EUROGRAPHICS '93 / R. J. Hubbold and R. Juan (Guest Editors), Blackwell Publishers

Volume 12, (1993), number 3

The Cone of Normals Technique for Fast Processing of Curved Patches

Leon A. Shirmun and Salim S. Abi-Ezzi

Advanced Technology Group
Integrated Media Platform Software
SunSoft Inc.
Mountain View, CA 94043, U.S.A.

Abstract

The cone of normals technique for curved surface patches allows to perform various quick tests at the patch level such as front- or backfacing test, light influence test, and existence of silhouette edges test. For a given patch, a truncated cone of normals is constructed at creation time, which contains all points and all normal directions of the patch. At traversal time, a simple scalar product test determines whether the whole patch is backfacing or frontfacing, so that the costly step of tessellating the patch is avoided in case of patch level face culling. In addition, the technique quickly determines which light sources have no influence on a patch, and which patches have no silhouette edges. The technique can also be used for other surface primitives, such as triangular strips and quadrilateral meshes,

Keywords: Surface patches, dynamic tessellation, cone of normals, backface rejection, face culling, silhouette, lighting.

1. Introduction

In this paper, we propose a technique for speeding up the graphical processing of curved patches. In our discussion we focus on Bézier patches, although the technique is applicable to all surfaces that have a well defined normal function.

There are two main approaches for the graphical processing of curved surfaces. The traditional *static* tessellation approach reduces a surface to a fixed bag of triangles, which then gets processed for each subsequent frame. In contrast, the *dynamic* tessellation approach tessellates the surface into properly sized triangles for each frame, based on viewing parameters [AbiEzzi91]. In this paper, we focus on dynamic tessellation which we consider to be the trend of the future.

The tessellation of curved patches is a fairly expensive operation, and hence it is desirable to apply operations at the patch level in order to avoid tessellation altogether, or to facilitate tessellation and processing of resulting triangles. The main idea of this paper is to construct a representative cone of a patch, at creation time, and then to apply efficient operations on this cone, at traversal time, in order to quickly reason about the patch.

After the patch is tessellated, each resulting triangle is passed to a triangle pipeline, such as the one shown in Fig. 1. A triangle is first tested whether it lies outside of the modeling clipping

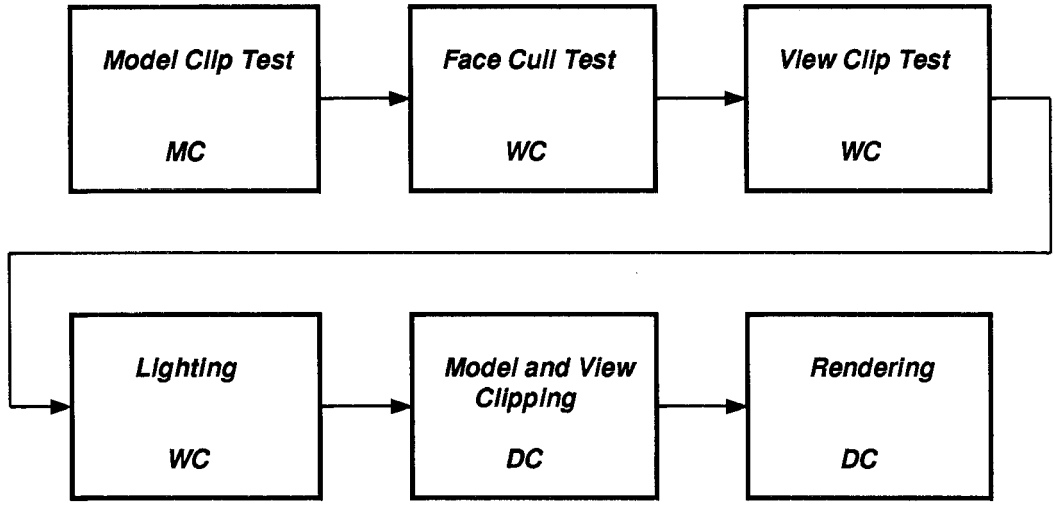

Figure 1: A simplified triangle pipeline.

region, the view clipping region, or faces away from the viewer. If one of these tests is true, the triangle is rejected before performing lighting computations, clipping, and rendering. Fig. 1 also shows in which coordinate system (Modeling, World, or Display) each operation conceptually takes place (although actual implementations usually perform these tasks in specialized coordinate systems to optimize performance).

The tests as applied to triangles above could also be applied to curved patches. Using the convex hull property, model clip and view clip tests can be easily accomplished by checking every control point of the patch. If the patch is not rejected, it is then tessellated and each triangle is processed by the triangle pipeline. In addition, if the patch is trivially accepted (i.e. it is not properly clipped), tests on the resulting triangles are not needed.

In this paper, we propose using extent techniques for directions similar to the way they are commonly used for points, as in the bounding box or convex hull approaches. We introduce the *cone ofnomals* of a Bézier patch that contains all normal directions of the patch. It allows performing face culling at the patch level, and it provides other information about the patch for speeding up its tessellation and the subsequent processing of its triangles.

The cone of normals of a patch has the property that any point on the patch and the normal at this point are contained in the cone. This allows the quick determination of whether:

- the whole patch is backfacing;
- the whole patch is frontfacing;
- a given light has no influence on the patch;
- a patch has no silhouette edges.

The relatively expensive construction of the cone of normals is view-independent and takes place only once at creation time. The actual tests, which are view-dependent and take place for every frame, are very fast and involve just a scalar product computation.

This paper is organized as follows: in section 2 we describe the computation of the *normal patch* and the construction of the *truncated cone* and the associated *frontfacing* and *backfacing regions* for Bézier patches. In section *3,* we show how frontfacing and backfacing regions are used

for efficient whole patch culling and for other purposes. Finally, in section 4, the effectiveness of the technique is discussed, as well as its application to other graphics primitives.

2. Construction of the Cone of Normals

The construction of the cone of normals proceeds in four stages. First, the *normal patch* of the given Bézier patch is computed analytically based on its control points. This gives us the control vectors of the normal patch. Second, we make use of the convex hull of these control vectors to construct the *floating cone,* which is guaranteed to contain all the normals to the given patch. We use the term *floating* since it is a cone of directions and therefore it has no fixed location in space. Third, the floating cone is translated in space and is truncated by two parallel planes orthogonal to the cone axis to contain the control points of the given patch. Finally, frontfacing and backfacing regions are constructed, which are used for the quick tests at the patch level.

All the constructions described in this section take place at surface creation time and are view-independent.

2.1. Computation of the Normal Patch

The normal function $\mathbf{N} = [N_x, N_y, N_z]$ of a patch S is the cross-product between the tangent vectors $\mathbf{S}_u$ and $\mathbf{S}_v$, as follows:

$$\mathbf{N} = \mathbf{S}_u \times \mathbf{S}_v = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ X_u & Y_u & Z_u \\ X_v & Y_v & Z_v \end{vmatrix}.$$

In the rational case, $\mathbf{S}_u$ is such that:

$$X_u = \frac{x_u w - x w_u}{w^2},\ Y_u = \frac{y_u w - y w_u}{w^2},\ Z_u = \frac{z_u w - z w_u}{w^2},$$

and analogously for $\mathbf{S}_v$. Since we only care about the direction of the normal, and in the equations above all the denominators are identical, we can eliminate these denominators and obtain the *scaled normal* $\mathbf{N}'$, which has the same direction as N:

$$\begin{aligned} N'_x &= (y_u z_v - z_u y_v)w + (y z_u - z y_u)w_v + (y_v z - z_v y)w_u, \\ N'_y &= (z_u x_v - x_u z_v)w + (z x_u - x z_u)w_v + (z_v x - x_v z)w_u, \\ N'_z &= (x_u y_v - y_u x_v)w + (x y_u - y x_u)w_v + (x_v y - y_v x)w_u. \end{aligned}$$

Thus, the normal function is a polynomial Bézier patch based on the above formulas. The reader is referred to [Farouki88] for discussion of differentiation and arithmetic operations for functions in Bézier form. In the rational case, the (u, v) degrees of the normal patch are $(3d_u - 1, 3d_v - 1)$, where (d_u, d_v) are the degrees of the original patch. In the polynomial case, $w_u = w_v = 0$, and multiplication with w is not necessary, so that the degrees of the normal patch are $(2d_u - 1, 2d_v - 1)$.

Multiplication of high degree polynomials in the Bézier basis is quite expensive. It turns out that finding the normal patch is the most expensive operation in the whole process of normal cone construction. For example, in case of rational bicubic patches, it takes about 75% of the total computational time.

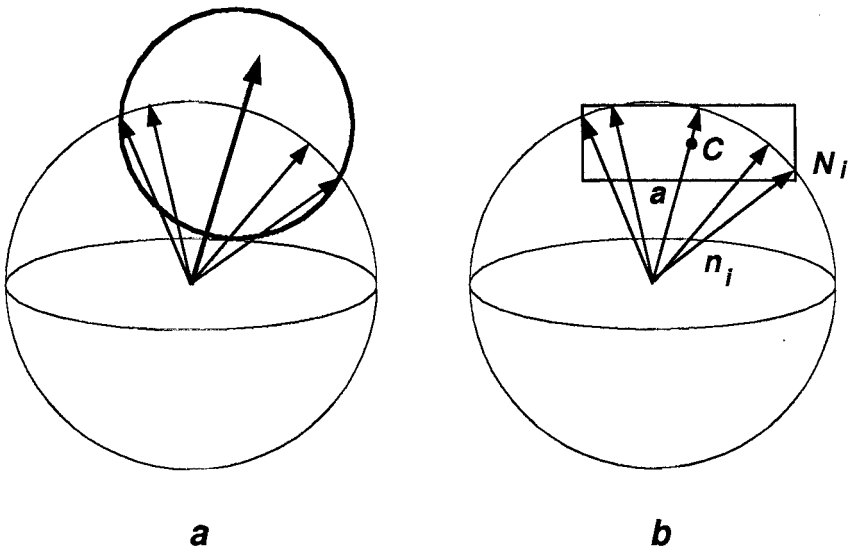

Figure 2: Construction of the floating cone.

2.2. Construction of the Floating Cone

Once the normal patch has been computed, the floating cone is constructed. The floating cone contains all the normal directions of the original patch, and it is floating since it has no position, only an orientation.

For the construction of the floating cone, we assume that all the control vectors are anchored at the origin. By the convex hull property, the tips of all the normals to the original patch when anchored at the origin, are contained in the convex hull of the tips of the control vectors of the normal patch. Assume that these control vectors (or their extensions) intersect the unit sphere, also centered at the origin, at points N_i. Then the smallest enclosing sphere of all the intersection points is constructed. The floating cone, when anchored at the origin, is defined to pass through the intersection circle of the two spheres, which is shown in Fig. 2a.

Finding the smallest enclosing sphere of a given set of points in space is a computational geometry problem. There are exact and iterative algorithms for its computation [Lawson65], but they are rather slow. Instead, we use a bounding box approach, which is very fast and in practical situations it produces a fairly close approximation to the enclosing sphere.

The idea of the method is very simple. A bounding box of all the points N_i is constructed. The center of the enclosing sphere is defined to be the center of the bounding box. The radius is then defined by the largest distance from the center to one of the points (Fig. 2b).

In practice we construct the cone without finding the enclosing sphere explicitly. We determine the cone axis to pass through the origin and the center of the enclosing sphere, C, where

$$C_x = \frac{1}{2}\left(\max_i N_{ix} - \min_i N_{ix}\right),$$
$$C_y = \frac{1}{2}\left(\max_i N_{iy} - \min_i N_{iy}\right),$$
$$C_z = \frac{1}{2}\left(\max_i N_{iz} - \min_i N_{iz}\right).$$

And we replace the step of constructing the enclosing sphere by constructing the floating cone through finding the maximum angle between the cone axis and control vectors of the normal patch. The cone semiangle α (the angle between the cone axis and its side) is defined by the minimum scalar product of the normalized cone axis direction a and normal patch control vectors $\mathbf{n}_i$:

$$\cos(\alpha) = \min_i (\mathbf{a} \cdot \mathbf{n}_i).$$

In case α is greater than $\pi/2$ there is no value in using the cone of normals technique, and this optimization is simply not applied. *Also,* we have assumed that all the control vectors of the normal patch are non-zero. If this is not the case, the patch may have a degeneracy, and the corresponding control vector has to be computed using higher derivatives or it may be undefined. In the latter case, again the cone of normals technique is not applied.

2.3. Determination of the Anchored Truncated Cone

In contrast to the floating cone, the anchored cone has a fixed position, and it is a translation of the floating cone so that it contains the patch itself. The construction described below translates the floating cone so that it encloses all the patch control points.

The anchored cone is determined as follows. First, the "bottom" control point B with respect to the cone axis is found (Fig. 3a). This point is defined by the smallest scalar product of the cone axis direction and directions from the origin to the patch control points. The bottom plane is then defined to pass through the bottom point, orthogonal to the cone axis. The apex of the floating cone is translated and anchored at the bottom point.

Second, for each patch control point P_i, which must be above the bottom plane, we check if it is enclosed by the translated cone. If it is not, then this point is projected onto the bottom plane. The projection takes place in the plane formed by the cone axis and the projected point along the line of intersection of the plane with the cone. The projected point P_i' can be computed as follows:

$$\vec{BP_i'} = r\left(\vec{BP_i} - h\,\mathbf{a}\right),$$
$$h = \vec{BP_i} \cdot \mathbf{a},$$
$$r = \frac{\sqrt{\|BP_i\|^2 - h^2} - h\tan(\alpha)}{\sqrt{\|BP_i\|^2 - h^2}}.$$

Here h is the height of P_i above the bottom plane, a is the normalized cone direction, and r is the ratio of the distances to the cone axis from P_i' and P_i.

After this operation has been done for all control points, the smallest circle that encloses all the points projected onto the bottom plane as well as the bottom point B is calculated. The two-dimensional equivalent of the enclosing sphere problem has been extensively studied. There are fast algorithms capable of computing the smallest enclosing circle in near-linear time using linear programming techniques [Reparata85]. However, the bounding box approach also works well in the planar case and efficiently produces a near-optimal circle. The cone is then translated so that this circle lies on the surface of the cone.

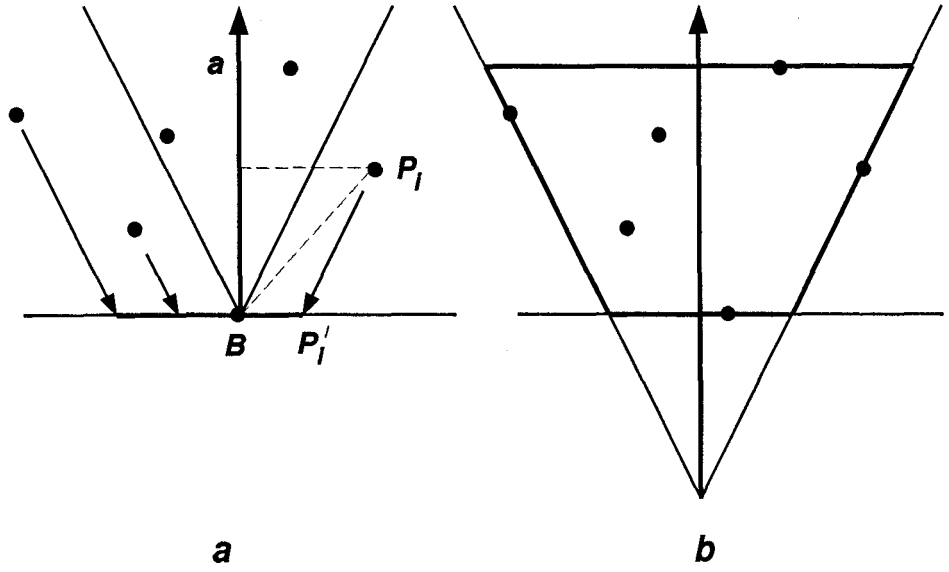

Figure 3: Construction of the anchored truncated cone.

Clearly, for any point S above the bottom plane and outside of the translated cone, a ray from S parallel to any cone side does not intersect the smallest enclosing circle. Therefore, since every projected point P_i' inside the circle was obtained by projecting P_i parallel to one of the cone sides, then P_i, and hence the whole patch, must lie inside the translated cone.

Finally, the top plane is determined in a similar fashion to obtaining the bottom plane. Both the bottom and the top planes are used to truncate the anchored cone as shown in Fig. 3b. Therefore, by construction the anchored truncated cone contains all the points of the patch and all the normals at these points.

2.4. Construction of Frontfacingand Backfacing Regions

After the above operations, we partition the space into a frontfacing region, a backfacing region, and a neutral region in relation to the given patch; see Fig. 4. We show the construction of these regions in a 2D slice of the cone through its axis, which then gets generalized to 3D in a straightforward fashion.

The truncated cone *ABCD* contains all the points of the patch and the normal directions at these points. The frontfacing region *EFG* and the backfacing region *KLM* are constructed by drawing perpendiculars *BG* and *DK* to side *AB*, and *CE* and *AM* to side *CD* as shown in Fig. 4. Point *F* is the intersection of rays *BG* and *CE*, while point *L* is the intersection of rays *DK* and *AM*.

For any point $\boldsymbol{P}$ and any direction n inside the truncated cone the following relationships hold:

- For any point $\boldsymbol{N}$ in the frontfacing region, the angle between $\boldsymbol{PN}$ and $\mathbf{n}$ is less than $\pi/2$.
- For any point $\boldsymbol{Q}$ in the backfacing region, the angle between $\boldsymbol{PQ}$ and $\mathbf{n}$ is greater than $\pi/2$.
- For any point $\boldsymbol{S}$ which lies in the neutral region (neither in the frontfacing nor in the backfacing regions), the angle between $\boldsymbol{PS}$ and n can generally range from 0 to π.

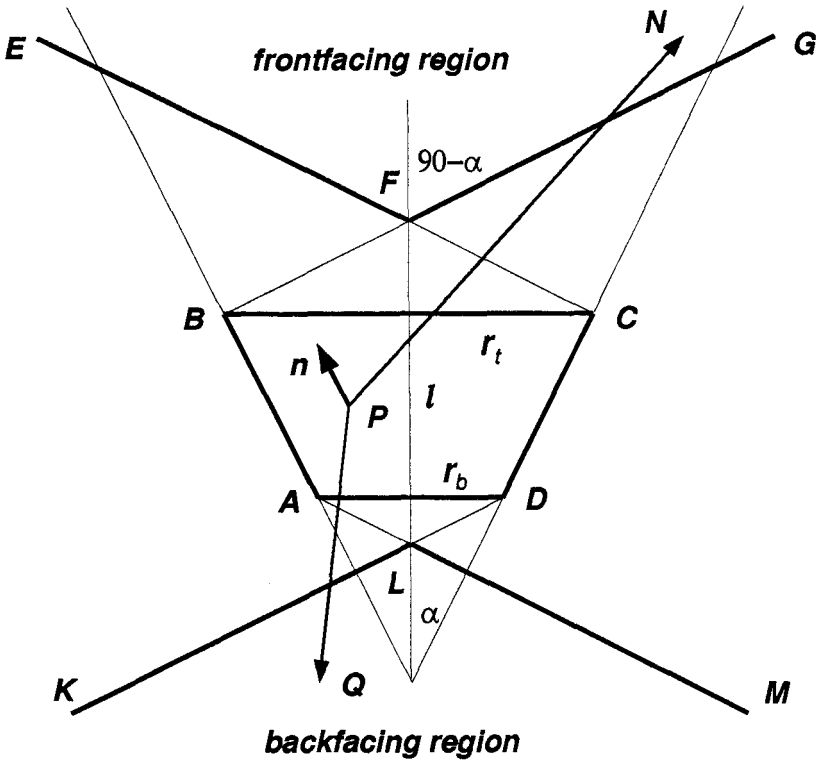

Figure 4: A cross-section of the cone through its axis to show the Frontfacing and backfacing regions.

As we will see in the next section, these relationships are crucial for fast patch level tests.

An imperative reasoning for the first of the above relationships is as follows: for a given fixed point P in the truncated cone, consider the set $S(P)$ of all points N such that the angle between PN and any direction n inside the truncated cone is less than $\pi/2$. This set is a cone centered at P with the axis parallel to the truncated cone axis and with the semiangle of $\pi/2 - a$, where a is the truncated cone semiangle. Since the above relationsip must hold for all points P in the truncated cone $ABCD$, the frontfacing region can be defined as an intersection of all sets $S(P)$ as P spans $ABCD$, which results exactly in EFG. Similar reasoning holds for the backfacing region.

By construction, the frontfacing and backfacing regions are actually *full cones* with apexes at F and L respectively. These apexes lie on the axis of the truncated cone. The semiangles of both cones are equal to $\pi/2 - \alpha$. The distance from the bottom plane to L is equal to $r_b \tan(a)$, where r_b is the radius of the bottom circle as determined in the previous section. Analogously, the distance from the top plane to F is equal to $r_t \tan(a)$, where r_t is the radius of the top circle, This radius can be expressed in terms of the radius of the bottom circle and the vertical length l of the truncated cone: $r_t = r_b + l \tan(\alpha)$.

As pointed out earlier, this construction only makes sense when a is less than $\pi/2$, since when the floating cone spans more than a halfspace, then obviously there will be no value in using this technique.

3. Using the Cone of Normals

All the construction steps described in the previous section take place at creation time in Modeling Coordinates. At traversal time, simple tests based on the cone can be used to determine various facts about the current patch in order to speed up its processing:

- If the eye point (viewing direction) is inside the backfacing region, the angle between any patch normal and the direction from any patch point to the eyepoint is greater than $\pi/2$. If backfacing primitives are culled, the whole patch is trivially rejected. Very significant savings are realized if this is the case, because there is no need to process this patch further (and tessellate it).
- If the eye point (viewing direction) is inside the frontfacing region then the whole patch is trivially accepted. In this case, face culling tests on each of the triangles resulting from the tessellation are avoided.
- If a light source position or direction is inside the backfacing region, this light source has no influence on the front face of the patch. Therefore, the light can be turned off which results in savings in lighting computations for each tessellated triangle of this patch.
- If the eye point (viewing direction) is in the frontfacing or in the backfacing region, then the patch has no silhouette edges because the angle between a patch normal and the direction to the eye point will either be less or greater than $\pi/2$, but never equal to. Detection and special processing of silhouette edges is an expensive operation, because all three normals of each triangle need to be checked whether they are frontfacing or backfacing. *Also*, finer tessellation is typically needed near silhouette edges. Flagging those patches for which these operations are not necessary is helpful.
- Similarly to silhouette edges, detection and special handling of highlights is expensive, Tests similar to the ones above can be devised to detect the existance or lack of highlights, but they are more complicated and go beyond the scope of this paper.

The cone inclusion test of a given point is a simple scalar product test. The test makes sure that the angle between the cone axis direction $\mathbf{a}$ and the direction from the cone apex to the point is less than the cone semiangle. This can be done by ensuring that the scalar product of the above normalized directions is larger than the cosine of the cone semiangle. In the case of the frontfacing and backfacing cones, the cosine of the semiangle is equal to the sine of the truncated cone semiangle.

In case of a perspective view (or positional light source), let $\mathbf{d}_f$ be the normalized vector from the frontfacing cone apex F to the eye point (light position), and let $\mathbf{d}_b$ be the normalized vector from the backfacing cone apex L to the eye point (light position). Then the test is

$$\mathbf{a} \cdot \mathbf{d}_f \geq \cos\left(\frac{\pi}{2} - \alpha\right) = \sin(\alpha)$$

for the frontfacing region and

$$-\mathbf{a} \cdot \mathbf{d}_b \geq \sin(\alpha)$$

for the backfacing region.

In case of a parallel view (or directional light source), the above tests can be used if we set

$$\mathbf{d}_f = \mathbf{d}_b$$

to be the view (light) direction from the object.

4. Discussion

In this section we discuss the value of the cone of normals technique in practice.

4.1. The Effectiveness of the Technique

We have tested the cone of normals technique on a fairly large volume of NURBS data sets. These data sets were produced by various *CAD* companies for industrial applications.

Typically, 20% to 45% of all Bezier patches were trivially rejected due to backface culling. The 20% rate was characteristic of cylindrical surfaces represented with four Bezier patches, so that the floating cone semiangle was $\pi/4$. Models with a large number of close to flat surfaces, which are common, had rejection rates approaching 50%. On the average, about 30% of Bezier patches were trivially rejected, and about the same proportion of patches were trivially accepted.

The 30% rejection rate means 30% increase in speed due to the fact that the rejected patches need not be processed further. In addition, trivially accepted patches (also 30% on the average) can be processed faster, because there is no need to perform face culling tests on the triangle level. *Also,* even if there is no face culling, face determination is still valuable, because proper (front or back) attributes can be applied to the triangles without doing an orientation test per triangle.

Finally, highly curved patches have their neutral regions encompass all of space, and hence their frontfacing and backfacing regions reduce to nothing. Fortunately, these types of patches are almost never used in industrial applications, which usually utilize B-spline surfaces consisting of a large number of Bezier patches with reasonable curvatures. In any case, highly curved patches can be detected at creation time and either the cone of normals technique is simply not applied to them or they can get subdivided, also at creation time, until their subpatches have reasonable floating cones.

Fig. 5 illustrates application of the cone of normals technique to backface culling of a dynamically tessellated cubic NURBS surface consisting of 9 Bezier patches. The snapshots were generated by *SurfTool,* a prototype implementation of the dynamic tessellation approach, The technique is also implemented in SunSoft's XGL graphics library.

In the first snapshot, the surface is roughly aligned along the viewing direction. The second snapshot shows the backface of the surface with backface culling disabled. The other four snapshots represent the views of the surface as it is rotated between the first two positions. As soon as the eye point moves inside the backfacing region of a Bezier patch, that patch is culled. The more of the surface backside is visible, the more patches are culled. The snapshots in Fig. 5 show the surface with 1, 3, 5, and 8 patches culled. Since the culled patches are immediately discarded and therefore not tessellated, they are not shown on the snapshots.

4.2. Application to Other Surface Prlmitives

So far, we have discussed the cone of normals technique to speedup the processing of Bezier patches. However, the method can be used for other surface primitives, such as triangular strips and quadrilateral meshes. These primitives are represented by a joined set of triangular or quadrilateral facets. In addition, vertex and facet normals are usually specified (Fig. 6).

Since facet normals are used for backface rejection, and vertex normals for lighting, both facet and vertex normals should be used in the construction of the cone of normals. These normal

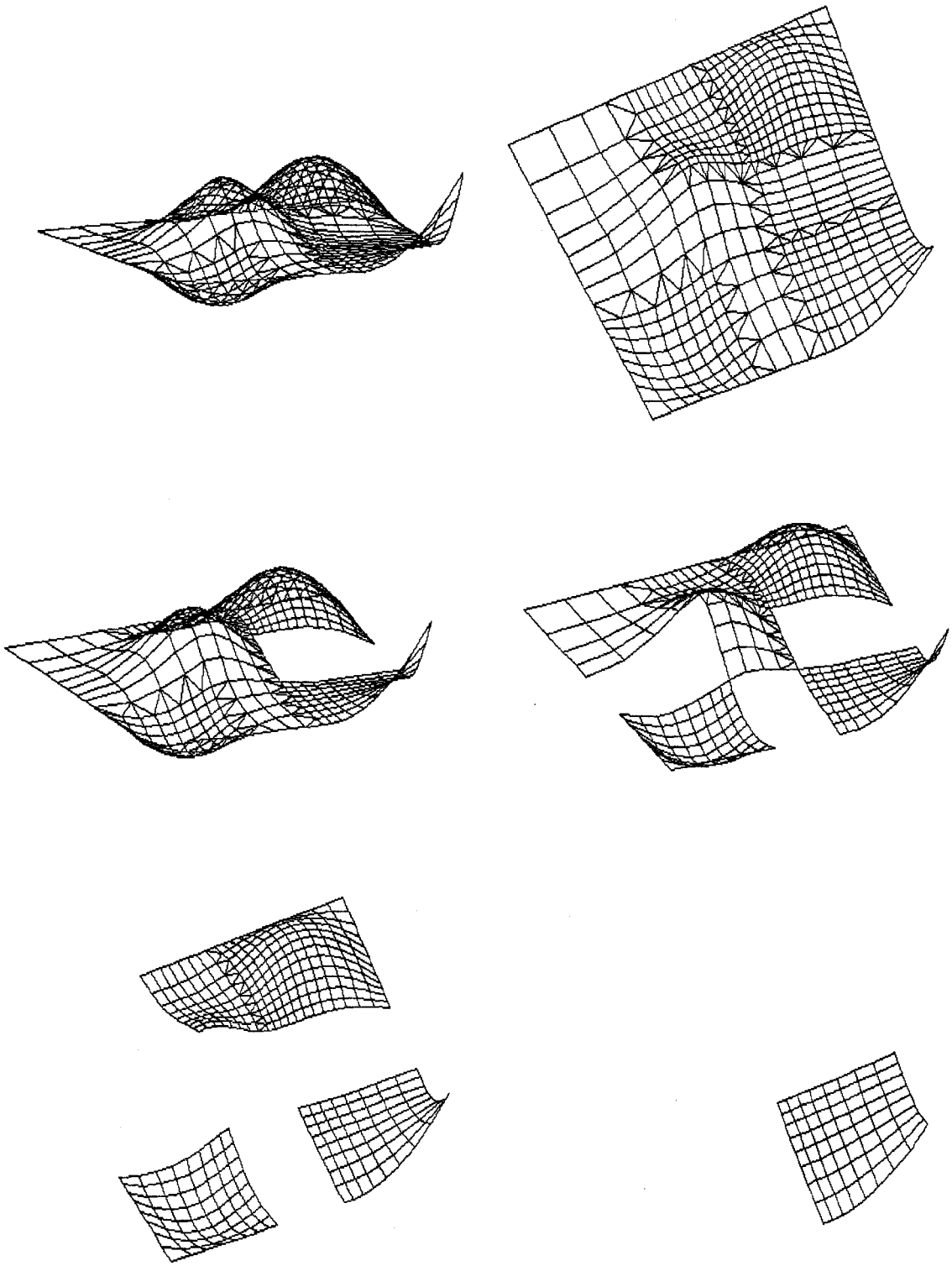

Figure 5: Backface culling at the patch level of a dynamically tessellated NURBS surface.

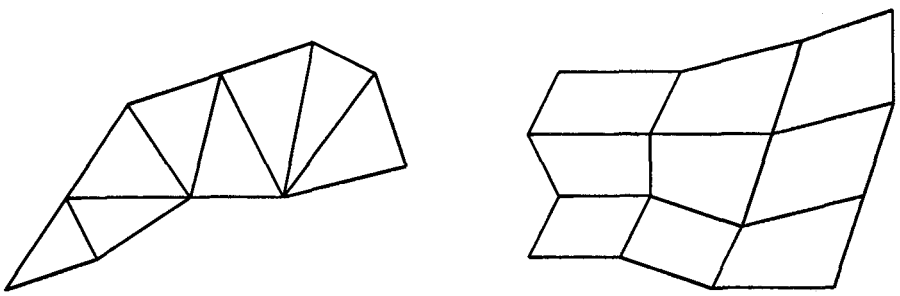

Figure 6: An example of a triangular strip and a quadrilateral mesh.

directions are used to construct the floating cone just like it was described in section 2, it is just that in this case relatively more numerous normals are used. The other steps of building the truncated cone and the frontfacing and the backfacing regions are identical to those for curved patches. The only difference for these simpler primitives is that there is no need to compute the normal patch.

Obviously, the major benefit of avoiding the tessellation in case of trivial rejection is not applicable in the case of these primitives. However, the other benefits remain for face cull tests (see Fig. 1), lighting computations, and silhouette edge detection.

4.3. The Case of Non-isotropic Modeling Transform

Since face culling and other patch-level tests take place in World Coordinates, where the eye point and light sources are specified, all the cone tests need to take place there. This would make the relatively expensive construction process depend on the modeling transformation. However, if the modeling transformation is *isotropic*, all computations can be done in MC by transforming the eye point (viewing direction) from WC to MC. Isotropic transforms are combinations of uniform scaling, rotation, and translation, and preserve aspect ratios and angles. In mathematical terms, a linear transform M is isotropic, if $M^T M = \lambda I$, where λ is some constant and I the identity matrix.

However, a non-isotropic modeling transformation may distort angles. Therefore, inclusion tests for frontfacing and backfacing regions have to be done in World, not Modeling, Coordinates. This situation is similar to lighting calculations — if the modeling transform is isotropic, they can be done in MC, otherwise, they have to be performed in WC.

An obvious way to proceed in case M is not isotropic is to transform the control vectors of the precomputed normal Bezier patch to WC, by using the inverse transpose of M, and then do all the remaining cone construction steps there. Hence, the most expensive operation — finding the normal patch — can still be done at creation time. Since the cone in WC is view-independent, it is reusable after view changes.

Usually, the modeling transformations are combinations of rotations, translations, and uniform scaling. Non-isotropic modeling transforms are used quite rarely. Therefore, a practical implementation may choose not to use the cone of normals technique in the case of a non-isotropic M at all, because complicating the code may not be worthwhile for optimizing a rarely occurring situation.

5. Conclusion

The cone of normals technique establishes a container of directions, analogously to what bounding box techniques do to points. We have described a way to construct a *cone of normals,* which contains all the normal directions of a patch. We have shown how to use the cone of normals in order to speed up important operations in the graphical processing of Bezier patches. While the demanding aspects of the technique take place at patch creation time, the benefit is achieved through quick tests at traversal time. This makes our approach very effective for highly interactive applications, where view/patch relationships change very frequently.

The cone of normals technique fits very well into the general approach of dynamic tessellation of surfaces [AbiEzzi91, AbiEzzi93]. In this approach, high performance is achieved by the compilation of graphics primitives at creation time into forms that can be processed very quickly during traversal. ***So,*** the construction of the cone of normals becomes a part of the compilation of a Bezier patch. Both the cone of normals and the dynamic tessellation techniques are successfully implemented in the XGL product from SunSoft.

Finally, an area for future work is that for certain patches it may be advantageous to construct *apyramid* of normals instead of a cone. For example, all the normals to a cylindrical surface are contained within a two-dimensional wedge; thus, a thin four-sided pyramid would be by far more effective than a cone.

6. References

1. [AbiEzzi91] Salim Abi-Ezzi and Leon Shiman: "Tessellationof Curved Surfaces under Highly Varying Transformations,"*Proc. Eurographics* '91,Vienna, Austria, pp. 385–397, September 1991.

2. [AbiEzzi93] Salim Abi-Ezzi and Leon Shirman: "The Scaling Behavior of Viewing Transformations,"to be published in *Computer Graphics and Applications,* May 1993.

3. [Farin90] Gerald Farin: ''Curves and Surfaces for Computer Aided Geometric Design," Academic Press, 1990.

4. [Farouki88] R.T. Farouki and V.T. Rajan: "Algorithms for Polynomials in Bernstein Form", *Computer Aided Geometric Design* 5, pp. 1–26, 1988.

5. [Lawson65] C. Lawson, "The Smallest Covering Cone or Sphere,"*SIAM Reviews* 7(3), pp. 415–417, 1965.

6. [Preparata85] Franco Preparata and Michael Shamos: "Computational Geometry: *An* Introduction,"Springer-Verlag, New York, 1985.