdivision surface [DeRose et al. 1998]. A general-purpose tool for placing arbitrary feature lines on any geometric model, based on a deformation, is described in [Singh and Fiume 1998].

To create an arbitrary a feature curve on a NURBS surface, using the algorithms presented in Sections 3 and 4, the feature curve is drawn as a parametric curve in parameter space of the surface, much like a trimming curve except that the curve need not be closed. The curve is then processed as discussed in Section 3: An axis-aligned bounding polygon is found, and topological and fitting operations are performed. The resulting control points can then be moved to create the desired feature.



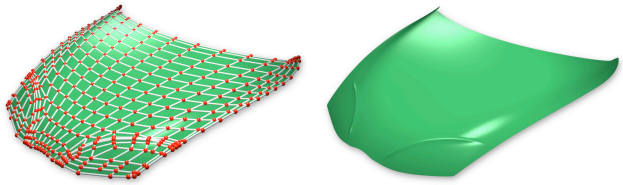
(a) Control grid. (b) Surface.

Figure 18: *NURBS Car Hood.*

We illustrate the procedure using the model of a NURBS car hood in Figure 18. Figure 19.a shows the control points that result from the process in Section 3, after they have been adjusted to create the desired feature lines, and Figure 19.b shows the resulting surface.

6 Discussion

The modeling tools presented in this paper extend the capabilities of T-Splines to express Booleans, fillets, and arbitrary feature curves in a single unified framework. The geometric models thus created are watertight, C^2 (or C^1 or C^0 if multiple knots are specified),



(a) Control grid. (b) Surface.

Figure 19: *T-Spline Car Hood with Detail.*

and can be exported without translation error as a finite collection of NURBS patches, making them compatible with CAD industry standards. The models are editable in that control points can be adjusted and the surface will remain C^2. These results can help to streamline the CAD modeling–analysis pipeline. In addition, these tools introduce new design workflows into the styling and CAD industries, allowing NURBS modelers to continue to style their models even after Booleans have been performed.

The paper also presents an enhanced merging capability involving an augmentation of the definition of T-Splines to support different knot intervals on opposite sides of the same control polygon face. This enables the merging of two NURBS or T-Splines surfaces whose mating curves are parametrized differently, a case not handled well in [Sederberg et al. 2003].

Our process for merging two trimmed NURBS surfaces into an untrimmed T-Spline involves a perturbation of the original surfaces. The perturbations can be limited to an arbitrarily narrow strip. Tighter tolerances demand more control points, making the resulting T-Spline more difficult to edit, although if the designer's intent is to ultimately edit the resulting T-Spline, a high initial tolerance may not be as crucial.

This paper invites future research on several fronts. Section 3 discusses possible lines of research for finding an efficient algorithm for computing the green control points, the problem we view as most pressing. Also called for is a rigorous analysis of approximation error. What is the relationship between ϵ, positional error, and normal-vector error? What is the convergence rate?

The paper focuses on two intersecting surfaces. Details of how to handle three or more intersecting surfaces are not presented and invite further study. The existing algorithms should extend readily to handle most cases where all intersections are to be computed simultaneously. The case where an untrimmed T-Spline that represents two intersecting surfaces is later intersected by a third NURBS or T-Spline is more challenging because it can involve the intersection of NU-NURBS or of faces next to extraordinary points. It is also not clear how the error might propagate upon repeated such intersections.

The paper only addresses non-singular intersection curves. Full treatment of intersection curves that self-intersect is another topic of further study.

7 Acknowledgements

Nicholas North and Adam Helps provided invaluable assistance with implementation and figures.

References

BJORCK, A. 1996. *Numerical Methods for Least Squares Problems*. SIAM.

DEROSE, T. D., KASS, M., AND TRUONG, T. 1998. Subdivision surfaces in character animation. In *Proceedings of SIGGRAPH 1998*, Computer Graphics Proceedings, Annual Conference Series, 85–94.

FAROUKI, R. T., HAN, C. Y., HASS, J., AND SEDERBERG, T. W. 2004. Topologically consistent trimmed surface approximations based on triangular patches. *Computer Aided Geometric Design 21*, 5, 459–478.

FAROUKI, R. T. 1999. Closing the gap between CAD model and downstream application (report on the SIAM Workshop on Integration of CAD and CFD, UC Davis, April 12–13, 1999). *SIAM News 32*, 5, 1–3.

HUNTER, G. M., AND STEIGLITZ, K. 1979. Operations on images using quad trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence 1*, 2 (April), 145–153.

KASIK, D. J., BUXTON, W., AND FERGUSON, D. R. 2005. Ten CAD model challenges. *IEEE Computer Graphics and Applications 25*, 2, 81–92.

KATZ, S., AND SEDERBERG, T. W. 1988. Genus of the intersection curve of two rational surface patches. *Computer Aided Geometric Design 5*, 253–258.

KRISHNAN, S., AND MANOCHA, D. 1996. Efficient representations and techniques for computing b-rep's of csg models with nurbs primitives. In *Proc. of CSG'96*, 101–122.

KRISHNAN, S., AND MANOCHA, D. 1997. An efficient surface intersection algorithm based on lower-dimensional formulation. *ACM Transactions on Graphics 16*, 1 (Jan.), 74–106.

KRISHNAN, S., MANOCHA, D., GOPI, M., AND KEYSER, J. 2001. Boole: A boundary evaluation system for Boolean combinations of sculptured solids. *International Journal on Computational Geometry and Applications 11*, 1, 105–144.

KRISTJANSSON, D., BIERMANN, H., AND ZORIN, D. 2001. Approximate Boolean operations on free-form solids. In *Proceedings of ACM SIGGRAPH 2001*, E. Fiume, Ed., Computer Graphics Proceedings, Annual Conference Series, 185–194.

KUMAR, S. 1996. *Interactive rendering of parametric spline surfaces*. PhD thesis, The University of North Carolina at Chapel Hill.

LITKE, N., LEVIN, A., AND SCHRÖDER, P. 2001. Trimming for subdivision surfaces. *Computer Aided Geometric Design 18*, 5 (June), 463–481.

LOOP, C. 2004. Second order smoothness over extraordinary vertices. In *Eurographics / ACM SIGGRAPH Symposium on Geometry Processing*, 165–174.

MORETON, H. 2001. Watertight tessellation using forward differencing. In *HWWS '01: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, ACM, New York, NY, USA, 25–32.

MÜLLER, K., REUSCHE, L., AND FELLNER, D. 2006. Extended subdivision surfaces: Building a bridge between NURBS and Catmull-Clark surfaces. *ACM Transactions on Graphics 25*, 2 (Apr.), 268–292.

1999. Planning Report: Interoperability Cost Analysis of the US Automotive Supply Chain. National Institute of Standards and Technology.

PATRIKALAKIS, N. M., AND MAEKAWA, T. 2002. Intersection problems. In *Handbook of Computer Aided Geometric Design*, North-Holland, G. Farin, J. Hoschek, and M.-S. Kim, Eds., 623–649.

PETERS, J. 2000. Patching Catmull-Clark meshes. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, 255–258.

SAMET, H. 1984. The quadtree and related hierarchical data structures. *ACM Computing Surveys 16*, 2, 187–260.

SEDERBERG, T. W., LI, X., LIN, H., AND FINNIGAN, G. T. Nonuniform NURBS. *In Preparation*.

SEDERBERG, T., ANDERSON, D., AND GOLDMAN, R. 1984. Implicit representation of parametric curves and surfaces. *Computer Vision, Graphics and Image Processing 28*, 72–84.

SEDERBERG, T. W., ZHENG, J., SEWELL, D., AND SABIN, M. A. 1998. Non-uniform recursive subdivision surfaces. In *Proceedings of SIGGRAPH 1998*, Computer Graphics Proceedings, Annual Conference Series, 387–394.

SEDERBERG, T. W., ZHENG, J., BAKENOV, A., AND NASRI, A. 2003. T-Splines and T-NURCCs. *ACM Transactions on Graphics 22*, 3 (July), 477–484.

SEDERBERG, T. W., CARDON, D. L., FINNIGAN, G. T., NORTH, N. S., ZHENG, J., AND LYCHE, T. 2004. T-spline simplification and local refinement. *ACM Transactions on Graphics 23*, 3 (August).

SINGH, K., AND FIUME, E. L. 1998. Wires: A geometric deformation technique. In *Proceedings of SIGGRAPH 1998*, Computer Graphics Proceedings, Annual Conference Series, 405–414.

SONG, Q., AND WANG, J. 2007. Generating g^n parametric blending surfaces based on partial reparameterization of base surfaces. *Comput. Aided Des. 39*, 11, 953–963.

SONG, X., SEDERBERG, T. W., ZHENG, J., FAROUKI, R. T., AND HASS, J. 2004. Linear perturbation methods for topologically consistent representations of free-form surface intersections. *Computer Aided Geometric Design 21*, 3, 303–319.

STAM, J. 1998. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. In *Proceedings of SIGGRAPH 1998*, Computer Graphics Proceedings, Annual Conference Series, 395–404.

WANG, W., POTTMANN, H., AND LIU, Y. 2006. Fitting B-spline curves to point clouds by curvature-based squared distance minimization. *ACM Transactions on Graphics 25*, 2 (Apr.), 214–238.