Accelerated Proximity Queries Between Convex Polyhedra By Multi-Level Voronoi Marching*

Stephen A. Ehmann **Ming C. Lin**

Department of Computer Science
University of North Carolina
Chapel Hill, NC 27599-3175
{ehmann,lin}@cs.unc.edu
http://www.cs.unc.edu/~geom/SWIFT/

Abstract

We present an accelerated proximity query algorithm between moving convex polyhedra. The algorithm combines Voronoi-based feature tracking with a multi-level-of-detail representation, in order to adapt to the variation in levels of coherence and speed up the computation. It provides a progressive refinement framework for collision detection and distance queries. We have implemented our algorithm and have observed significant performance improvements in our experiments, especially on scenarios where the coherence is low.

1 Introduction

Proximity queries, i.e. distance[1] computations and the closely related collision detection problems, are ubiquitous in robotics, design automation, manufacturing, assembly and virtual prototyping. The set of tasks include motion planning, sensor-based manipulation, assembly and disassembly, dynamic simulation, maintainability study, simulation-based design, tolerance verification, and ergonomics analysis.

Proximity queries have been extensively studied in robotics and several specialized algorithms have been proposed for convex polyhedra as well as hierarchical approaches for general geometric models. In this paper, we present a novel algorithm that precomputes a hierarchy composed of a series of bounded error levels-of-detail (LODs) and uses them to accelerate proximity queries. Algorithms to generate LODs for polygonal models have been widely used for rendering acceleration, animation and simulation applications [8, 22, 24]. One of our goals is to take advantage of multiresolution representations commonly used in rendering applications and use them for proximity queries as well.

Given a convex polyhedron, our algorithm precomputes a bounded error level-of-detail (LOD) hierarchy. It constructs a series of bounding volumes that enclose the original polyhedron. Then, it establishes parent-child relationships between the features of adjacent levels. At runtime, the hierarchy is traversed according to the type of query being performed. Within each level, the surface of an object is "marched" across using a modified Lin-Canny [16] closest feature tracking algorithm based on Voronoi regions. We refer to such a technique as "Voronoi marching" in this paper. Spatial locality and temporal coherence is captured by both the Voronoi regions and the hierarchy of LODs.

The algorithm has been implemented and analyzed using various benchmarks. Experiments were performed using various shapes of objects and various multiresolution options. We observe significant (up to nearly one order of magnitude in some cases) speedups for each type of query.

1.1 Main Results

In this paper, we present an accelerated proximity query algorithm between moving convex polyhedra which exploits multiresolution representations. Our main contributions are:

- An accelerated proximity query algorithm between convex polyhedra using multi-level Voronoi marching. The substantial performance improvements are mainly due to the use of a multiresolution representation and a faster Voronoi marching algorithm.

- A progressive refinement algorithmic framework for

*Supported in part by ARO DAAG55-98-1-0322, NSF EIA-9806027, NSF DMI-9900157, NSF IIS-9821067 and Intel

[1] Distance is commonly defined as the Euclidean separation distance by many applications, and we adopt the same definition in this paper.

proximity computation. For many applications including motion planning and tolerance verification, a relatively inexpensive *approximate* distance computation with bounded error is sufficient. Our framework allows applications to progressively refine the distance estimate to suit their needs, while minimizing overall computation cost.

- A better understanding of the issues involved in designing suitable level-of-detail representations for proximity queries. These issues include level of coherence, object aspect ratios, and contact scenarios.

1.2 Organization

The rest of the paper is organized as follows. Section 2 gives a brief survey of related work. Section 3 presents an overview of our approach. The design and computation of the multi-level-of-detail representation is described in section 4. Next, a proximity query algorithm using the multiresolution representation is described in section 5 along with other ways to accelerate queries. Section 6 describes our prototype implementation and shows the performance of our system. Finally, we conclude with future research directions in section 7.

2 Previous Work

Distance computation and intersection (collision) detection problems have been fundamental subjects of study in robotics, computational geometry, simulation, and physical-based modeling. There is a wealth of literature on both analyzing the theoretical complexity of proximity queries and on designing algorithmic solutions to achieve interactive performance. We will limit the scope of the discussion in this paper to mostly rigid convex polyhedra, although some of the techniques may be applicable to other domains and model representation as well.

2.1 Proximity Queries for Convex Polyhedra

Most of the earlier work has focused on algorithms for convex polyhedra. A number of algorithms with good asymptotic performance have been proposed in the computational geometry literature [5]. Using hierarchical representations, an $O(log^2 n)$ algorithm is given in [3] for the convex polyhedral overlap problem, where n is the number of vertices. This elegant approach is difficult to implement robustly in 3D, however.

Good theoretical and practical approaches based on the linear complexity of the linear programming problem are known [18, 23]. Minkowski difference and convex optimization techniques are used in [10] to compute the distance between convex polyhedra.

Erickson, et al [7] recently proposed a new class of kinetic data structures for collision detection between convex polyhedra. This class of hierarchical representations has only been analyzed for the 2D case however.

In applications involving rigid motion, geometric coherence has been exploited to design algorithms for convex polyhedra based on either traversing features using locality or convex optimization [2, 4, 16, 15, 19]. These algorithms exploit the spatial and temporal coherence between successive queries and work well in practice.

2.2 Hierarchical Representations

Bounding volume hierarchies are presently regarded as one of the most general methods for performing proximity queries between general polyhedra. Specifically, sphere trees, cone trees, axis-aligned box trees, oriented box trees, k-d trees and octrees, trees based on S-bounds, and k-dops have been used for fast intersection queries for general polyhedra, as well as polygon soups [13, 20, 11, 21, 1, 14]. For most scenarios, these hierarchies excel at intersection detection but do not do so well when it comes to distance computation.

2.3 Multiresolution Techniques

Multiresolution modeling techniques, such as model simplification, have been proposed to extract the shape of the underlying geometry [25]. A recent survey on polygonal model simplification is available [17]. The main idea behind using a multiresolution hierarchy is to compute and utilize a correspondence between the original model and a simplified one. We will discuss the use of a vertex removal simplification algorithm due to Garland and Heckbert [9].

For proximity query, Guibas, et al. [12] proposed an elegant approach that exploits both coherence of motion and hierarchical representation for faster distance computation. Our approach differs from their *H-Walk* algorithm in that our algorithm can easily compute an approximated distance with a guaranteed error tolerance without always descending and/or ascending the entire hierarchy.

3 Algorithm Overview

Our algorithm operates on orientable 2-manifolds that are closed and represented using triangles. The polyhedra must be convex and can only undergo rigid motion. We will use the terms “polyhedron” and “object” interchangeably.

Multiresolution techniques typically consist of two main components. They involve the precomputation of a hierarchical representation upon which subsequent queries are performed. We wish to compute a multiresolution representation that supports proximity queries.

In the preprocessing stage, we compute a level-of-detail representation for each object. This hierarchy is organized as a sequence of convex polyhedra P_0, P_1, P_2, $\cdots$, P_k where P_0 is the input polyhedron. Each successive polyhedron is composed of fewer features and has the property that it bounds the original object. Furthermore, a correspondence is established between successive levels in each direction of the sequence. More details of the actual construction of this representation are given in section 4.

A query using this hierarchy is performed in much the same way for each type of proximity query. Basically, as long as certain levels of two objects are intersecting, then the hierarchy is refined. When two levels are found to be disjoint, then it may be possible to end the query at a subsequent point of refinement. Furthermore, for queries other than intersection, a tolerance may be provided which specifies how close objects must be in order to answer the query. In section 5 the hierarchical query is described in more detail.

4 Hierarchical Representations

The multi-level-of-detail representation of each object that is constructed in the algorithm's preprocessing stage must have certain properties in order to be useful and efficient for performing proximity queries. Next, we describe desirable characteristics for the representation, discuss the decisions involved in the design of the hierarchy, and touch upon various tradeoffs along the way.

4.1 Terminology

Recall that the hierarchy is organized as a sequence of convex polyhedra P_0, P_1, P_2, $\cdots$, P_k where P_0 is the input polyhedron. We will call P_0 the finest level and P_k the coarsest level. We will call the level P_{i-1} the "child" of the level P_i and the level P_{i+1} the "parent" of P_i. We use the convention that the finest object is at the bottom of the hierarchy so the term *moving up* the hierarchy means moving to a coarser level. Moving to a finer level is termed *moving down* the hierarchy.

The maximum deviation of P_i from P_0 is represented by ϵ_i. The computed distance between certain levels of two objects that are found to be disjoint is δ and the maximum error associated with the distance is ϵ. The distance tolerance given by the application is δ_{max} and the error tolerance is ϵ_{max}.

4.2 Desired Features

To design a multi-level representation for accelerating distance computation while preserving locality and coherence, our goal is to produce a hierarchy that provides the following characteristics:

- **Simplified Combinatorial Complexity:** Each level of representation in the hierarchy should have less combinatorial complexity than its child and higher combinatorial complexity than its parent. Ideally we would like to create $k = O(\log n)$ levels, where n is the number of vertices in the original polyhedron P_0. This is possible if a constant fraction of features are eliminated for each level. In addition, it is wise to stop the hierarchy construction when a certain number of levels or features has been reached.

- **Bounding Volumes:** If we wish to have a mechanism to compute approximate distances with bounded error tolerances ϵ_{max}, then we can make use of the levels P_i of the hierarchy which bound the original polyhedron P_0 with a global surface deviation $\epsilon_i \leq \epsilon_{max}$. Query performance can be further improved by aiming to create a hierarchy where the bounding volumes are kept as small as possible.

- **Local Correspondence:** For each level of the hierarchy P_i for $i > 0$ that bounds P_0, there must also be spatial coherence between it and its adjacent levels. This implies that a feature on polyhedron P_i will have a corresponding feature on P_{i+1} and on P_{i-1} which are proximate in position and orientation. These are called the parent and child features respectively.

4.3 Construction

The levels of the hierarchy are constructed in order starting with P_1. In particular, when constructing P_i, P_{i-1} provides the topological and geometric information required to reduce the number of features while P_0 provides geometric information to satisfy the bounding criterion. The process of constructing a level in the hierarchy is given by the following steps to create P_i:

1. Create P_i as a high quality simplification of P_{i-1}.

2. Make P_i convex by computing its convex hull.

3. Compute the center of mass of P_i and translate P_i such that its center of mass coincides with the P_{i-1}'s center of mass.

4. Scale P_i from its center of mass so that it barely bounds P_0 [2].

5. Compute the maximum deviation ϵ_i of P_i from P_0. This is the same as computing the Hausdorff distance between P_i and P_0.

6. Assign the feature correspondences from P_i to P_{i-1} and from P_{i-1} to P_i.

For the first step, any simplification algorithm may be used as long as the topology is maintained. In our implementation, we used the QSlim system which is a publicly available implementation of Garland and Heckbert's quadric error metric vertex removal algorithm [9]. For the second step, we used the QHull convex hull library that is also publicly available.

The computation of the center of mass is straightforward for a closed polyhedron. To scale P_i in step four, the maximum scaling required over all of the faces of P_i is found and applied. The scaling required for a face is computed by finding the extremal vertex on P_0 in the direction of the (outward pointing) face normal and computing the scaling factor required to cause the vertex to coincide with the new face's supporting plane.

The Hausdorff distance (ϵ_i) can be computed in this context by computing the maximum deviation over all the vertices of P_i. The deviation of a vertex of P_i is computed by computing its distance from P_0. The vertices are the only points on P_i that have to be checked because they represent local maxima of the deviation function over P_i.

To assign children (features of P_{i-1}) to the features of P_i, the nearest feature of P_{i-1} is found for each vertex of P_i and the nearest vertex is selected from this nearest feature. Each neighbor of each vertex of P_i (including the vertex) is assigned the nearest vertex as its child feature. To assign parents (features of P_i) to the features of P_{i-1}, the extremal vertex of P_i is found for each face of P_{i-1}. Each face as well as its edges and vertices are assigned the extremal vertex as their child. These feature correspondence schemes were chosen for their spatial coherence. Of course, there are other schemes that may work better but it is unlikely that a sizeable overall improvement would be gained.

In our implementation, there is a triangle count cutoff that determines when to stop building the hierarchy. In addition, there is a constant factor that determines what the feature reduction rate is across levels. These parameters affect the query performance and are discussed later.

[2] This was incorrectly stated as "P_{i-1}" in the published IROS 2000 paper.

5 Proximity Query

Proximity queries can be in the form of intersection detection, error-bounded approximate distance computation, exact distance computation, or contact determination. The hierarchy is queried in much the same way for all four of these query types. The basis is a multiresolution extension on the algorithm proposed by Lin and Canny [16].

In the case of intersection detection, the search normally starts at a pair of features related to the closest features from the previous query so that coherence may be exploited. Like in Lin-Canny, the distance between two convex polyhedra is minimized by marching across their surfaces. If an intersection is detected, a finer level of the hierarchy is traversed to. If at *any* level, we reach a minimum in the distance function and the objects are disjoint, then this is reported. With no additional cost, an approximate distance can be provided as δ with the error ϵ, when the objects are disjoint. δ is the distance between the levels of the two objects; $\epsilon = \epsilon_i + \epsilon_j$ is an associated error and the sum of the two level deviations. The pair of closest features are used to find the features to be used for the next query. This is done by traversing their parent pointers, using the deviations ϵ_i, and keeping track of the distance δ that is decreased by the differences in deviation. If an intersection has been detected and P_0 has been reached for both objects, then intersection is reported. The pair of intersecting features is stored for the next query.

For error-bounded approximate distance computation, the application can provide a distance tolerance δ_{max} and an error tolerance ϵ_{max}. If the levels of the two objects intersect all the way to the finest levels of both objects, then intersection is reported and the pair of features is stored. Otherwise, two levels were found to be disjoint during the refinement. The disjoint features are used as in the intersection case to determine the features to be used for the next query. If at any point $\delta > \delta_{max}$ the query is terminated and nothing is reported. If at any point, $\epsilon \leq \epsilon_{max}$ then the error tolerance has been met and δ and ϵ are reported. This means that the exact distance is in the range $[\delta, \delta + \epsilon]$.

To compute exact distance, the same method is followed but only the distance tolerance δ_{max} is specified. As long as $\delta \leq \delta_{max}$, the distance is refined until the finest levels are reached at which point δ is reported. Finally, contact determination (closest points) queries can be handled the same as exact distance ones.

6 Experimental Results

The models used to test our query algorithm were empirically created by taking the convex hull of randomly

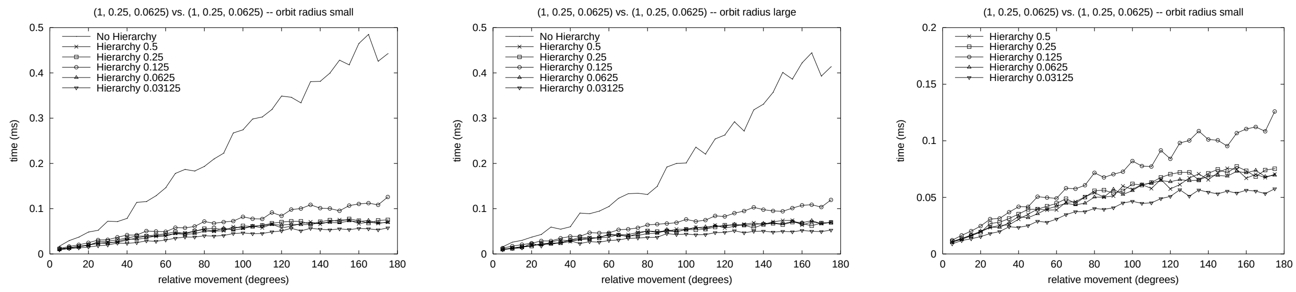

Figure 1: Performance of Intersection Detection

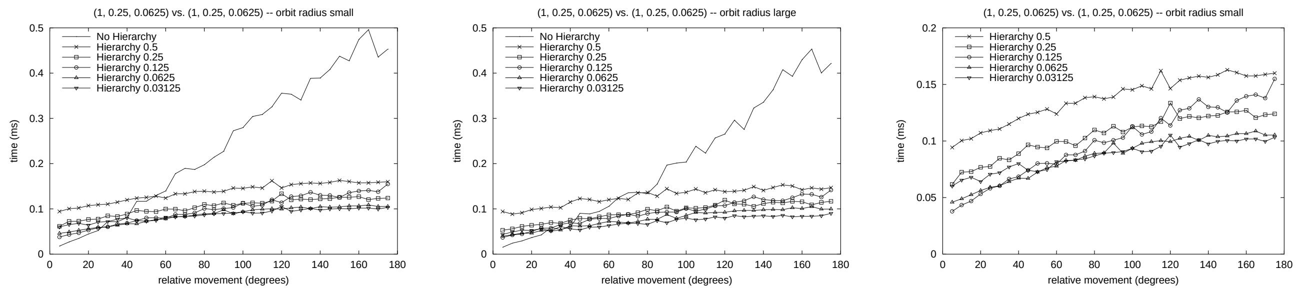

Figure 2: Performance of Exact Distance Computation

sampled points on objects of various aspect ratio. Four different ellipsoidal models were created, each with 2000 triangles. Their aspect ratios are (1,1,1), (1,1,0.1), (1,0.1,0.1), and (1,0.25,0.0625). Each of the models was normalized to have a bounding sphere of radius r_0.

Following the performance benchmarking algorithm due to Mirtich [19] and used in [12], we created 20 scenarios. The algorithm has one object remain fixed while another orbits about it in an elliptical trajectory. An orbit radius is defined as the distance between the centers of the models when they are closest in the orbit. All possible pairs of the four models (10) were used and their orbit radius set to be either $2r_0$ (small) or $3r_0$ (large). Thus, the models either just touch or are always separated by a distance of at least r_0.

We obtained empirical observations for our implementation by running programs on an SGI Reality Monster using a single 300 MHz MIPS R12000 CPU. Our implementation of the algorithm that marches within a level was compared to *V-Clip* [19] and found to faster[3]. No hierarchical or lookup table enhancements were used in the comparison.

[3] In the published IROS 2000 paper we stated that our implementation was nearly twice as fast but we have since gathered more data and found that to be an inaccurate assessment.

6.1 Impact of the Multiresolution Hierarchy

The level-of-detail hierarchy we have implemented can be used to achieve further speedups for both intersection detection and exact distance computation. Shown in Figure 1 is the performance of various intersection detection queries. Likewise, Figure 2 shows exact distance computation performance. The titles on the graphs reflect the object pairs that were used as well as whether the orbit was small or large. The two leftmost graphs in each figure compare various hierarchy configurations against no hierarchy. The rightmost graphs in each figure leave out the data when using no hierarchy so that the differences in hierarchy parameters are easier to distinguish. The hierarchies were built with various triangle reduction rates. For example, the hierarchy with a 0.25 reduction rate was constructed by keeping a quarter of the triangles between levels. The hierarchies were created by using the associated triangle reduction factor down to a minimum of 100 triangles. At least two levels were created for each hierarchy.

The reason for the difference in the two query types is due to the fact that in exact distance computation, we cannot stop once the models are found to be disjoint like we can in intersection detection. It can be seen from the profiles of our timing curves that our multi-level Voronoi marching algorithm is able to keep the impact of the relative motion of the objects under control. Approximate distance computation is highly dependent upon the application. It can be shown that its cost lies in between the cost of intersection detection and exact distance computation.

Since QSlim builds such tight simplifications, it seems sufficient to simply build a couple of coarser levels for collision detection. The coarser the levels are, the better the performance. This has been noticed for even coarser simplifications than those shown. Hierarchy 0.125 is anomalous because its coarsest level consists of 250 triangles while the other hierarchies have fewer. For exact distance computation, the winners appear to be those hierarchies that have only two levels. We conjecture that the objects we used have only a modest complexity of 2000 triangles and the sum of the overhead associated with minimizing the distance function at each pair of levels can offset the benefit of making a good rough guess, if too many intermediate levels in each hierarchy were created.

7 Conclusion

We have presented a new algorithm for accelerating proximity queries between convex polyhedra using level-of-detail representations. We described the construction of the required hierarchical data structures and discussed various design issues involved. The runtime algorithm along with its issues were addressed and the performance presented. Our technical report [6] presents more detail about various performance issues, including the use of a directional lookup table and more diverse scenarios.

Our implementation has been shown to have good performance. The source code is available to the public and provides a complete and entire proximity query library. It is available at

http://www.cs.unc.edu/∼geom/SWIFT/

We have shown that a hierarchical representation integrated with Voronoi walking offers a progressive refinement framework for (approximate) distance computation, which optimizes performance while meeting the application's need for bounded error computation.

This research is the first step toward the design of distance query algorithms using multiresolution representations. There are many potential directions for future research. There is a need to develop a suitable metric for measuring or quantifying the optimality of the hierarchies, coherence levels, and level relationships. On the theoretical side, there is the interesting problem of proving a better bound on the computation of the distance between two convex polyhedra. The extension of this framework to non-convex objects and deformable models remains a very challenging problem.

References

[1] S. Cameron. Approximation hierarchies and s-bounds. In *Proceedings. Symposium on Solid Modeling Foundations and CAD/CAM Applications*, pages 129–137, Austin, TX, 1991.

[2] S. Cameron. Enhancing GJK: Computing minimum and penetration distance between convex polyhedra. *Proceedings of International Conference on Robotics and Automation*, pages 3112–3117, 1997.

[3] D. P. Dobkin and D. G. Kirkpatrick. Determining the separation of preprocessed polyhedra – a unified approach. In *Proc. 17th Internat. Colloq. Automata Lang. Program.*, volume 443 of *Lecture Notes Comput. Sci.*, pages 400–413. Springer-Verlag, 1990.

[4] P. Dworkin and D. Zeltzer. A new model for efficient dynamics simulation. *Proceedings Eurographics workshop on animation and simulation*, pages 175–184, 1993.

[5] H. Edelsbrunner. Computing the extreme distances between two convex polygons. *J. Algorithms*, 6:213–224, 1985.

[6] S. A. Ehmann and M. C. Lin. Accelerated proximity queries between convex polyhedra by multi-level Voronoi marching. Technical Report TR00-026, Computer Science Department, University of North Carolina at Chapel Hill, 2000.

[7] J. Erikson, L. Guibas, J. Stolfi, and L. Zhang. Separation-sensitive collision detection for convex objects. *Proc. of SODA*, 1999.

[8] G. Hirota, R. Maheshwari, and M. Lin. Fast volume-preserving free-form deformation using multi-level optimization. *Proc. Symposium on Solid Modeling and Applications*, 1999.

[9] M. Garland and P. Heckbert. Surface simplification using quadric error bounds. *Proc. of ACM SIGGRAPH*, pages 209–216, 1997.

[10] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between objects in three-dimensional space. *IEEE J. Robotics and Automation*, vol RA-4:193–203, 1988.

[11] S. Gottschalk, M. Lin, and D. Manocha. Obb-tree: A hierarchical structure for rapid interference detection. In *Proc. of ACM Siggraph'96*, pages 171–180, 1996.

[12] L. Guibas, D. Hsu, and L. Zhang. *H-Walk*: Hierarchical distance computation for moving convex bodies. *Proc. of ACM Symposium on Computational Geometry*, 1999.

[13] P. M. Hubbard. Interactive collision detection. In *Proceedings of IEEE Symposium on Research Frontiers in Virtual Reality*, October 1993.

[14] J. Klosowski, M. Held, Joseph S. B. Mitchell, K. Zikan, and H. Sowizral. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Trans. Visualizat. Comput. Graph.*, 4(1):21–36, 1998.

[15] M.C. Lin. *Efficient Collision Detection for Animation and Robotics*. PhD thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley, December 1993.

[16] M.C. Lin and John F. Canny. Efficient algorithms for incremental distance computation. In *IEEE Conference on Robotics and Automation*, pages 1008–1014, 1991.

[17] D. Luebke and C. Erikson. View-dependent simplification of arbitrary polygon environments. In *Proc. of ACM SIGGRAPH*, 1997.

[18] N. Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM J. Computing*, 12:pp. 759–776, 1983.

[19] Brian Mirtich. V-Clip: Fast and robust polyhedral collision detection. *ACM Transactions on Graphics*, 17(3):177–208, July 1998.

[20] S. Quinlan. Efficient distance computation between non-convex objects. In *Proceedings of International Conference on Robotics and Automation*, pages 3324–3329, 1994.

[21] H. Samet. *Spatial Data Structures: Quadtree, Octrees and Other Hierarchical Methods*. Addison Wesley, 1989.

[22] P. Schröder and D. Zorin. Subdivision for modeling and animation. *ACM SIGGRAPH Course Notes*, 1998.

[23] R. Seidel. Linear programming and convex hulls made easy. In *Proc. 6th Ann. ACM Conf. on Computational Geometry*, pages 211–215, Berkeley, California, 1990.

[24] E. Stollnitz, T. Derose, and D. Salesin. *Wavelets for Computer Graphics*. Morgan Kaufmann Publishers, 1996.

[25] Tiow-Seng Tan, Ket-Fah Chong, and Kok-Lim Low. Computing bounding volume hierarchies using model simplification. In *ACM Symposium on Interactive 3D Graphics*, pages 63–70, 1999.