Spatialized Normal Cone Hierarchies

David E. Johnson and Elaine Cohen
{dejohnso,cohen}@cs.utah.edu
School of Computing, University of Utah

Abstract

We develop a data structure, the *spatialized normal cone hierarchy*, and apply it to interactive solutions for model silhouette extraction, local minimum distance computations, and area light source shadow umbra and penumbra boundary determination. The latter applications extend the domain of surface normal encapsulation from problems described by a point and a model to problems involving two models.

Keywords

Collision Detection, Computational Geometry, Non-Realistic Rendering, Shadow Algorithms.

1 Introduction

Many computer graphics applications depend upon the surface normal at a point on a model. Data structures to encapsulate sets of these surface normals have accelerated backface culling, lighting, model simplification, and silhouette extraction. In this paper, we expand the application of one such data structure, the *spatialized normal cone hierarchy*, to interactively solve seemingly disparate problems: local minimum distance computation and area light source shadow boundary computations. In addition, we introduce *variable precision silhouette* extraction through application of spatialized normal cones.

While there exist previous methods for encapsulating surface normal information, they typically have been applied to problems that can be described in terms of a point, such as a viewpoint or point light source, and a single model. We apply the spatialized normal cone data structure to problems involving two models, such as between a polyhedral light source and a model. This extension enables application to a much richer set of problems.

2 Background

In this section we review previous work on normal encapsulation. Some additional background material will appear in the sections covering particular problem domains.

The work most related to ours is Luebke and Erikson's work on view-dependent simplification [1]. They associate a cone, which encapsulates normals, with a sphere, which bounds space. This hierarchical data structure is used to refine polygonal models near their silhouettes, where artifacts from approximation are more apparent.

Shirman and Abi-Ezzi use a slightly different "cone of normals" technique for backface culling of Bézier patches [2]. They position a cone that encapsulates the range of normals such that the associated geometry also falls within the cone. Backface culling occurs when the viewpoint falls within a backface cone derived from the spatialized cone.

Other backface culling methods include Kumar *et al.*'s work on hierarchical clustering [3]. This method encapsulates the front-and-back facing regions of a polygonal model in a collection of half-space clusters. Zhang and Hoff reduce the normal clustering problem to that of a "normal mask", which allows very efficient per-triangle testing of normal direction against a backface bit mask with a single logical AND operation [4].

Surface normal encapsulation is also used in silhouette edge detection. Benichou and Elber [5] store the arcs associated with polygon edges on a Gaussian sphere [6]. Gooch *et al.* [7] store the arcs on the Gaussian sphere and build a hierarchy of spherical triangles for efficient search. Neither of these methods associates regions of Euclidean space with the normals and thus are suitable only for orthogonal projection viewing space.

Sander *et al*. [18] extended [2] to large, polygonal models with a hierarchy of "anchored cones". The cones conservatively bounded the front and back-facing regions of the triangles inside a node.

These techniques were all developed for problems where the encapsulated normals interact with a single point, such as a light source or viewpoint. This limits their application to the problems described above: backface culling, lighting, and silhouette edge extraction. In the following sections we show that spatialized, encapsulated normals apply to a broader set of problems.

3 The Spatialized Normal Cone Hierarchy

The following description of the data structure is as it applies to a polygonal mesh, although it can be adapted to other model representations. The data structure is simple, consisting of a normal cone, represented by a cone axis vector and a cone semiangle; and a Euclidean bounding volume, in this case a sphere, represented by a center and a radius (Figure 1.A). The bounding volume associates the normals with a particular region of model space, or *spatializes* the normal cone. Each node of the data structure also contains pointers to two child nodes, and a pointer to an underlying triangle if it is a leaf node.

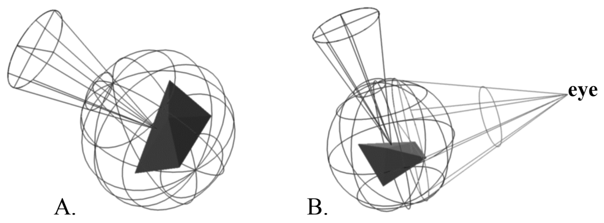


Figure 1.A: The spatialized normal cone encompasses the range of normals and bounds the geometry. 1.B: The view cone starts at the eye and encloses the bounding sphere.

We apply this data structure to a polygonal mesh structure of vertices, edges, and triangles. Note that the connectivity of the triangles is not important for the operation of our algorithms, except for the construction of appropriate normals for each of the primitives.

3.1 Constructing the hierarchy

The first step in the process constructs a spatial bounding volume hierarchy. We use the publicly available PQP code (http://www.cs.unc.edu/~geom/SSV/) to construct the Euclidean bounding volume hierarchy [8]. The geometry at each node is fit with a bounding sphere for use in the spatialized normal cone data structure.

The second step computes a normal cone for each node of the Euclidean bounding volume. This is done by finding an average normal from the triangle normals contained in that node. The cone axis vector is set to the average normal. The cone angle is the maximum angle between the cone axis and the normals.

4 Applications

In the following sections, we demonstrate use of the spatialized normal cone hierarchy on different problems. We start with its use in silhouette extraction, the simplest of the applications we present in this paper.

4.1 Silhouette extraction

Silhouettes occur when a triangle faces towards the eye and a neighbor triangle across an edge faces away from the eye. So for two triangles T_1 and T_2 with normals N_1 and N_2 and a view vector $\vec{V}$, a shared edge E is a silhouette edge if [4]

$$\left(N_1 \cdot \vec{V}\right)\left(N_2 \cdot \vec{V}\right) \le 0.$$

Another way of characterizing this is that the span of the two neighboring triangle normals over an edge must contain a vector orthogonal to the view vector. In the case of orthogonal projection, the view vector is always the same. Under perspective projection, there are a range of possible view vectors from the eye to the geometry contained within the bounding sphere for that node. We use the sphere as a conservative bound on that set of positions. The set of vectors between the eye and the geometry is bound by a cone from the eye to the bounding sphere. We call this cone the *view cone* (see Figure 1.B). The view cone axis runs from the eye to the center of the spherical bounding volume. The view cone angle is the arcsine of the ratio of the bounding volume radius to the view cone axis length.

If a vector contained within the view cone is orthogonal to a vector in the node's normal cone, there may be a silhouette in that node. It is simple to test if no vectors are orthogonal by computing the angle between the normal and view cone axes, and expanding and contracting that angle by the sum of the cone half-angles. Nodes that pass this test are recursively tested to the leaf level where an exact edge test is applied.

4.1.1 Results

We tested our perspective silhouette extraction method on several different models and compared the results against an exhaustive silhouette search method. All tests were run on an SGI Onyx2 computer with a 195 MHz MIPS R10000 processor.

Model	*# triangles*	*Exhaustive*	*Normal Cone*	*Ratio*
Bunny	23,000	5Hz	27Hz	5x
Sphere	33,000	11Hz	340Hz	31x

Figure 2: Using a spatialized normal cone provides a several time speedup over an exhaustive search for silhouettes.

The sphere showed a larger increase in update rates relative to a brute force approach. For the bunny model, the bumpy surface and increased complexity of the silhouette meant increased difficulty in pruning.

4.1.2 Variable precision silhouettes

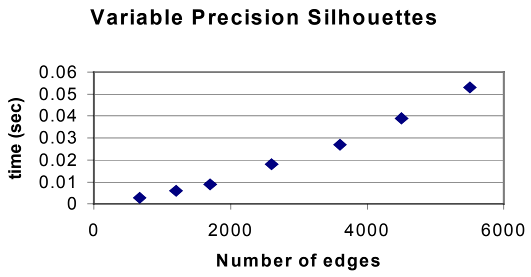
On models with fine detail, the silhouette can contain a surprisingly large number of edges. Similarly, for high-resolution models, the silhouette edges may be below pixel size. In both cases, there is wasted computational effort. In the first case, the edges are largely redundant as they project to the same part of the view plane. In the second case, the small silhouette edges represent detail that is impossible to see.

The spatialized normal cone hierarchy provides a means to attack these issues. We can stop descending the normal cone hierarchy when the normal cone angle falls below a threshold level. The challenge is to produce a variable precision silhouette for that node that replaces the numerous exact silhouettes.

Our approximation is to compute a line by projecting the node's normal cone axis to the view plane and finding the cross product of that and the view cone axis. This line is placed at the center of the node's bounding sphere and clipped to the bounding sphere.

4.1.2.1 Variable precision silhouette discussion

We tested the 69,000 triangle bunny model at various threshold angle values, measuring the number of edges found and the time to extract the edges. The silhouette remained visually pleasing even with less than half the edges of the full silhouette. At low numbers of edges, the bunny shape was still recognizable and silhouette extraction was 20 times faster than for the full silhouette (Figure 3).



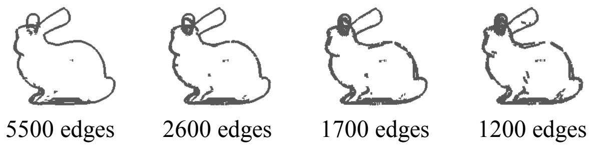


Figure 3: Approximate silhouettes reduce the number of edges examined and the number of edges drawn.

By using variable precision silhouettes, models can render in time sub-linear to the number of actual silhouette edges. On large models, this can result in substantial speed-ups. This technique also matches well with artistic rendering methods, such as [17], that do a post-process on the silhouette edges to produce a drawn look.

4.2 Local minimum distance

In the previous section, we applied spatialized normal cone hierarchies to a much-studied area, silhouettes, and demonstrated extending standard silhouette extraction to extract variable precision silhouettes. In this section, we apply spatialized normal cone hierarchies to find the minimum distance between models. Our solution finds all the local minima between models as well as the usual global minimum distance.

4.2.1 Background

Minimum distance computations have been used in robotics [9], computer graphics [10], and haptics [11]. Distance measures are important because they are predictive – they warn of potential events, such as contact, in advance.

Almost all minimum distance literature, especially for polygonal models, treats the problem as primarily an Euclidean one. Approaches typically partition the model into a hierarchical spatial bounding volume. Nodes of the hierarchy may be pruned by obtaining a lower bound on the distance to the contained geometry and comparing that distance to an upper bound on the minimum distance obtained through a depth-first descent in the hierarchy [12,8] or a sample point on the surface [11]. The primary research thrust in this area has focused on methods for quickly determining the lower bound on the distance at a node.

A different approach has been found in work on sculptured surfaces, such as B-splines [13,10]. Often a local solution to the minimum distance is desired, rather than a global solution. The distance between two parametric surfaces, *F(u,v)* and *G(s,t)*, may be described by

$$D^2(u,v,s,t) = \left\| F(u,v) - G(s,t) \right\|^2 .$$

Extrema of the distance can be found by differentiating and finding the roots of the resulting set of equations. We remove the parameters for clarity and denote the partial derivative of the surface with a subscript.

$$\begin{aligned} (F-G)\cdot F_u &= 0 \\ (F-G)\cdot F_v &= 0 \\ (F-G)\cdot G_s &= 0 \\ (F-G)\cdot G_t &= 0. \end{aligned}$$

The roots of these equations yield a set of local extrema in distance between the two models. Choosing the minimum solution provides the global minimum distance solution. This approach is very different from the Euclidean pruning method used for polygonal models and is the basic approach adapted in this research by using the spatialized normal cone hierarchy.

4.2.2 Local minimum distance for polygonal models

We start by describing the solution to the minimum distance between a space point and a polygonal model. Ultimately, we wish to find all the surface features, such as faces, edges, or vertices, on the model such that the range of normals for the feature contains the vector from the space point to the closest point on the feature (Figure 4).

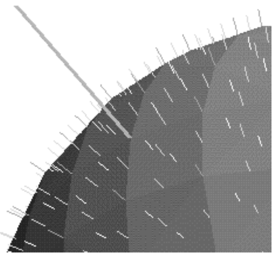


Figure 4: The line between the space point and the feature is in line with the normal at the feature.

The minimum distance is the length of the shortest such vector to a feature on the model. Note that the solution, couched in terms of normals, is the same technique we used for perspective silhouette extraction. Indeed, the nodes of the normal cone hierarchy can be pruned in the same manner as for silhouettes, just by changing the test from orthogonal view and normal cone to a test for parallel minimum distance and normal cone vectors.

Also, it is not necessary to select only the shortest line as the solution. Instead, all the lines that satisfy the criteria may be used. These are the local minimum distance solutions from the space point to the model (Figure 5), a problem that is difficult to even formulate with the standard techniques for global minimum distance.

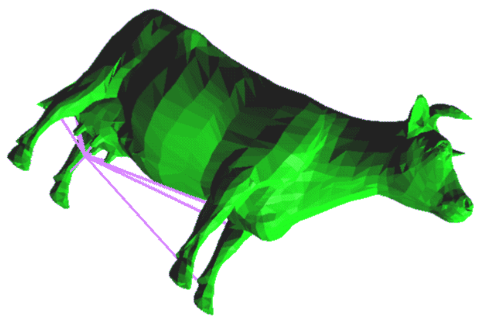


Figure 5: All the solutions satisfying the normal cone approach. The local minimum distance lines extend to each leg, the cow's udder, and tail.

4.2.3 Leaf tests for local minimum distance

The leaf test for the local minimum distance is much more complicated than the leaf test for silhouettes, or even for global minimum distance. In a global minimum leaf test, the minimum distance between the space point and the leaf triangle is computed and compared to previous leaf distances.

To test for a locally minimum distance, the leaf test finds the closest point on the triangle to the space point. The feature associated with this closest point must contain a normal that is parallel with the vector from the closest point to the space point, *the minimum distance vector*. The closest point may lie on a triangle face, edge, or vertex, each with a characteristic range of normals.

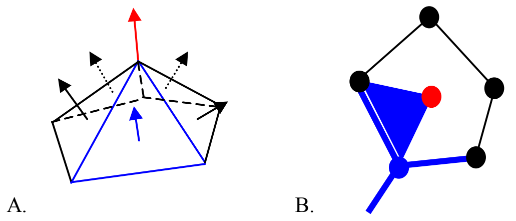


Figure 6: A. The vertex normal is surrounded by triangles. B. Each triangle normal maps to a point on the Gauss sphere and each edge maps to an arc. We share the normal range for a vertex among each surrounding triangle.

If the closest point lies on the triangle face, one must only compare the triangle normal with the minimum distance vector. If the closest point lies on a triangle edge, the minimum distance vector must be compared against the span of normals from the triangle face to the edge normal.

When the closest point matches up with a triangle vertex, the minimum distance vector must be compared against the span of normals for the vertex. Each vertex normal covers an area on the Gauss sphere defined by the surrounding triangle face normals (Figure 6). This area is divided into non-overlapping regions to associate with each surrounding triangle for computational efficiency and to prevent redundant solutions. A possible region would be to take the span from the triangle normal to the vertex normal and then half of each edge span. However, that produces a quadrilateral on the sphere that may cross itself in complex ways. Instead, the triangular span from the triangle normal to the vertex normal to the left neighbor's face normal (Figure 6.B.) is associated with each triangle vertex. Given this association, it is easy to determine if the minimum distance vector falls within the triangular range of normals.

4.2.4 Local minimum distance for two models

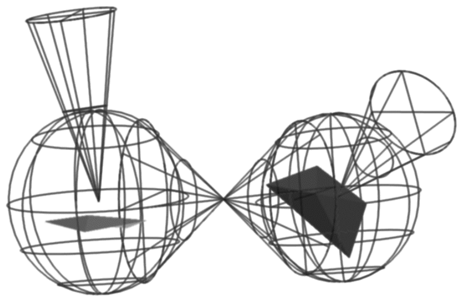


Figure 7: The view cone becomes the dual view cone – encompassing all possible normals between the models.

We are now ready to extend this approach to problems involving two different models, such as the local minimum distance between two polygonal models. Essentially, when there are two models, we must account for a range of vectors between nodes on the two models, not just the range of vectors between a space point and a node of a single model. The view cone now becomes the *dual view cone*, the span of vectors possible between the two bounding spheres (Figure 7). The pruning test for a node checks if the normal cones are parallel to the dual view cone and inward facing. This extension allows pruning of each model's normal cone tree down to leaf nodes that may meet the local minimum distance requirement. The leaf test is just the leaf test for a single model applied twice, once to the triangle in each leaf.

4.2.5 Results

First, we tested the space point to model method with a variety of models. The models rotated randomly while the space point stayed fixed a short distance above the surface.

	Sphere	*Holes3*	*Small bunny*	*Large bunny*
# triangles	32,700	11,800	10,800	69,500
Time (ms)	0.06	0.05	0.1	0.2

Figure 8: The local minimum distance from a point in space to a model is fast for a range of model types.

The local minimum distances (including the global minimum distance) from a point in space to a model (Figure 8) are computed in sub-millisecond time. This makes the presented approach appropriate for a number of tasks, including haptic rendering of polygonal models [19] and volumetric conversion [20].

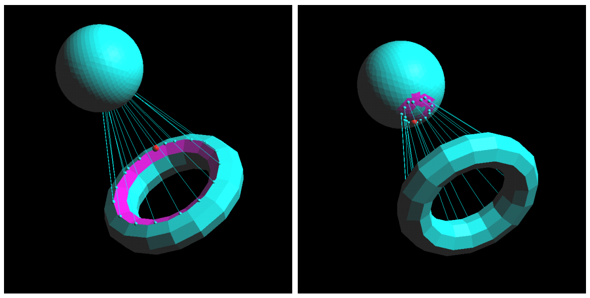


Figure 9: The local minimum distances between a torus and sphere. The highlighted color indicates a tested leaf node and the lines show local solutions.

For testing this method for the distance between two models, we compared the speed of the local minimum distance method with the global minimum distance method from the PQP package.

	1 sphere 2 torus	1 torus 2 cow	1 torus 2 bunny
# Triangles 1	8192	4096	4096
# Triangles 2	4096	5804	69451
PQP (secs)	0.007	0.006	0.0057
Normal cone	0.004	0.008	0.0059

Figure 10: Timing results for finding the distance between models.

The normal cone method worked best on fairly smooth surfaces. This suggests it may be a good choice for certain classes of models, such as those derived from subdivision surfaces. It had more difficulty on surfaces with a lot of fine detail, such as the bunny. The small bumps on the bunny translate into a wide range of normals in a small area, as well as producing numerous local minima. The normal cone method is competitive with the global method, and is returning all the local minima as well, as in Figure 11.

4.2.6 Discussion

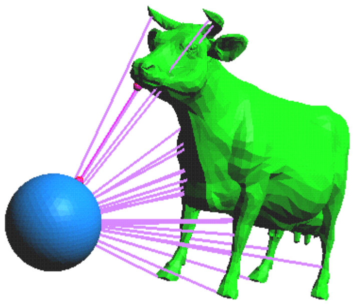


Figure 11: The local minimum distance returns more information about the distance between models than the global minimum distance.

This project was motivated by the need for a predictive contact algorithm in a haptic rendering system [17]. The haptic system needed to initialize local tracking methods in order to provide high response feedback. A global method will only report on one location, possibly missing other nearby regions of interest. Additionally, a simple modification to a global algorithm to return all pairs of triangles within a certain distance can result in thousands of pairs to interpret. Finding the local minima returns enough potential contact points to be useful without overloading the system.

In Figure 11, the normal cone method finds potentially interesting points on the legs, head and body of the cow compared to the one point provided by the global minimum. We expect that these local minimum distances may provide additional useful guidance for tasks such as path planning, simulation, and haptics.

4.3 Shadow boundary computations

The local minimum distance method found points on the models with normals collinear with each other and with the minimum distance vector. Another interesting case is obtained by finding points on the models with surface normals collinear with each other but orthogonal to their connecting vector.

If the surface normals point in the same direction, then the set of solution lines defines an envelope that is mutually tangent to each model. If one model is considered an area light source and the other model considered a light blocker, then this envelope defines the boundary of the hard shadow, or umbra, of these two objects. If the surface normals point in opposite directions, but still orthogonal to their connecting vector, then the envelope defines the boundary of the soft shadow, or penumbra (Figure 12).

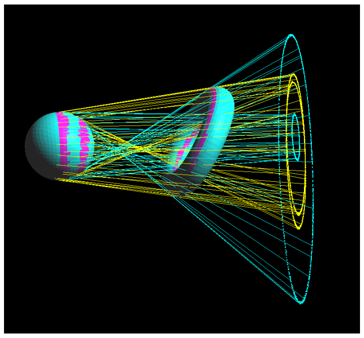


Figure 12: The shadow umbra and penumbra boundaries are defined as lines mutually tangent to each model. The spatial normal cone hierarchy enables interactive computation of these boundaries.

The leaf test for shadow boundaries uses the same normal span division structure as the local minimum distance leaf test. In this case, though, we are looking for edges and vertices with a shared normal direction and with normals orthogonal to the line between them.

4.3.1 Results

We tested the speed of the spatialized normal cone hierarchy approach using a sphere light source with 2000 triangles and a torus consisting of 4000 triangles. The method computed the set of umbra and penumbra solution lines at an update rate of 4Hz. A considerable portion of this time was spent moving the thousands of solution lines into the view space out of their model object spaces.

Reconstructing the exact shadow is a higher dimensional process [14]. However, an approximate shadow can be obtained using these boundaries. Other problem areas, such as discontinuity meshing in radiosity solutions [15], make use of the boundaries of the shadows to guide effort in solving for global illumination.

5 Discussion and Conclusion

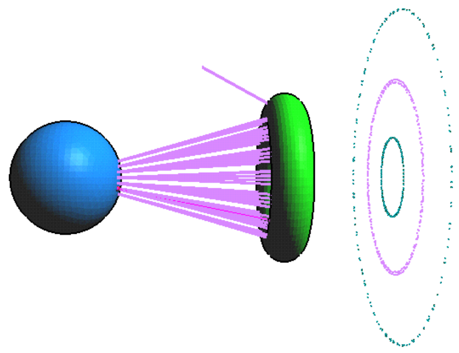


Figure 13: Silhouettes, local minimum distances for pt-model and model-model, and shadow umbra and penumbra boundaries all being computed simultaneously using spatialized normal cone hierarchies at interactive rates.

As Figure 13 shows, a spatialized normal cone hierarchy is a flexible data structure and pruning technique useful in a variety of applications. Prior techniques for using orientation information in geometric computations for polygonal models have been limited to interactions between a point source and the model. Backface culling and silhouette edge extraction are important examples of this problem domain. We have developed two new extensions to the point-model normal domain: variable precision silhouettes and local minimum distance. In addition, we have extended the domain of problems over which surface normal information plays a role into problems that depend on the interplay between two models. Using this new formulation, we are able to efficiently solve two new problems in the polygonal domain: the local minimum distance between two models and shadow boundary computation.

6 Acknowledgements

The authors would like to thank the members of the Utah graphics and virtual prototyping groups for their support. This work was supported by NSF grant DMI 9978603 and the NSF STC for Computer Graphics and Visualization.

7 REFERENCES

[1] Luebke, D. and Erickson, E., "View-Dependent Simplification of Arbitrary Polygonal Environments", Computer Graphics Proceedings, Annual Conference Series, SIGGRAPH 1997, pp. 199-208. 1997.

[2] Shirman, L. and Abi-Ezzi, S., "The Cone of Normals Technique for Fast Processing of Curved Patches," EUROGRAPHICS'93, Vol. 12.3., pp. 261-272. 1993.

[3] Kumar, S., Manocha, D., Garrett, W. and Lin, M., "Hierarchical Backface Computation," in Proc. of 7th Eurographics Workshop on Rendering, pp. 231-240. 1996.

[4] Zhang, H. and Hoff, K., "Fast Backface Culling Using Normal Masks," In *Proc. 1997 Symposium on Interactive 3D Graphics*, pp.103-106. April, 1997.

[5] Benichou, F. and Elber, G., "Output Sensitive Extraction of Silhouettes from Polygonal Geometry," in The Seventh Pacific Conference on Computer Graphics and Applications. Seoul, Korea. October 5 - 7, 1999.

[6] DoCarmo, M. *Differential Geometry of Curves and Surfaces*. Prentice-Hall. 1976.

[7] Gooch, B., Sloan, P.-P., Gooch, A., Shirley, P., and Riesenfeld, R., "Interactive Technical Illustration," in *1999 ACM Symposium on Interactive 3D Graphics*, pp. 31-38, ACM SIGGRAPH, April 1999.

[8] Larsen, E., Gottschalk, S., Lin, M., and Manocha, D., "Fast Proximity Queries with Swept Sphere Volumes," Technical report TR99-018, Department of Computer Science, University of N. Carolina, Chapel Hill.

[9] Bobrow, J.E., "Optimal robot path planning using the minimum-time criterion," IEEE Journal of Robotics and Automation, 4(4), pp. 443-450, Aug. 1988.

[10] Snyder, J., "An Interactive Tool for Placing Curved Surfaces without Interpenetration," in *Proceedings of Computer Graphics*, SIGGRAPH 1995 pp. 209-218, 1995.

[11] Johnson, David E. and Cohen, Elaine, "A framework for efficient minimum distance computations," *Proc. IEEE Intl. Conf. Robotics & Automation*, Leuven, Belgium, May 16-21, 1998, pp. 3678-3684.

[12] Quinlan, Sean. "Efficient Distance Computation between Non-Convex Objects," *IEEE Int. Conference on Robotics and Automation*, pp. 3324-3329, 1994.

[13] Lin, Ming and Manocha, Dinesh. "Fast Interference Detection Between Geometric Models," The Visual Computer, pp. 542-561, 1995.

[14] Stark, M., Cohen, E., Lyche, T., Riesenfeld, R., "Computing Exact Shadow Irradiance Using Splines," in *SIGGRAPH 99 Conference Proceedings*, pp. 155-164. 1999.

[15] Lichinksi, D., Tampieri, F., and Greenberg, D., "Discontinuity Meshing for Accurate Radiosity," *IEEE Computer Graphics and Applications*, Vol. 12, No. 6, November 1992, pp. 25-39.

[16] Northrup, J.D. and Markosian, L., "Artistic Silhouettes: A Hybrid Approach," NPAR'2000, to appear.

[17] Nelson, D., Johnson, D., and Cohen, E., "Haptic Rendering of Surface-to-Surface Sculpted Model Interaction," in Proc. 8th Annual Symp. on Haptic Interfaces for Virtual Environment and Teleoperator Systems, (Nashville, TN), ASME, November 1999.

[18] Sander, P., Gu, X., Gortler, S., Hoppe, H., Snyder, J., "Silhouette Clipping." in Computer Graphics Proceedings, SIGGRAPH 2000. New Orleans. pp. 327-334.

[19] Ruspini, D., Kolarov, K., and Khatib, O., "The Haptic Display of Complex Graphical Environments," in Computer Graphics Proceedings, SIGGRAPH 1997. Aug. 3-8. pp. 345-352.

[20] Whitaker, R. and Breen, D., "Level-Set Models for the Deformation of Solid Objects," Proceedings of the 3rd International Workshop on Implicit Surfaces, Eurographics Association, June 1998, pp. 19-35.