Robust Tessellation of Trimmed Rational B-Spline Surface Patches

Wonjoon Cho[1] **Takashi Maekawa**[1] **Nicholas M. Patrikalakis**[1]
Jaime Peraire[2]

Massachusetts Institute of Technology
[1]Department of Ocean Engineering, Design Laboratory
[2]Department of Aeronautics and Astronautics, Fluid Dynamics Research Laboratory
Cambridge, MA 02139-4307, USA

Design Laboratory Memorandum 97-16

November 14, 1997

Abstract

We present an unstructured triangular mesh generation algorithm that approximates a set of mutually non-intersecting simple trimmed rational B-spline surface patches within a user specified geometric tolerance. The proposed method uses numerically robust interval geometric representations/computations and also addresses the problem of topological consistency (homeomorphism) between the exact geometry and its approximation. Those are among the most important outstanding issues in geometry approximation problems. Our surface tessellation algorithm is based on the unstructured Delaunay mesh approach which leads to an efficient adaptive triangulation. A robust decision criterion is utilized to prevent possible failures in the conventional Delaunay triangulation. To satisfy the prescribed geometric tolerance, an adaptive node insertion algorithm is employed. Unstructured triangular meshes for free-form surfaces frequently involve triangles with high aspect ratio and accordingly, result in ill-conditioned meshing. Our proposed algorithm constructs 2D triangulation domains which sufficiently preserve the shape of triangles when mapped into 3D space and furthermore, the algorithm provides an efficient method that explicitly controls the aspect ratio of the triangular elements.

Keywords : trimmed rational B-spline patch; well-conditioned unstructured meshing; homeomorphism; robustness.

1 Introduction

Modern CAD/CAM systems allow users to access specific application programs for performing several tasks including finite element or boundary element meshing for analysis, fundamental geometric calculations such as integrals, and rendering objects on a graphical display. These application programs often operate on approximate piecewise lower order representations of the exact geometric definition. For this reason, and in order to support reliable data exchange between various geometric modelers or design and fabrication systems, a *robust piecewise linear approximation method* of high order and procedural surfaces is necessary. For complicated 3D curved objects, a boundary representation (B-rep) model coupled with constructive solid geometry (CSG) is regarded as the most promising representation method in the current CAD/CAM systems. Rational B-spline curves and trimmed rational B-spline surfaces play a fundamental role in B-rep since they represent edges and faces of the complicated 3D curved objects [26]. Furthermore, they have become the standard for curve and surface description in various fields such as visual arts and virtual reality applications [27]. One of the most appropriate ways to accomplish a piecewise linear approximation of trimmed rational B-spline patches is a *triangular* tessellation within a specified geometric tolerance. Triangular tessellation allows topological simplicity which enables *local mesh adaptivity* in finite element analysis [21, 5, 25] and it also provides a unique database i.e., the same triangular facets can be used for rendering [30] as well as for other geometric computations or general analysis. Current solid free-form fabrication (SFF) methods [20] for example, use triangular faceting for data exchange and manufacturing. There exist several triangulation algorithms for trimmed parametric surface patches such as those proposed by Sheng and Hirsch [31], Shimada and Gossard [32], Gürsoy and Patrikalakis [13, 14], Boender *et al.* [6], Piegl and Richard [26], Klein and Straßer [19], Austin *et al.* [3]. Current surface tessellation algorithms, however, suffer from lack of *robustness* in geometric modeling systems that operate in floating point arithmetic; and grid generation systems with considerable emphasis on local differential geometric methods do not have a *topological* concept that is crucial for fully automatic mesh generation. It is also well-known that triangular tessellation for free-form surface patches often results in *ill-conditioned* meshing caused by elements of high aspect ratio.

Automatic mesh generators are designed so that they can be used as a black box and hence can be integrated into larger systems. One of the most important issues concerning existing generators is *numerical robustness and topological consistency.* For complex three-dimensional geometries, the sources of failure fall into the following two categories: (1) numerical inaccuracies in the geometric definition/computation and (2) topological inconsistency between the input geometry and its approximation. Current state-of-the-art CAD systems used to create and interrogate curved objects are based on geometric solid modeling systems that operate in floating point arithmetic (FPA). These systems will frequently fail as a result of the limited precision of geometrically crucial computations, which is inherent to the internal representation of floating point numbers. In our approach, we employ a new representation scheme based on *interval* parametric curves and surfaces [22, 15, 16, 17, 18, 34] implemented in *rounded interval arithmetic* (RIA) [24, 1] which insures numerical robustness and provides results with verifiable numeric certainty. For example, we have developed a robust interval Delaunay test [9] to overcome serious problems such as degeneracy, non-conformance and topologically invalid triangulations in the conventional Bowyer-Watson algorithm operating in FPA. The second problem, possible topological inconsistency between the input geometry and its approximation, should be carefully identified and eliminated since such a topological inconsistency immediately results in *system failure* e.g., during solution of computational mechanics problems in finite element analysis, or in the solid free-form fabrication (SFF) methods. For simple composite surfaces which do not have any inappropriate intersection, corresponding approximants also should not have any improperly intersecting elements to achieve a *homeomorphism* [2] between the exact input geometry and its approximation [7]. Therefore, a robust intersection test relying on RIA and the ability to perform local refinement of the approximation is required to avoid inappropriate intersections. Unstructured triangular meshes for free-form surfaces frequently suffer from ill-shaped elements with *high aspect ratio* (AR) [26]. If such ill-conditioned meshes are used for further geometric computations or general analysis, they may cause serious numerical problems [33, 4]. Many existing triangulation algorithms for trimmed parametric surfaces perform a triangulation in the original parametric space, and map each triangle vertex into 3D space through surface equations. It is well-known that such a mapping will inevitably produce *distortion and/or stretching* in general. Our proposed surface meshing algorithm constructs 2D triangulation domains [10] which sufficiently preserve the shape of triangles during mapping process and furthermore, it provides an efficient method that explicitly controls the AR of the triangular elements.

In our earlier work [11], we have considered topologically reliable approximation of trimmed *polynomial* surface patches. In the present paper we extend the surface tessellation algorithm to more general *rational B-spline* surface patches. Major differences are: In [11], (1) geometrically significant points of input geometry were computed for use in the approximation but in this paper this step is avoided for computational simplicity; (2) tight upper bound of the approximation error was employed but in this paper a simpler error measure is used for rational B-spline patches for reasons of computational convenience and efficiency. This paper is structured as follows: In Section 2, we describe auxiliary algorithms developed for our surface meshing method. Section 3 presents an initial triangulation algorithm which reflects the boundary information of input surface patches. In Section 4, we refine the initial approximation until every approximating triangle satisfies the prescribed geometric tolerance. Section 5 presents a method to improve the aspect ratio of the approximating triangles. Section 6 provides a technique to detect and eliminate inappropriate intersections of the approximating triangles. In Section 7 we present some examples and illustrate the performance of the method. Finally, Section 8 concludes the paper with some recommendations for further work.

2 Auxiliary Algorithms

2.1 Construction of Triangulation Domain

Many existing surface meshing algorithms perform the triangulation in the parametric space, typically a unit square, and map the generated vertices into 3D space through surface equations. Even if the quality of triangular elements is assumed to be sufficiently high in the parametric space, it can easily deteriorate through the mapping process due to distortion and/or stretching. To overcome this adverse effect, we introduced an auxiliary *planar domain of triangulation* for trimmed surface patches [8, 10]. Approximate locally isometric mapping between the input trimmed surface patch and its planar triangulation domain is achieved by minimizing a locally isometric mapping error function E. Our algorithm also guarantees a homeomorphism between the triangulation domain and parametric space (and hence given surface patch) by robustly removing the possibility of self-intersection of the approximately developed surface patch. In case the minimized mapping error $min(E)$ is unsatisfactory, we bisect the input surface patch and repeat the developing process until $min(E)$ is within a certain threshold, as detailed in [8, 10]. As an example, we consider a trimmed NURBS surface patch of degrees 2 and 3 in the u, v parametric directions respectively, as shown in Figure 1. The corresponding triangulation domain is shown in Figure 1-(f).

2.2 Linear Approximation of Boundary Curves

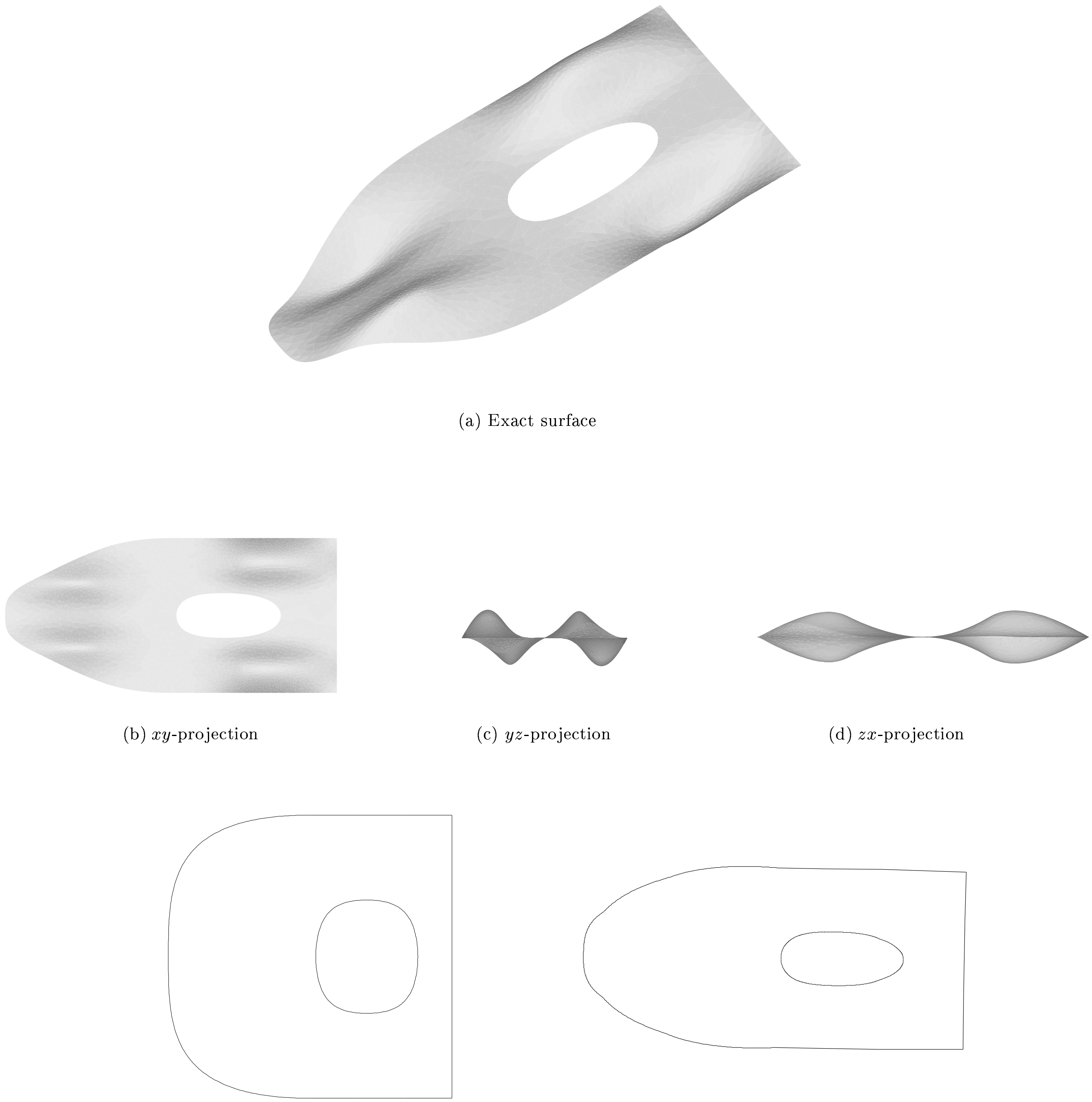
To determine initial boundary nodes, we perform a *topologically reliable linear approximation* of the exterior and interior boundary loops of the input rational B-spline surface patches within a prescribed approximation tolerance [7, 8]. For simplicity, we skip the preliminary approximation step of our existing algorithm [7] that computes the geometrically significant points of input boundary curves for use in the approximation. Then, we refine the approximation until every approximating segment satisfies the user specified approximation tolerance – *main approximation step* in [7]. Also for convenience, we just measure the maximum positional difference between the approximating segment and the corresponding surface trimming curve at a finite number of isoparametric points to determine the approximation error instead of using the convex hull property described in [7]. Finally, for each pair of linear approximating segments, a robust intersection test coupled with an efficient bucketing technique is performed to identify possible inappropriate intersections. If these intersections exist, further local refinement of the approximation is performed to guarantee a homeomorphism between a set of exact input curves and their approximation.

2.3 Robust Interval Planar Delaunay Test

Our surface tessellation algorithm is based on the unstructured Delaunay mesh approach which leads to an efficient adaptive triangulation. One of the floating point operations invoked in the Delaunay algorithm in E^2 is the *planar Delaunay test* which consists of determining whether a newly inserted point P lies inside a circumcircle of an existing triangle i.e.,

$$d_p^2 = (x_p - x_i)^2 + (y_p - y_i)^2 \overset{?}{<} r_i^2 \tag{1}$$

where (x_p, y_p) are the coordinates of point P and r_i is the radius of a circumcircle centered at Voronoi vertex V_i with coordinates (x_i, y_i). However, owing to floating point imprecision, the Delaunay test may incorrectly recognize P as falling inside or outside circumcircle. Such an incorrect decision may cause two types of failures: (1) generation of degenerate triangles, and (2) non-conforming triangulations. Degenerate triangles will cause serious numerical problems, and in case of free-form surface meshing via auxiliary planar domain of triangulation e.g., parametric space, the non-conforming triangulation will generate inappropriate *gaps* (topological inconsistency) when each vertex is mapped into three-dimensional space. To prevent these possible failures, we have introduced a new decision criterion named *interval Delaunay test* [8, 9]:



(a) Exact surface

(b) xy-projection

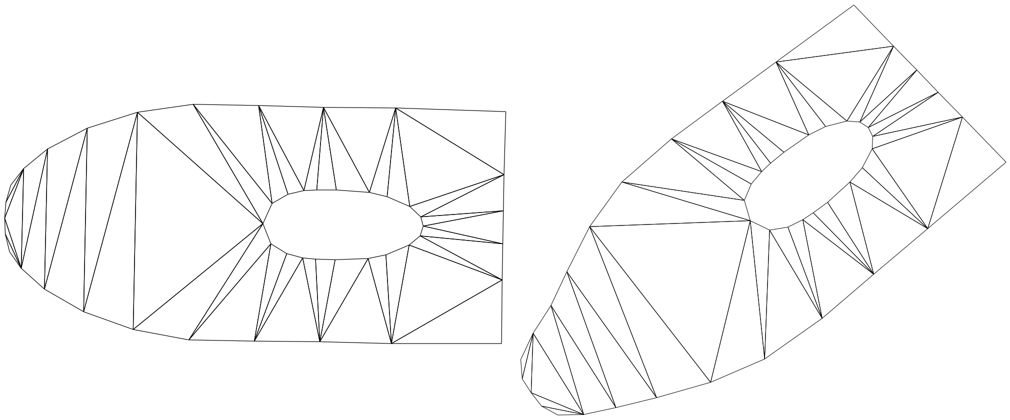
(c) yz-projection

(d) zx-projection

(e) Parametric space

(f) Triangulation domain

Figure 1: Trimmed NURBS patch and its triangulation domain



(a) Triangulation domain (b) 3D space

Figure 2: Initial triangulation ($N_T = 47$)

Definition 2.1 *Given a newly inserted point $P(x_p, y_p)$, Voronoi vertex $V_i(x_i, y_i)$ and the corresponding Delaunay triangle $T_i(ABC)$, we compute d_p^2 and r_i^2 defined in Eq.* (1) *using rounded interval arithmetic (RIA) and if the following two conditions* **a**, **b** *are satisfied, then we decide P is within the circumcircle of T_i :*

a. $(d_p^2)_l \leq (r_i^2)_u$,

b. $\begin{cases} \textbf{1.}\ \textit{none of the triangles } \triangle ABP,\ \triangle BCP \textit{ and } \triangle CAP \textit{ will degenerate, } \underline{\textbf{or}} \\ \textbf{2.}\ \textit{if a triangle degenerates e.g., } \triangle BCP, \textit{ then } P \textit{ should be within edge } BC, \end{cases}$

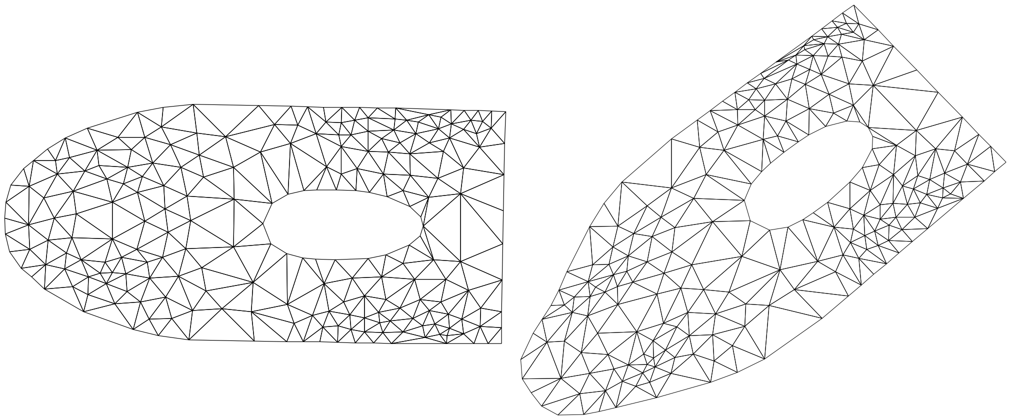
where $(d_p^2)_l$, $(r_i^2)_u$ are the lower and upper value of the rounded interval numbers d_p^2, r_i^2.

When performing the degeneracy test in condition **b**, we compute the signed area S of a triangle using RIA and if S contains zero in its interval, then we decide the triangle degenerates.

3 Initial Triangulation

For each input trimmed surface patch, we perform an approximate locally isometric mapping as described in Section 2.1 to construct the *triangulation domain.* As stated in Section 2.2, *boundary nodes* of each input patch are determined by a piecewise linear approximation of the exterior and interior boundary loops within a prescribed approximation tolerance. Notice that for a common boundary curve between two adjacent patches, we perform the linear approximation once and transfer the corresponding boundary data to the mate boundary segment by utilizing half-edge data structure [23] coupled with input boundary representation (B-rep). By mapping those nodes onto each triangulation domain, we obtain the initial boundary nodes for the surface triangulation. This is followed by boundary-conforming Delaunay triangulation coupled with the robust interval Delaunay test described in Section 2.3.

Two different types of triangles can be identified after the initial triangulation – the triangles internal to the domain of triangulation and the triangles external to it. The external triangles belong to the trimmed part of the triangulation domain or in the background region outside the exterior polygonal boundary loop. Figure 2 illustrates an initial triangulation after removing the external triangles for our example shown in Figure 1. 47 triangles are generated from the linear approximation of boundary loops with normalized approximation tolerance 10^{-2}.



(a) Triangulation domain

(b) 3D space

Figure 3: ϵ-triangulation ($N_T = 397$)

4 ϵ-Triangulation

For each approximating triangle resulting from the initial triangulation, we compute an approximation error. In practice, we define the approximation error δ by the maximum of the absolute position difference at a finite number N of isoparametric points $\mathbf{s}_{i,j,k}$ on the approximating triangle and the corresponding points $\mathbf{r}_{i,j,k}$ on the exact rational B-spline patch i.e.,

$$\delta := \max_{i,j,k} |\mathbf{r}_{i,j,k} - \mathbf{s}_{i,j,k}|, \tag{2}$$

where integers $i, j, k \geq 0$, $i + j + k = L$, $N = \frac{(L+1)(L+2)}{2}$, and L is a user-input. The triangle with largest approximation error δ_{max} is then chosen from the triangulation domain. If δ_{max} is not within a given approximation tolerance ϵ, a new node is placed on the *Voronoi vertex* of the triangle and the triangulation is updated using constrained Bowyer-Watson algorithm based on interval Delaunay test [9]. We next compute δ for each newly generated triangle and repeat the same procedure until every approximating triangle satisfies ϵ-tolerance. We call this procedure *ϵ-triangulation.* A similar Voronoi-vertex point insertion algorithm was introduced by Rebay [29] in his planar mesh generation algorithm. We notice the node insertion at the Voronoi vertex of a triangle clearly eliminates the triangle from the existing Delaunay structure and refines the mesh.

Note that a Voronoi vertex of a triangle may exist outside the triangulation domain. This usually happens if the *longest* edge of an *obtuse* triangle is on the boundary loop. In such a case, instead of Voronoi vertex, we insert a node at the midpoint of the corresponding *exact* boundary segment and update the triangulation utilizing the *node repositioning* technique described in [8, 11]. Otherwise, we just insert a new node at the midpoint of the longest edge of the triangle and update the triangulation.

Figure 3 shows the result of ϵ-triangulation refined from Figure 2 with normalized $\epsilon = 10^{-2}$. Total number of triangles (N_T) is 397 and the worst and average approximation error are 0.99×10^{-2} and 0.59×10^{-2}, respectively.

5 AR-Improving Triangulation

As stated before, unstructured triangular meshes for free-form surfaces frequently suffer from triangles with high aspect ratio [26]. If such low quality elements are used for further geometric computations or general analysis, they may cause serious numerical problems [4, 33]. The *aspect ratio* (AR) of a triangle is defined by *the ratio of the length of the longest edge to the height,* where the height is the minimum distance from a vertex to its opposite side. It is obvious that the *equilateral* triangle has the *least* AR $= \frac{2}{\sqrt{3}} \approx 1.155$. Our

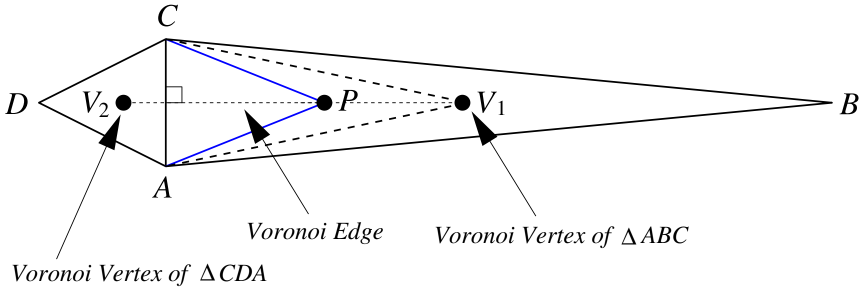


Figure 4: Node insertion in AR-improving triangulation

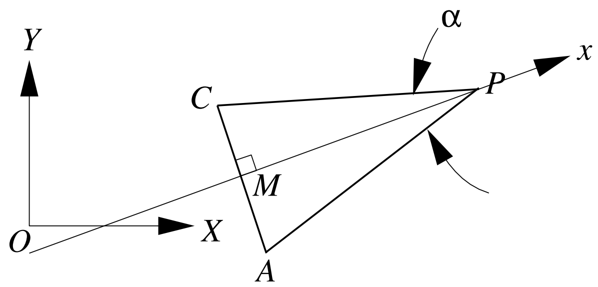


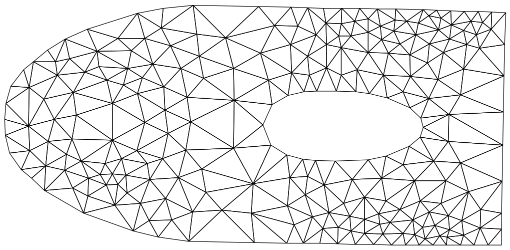
Figure 5: Isosceles triangle APC

free-form surface meshing algorithm constructs planar domains of triangulation which sufficiently preserve the shape of triangular elements when mapped into three-dimensional space. This feature enables us to achieve a well-conditioned triangulation in three-dimensional space by improving AR of triangles *on the planar domain of triangulation.*

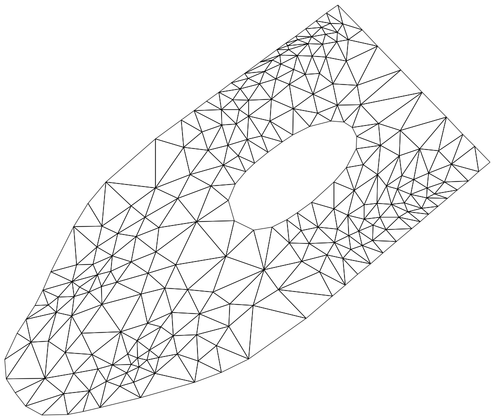
Our AR-improving algorithm utilizes the concept of *Voronoi-edge point insertion algorithm* [29], where a new node is repeatedly inserted on the Voronoi edge to generate a new triangle whose circumcircle radius is identical to the prescribed limit. We now illustrate the procedure of *AR-improving triangulation* in detail. After the ϵ-triangulation has been constructed, the internal triangles on the triangulation domain are further classified into two types of triangles depending on their AR – triangles already satisfying and not yet satisfying the *prescribed threshold* (τ_{AR}) for AR. These two types of internal triangles are referred to as done and undone triangles, respectively. Since we will insert a node by considering an undone triangle whose neighbors are done or external triangles, we need to introduce a further classification of the undone triangles namely, active and waiting triangles. An active triangle is an undone triangle having at least one done or external triangle among its neighbors. Finally, all the undone triangles that are not active are called waiting triangles. Therefore, a waiting triangle is surrounded by active or other waiting triangles. Before we start improving AR of triangles, a *feasibility test* is performed for each triangle T^* two of whose edges correspond to the boundary loop segments. For a triangle whose least internal angle $\theta \ll 1$, its AR $\approx \frac{1}{\sin\theta} \approx \frac{1}{\theta}$. Therefore, we compute the internal angle θ made by two boundary loop segments of T^* and if $\theta < \frac{1}{\tau_{\mathrm{AR}}}$ then, we regard the AR of T^* as *not improvable* and mark it to be done, in other words we do not attempt to improve the AR of such T^*.

After classifying each internal triangle obtained from the ϵ-triangulation, we pick up the triangle with the largest AR among the active triangles. A new node is inserted by considering the Voronoi edge shared by the chosen active triangle and one of its done or external neighboring triangles. If more than one done or external neighbors exist, the Voronoi edge associated with the *shortest* edge of the active triangle is chosen. As shown in Figure 4, the position of new node P along the Voronoi edge V_1V_2 is chosen in an attempt to generate an *isosceles* triangle APC whose AR is equal to the prescribed threshold τ_{AR} ($\geq \frac{2}{\sqrt{3}}$). Furthermore, we assume edge CA to be the shortest edge of the newly generated isosceles triangle APC, which means an angle α between two edges AP and PC should not be greater than 60°. Hence, with reference to Figure 5, the position of P is given by,

$$P : (P_X, P_Y) = \left(\frac{A_X + C_X - (A_Y - C_Y)\nu}{2}, \frac{A_Y + C_Y + (A_X - C_X)\nu}{2}\right), \tag{3}$$



(a) Triangulation domain



(b) 3D space

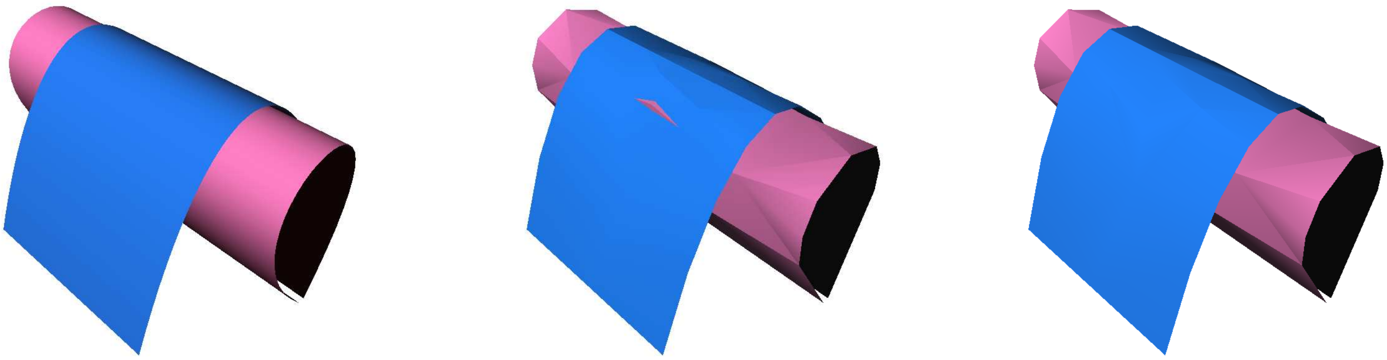
Figure 6: AR-improving triangulation ($N_T = 433$)

where $\nu = \tau_{\text{AR}} + \sqrt{\tau_{\text{AR}}^2 - 1}$ and (A_X, A_Y), (C_X, C_Y) are coordinates of points A and C, respectively.

Notice that new node P computed by Eq. (3) may not be within the Voronoi edge between the chosen active triangle and the neighboring done or external triangle. In other words, this is the case of Voronoi edge V_2V_1 being too short with respect to V_2P. In such a case, we insert a new node at Voronoi vertex V_1 of triangle ABC to delete the currently worst active triangle from the existing Delaunay structure. Another exceptional case occurs if a newly inserted node lies outside the triangulation domain – same treatment can be applied as in Section 4.

We note that the AR-improving procedure *may* generate a number of small triangles to meet a prescribed τ_{AR}, if the ϵ-triangulation results in *globally* very ill-shaped meshing and τ_{AR} is *too ideal.* To remedy this problem, it is desirable to introduce a threshold τ_r which bounds the *smallest size* of a triangle in AR-improving procedure. The most appropriate quantity which measures the size of a triangle is the circumcircle radius r of a triangle. Apart from its close geometric relation to the size of a triangle, it is introduced for the efficiency of the algorithm – r is already computed in Bowyer-Watson algorithm and hence, no additional computations are required to determine r. The threshold τ_r could be an *arbitrary* user-input or may have a dynamic relation with the approximation tolerance ϵ, as detailed in [8]. Given τ_r, if the currently chosen worst active triangle T has $r < \tau_r$, then T is marked as done and the second worst active triangle is chosen to repeat the AR-improving procedure.

Based on the node insertion strategy described so far, we repeatedly place a node on the triangulation domain and update the triangulation until every internal triangle becomes done. Figure 6 shows the result of AR-improving procedure refined from the ϵ-triangulation shown in Figures 3. We input τ_{AR} to be 3, which guarantees that any internal angle θ of a triangle on the triangulation domain should be $20° < \theta < 113°$. We also set normalized $\tau_r = 10^{-2}$. Total number of approximating triangles (N_T) is 433 and the worst (AR_{max}) and average (AR_{avg}) aspect ratio of triangles is 2.97, 1.77 on the triangulation domain, and 3.76, 1.90 in the three-dimensional space, respectively. The number of triangles (N_{fail}) whose AR's are greater than τ_{AR} is 0(4) on the triangulation domain (in the three-dimensional space), respectively. In other words, every triangle on the triangulation domain satisfies τ_{AR} and furthermore, only 1% of the approximating triangles in the three-dimensional space do not meet τ_{AR}. In the ϵ-triangulation: $N_T = 397$, $\text{AR}_{max} = 9.44$, $\text{AR}_{avg} = 1.93$ on the triangulation domain; $\text{AR}_{max} = 7.67$, $\text{AR}_{avg} = 1.96$ in the three-dimensional space; and $N_{fail} = 20$ on the triangulation domain and $N_{fail} = 20$ in the three-dimensional space. *Our satisfactory result after AR-improving triangulation is due to the coupled effects of the (1) features of Delaunay based triangulation; (2) AR-improving procedure on the planar domain of triangulation and; (3) preservation of triangles' shape during mapping process from the triangulation domain into three-dimensional space.*



(a) Exact surface (b) Topologically invalid approximation (c) Final triangulation

Figure 7: Intersection test and final triangulation ($\epsilon = 10^{-1},\ \tau_{\mathrm{AR}} = 3$)

6 Intersection Test and Final Triangulation

As proven in [8], a simple composite surface X comprising of m conforming surface patches is homeomorphic to the corresponding set Y of n planar approximating triangles which do not have any inappropriate intersection nor have any inappropriate gap. During the surface triangulation procedure based on the robust *interval* Delaunay test [9], the *inappropriate gap* is prevented by utilizing the half edge data structure coupled with *interval* boundary representations/evaluations [16, 17]. However, *inappropriate intersections* may exist between the approximating triangles in E^3 e.g., in case of global distance function features of the input surface such as *constrictions* as illustrated in Figures 7-(a,b). If such inappropriate intersections exist, they necessitate further local refinement of the approximation until every approximating triangle does not have any inappropriate intersection – *final triangulation*.

To perform an intersection test based on the bucketing technique [12, 7], we first construct $\mathcal{O}(n)$ uniform buckets that will contain n approximating triangles. The procedure of constructing buckets is analogous to what was described in [7]. Once the buckets are constructed, each approximating triangle is transformed with respect to the bucket coordinates. To associate each triangle with the buckets, we utilize an efficient and robust method given by Ratscheck and Rokne [28]. A robust intersection test computed in rounded interval arithmetic (RIA) [24, 1] is next performed for each triangle in the same bucket to identify possible inappropriate intersections. We need to keep in mind the intersection algorithms operating in *floating point arithmetic* may *fail* even for *linear* geometry. If a pair of intersecting triangles are found, we pick up the corresponding triangles T_p and T_q in the triangulation domain. Additional nodes are inserted on the midpoint of the longest edge of each T_p, T_q and a local refinement of the approximation is performed by bisecting T_p and T_q together with their neighboring triangles. This procedure is then repeated until every approximating triangle does not inappropriately intersect each other. The *final triangulation* provides a *topologically consistent* approximation to the exact input composite surface, as shown in Figure 7-(c).

7 Examples

We have implemented the proposed surface tessellation algorithm on a graphics workstation running at 150 MHz. The approximation tolerance ϵ is normalized by the input surface length scale defined as the cubic root of the rectangular box bounding the control polyhedra of the surface patches in the given coordinate system.

Our first example shown in Figure 8 is a trimmed bi-cubic rational B-spline surface patch. Table 1 together with Figures 8-(e,f,g) show the result of meshing depending on the approximation tolerance $\epsilon = 10^{-1}, 10^{-2}, 10^{-3}$ with a given AR-threshold $\tau_{\mathrm{AR}} = 4$. If the approximation tolerance ϵ is not sufficiently tight, most of the time cost is spent on the initial triangulation. Time cost C_{init} of the initial triangulation is not so sensitive to the change of ϵ since the dominant procedure is the construction of triangulation domain. As ϵ becomes tighter, the time cost C_ϵ of the ϵ-triangulation becomes compara-

ϵ	CPU (sec)					N_{total}	$\frac{N_{fail}}{N_{total}} \times 100\,(\%)$	AR_{avg}	AR_{max}
	C_{init}	C_ϵ	C_{AR}	C_{fin}	C_{total}				
10^{-1}	9.05	0.1	0.02	0.03	9.2	30	3.3	2.3	4.5
10^{-2}	9.32	4.37	0.37	0.47	14.53	248	1.6	2.1	4.9
10^{-3}	10.03	82.6	0.92	5.64	99.19	2200	0.1	1.8	4.8

Table 1: Trimmed bi-cubic NURBS surface patch; $\tau_{\mathrm{AR}} = 4$

τ_{AR}	CPU (sec)	N_{AR}	$\frac{N_{fail}}{N_{total}} \times 100\,(\%)$	AR_{avg}	AR_{max}
3.5	0.48	31	3.1	2.0	4.9
4	0.37	26	1.6	2.1	4.9
4.5	0.32	22	1.2	2.1	4.9

Table 2: Trimmed bi-cubic NURBS surface patch; AR-improving triangulation, $\epsilon = 10^{-2}$, $N_\epsilon = 222$

ble and dominant with respect to C_{init}. Also in Table 1, C_{AR}, C_{fin} and C_{total} show the time cost of the AR-improving procedure, intersection test/final triangulation, and whole process, respectively. For a moderate approximation tolerance, the other procedures are relatively quick compared with the initial and ϵ-triangulations. Total number N_{total} of approximating triangles follows $N_{total} \sim \frac{1}{\epsilon}$ relation – see [8, 11] for detailed complexity analysis.

Table 1 also shows the AR of the approximating triangles for a given AR-threshold $\tau_{\mathrm{AR}} = 4$, where N_{fail} denotes the number of triangles whose AR $> \tau_{\mathrm{AR}}$. It is worthwhile to mention that the satisfactory average aspect ratio AR_{avg} results from the coupled effects of the features of Delaunay based point insertion algorithm; AR-improving procedure in the planar domain of triangulation; and preservation of triangles' shape during the mapping process from the triangulation domain into three-dimensional space.

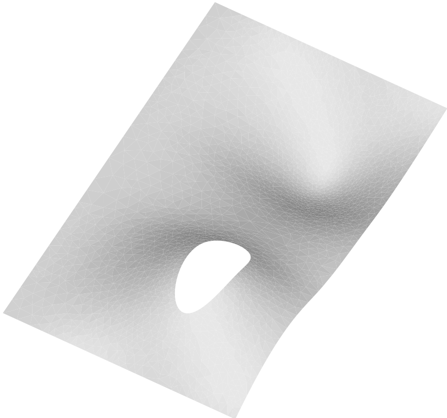
Table 2 shows the dependency of AR-improving procedure upon the AR-threshold τ_{AR} for a fixed approximation tolerance $\epsilon = 10^{-2}$. In this example, 3% of N_{total} have AR $> \tau_{\mathrm{AR}} = 3.5$ due to slight deformation occurred in the process of approximate locally isometric mapping. In general, N_{fail} increases as τ_{AR} becomes more ideal since we fix the limit τ_r of circumcircle radius described in Section 5, regardless of τ_{AR}. Table 2 also shows how many triangles N_{AR} are generated in AR-improving procedure after ϵ-triangulation, where the number N_ϵ of triangles is 222.

We have also performed the triangulation in fully rounded interval arithmetic (RIA). Table 3 shows the result. As far as the time cost is concerned, RIA is one order of magnitude more expensive than FPA. However, we need to keep in mind that crucial computations such as *intersection test* between the edges of an approximately developed surface net, *point mapping* between the triangulation domain and the parametric space, *Delaunay test* and *self-intersection check* of the approximating triangles, may result in *system failure* if the algorithm is operating purely in FPA.

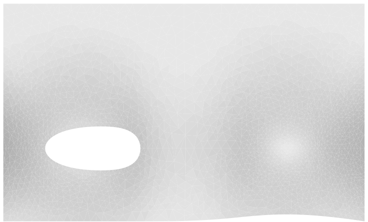
Global behavior of the other realistic examples shown in Figure 9 is consistent with that of our first example, where we set $\epsilon = 10^{-3}$ and $\tau_{\mathrm{AR}} = 4$. Figure 9-(a) shows the result of meshing ($N_{total} = 7832$) for a surface of revolution, a typical example of rational B-spline surfaces. We have also implemented the method with a composite integral B-spline surface representing a part of a ship hull with a bulbous bow – see Figure 9-(b), where $N_{total} = 2580$. The next example is another composite integral B-spline surface that represents an airfoil. Figure 9-(c) illustrates the performance ($N_{total} = 3310$). In these examples, we can verify the well-conditioned meshing as well as local adaptivity near high curvature regions.

CPU (sec)					N_{total}
C_{init}	C_ϵ	C_{AR}	C_{fin}	C_{total}	
128	57	2	5	192	248

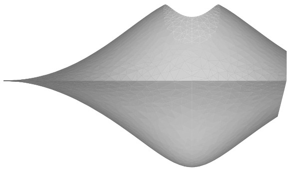
Table 3: Trimmed bi-cubic NURBS surface patch; **RIA**, $\epsilon = 10^{-2}$, $\tau_{\mathrm{AR}} = 4$



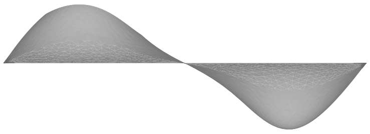
(a) Exact surface



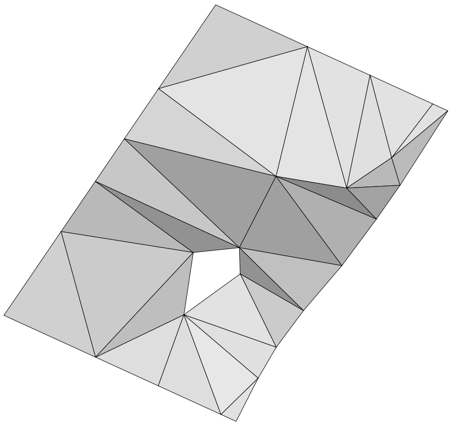
(b) xy-projection



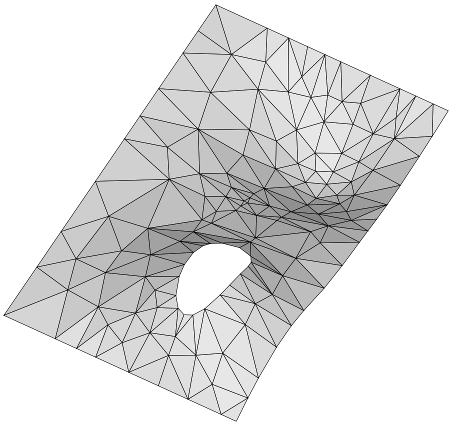
(c) yz-projection



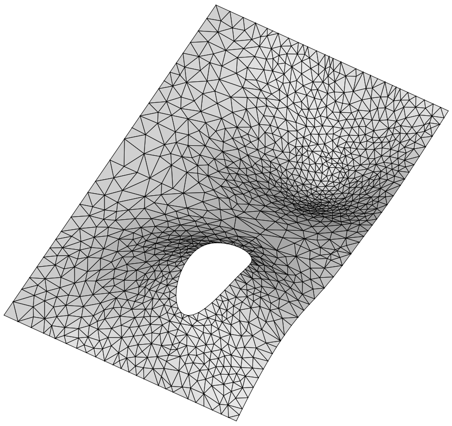
(d) zx-projection



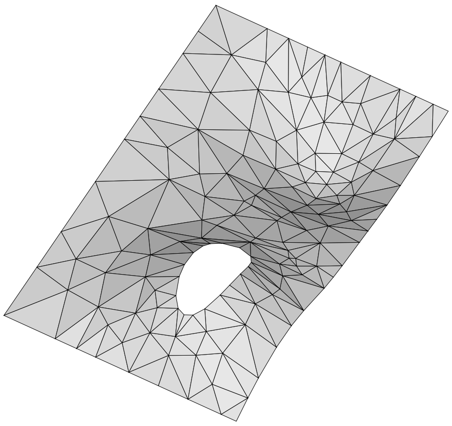
(e) $\epsilon = 10^{-1}$, $\tau_{\mathrm{AR}} = 4$



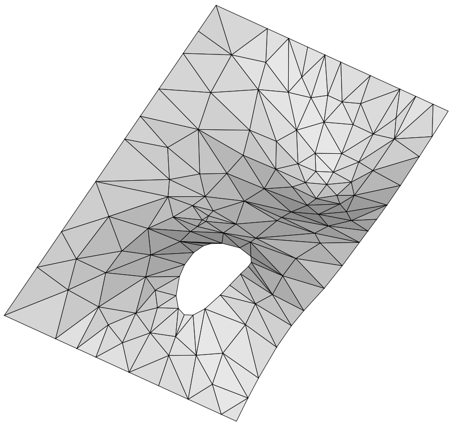
(f) $\epsilon = 10^{-2}$, $\tau_{\mathrm{AR}} = 4$



(g) $\epsilon = 10^{-3}$, $\tau_{\mathrm{AR}} = 4$

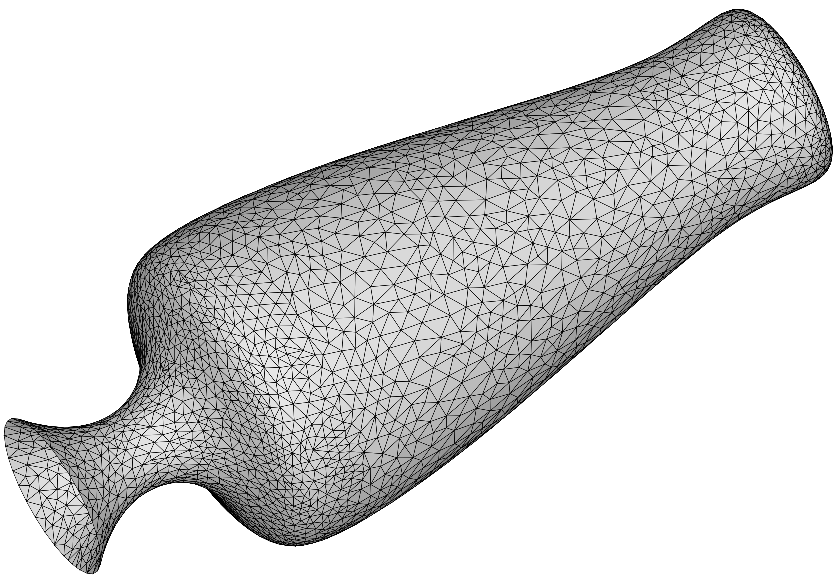


(h) $\epsilon = 10^{-2}$, $\tau_{\mathrm{AR}} = 3.5$

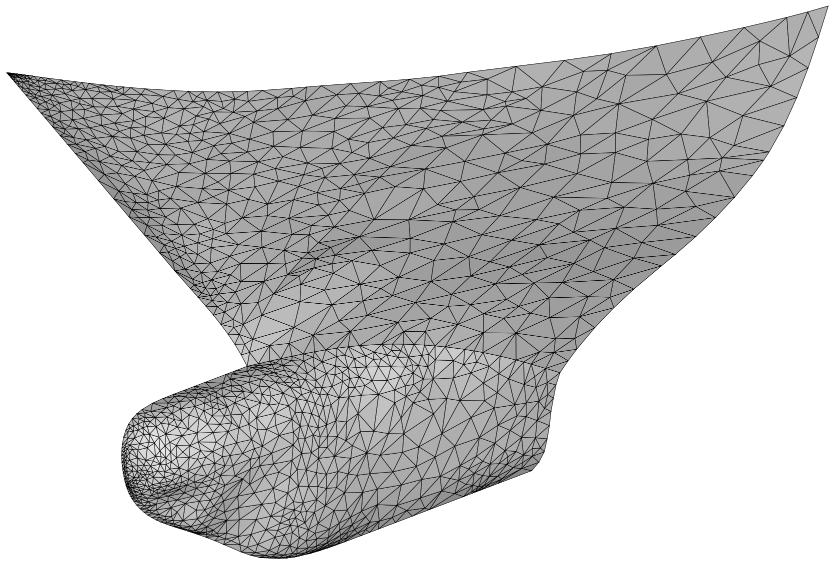


(i) $\epsilon = 10^{-2}$, $\tau_{\mathrm{AR}} = 4.5$

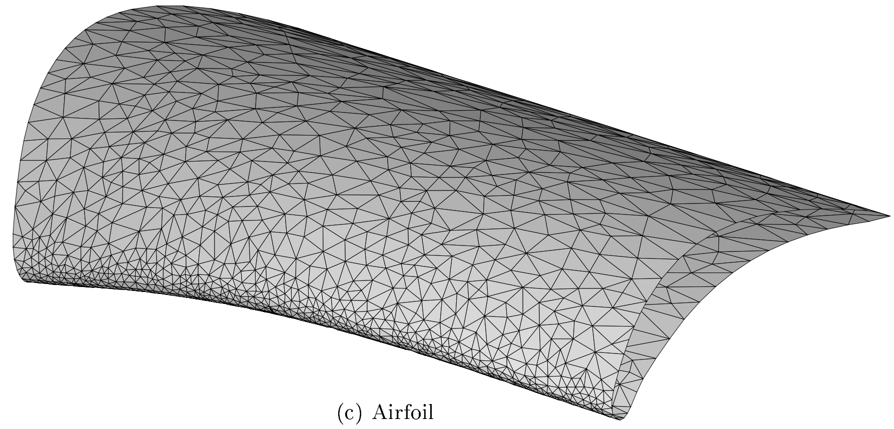
Figure 8: Trimmed bi-cubic NURBS surface patch



(a) Surface of revolution



(b) Stem with a bulbous bow



(c) Airfoil

Figure 9: Example surface meshes; $\epsilon = 10^{-3}$, $\tau_{\mathrm{AR}} = 4$

8 Conclusions and Recommendations

In this paper, we have developed a piecewise linear approximation method of trimmed high order rational B-spline composite surfaces, which include: the use of *numerically robust* interval geometric representations and computations; an *efficient* and *adaptive* approximation method to satisfy the prescribed geometric tolerance; a surface meshing algorithm which generates *well-conditioned* triangulation in three-dimensional space; and an efficient method to identify and remove the possible *inappropriate intersections* of the approximating triangles, ensuring the existence of a *homeomorphism* between the approximants and their actual nonlinear surfaces.

For future research, we will consider the extension of the proposed method to the topologically reliable piecewise lower order non-linear approximation of trimmed high order rational B-spline surface patches. This procedure may be followed by a topologically reliable adaptive tetrahedral meshing of a B-rep solid or its complement relying on the result of surface triangulation.

Acknowledgements

Funding was obtained in part from ONR and NSF under grant numbers N00014-96-1-0857 and DMI-9500394.

References

[1] Abrams, S. L., Cho, W., Hu, C.-Y., Maekawa, T., Patrikalakis, N. M., Sherbrooke, E. C. and Ye, X. (1997), Efficient and reliable methods for rounded interval arithmetic, Computer Aided Design, To appear.

[2] Agoston, M. K. (1976), *Algebraic Topology,* Marcel Dekker Inc., New York.

[3] Austin, S. P., Jerard, R. B. and Drysdale, R. L. (1997), Comparison of discretization algorithms for NURBS surfaces with application to numerically controlled machining, Computer Aided Design, 29(1):71–83.

[4] Babuška, I. and Aziz. A. (1976), On the angle condition in the finite element method, SIAM Journal of Numerical Analysis, 13:214–227.

[5] Baker, T. J. (1989), Automatic mesh generation for complex three-dimensional regions using a constrained Delaunay triangulation, Engineering with Computers, 5:161–175.

[6] Boender, E., Bronsvoort, W. F. and Post, F. H. (1994), Finite-element mesh generation from constructive-solid-geometry models, Computer Aided Design, 26(5):379–392.

[7] Cho, W., Maekawa, T. and Patrikalakis, N. M. (1996), Topologically reliable approximation of composite Bézier curves, Computer Aided Geometric Design, 13(6):497–520.

[8] Cho, W. (1997), *Topologically reliable approximation of curves and surfaces,* Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, January.

[9] Cho, W., Patrikalakis, N. M. and Peraire, J. (1997), Robust interval planar Delaunay test, Design Laboratory Memorandum 97-2, Cambridge, MA. Submitted for publication.

[10] Cho, W., Patrikalakis, N. M. and Peraire, J. (1997), Approximate development of trimmed patches for surface tessellation. Design Laboratory Memorandum 97-3, Cambridge, MA. Submitted for publication.

[11] Cho, W., Maekawa, T., Patrikalakis, N. M. and Peraire, J. (1997), Topologically reliable approximation of trimmed polynomial surface patches, Design Laboratory Memorandum 97-4, Cambridge, MA. Submitted for publication.

[12] Cormen, T. H., Leiserson, C. E. and Rivest, R. L. (1990), *Introduction to Algorithms,* MIT Press, Cambridge, MA.

[13] Gürsoy, H. N. and Patrikalakis, N. M. (1992), An automated coarse and fine surface mesh generation scheme based on medial axis transform, part I: Algorithms, Engineering with Computers, 8(3):121-137.

[14] Gürsoy, H. N. and Patrikalakis, N. M. (1992), An automated coarse and fine surface mesh generation scheme based on medial axis transform, part II: Implementation, Engineering with Computers, 8(4):179-196.

[15] Hu, C.-Y., Maekawa, T., Sherbrooke, E. C. and Patrikalakis, N. M. (1996), Robust interval algorithm for curve intersections, Computer Aided Design, 28(6/7):495–506.

[16] Hu, C.-Y., Patrikalakis, N. M. and Ye, X. (1996), Robust interval solid modeling, Part I: Representations, Computer Aided Design, 28(10):807–817.

[17] Hu, C.-Y., Patrikalakis, N. M. and Ye, X. (1996), Robust interval solid modeling, Part II: Boundary evaluation, Computer Aided Design, 28(10):819–830.

[18] Hu, C.-Y., Maekawa, T., Patrikalakis, N. M. and Ye, X. (1997), Robust interval algorithm for surface intersections, Computer Aided Design, 29(9):617–627.

[19] Klein, R. and Straßer, W. (1995), Large mesh generation from boundary models with parametric face representation, in: Hoffmann, C. and Rossignac, J., eds, *Third Symposium on Solid Modeling and Applications*, 431–440. ACM Press.

[20] Kumar, V. and Dutta, D. (1997), An assessment of data formats for layered manufacturing, Advances in Engineering Software, 28(3):151–164.

[21] Löhner, R. (1987), Finite elements in CFD: what lies ahead, International Journal for Numerical Methods in Engineering, 24:1741–1756.

[22] Maekawa, T. and Patrikalakis, N. M. (1994), Interrogation of differential geometry properties for design and manufacture, The Visual Computer, 10(4):216–237.

[23] Mäntylä, M. (1988), *An Introduction to Solid Modeling*, Computer Science Press, Rockville, MD.

[24] Moore, R. E. (1966), *Interval Analysis,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

[25] Peraire, J., Perio, J. and Morgan, K. (1992), Adaptive remeshing for three dimensional compressible flow computations, Journal of Computational Physics, 103(2):269–285.

[26] Piegl, L. A. and Richard, A. M. (1995), Tessellating trimmed NURBS surfaces, Computer Aided Design, 27(1):16–26.

[27] Piegl, L. A. and Tiller, W. (1997), *The NURBS Book,* 2^{nd} *edition*, Springer-Verlag, New York.

[28] Ratschek, H. and Rokne, J. (1993), Test for intersection between plane and box, Computer Aided Design, 25(4):249–250.

[29] Rebay, S. (1993), Efficient unstructured mesh generation by means of Delaunay triangulation and Bowyer-Watson algorithm, Journal of Computational Physics, 106:125–138.

[30] Rockwood, A., Heaton, K. and Davis, T. (1989), Real-time rendering of trimmed surfaces, ACM Computer Graphics, 23(3):107–116.

[31] Sheng, X. and Hirsch, B. E. (1992), Triangulation of trimmed surfaces in parametric space, Computer Aided Design, 24(8):437–444.

[32] Shimada, K. and Gossard, D. (1992), Computational methods for physically-based FE-mesh generation, in: Olling, G. and Kimura, F., eds, *Human Aspects in Computer Generated Manufacturing*, 411–420. Elsevier Science Publishers B. V. (North Holland).

[33] Strang, G. and Fix, G. J. (1973), *An Analysis of the Finite Element Method*, Prentice-Hall, Inc., New Jersey.

[34] Tuohy, S. T., Maekawa, T., Shen, G. and Patrikalakis, N. M. (1997), Approximation of measured data with interval B-splines, Computer Aided Design, To appear.