EUROGRAPHICS 2004 / M.-P. Cani and M. Slater
(Guest Editors)

Volume 23 (2004), Number 3

Interference Detection for Subdivision Surfaces

Xiaobin Wu and Jörg Peters

University of Florida †

Abstract

Accurate and robust interference detection and ray-tracing of subdivision surfaces requires safe linear approximations. Approximation of the limit surface by the subdivided control polyhedron can be both inaccurate and, due to the exponential growth of the number of facets, costly.

This paper shows how a standard intersection hierarchy, such as an OBB tree, can be made safe and efficient for subdivision surface interference detection. The key is to construct, on the fly, optimally placed facets, whose spherical offsets tightly enclose the limit surface. The spherically offset facets can be locally subdivided and they can be efficiently intersected based on standard triangle-triangle interference detection.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

For animation and simulation, subdivision surfaces fill a gap between polyhedral and spline modeling and have matured to an important high-end modeling tool. Yet, while both polyhedral and spline modeling techniques have well-established (and largely separate) intersection toolkits, subdivision surfaces lack safe and efficient interference detection algorithms needed for accurate animation and physical simulation.

Since subdivision surfaces are typically represented by a control mesh composed of triangles or quadrilaterals, a straightforward approach is to reduce the intersection testing to polyhedral intersection testing between the control meshes. However, Figure 1 shows that such tests are *neither accurate nor safe*: interference of the control meshes does not imply that the limit surfaces intersect, and separation of the control meshes does not imply that the limit surfaces are disjoint!

One can increase the accuracy and safety of polyhedral testing by comparing finely subdivided meshes but this raises the question: how many times do we need to subdivide to guarantee a given maximal distance to the limit? An exact estimate is crucial since the number of faces grows exponentially with the number of subdivision steps and so does the space and time needed for the detection task. The related argument, that we should consider the control mesh as the final polygonal surface after a few, fixed number of refinement steps, ignores both rendering artifacts when zooming in and collision response artifacts due to sharp edges in place of smooth transitions.

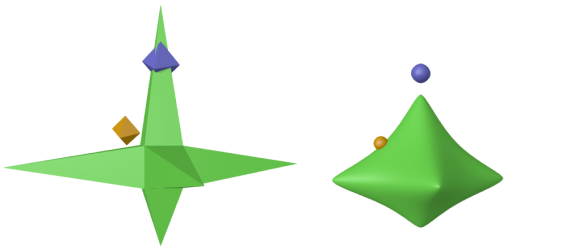

Figure 1: *Interference detection based on the control mesh rather than the limit surface is neither safe nor accurate. The green and the blue surfaces are disjoint but their control meshes collide. The green and the orange surfaces collide but their control meshes are disjoint.*

This paper now shows that we can get highly accurate and safe intersection testing for the *subdivision limit surface* at the cost of working with a triangle hierarchy – using stan-

† {xwu, jorg}@cise.ufl.edu

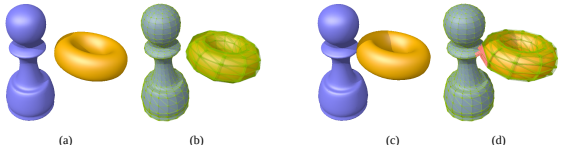

Figure 2: *(a) The limit surface is separated and (b) the bounding interval triangle detects the separation. (c) The limit surfaces intersect and (d) the interval triangle detects the intersection.*

dard tools such as an OBB tree and a triangle-triangle interference test at the fine levels. The approach is *safe* in that no contacts are missed and *accurate* in that possible intersection will only be reported if the limit surface is within the prescribed tolerance.

The key is a novel bounding *3D interval triangle* that
- ▷ tightly sandwiches the limit surface,
- ▷ can be efficiently computed,
- ▷ can be adaptively refined to the desired accuracy,
- ▷ can be efficiently tested against another, and
- ▷ can be easily imported into an existing triangle hierarchy.

1.1. Review and comparison to prior work

The proposed approach combines ideas from bounding hierarchies [GLM96, LGLM99], envelopes for ray-tracing [Kob98], tight bounds on splines [LP01b, LP01a], and triangle intersection testing [Möl97]. It extends these ideas to subdivision surfaces [WW02, Sch98]. As in [Kob98, GS01], we illustrate our approach with Loop's subdivision scheme [Loo87]. For Loop subdivision, all facets are triangles and mesh nodes with valence other than 6 are called extraordinary nodes.

The approach matches one interval triangle (hierarchy) to each control facet, also near extraordinary nodes. The sophisticated eigendecomposition in [Sta98] and [GS01] allows viewing the neighborhood of extraordinary nodes as one entity rather than an infinite, nested sequence of polynomial surface rings. Interval triangles do not require such an eigendecomposition. This makes it simpler and allows a larger range of subdivision algorithms than [GS01]. Also, we do not use axis-aligned bounding boxes (AABBs) on the control mesh. AABBs were considered sufficiently efficient for the intersection in [GS01] but not in [GLM96], reflecting a different weighing of intersection accuracy and speed. In contrast to [Sta98], which parametrizes over the unit square, the parametrization used to construct interval triangles near extraordinary nodes has n-gon symmetry.

We follow [Kob98] in that we use a min-max expression of interval arithmetic to create bounds. Our overall approach differs, however. We do not bound just the normal direction and we do not build a complete inner and outer shell pair to support the intersection testing. Constructing the shells turns out to be unnecessary, expensive and error-prone. Bounding in just the normal direction can lead to subtle errors and [PW] gives examples where envelopes constructed according to [Kob98] fail to properly enclose the surface. Interval triangles guarantee that all parts of the surface are enclosed.

To improve accuracy and efficiency, we take a cue from [LP01b] in that we remove linear components before sandwiching surface pieces. There is, however, no need to compute completely new 'antidifference' basis functions or second derivatives near extraordinary nodes.

We tested the approach by making a small but important change to the oriented bounding box (OBB) tree code [GLM96] that the authors generously make available. This change guarantees that subdivision limit surfaces are accurately and safely tested at the same cost as testing control meshes. In essence, we replace mesh triangles by optimally placed triangles whose constant offset encloses the surface tightly. Such offset triangles are known as s-topes [HKT92] and the approach is very similar to [LGLM99] except that we cover a curved, recursively computed manifold rather than a cluster of triangles. For efficient intersection computations at the leaves of the tree, we modify the fast triangle-triangle intersection test of [Möl97].

1.2. Overview

Our method has two major steps, reported in Section 2 and Section 3, respectively. First, we build local bounding volumes, called *interval triangles*, that tightly sandwich the limit surface. These interval triangles are created using linear upper and lower bounds on each of the x,y,z components of the limit surface. Each interval triangle bounds one piece of the limit surface, and the union of interval triangles encloses the whole surface. If a pre-defined accuracy bound ε is given, the interval triangles are locally, adaptively refined until the thickness is less than ε.

In the second step, an intersection hierarchy is constructed based on the interval triangles created in the first step. We enlarge each interval triangle slightly to a triangle with sphere offset. Then we build an OBB tree intersection hierarchy with the offset triangles.

In Section 4, we test the space and time performance in a collision scenario and in Section 5, we show how one can leverage interval triangles to construct an explicit conforming inner and outer hull of the surface.

2. Interval triangles

In this section, we describe the efficient construction of an interval triangle that sandwiches a piece of the subdivision surface: we bound the x, y and z component of the limit

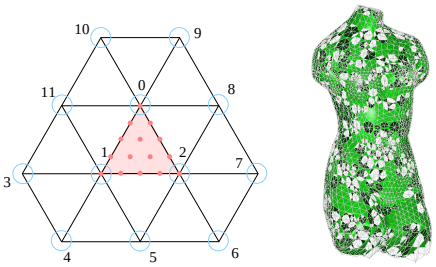

Figure 3: *(left) The neighborhood of a regular patch and (right) regular patches on the Venus shape (dark if back-faced).*

surface by forming a linear combination of pre-computed bounds on the basis functions. The construction is straightforward except for some subtle considerations during a one-time pre-computation.

We focus on Loop's subdivision. Extending the approach to other, say interpolatory subdivision schemes, requires only different tables of bounds than those computed in Section 2.4.

2.1. Patches

Given a triangulation, we can associate each triangle with one *Loop-patch*. A Loop patch is a piece of the limit surface under Loop subdivision applied to the triangle and its one-ring of neighbors (see Figures 3 and 4). If each of the vertices of the triangle has six neighbors, it is called a *regular* patch. Otherwise, it is irregular. We assume, for now, that at most one of the vertices has $\mathsf{n} \neq 6$ neighbors.

2.2. Component bounds

The x-component of the Loop patch can be expressed as

$$x(u,v) = \sum_{i=0}^{\mathsf{n}+5} c_i b_i(u,v)$$

where $b_i(u,v)$ is the Loop patch corresponding to the control polygon that is 1 at i-th vertex and 0 elsewhere, and c_i is the x component of the ith control point. Although the b_i have a closed form only if $\mathsf{n} = 6$, Section 2.4 shows how to compute linear functions b_i^+ and b_i^+ such that $b_i^- \leq b_i \leq b_i^+$ over the domain Ω_n of a Loop patch. (Ω_n is a subset of a triangle Δ_n as depicted in Figure 4; it is defined in Section 2.4). Then, on Ω_n,

$$x^+(u,v) := \sum_{i=0}^{\mathsf{n}+5} \max\{c_i,0\}\; b_i^+(u,v) + \min\{c_i,0\}\; b_i^-(u,v)$$

is an upper bound for x. Since linear functions are their own best upper and lower bound, we can extract a linear function ℓ interpolating c_0, c_1, c_2. With d_i the difference between

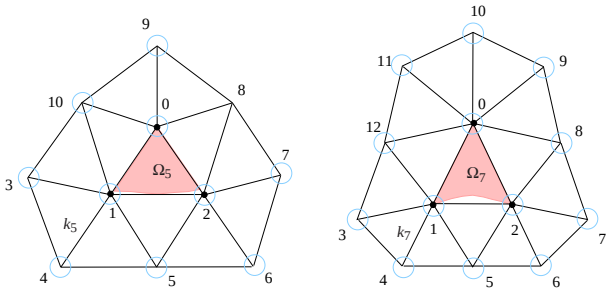

Figure 4: *Parametrization around an extraordinary node, $\mathbf{u}_0$, of valence $\mathsf{n} = 5$ respectively $\mathsf{n} = 7$. The triangles attached to $\mathbf{u}_0$ form a regular n-gon. Only the indices of the $\mathbf{u}_i$ are shown. The vertex $\mathbf{u}_4$ is defined by $\mathbf{u}_4 - \mathbf{u}_0 = k_\mathsf{n}(\mathbf{u}_1 - \mathbf{u}_0)$ and $\mathbf{u}_5$ as the average of $\mathbf{u}_4$ and $\mathbf{u}_6$. Nodes of type $\mathbf{u}_7$ are the reflection of $\mathbf{u}_5$ across the line through $\mathbf{u}_0$ and $\mathbf{u}_6$. The patch domain Ω_n is shaded. It always fits tightly inside the triangle Δ_n with vertices $\mathbf{u}_0$, $\mathbf{u}_1$ and $\mathbf{u}_2$.*

$\ell(u_i,v_i)$ and c_i,

$$x(u,v) = \ell(u,v) + \sum_{i=3}^{\mathsf{n}+5} d_i b_i(u,v).$$

The d_i are linear combinations of the control points c_i; by construction, $d_i = 0$ for $i = 0,1,2$. For regular patches, shown in Figure 3, there are two cases exemplified by $d_5 := c_5 - (c_1 + c_2 - c_0)$ and $d_6 := c_6 - (2c_2 - c_0)$. Then, on Ω_n,

$$x^- \leq x \leq x^+ \qquad \text{where} \tag{1}$$

$$\begin{aligned} x^+ &:= \ell + \textstyle\sum_{i=3}^{\mathsf{n}+5} \max\{d_i,0\}\; b_i^+ + \min\{d_i,0\}\; b_i^-, \\ x^- &:= \ell + \textstyle\sum_{i=3}^{\mathsf{n}+5} \max\{d_i,0\}\; b_i^- + \min\{d_i,0\}\; b_i^+. \end{aligned}$$

The linear bounds y^-, y^+, z^- and z^+ are determined analogously.

2.3. Constructing interval triangles

Let Δ_n be the triangle $\mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2$ in the domain with $\Omega_\mathsf{n} \subset \Delta_\mathsf{n}$ as shown in Figure 4 (The domain Ω_n is described in detail in Section 2.4 below). On Δ_n, the convex hull of the eight combinations of upper and lower component bounds,

$$p_{\alpha,\beta,\gamma} := (x^\alpha, y^\beta, z^\gamma), \qquad \alpha,\beta,\gamma \in \{-,+\},$$

forms an interval triangle that is guaranteed to enclose the patch. We can reduce its apparent complexity by observing that the interval triangle can be generated as the convex hull of three boxes $\square_i$, each corresponding to one domain vertex $\mathbf{u}_i$, $i = 0,1,2$. The eight corners of each $\square_i$ are

$$(x_i^\alpha, y_i^\beta, z_i^\gamma) := (x^\alpha(\mathbf{u}_i), y^\beta(\mathbf{u}_i), z^\gamma(\mathbf{u}_i)), \qquad \alpha,\beta,\gamma \in \{-,+\}.$$

This construction is illustrated in Figure 6.

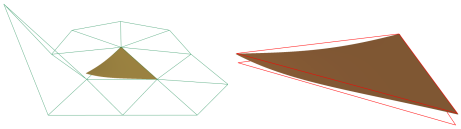

Figure 5: *(left) Basis function b_3 for $\mathsf{n} = 7$ and its control polygon, (right) The upper and lower bound (red polygon) of b_3 (enlarged).*

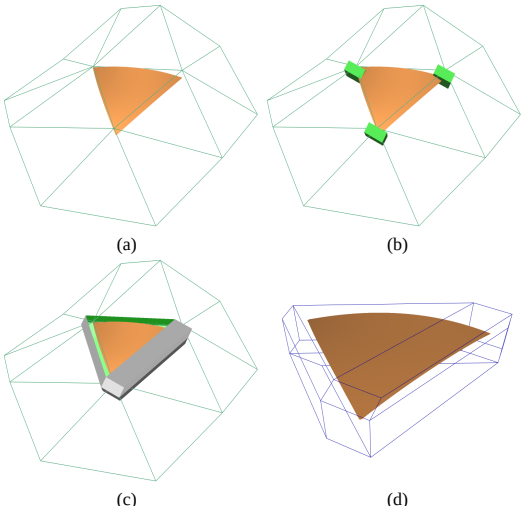

Figure 6: *(a) Loop patch; (b) corner boxes $\square_i$; (c) convex hull of the $\square_i$ forming an interval triangle (to display the inside, the top facet is removed); (d) the enlarged patch inside its wire-frame interval triangle.*

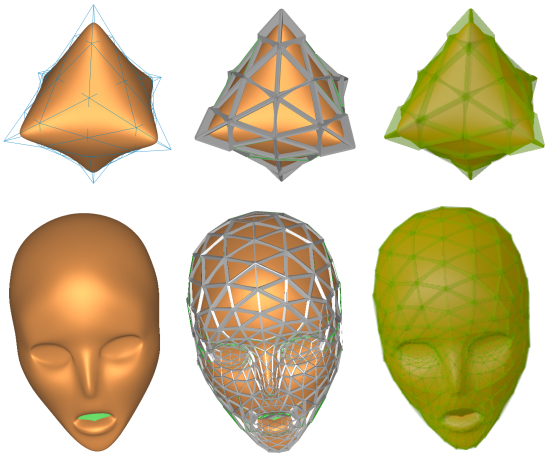

Figure 7: *From left to right: The subdivision surface; the surface and its interval triangles (with the top facet removed); the surface with semi-transparent interval triangles.*

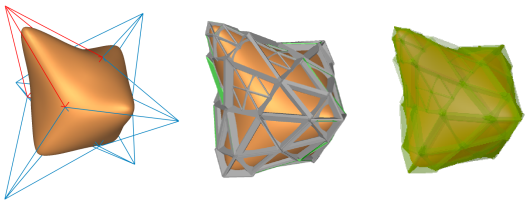

Figure 8: *Subdivision surface with semi-sharp creases (red = crease value 1.0).*

2.4. Bounding the basis functions

This section contains technical details for correctly bounding the basis functions b_i. Since we pre-computed and tabulated the bounds as a list of values at the vertices of the central triangle, in principle, no user of the approach needs to ever be concerned with the derivation: at runtime, the vertex values are simply read from a file [WP04], scaled by the d_i and summed with the linear function ℓ to yield upper and lower bounds.

For completeness, here are the details. A regular Loop patch is piecewise polynomial of degree 4 and defined by 12 control points (Figure 3, *left*). However, irregular patches consist of an infinite sequence of polynomial pieces so that the usual bounds, say those of [FMM86], do not work. To avoid dealing with this infinite sequence directly, we could compute the eigen-decomposition [Sta98, GS01]. But a simpler approach, akin to [Kob98], can bound the limit surface with one piece per mesh triangle. For simplicity, we assume that extraordinary nodes are separated by at least one ordinary, valence six node. This is always the case after one locally executed (!) subdivision step. Alternatively, we can use more tables ($\binom{\mathsf{n}-3}{3}$ *vs.* $\mathsf{n}-3$) to cover all possible combinations of extraordinary extraordinary nodes for valence 3 to n. For the current implementation, we used the first approach.

For the (one-time) bounding, for each n, each basis function is subdivided many times (we subdivided 7 times). Due to the convex hull property of Loop's subdivision, we obtain correct upper (lower) bounds if we fit a triangle so that it sits above (below) the subdivided control net.

The correct association of (u,v)-abscissae with the values is crucial so that, if the patch is linear, the upper and the lower triangle fall together and the interval triangle has thickness zero. In particular, we must reconstruct ℓ exactly as a linear combination of the b_i. To ensure this *linear precision*, we apply Loop subdivision also to the (u,v)-abscissae as we refine the values. Then the domain Ω_n of the Loop patch is the limit of the subdivision applied to the initial mesh of the abscissae $\mathbf{u}_i$. We choose the abscissae mesh to be symmetric with respect to the extraordinary node. Then Ω_n falls into the sector formed by the initial abscissa triangle Δ_n with vertices $\mathbf{u}_0$, $\mathbf{u}_1$, $\mathbf{u}_2$ (Figures 3, 4).

The shape of Ω_n, shown as shaded areas in Figures 3 and 4, changes with n. Symmetry at the extraordinary node

leaves exactly two degrees of freedom: the ratio $k_n := \frac{\mathbf{u}_4 - \mathbf{u}_0}{\mathbf{u}_1 - \mathbf{u}_0}$, and the ratio $(\mathbf{u}_1 + \mathbf{u}_2) - \mathbf{u}_0 : \mathbf{u}_5 - (\mathbf{u}_1 + \mathbf{u}_2)$. The latter is fixed by choosing $\mathbf{u}_5 := (\mathbf{u}_4 + \mathbf{u}_6)/2$. The domain agrees with Δ_n exactly if $n = 6$. For $n > 6$, the domain bulges inward, towards the extraordinary node. For $n < 6$, it bulges outward. Since the upper and the lower bounding triangle are naturally parametrized over Δ_n, the bound is only safe if the patch domain lies inside Δ_n. On the other hand, the patch domain should cover Δ_n as much as possible, since a larger gap implies a larger overestimate. Having the bulging triangles touch $(\mathbf{u}_1 + \mathbf{u}_2)/2$ and the triangles for $n \leq 6$ meet both points $\mathbf{u}_1$ and $\mathbf{u}_2$ yields

$$k_n := \begin{cases} -4(c^2-2)/(1+2c^2) & \text{if} \quad n \geq 6, \\ -6(2c^2-7)/(15+2c^2) & \text{if} \quad n < 6, \end{cases} \qquad c := \cos\frac{\pi}{n}.$$

2.5. Semi-sharp Creases and Boundaries

For enhanced realism, standard subdivision is often enhanced with directional, anisotropic semi-sharp crease rules. For example, [DKT98] proposes to apply, along marked mesh lines, a few steps of the sharp crease rules from [HDD*94], followed by one optional step of blending and infinitely many steps of standard subdivision.

There are two strategies for creating bounding interval triangles for surfaces with such semi-sharp creases. The first is to bound the basis functions corresponding to all different configurations of crease edges. This is similar to [BS02], but, since we only need upper and lower *bounds*, we need not generate bounds for every combination of two subdivision rules as in [BS02]. If one rule results consistently in higher values than the other, say a sharp crease rule or a boundary rule based on univariate splines, and a generic subdivision rule, then it suffices to bound the upper function from above and the lower function from below to enclose the whole range of combinations. Nevertheless, this strategy leads to a large number of tables.

The alternative strategy, used in Figure 8, is to apply subdivision with the sharp crease rule on the patches influenced by the crease edge until only smooth subdivision steps are left and the standard bounds apply. If a boundary is the result of generating one extra layer of nodes on the fly, i.e. by copying vertices or reflecting vertices, or if boundaries contract by one layer, interval triangles can be used without modification.

3. Collision Detection using offset triangles

3.1. Pairwise Interference Detection

Since one interval triangle can have up to 19 facets, (see Figure 6), comparing all facets to detect interference is not an efficient approach. Instead, we reduce the task to a triangle-triangle intersection test by slightly enlarging the interval triangle to an offset triangle, i.e. trading intersection cost for accuracy.

The *base triangle* of the offset triangle is defined as

$$(\frac{x^- + x^+}{2}, \frac{y^- + y^+}{2}, \frac{z^- + z^+}{2}) \tag{2}$$

restricted to Δ_n. The length of the half-diagonal of the ith corner cube is

$$l_i := \sqrt{(\frac{x_i^+ - x_i^-}{2})^2 + (\frac{y_i^+ - y_i^-}{2})^2 + (\frac{z_i^+ - z_i^-}{2})^2}.$$

Therefore, if we run (the center of) a sphere with radius

$$\rho := \max_{i=0,1,2} \{l_i\} \tag{3}$$

over the base triangle, we create an offset triangle that is guaranteed to enclose the Loop patch. The offset triangle is not optimally tight but reduces the task to a triangle-triangle intersection test of the base triangles $\mathcal{T}_1$, $\mathcal{T}_2$ with an error bound set to the sum of the two radii, ρ_1, ρ_2:

$$\text{dist}(\mathcal{T}_1, \mathcal{T}_2) \leq \rho_1 + \rho_2.$$

For the test, we adapt Möller's method [Möl97]: if one triangle lies to one side of the plane containing the other triangle non-intersection is reported; otherwise the common line passing through the two triangles is examined and non-intersection reported if the intersection line intervals do not overlap. We need only change to $\rho_1 + \rho_2 + \varepsilon$ the tolerance ε that Möller uses to stabilize the computation. This adds three additions to the approximately 100 operations of Möller's test.

3.2. Intersection Hierarchy

Most intersection applications have an associated tolerance ε so that surfaces are considered disjoint if their distance is more than ε. To enforce the tolerance, we split each patch into four by local Loop subdivision, until the offset triangle has a radius ρ less than ε. It is straightforward to modify any triangle hierarchy to use offset triangles. We modified the OBB tree code of [GLM96] by enlarging the dimension of the OBB until it contains each base triangle within the radius tolerance.

4. Performance evaluation

A major challenge when introducing a new collision detection technique is to conduct a fair performance experiment to measure space and time requirements in a realistic rather than just a worst case setting. Our task is made easier, in that we need not compare different types of hierarchies but only the test.

In our context, the performance criteria are the number of primitives needed to achieve a given accuracy (space), and the hierarchy initialization and the average intersection cost (time). We compare the performance of an offset triangle-based hierarchy to a similar hierarchy based on a control mesh that has been uniformly subdivided to lie within the

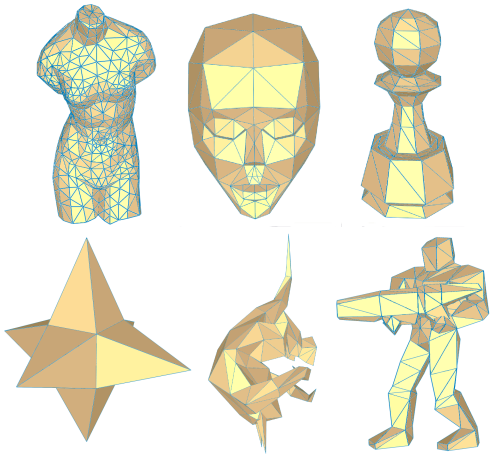

Figure 9: *Models used for performance evaluation.*

same prescribed error tolerance. As pointed out earlier (Figure 1) such a mesh does not guarantee correct intersection testing but would nevertheless be accepted in many practical situations.

We did not compare to an adaptively refined mesh because no efficient adaptive refinement based on the maximum norm is available. AABBs overestimate so much that the adaptive scheme is not competitive. The efficient alternative is to use interval triangles to drive adaptation – but then we are almost back to the proposed new solution. We also confirmed in our test scenario, which rigidly transforms the objects, that AABB trees built on the convex hull of the control polygon do not perform well compared to the OBB tree.

4.1. Experiment Set up

We used six different models of small to medium size as is suitable for further subdivision. The models range from 24 triangles (Star) to 1418 (Venus) and are shown in Figure 9. In the spirit of [KMSZ98], we place always two of these models into a cube or room, each with random position and random orientation. All models are scaled to tightly fit into a bounding box of size 1.

The interference test returns separation, or possible collision and only the first collision pair. Since the intersection cost depends on the position and orientation of the input models, we ran 150,000 random tests for each pair and report the average time. We also varied the room size to modify the percentage of intersecting cases.

4.2. Results

In the following tables, we label the new offset triangle-based method "o-t" and the intersection based on subdivided meshes "Loop". The timing is measured on a single P4 2.4G HZ CPU machine with 1G RAM.

ε	2%		1%		0.5%	
	o-t	Loop	o-t	Loop	o-t	Loop
Venus	5705	22688	6098	22688	7214	90752
Head	1499	3200	2474	12800	5990	51200
Pawn	1216	1216	2368	4864	4732	4864
Star	384	384	888	1536	1536	1536
Demon	1258	4384	2734	4384	4897	17536
Quake	1728	6528	4218	6528	6636	26112

Table 1: *Number of triangles needed to test intersection within the given error bound.*

Tolerance	2%		1%		0.5%	
	o-t	Loop	o-t	Loop	o-t	Loop
Venus	180	561	180	551	240	2414
Head	40	70	80	310	180	1342
Pawn	30	20	60	110	140	110
Star	10	10	20	30	40	40
Demon	40	100	90	170	150	431
Quake	50	150	100	100	200	641

Table 2: *Intersection hierarchy (OBBtree) creation time in milli seconds for different tolerances.*

Table 1 lists the number of triangles needed to guarantee a given accuracy. That is, a possible intersection is announced if the limit surfaces within the cubicle are closer than 2%, 1%, respectively 0.5% of the object size. The offset triangle-based method requires fewer triangles in all cases, indicating that the tight bounds pay off.

Table 2 lists the time used to build the OBB hierarchy. This includes the computation of all interval triangles in the case of the offset triangle-based method. Nevertheless, the offset triangle-based method requires less time in most cases. The average savings are 30%.

Table 3 lists the average intersection time for both approaches. For a room of size 6^3, offset triangles improve only marginally over the Loop method because most rejections occur in high level box-box tests of the OBB tree. However, for a room of size 3, more box-box tests are needed for Loop than for offset triangles. That is, the safe offset triangle-based test is cheaper than the unsafe control mesh-based test!

5. Inner and outer hull

For intersection testing, it is not necessary to build an explicit conforming inner and outer hull, since the union of the

Tolerance	2%		1%		0.5%	
	o-t	Loop	o-t	Loop	o-t	Loop
Room size 6.0: 3 percent of the models collide						
Venus/Head	22	24	22	26	22	27
Star/Pawn	11	12	11	12	12	12
Demon/Quake	13	13	13	14	14	14
Room size 3.0: 30 percent of models collide						
Venus/Head	147	178	144	178	148	187
Star/Pawn	81	91	84	91	84	90
Demon/Quake	105	116	107	117	106	118

Table 3: *Intersection cost in ms for different tolerances and room sizes.*

interval triangles encloses the surface. However, for other applications such as manufacturing with tolerances, intersection and overlap of the interval triangles is not acceptable. Instead, we need a pair of triangulations that sandwich the limit surface. To this end, we first select from the eight choices

$$p_{\alpha,\beta,\gamma} := (x^{\alpha}, y^{\beta}, z^{\gamma}), \qquad \alpha,\beta,\gamma \in \{-,+\},$$

computed in Section 2, for each mesh triangle two planes that tightly enclose the limit surface. Then, we shrink-wrap the individual planes around each node.

The two planes are selected as follows. Compute the normal of the linear functions $p_{\alpha,\beta,\gamma}$. If the signs of the *xyz*-components of the normal agree with (α,β,γ) then the plane is the outer plane. If the signs agree with $(-\alpha,-\beta,-\gamma)$ then the plane is the inner plane. In theory, there could be cases where we need to subdivide to assure uniqueness, but in practice we have never encountered such a case. All the points in the triangle cell lie between these two planes because their inner product with the outer plane is negative and their inner product with the inner plane is positive.

In the second step, we compute, for each node, an outer triangulation vertex. The vertex is the point of least distance to the node's limit position and such that it lies outside the outer planes of the facets surrounding the node. This is a simple local quadratic optimization problem that can be solved by enumeration. Connecting the outer triangulation vertices according to the inherited connectivity of the input mesh completes the construction (see Figure 10).

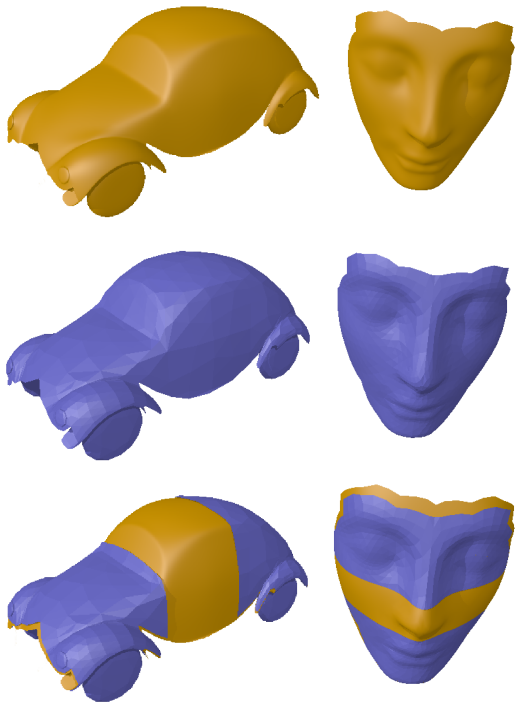

Figure 10: *(from top) The limit surface, the outer hull and a superposition of the limit surface and the outer hull with a cutout to show the position of the limit surface.*

6. Conclusion and future work

Safe, accurate and efficient intersection testing of subdivision limit surfaces can be based on a new bounding volume that is a spherical offset of a well-chosen base triangle. Alternative approaches, such as comparing refined control meshes, prisms [Kob98] or hierarchies of AABBs fail either to be efficient, or to be correct or both. The proposed algorithm is simple: read pre-tabulated data, form linear combinations according to (1), determine ρ and the base triangle according to (2), (3) and insert the base triangle with the ρ tolerance into the OBB hierarchy. This approach requires only a simple modification of available software.

A closer look at Section 2 shows that the approach works for any surface parametrization that can be bounded and is adaptively refinable. Therefore it can be applied to NURBS surfaces (if the denominator is bounded away from zero) and to other subdivision schemes, say interpolatory subdivision. The key ingredient in each case, is the one-time generation of accurate bounds analogous to Section 2.4. If the functions b_i are sufficiently smooth, we can also bound the components of the partial derivatives of the subdivision surface and estimate the range of the normal. That is, we can replace AABBs and eigendecomposition in the self-intersection test of [GS01] by interval triangles on the partial derivatives.

We are in the process of adapting interval triangles to other hierarchies, discrete orientation polytopes (k-dops) [KMSZ98] in particular; and we want to optimally regenerate the hierarchy on the fly for articulated characters. For

such continuously deforming objects, only a local subset of interval triangles needs to be regenerated but, off hand, the intersection hierarchy needs to be updated from bottom to top. Tabulated upper and lower bounds of b_i for $\mathsf{n} = 3, \ldots, 16$ and sample routines for constructing an x^+ and x^- pair are available via the conference CD and our website [WP04].

Acknowledgements: We thank MinHo Kim for helping with the runtime comparisons, and Andy Shiue and Ulrich Reif for helpful comments on an early draft. This research was made possible by NSF grant 9457806-CCR.

References

[BS02] BOLZ J., SCHRÖDER P.: Rapid evaluation of catmull-clark subdivision surfaces. In *Proceedings of the Web3D 2002 Symposium* (2002), ACM Press, pp. 11–18. 5

[DKT98] DEROSE T., KASS M., TRUONG T.: Subdivision surfaces in character animation. In *Siggraph 98* (1998), Cohen M., (Ed.), pp. 85–94. 5

[FMM86] FILIP D., MAGEDSON R., MARKOT R.: Surface algorithms using bounds on derivatives. *Computer Aided Geometric Design 3*, 4 (1986), 295–311. 4

[GLM96] GOTTSCHALK S., LIN M. C., MANOCHA D.: OBBTree: A hierarchical structure for rapid interference detection. *Computer Graphics 30* (1996), 171–180. 2, 5

[GS01] GRINSPUN E., SCHRÖDER P.: Normal bounds for subdivision-surface interference detection. In *Proc Visualization* (2001), Ertl T., Joy K.,, Varshney A., (Eds.), IEEE, pp. 333–340. 2, 4, 7

[HDD*94] HOPPE H., DEROSE T., DUCHAMP T., HALSTEAD M., JIN H., MCDONALD J., SCHWEITZER J., STUETZLE W.: Piecewise smooth surface reconstruction. *Computer Graphics 28*, Annual Conference Series (July 1994), 295–302. 5

[HKT92] HAMLIN G., KELLEY R., TORNERO J.: Efficient distance calculation using the spherially-extended polytope (s-tope) model. In *IEEE International Conference on Robotics and Automation* (1992), pp. 2502–2507. 2

[KMSZ98] KLOSOWSKI J. T., MITCHELL J. S. B., SOWIZRAL H., ZIKAN K.: Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics 4*, 1 (Jan. 1998), 21–36. 6, 7

[Kob98] KOBBELT L.: Tight bounding volumes for subdivision surfaces. In *Pacific-Graphics'98* (1998), Werner B., (Ed.), IEEE, pp. 17–26. 2, 4, 7

[LGLM99] LARSEN E., GOTTSCHALK S., LIN M., MANOCHA D.: Fast proximity queries with swept sphere volumes, 1999. Tech. Rep. TR99-018, Department of Computer Science, University of North Carolina. 2

[Loo87] LOOP C. T.: Smooth subdivision surfaces based on triangles, 1987. Master's Thesis, Department of Mathematics, University of Utah. 2

[LP01a] LUTTERKORT D., PETERS J.: Optimized refinable enclosures of multivariate polynomial pieces. *Comput. Aided Geom. Design 18*, 9 (2001), 851–863. 2

[LP01b] LUTTERKORT D., PETERS J.: Tight linear bounds on the distance between a spline and its B-spline control polygon. *Numerische Mathematik 89* (May 2001), 735–748. 2

[Möl97] MÖLLER T.: A fast triangle-triangle intersection test. *Journal of Graphics Tools: JGT 2*, 2 (1997), 25–30. 2, 5

[PW] PETERS J., WU X.: Sleves for planar spline curves. *Computer Aided Geometric Design*, x+1:x+24. to appear. 2

[Sch98] SCHRÖDER P.: Subdivision for modeling and animation, 1998. Course notes, ACM Siggraph. 2

[Sta98] STAM J.: Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *SIGGRAPH 98 Proceedings* (1998), Cohen M., (Ed.), Addison Wesley, pp. 395–404. 2, 4

[WP04] WU X., PETERS J.: The SubLiME package (Subdividable Linear Maximum-norm Enclosure), 2004. http://www.cise.ufl.edu/research/SurfLab/SubLiME. 4, 8

[WW02] WARREN J., WEIMER H.: *Subdivision Methods for Geometric Design*. Morgan Kaufmann Publishers, New York, 2002. 2