Interactive Motion Planning Using Hardware-Accelerated Computation of Generalized Voronoi Diagrams*

Kenneth Hoff III Tim Culver John Keyser Ming C. Lin Dinesh Manocha
Department of Computer Science
University of North Carolina
Chapel Hill, NC 27599-3175
{hoff,culver,keyser,lin,dm}@cs.unc.edu
http://www.cs.unc.edu/~geom/voronoi/planner

Abstract

We present techniques for fast motion planning by using discrete approximations of generalized Voronoi diagrams, computed with graphics hardware. Approaches based on this diagram computation are applicable to both static and dynamic environments of fairly high complexity. We compute a discrete Voronoi diagram by rendering a three-dimensional distance mesh for each Voronoi site. The sites can be points, line segments, polygons, polyhedra, curves and surfaces. The computation of the generalized Voronoi diagram provides fast proximity query toolkits for motion planning. The tools provide the distance to the nearest obstacle stored in the Z-buffer, as well as the Voronoi boundaries, Voronoi vertices and weighted Voronoi graphs extracted from the frame buffer using continuation methods. We have implemented these algorithms and demonstrated their performance for path planning in a complex dynamic environment composed of more than 140,000 polygons.

1 Introduction

Motion planning is one of the fundamental problems in robotics and automation. Most of the earlier work has focused on the classic Piano Mover's problem. Besides robotics, this problem also arises in motion control and planning of digital actors or autonomous agents [KKKL94] in computer animation, maintainability study in virtual prototyping [CL95], drug design [FKL+97] and robot-assisted medical surgery [STK+94, TAL99]. This problem has been well studied for decades and a number of algorithms have been proposed. Most of them can be classified into global or local methods. Some of the well-known approaches include roadmap algorithms, exact and approximate cell-decomposition, and potential field methods [Lat91].

1.1 Related Work

Several algorithms have been proposed based on generalized Voronoi diagrams [CD87, ÓSY83, CB94, CB95a, CB95b, CB96, CKR97a, CKR97b, CMB97, Cho97, KC97, SAW99b, SAW99a]. The underlying idea is that the boundaries of generalized Voronoi diagrams or simplified Voronoi diagrams provide paths of maximal clearance between the robot and the obstacles. This characteristic of the paths generated by Voronoi-based algorithms is similar to those generated by the potential field based roadmap methods [BL91, CL90, CL93, Lat91]. However, due to the practical complexity of computing generalized Voronoi diagrams, the applications of such planners have been limited to environments composed of a few simple obstacles.

Our approach also treats Voronoi diagrams as paths of maximal clearance. However, we accelerate the computation by designing algorithms that make use of graphics hardware. Polygon rasterization graphics hardware has been used in geometric computation [GMTF89, RMS92, RR86, HCK+99a], and in motion planning for constructing configuration space [LRDG90]. Our method imposes no restrictions on input size or primitive type, is efficient for planning in dynamic environments, and is easy to implement. Though the computation is discrete, we enumerate all sources of errors and generate output within a specified tolerance.

1.2 Main Contribution

In this paper, we present techniques for fast motion planning that use a discrete approximation of the generalized Voronoi diagram computed with graphics hardware. We

*Supported in part by ARO Contract DAAH04-96-1-0257, NSF Career Award CCR-9625217, NSF grants EIA-9806027 and DMI-9900157, ONR Young Investigator Award and Intel.

show how to utilize rasterization hardware to compute the following information for path planning in complex environments with stationary and moving obstacles and other robots at *interactive* rates:

- Approximate distance functions with bounded error, suitable for not only classical motion planning in a static environment, but also for planning in a dynamic environment and for sensor-based planning.

- Voronoi neighbors, Voronoi boundaries, and Voronoi vertices, used to identify potential paths with important "junction points" or "meet points" to ensure the correct topological connection of paths.

- Color and distance buffers to provide "weights" for all Voronoi edges. These values can be further used to estimate potential "narrow" passages in configuration space, reduce the search space, establish milestones, or bias paths based on length, clearance, or other constraints.

We demonstrate their effectiveness with our prototype implementation of a potential field based planner in a three-dimensional configuration space. We show that it is feasible to plan motions of autonomous robots based on generalized Voronoi diagrams for highly complex environments composed of hundreds of thousands of primitives at interactive rates. Our approach is complementary to other techniques proposed for computing roadmaps. However, this technique is simple to implement and uses graphics hardware capabilities to achieve high performance.

1.3 Organization

The rest of the paper is organized as follows: In Section 2, we describe an overview of our approach. Section 3 presents our algorithm for computing the generalized Voronoi diagram using graphics hardware. Section 4 discusses the use of the discrete generalized Voronoi diagram for motion planning. We demonstrate our prototype system implementation in Section 5. Finally, we conclude with future research directions.

2 Algorithm Overview

In this section, we briefly describe the basic ideas of our approach, giving an overview of generalized Voronoi diagrams and polygon rasterization hardware. Next, we summarize how we accelerate the computation of generalized Voronoi diagrams with graphics hardware and use them for motion planning.

2.1 Generalized Voronoi Diagram

Let the set of input sites be denoted as $s_1, s_2, \ldots, s_n$. For each site s_i, define a distance function $d_i(\mathbf{x}) = \mathrm{dist}(s_i, \mathbf{x})$. The *Voronoi region of* s_i is the set $V_i = \{\mathbf{x} \mid d_i(\mathbf{x}) \leq d_j(\mathbf{x})\ \forall j \neq i\}$.

The collection of regions $V_1, \ldots, V_n$ is called the *generalized Voronoi diagram* or *GVD*, which partitions the space into cells suitable for proximity queries.

The (ordinary) Voronoi diagram corresponds to the case when each s_i is an individual point. The boundaries of the regions V_i are called *Voronoi boundaries*, which are loci of points equidistant to at least two sites. The *Voronoi vertices* are locations equidistant to at least three Voronoi sites. For sites such as points, lines, polygons, and splines, the Voronoi boundaries are portions of algebraic curves or surfaces.

2.2 Graphics Rasterization Hardware

Graphics hardware has readily available, and is often provided with desktop computers. To take advantage of advances in hardware development, we make use of standard Z-buffered raster graphics hardware for rendering polygons. The color buffer stores the attributes (intensity or shade) of each pixel in the image space; the depth buffer (Z-buffer) stores the depth of every visible pixel. Given the vertices of a triangle, the rasterization hardware interpolates depth linearly across the triangle's interior. All raster samples covered by a triangle have an interpolated depth.

2.3 Key Concept

We compute a discrete Voronoi diagram by rendering a three-dimensional distance mesh for each site. A site may be a point, line segment, polygon, polyhedron, curve, or curved surface. The 3D polygonal distance mesh is a bounded-error approximation of a possibly non-linear distance function over a plane. Each site is assigned a unique identifying color, and the corresponding distance mesh is rendered in that color using a parallel projection. The graphics system performs a depth test for each pixel in order to resolve the visibility of surfaces. The depth buffer keeps a running minimum depth as polygons are rendered. When the minimum depth is updated, the frame buffer is also updated with the pixel's color. Thus, the rasterization provides, for each pixel, the identity of the nearest site (encoded as a color) and the distance to that site (encoded as a depth value). The error in the mesh is bounded to be smaller than the half the distance between diagonally adjacent pixel samples, in order to maintain an accurate Voronoi diagram.

2.4 Motion Planning Using GVD

The depth buffer stores the distance values needed for many motion planning algorithms. The distance gradient is easily computed by finite differences. The color ID for each Voronoi site is used to identify the nearest neighbors, the Voronoi boundaries, the Voronoi vertices, and the Voronoi graph. By traveling on the Voronoi boundaries, the robot steers away from all obstacles. The Voronoi boundaries can also provide "hints" for sampling the configuration space for probabilistic roadmap methods. Furthermore, we can use the distance information to eliminate paths that are clearly not feasible, or to bias the robot toward regions of with sufficient clearance or short path length. Since we compute the Voronoi diagram of the environment at interactive rates, these techniques are useful for *dynamic environments* where obstacles or other robots are moving, and for *sensor-based planning* where the robot constructs a map of the partially known environment as it explores with sensor information.

3 Computing Generalized Voronoi Diagrams Using Graphics Hardware

In 2D, the distance function for a point is a circular cone. Our algorithm approximates this cone with a fan of narrow radial triangles. In this section, we describe distance functions for points and other site types in the plane, and also in a planar slice of three-space. We also present techniques for computing error-bounded polygonal approximations to these distance functions. More details are given in [HCK$^+$99b].

3.1 2D Voronoi Diagrams

For a *point* in 2D, the conical distance function is approximated with a fan of radial triangles. The maximum error in this approximation is at the mesh edge. In the general case, the radius of the mesh must be equal to the diameter of the scene (though in specific cases, it can often be made much shorter). The number of triangles in the mesh is chosen so as to commit the maximum allowable error ϵ at the mesh edge. A reasonable value for ϵ is half the distance between diagonally adjacent pixels, measured in scene coordinates. Under this assumption, a simple calculation [HCK$^+$99a] shows that the distance mesh for a point requires 60 triangles for a 512×512 display resolution, or 85 triangles for a 1024×1024 display resolution. In practice, the radius of the mesh can safely be taken to be significantly less than the scene diameter, allowing for a smaller mesh with fewer triangles.

An *open line segment* in 2D has a tent-shaped distance function. Since the function is linear, it can be meshed without error. The algorithm draws two quadrilaterals. A *polygonal chain* in 2D is approximated by a pair of quadrilaterals for the interior of each edge, together with a partial cone at each vertex on its convex side.

3.2 3D Voronoi Diagrams and Other Generalizations

Our hardware-based algorithms have been extended to compute 3D Voronoi diagrams. The implementation of our motion planner presented in this paper works in three degrees of freedom based on a 2D Voronoi diagram. Our extended algorithm, however, could be applied to planners with more degrees of freedom. The 3D algorithm is described in [HCK$^+$99a, HCK$^+$99b] and has been used successfully for planning of a free-flying rigid robot in a 3D environment [PHLM00]. The basic idea is to compute the diagram in a series of parallel 2D slices.

So far we have described methods for linear sites. For a curved site such as a Bézier curve, the distance function is a high-degree algebraic function. We tessellate a Bézier curve into a piecewise linear approximation. The tessellation error is independent of the meshing error, so the two errors combine by addition. Bounded-error tessellation methods for parametric curves and surfaces can be found in [FG87] and [Kum96].

Our method also generalizes easily to *additively-weighted*, *multiplicatively-weighted* and *farthest-site* Voronoi diagrams. Each of these corresponds to a simple transformation of the distance mesh for a site. Note that scaling the distance function for a multiplicatively-weighted diagram also scales the meshing error.

4 Interactive Motion Planning

The computation of a generalized Voronoi diagram using graphics hardware provides us discrete information in two buffers: the depth buffer and the color buffer. Both are used for motion planning in a two-dimensional scene.

4.1 Use of Depth Buffer

The depth buffer gives the distance from each sample point to its nearest obstacle. Distance information is often used for proximity queries. Potential-field motion planners use a combination of an attractive force to the goal and a repulsive force away from the obstacles in order to plan the motion of the robot. The strength of the repulsive force is often a function of the distance to the nearest obstacle, and the direction of force is based on the gradient of the distance.

We begin by computing the discrete Voronoi diagram of the obstacles in the scene. We determine the repulsive force acting on the robot by examining the distance buffer. At a point on the robot, we interpolate a distance and compute a gradient by finite differences.

In order to compute the force acting on a robot, we sample the robot's geometry. We determine the repulsive force acting at each sample point on the robot. Following rigid-body dynamics, we decompose these forces into those acting on the center of mass and those applying torque. An alternative approach, useful in configuration space or for disk robots, is to apply the force on only the center of mass of the object, ignoring torque.

Hardware-accelerated Voronoi computation is especially useful for motion planning in dynamic environments where no *a priori* information is available about the motion of obstacles or multiple mobile robots. As obstacles move, the distance buffer is dynamically recomputed, and the repulsive forces on the robot change. In most cases, the robot will avoid the moving obstacles since the distance to each obstacle is dynamically updated and thus the robot will be pushed away from the obstacles. In this way, the fast computation of robot-to-obstacle distance using the hardware enables local motion planning through dynamic environments.

4.2 Use of Color Buffer

In the continuous domain, the Voronoi boundary represents the set of points that are equidistant from the two or more nearest obstacles and the Voronoi vertices are the points that are equidistant from three or more closest obstacles. A robot which moves along a Voronoi boundary follows the maximally clear path between two obstacles. The Voronoi vertices determine the branching points of these maximally clear paths, providing alternative paths to the goal. Our method finds approximate Voronoi boundaries by analyzing the rendered output in the color buffer.

The color buffer gives the ID of the nearest obstacle to each sample point. A magnified discrete Voronoi diagram is shown in Figure 1. The pixels are treated as squares tiling the plane. The squares' sides and corners form a regular 4-valent graph. An edge of this graph is said to be a member of the *discrete Voronoi boundary* if its two adjacent pixels are colored differently. A *discrete Voronoi vertex* is a node with three or four incident boundaries. The discrete Voronoi boundaries and vertices form the *discrete Voronoi graph*. This graph can be qualitatively different from the actual Voronoi diagram. For example, the discrete Voronoi graph in Figure 1 has a two-cycle, which cannot occur in the continuous Voronoi diagram.

From the color buffer, we compute the discrete Voronoi graph by using a continuation method. We begin by searching the outside edge of the color buffer for a pair of adjacent, different-colored pixels and then trace out the rest of the component by repeatedly examining adjacent pixels. Discrete Voronoi vertices are inserted into the graph as they are covered by the tracing algorithm. The edges of the graph are formed by boundary chains.

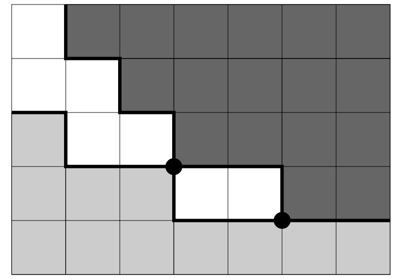

Figure 1. A portion of the discrete Voronoi graph of three sites.

The use of the color buffer places some additional restrictions on the representation of the obstacles. Large, non-convex obstacles may need to be broken up into smaller obstacles, so that the Voronoi diagram reveals pathways needed for planning. However, breaking a large obstacle into a great number of small obstacles is to be avoided when possible, as it makes the Voronoi diagram unnecessarily complex and cluttered, and increases the risk of resolution error.

As with the distance buffer, the fact that we can quickly recompute the color buffer allows us to plan through dynamic scenes. In this case, the update of the color buffer and the associated Voronoi graph can be computed at interactive rates. This helps to identify new potential paths in a dynamic environment.

4.3 Utilizing Both Buffers

The distance and color buffers can be used together in a motion planner. Here we describe two techniques that we have developed and implemented successfully.

The first algorithm plans motion along the discrete Voronoi boundary, computed from the color buffer. Along each arc of the boundary, the distance to the nearest obstacle is given in the distance buffer. When the discrete Voronoi graph is computed, each edge is stored along with its minimum and maximum clearances. These clearances are used to determine a weight for each edge. For instance, if each edge is weighted with the reciprocal of the minimum distance, then a shortest-path graph algorithm can find a maximally-clear path for the robot.

The second motion planning algorithm we present is designed for dynamic environments. The distance buffer is quite useful in local motion planning through a dynamic scene, but it needs a sequence of subgoals or "milestones."

We use the discrete Voronoi graph, obtained from the color buffer, to determine the milestones. At each time step, we compute the entire Voronoi graph, and weight the edges by a combination of boundary arc length and clearance. The nearest Voronoi vertex to the robot is chosen as the starting point. An optimal path is found in this graph from the starting point to the goal. The next Voronoi vertex in the optimal path is chosen as a milestone. A force is applied to the robot that attracts it toward the milestone. This force is combined with the other forces on the robot to determine its motion for that time step.

4.4 Sources of Error

Our techniques derive their efficiency from a uniform discretization of space. The discretization implies several different kinds of error, which we classify into *distance error* and *combinatorial error*.

Distance error is simply the error in the distance buffer. Such error arises primarily from two sources: meshing error, as from approximating a cone by a fan of triangles, and tessellation error, as from replacing a curved site by a polygonal approximation. Distance error is easily bounded, as discussed in Section 3.

Combinatorial error is qualitative rather than quantitative. For instance, a discrete Voronoi boundary is found between two sites that are not Voronoi neighbors, or two Voronoi vertices are merged into one. A discrete Voronoi vertex may be arbitrarily far from its corresponding vertex in the continuous domain. Combinatorial error is usually due to insufficient spatial sampling (as determined by display resolution). The error can be alleviated by local magnification.

5 System Implementation and Performance

To illustrate the application of our techniques, we have implemented a simple motion planner using our system for computing generalized Voronoi diagrams. We demonstrate its effectiveness on a complex environment—the interior of a house (figure 4.4)—composed of over 140,000 polygons. Initially we consider a static environment, but later allow dynamic obstacles. The robot has three degrees of freedom, x- and y-translation along the ground and rotation about the z-axis.

The approach we use is similar to the one outlined in Section 4.3. Each obstacle is assigned a unique identifying color. For our house example, each piece of furniture is modeled separately and gets its own color. Unfortunately, the walls of the house were modeled as a single object. If all walls are given the same color, then there will be no Voronoi boundary between opposite walls, so it was necessary to manually divide the house into several wall sections. Running our Voronoi algorithm on the 2D projection of these obstacles generates a color image (Figure 4.4a) in the frame buffer on which we run our boundary finding and graph building algorithm, as described in Section 4.2. Figure 4.4b gives a picture of the distance buffer generated. Figure 4.4c shows the graph connecting the Voronoi vertices, which were derived from the color buffer.

We find the nearest node in the Voronoi graph to the current position as well as the nearest node to the goal configuration, and perform a graph search over the Voronoi graph edges, finding the path of minimum weight. The weight is determined for each edge by a combination of two factors: the arc length (in the L^∞ "Manhattan" metric) of the Voronoi boundary between the nodes, and the inverse of the minimum clearance along that edge. We take the next node in the generated path to be our next milestone. In general, the path between each two milestones will be relatively straight and wide.

Planning the path to the next milestone is accomplished using the potential-field based approach. The repulsive force is calculated using the distance values obtained from the distance buffer. These forces and the resulting torques cause the robot to avoid the obstacles locally, possibly inducing rotation. The automated selection of milestones described earlier prevents many of the problems associated with local minima.

All computations, including the generation of the GVD, the building and searching of the graph, and the planning of the next step in the motion path, occur *interactively* (this entire cycle runs about 30 times per second). It is important that the computation be performed at interactive rates since the motion of the furniture in a dynamic scene can change the path, both locally and globally.

6 Summary and Future Work

We have described several techniques that exploit the fast computation of a generalized Voronoi diagram using graphics hardware for robot motion planning in complex static and dynamic environments. We have also demonstrated some promising preliminary results in a prototype implementation. Our algorithms and implementation presented in this paper is limited to a three-dimensional workspace for rigid robots. We recently designed some better sampling strategies based on Voronoi boundaries for randomized potential field planning or probabilistic roadmap methods [KLH98, KL94, KSLO96] and the details of this work can be found in [PHLM00]. We conjecture that this approach can be extended to flexible robots or articulated robots. There are several interesting research issues that we are planning to investigate next:

- Develop smart biasing techniques using weighted Voronoi diagrams to indicate "preferred" paths or directions, when planning using generalized Voronoi di-

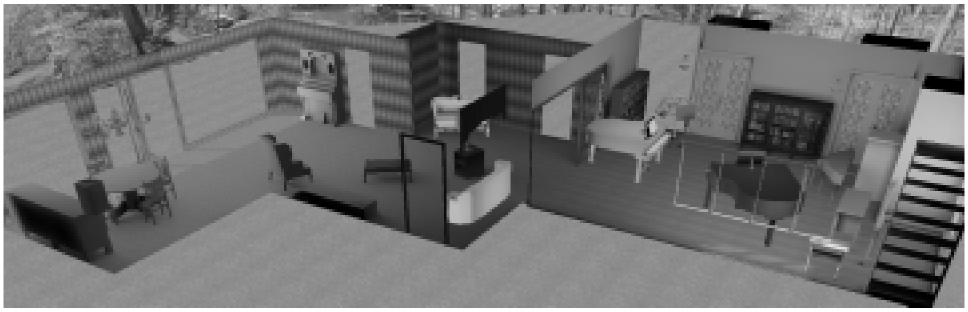

Figure 2. The house model, consisting of over 140,000 polygons.

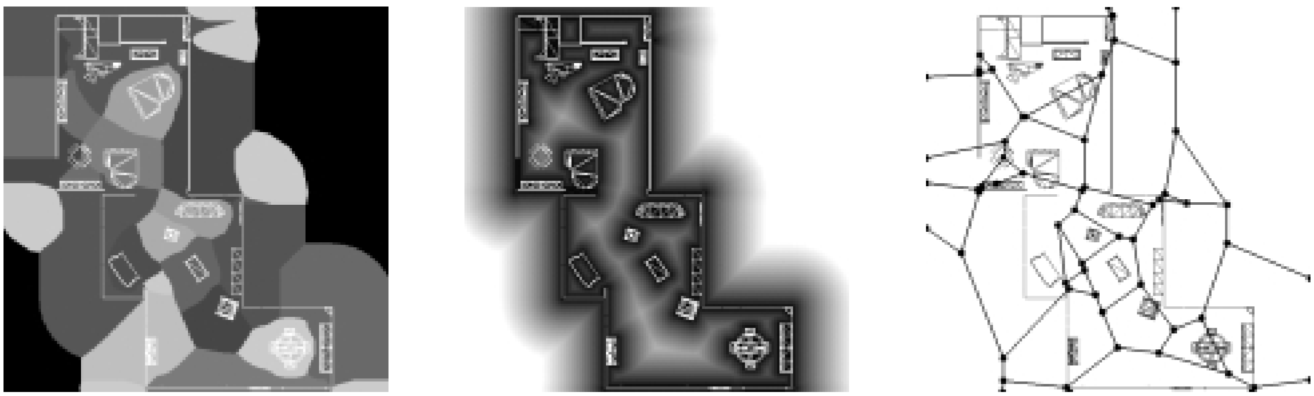

Figure 3. The color buffer, depth buffer, and Voronoi graph for the house model.

agrams.

- Investigate the use of approximate generalized Voronoi diagrams for articulated and deformable robots, as well as planning with (non-holonomic, visibility, etc.) constraints.
- Integrate the resulting motion planning algorithms with six-degree-of-freedom haptic rendering for maintainability studies.

References

[BL91] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *Int. J. Robotic Research*, 1991.

[CB94] H. Choset and J. Burdick. Sensor based planning and nonsmooth analysis. *IEEE Conference on Robotics and Automation*, 1994.

[CB95a] H. Choset and J. Burdick. Sensor based planning, part i: The generalized voronoi graph. *IEEE Conference on Robotics and Automation*, 1995.

[CB95b] H. Choset and J. Burdick. Sensor based planning, part ii: Incremental construction of the generalized voronoi graph. *IEEE Conference on Robotics and Automation*, 1995.

[CB96] H. Choset and J. Burdick. Sensor based planning: The hierarchical generalized voronoi graph. *Workshop on Algorithmic Foundations of Robotics*, 1996.

[CD87] J. Canny and B. R. Donald. Simplified Voronoi diagrams. In *Proc. 3rd Annu. ACM Sympos. Comput. Geom.*, pages 153–161, 1987.

[Cho97] H. Choset. Nonsmooth analysis, convex analysis and their applications to motion planning. *International Journal of Computational Geometry and Applications*, 1997.

[CKR97a] H. Choset, I. Konukseven, and A. Rizzi. Sensor based planning: A control law for generating the generalized voronoi graph. *IEEE Conference on Robotics and Automation*, 1997.

[CKR97b] H. Choset, I. Konukseven, and A. Rizzi. Sensor based planning: Using a honing strategy and local map method to implement the generalized voronoi graph. *SPIE Mobile Robotics*, 1997.

[CL90] J. F. Canny and M. C. Lin. An opportunistic global path planner. *Proceedings of International Conference on Robotics and Automation*, pages 1554–1559, 1990.

[CL93] J. F. Canny and M. C. Lin. An opportunistic global path planner. *Algorithmica*, 10:102–120, 1993.

[CL95] H. Chang and T. Li. Assembly maintainability study with motion planning. In *Proceedings of International Conference on Robotics and Automation*, 1995.

[CMB97] H. Choset, B. Mirtich, and J. Burdick. Sensor based planning for a planar rod robot: Incremental construction of the planar rod-hgvg. *IEEE Conference on Robotics and Automation*, 1997.

[FG87] D. Filip and R. Goldman. Conversion from bezier-rectangles to bezier-triangles. *CAD*, 19:25–27, 1987.

[FKL+97] P.W. Finn, L.E. Kavraki, J.C. Latombe, R. Motwani, C. Shelton, S. Venkatasubramanian, and A. Yao. Rapid: Randomized pharmacophore identification for drug design. *Proc. of 13th ACM Symp. on Computational Geometry (SoCG'97)*, 1997. A revised version of this paper also appeared in Computational Geometry: Theory and Applications, 10, pp. 263-272, 1998.

[GMTF89] J. Goldfeather, S. Molnar, G. Turk, and H. Fuchs. Near real-time csg rendering using tree normalization and geometric pruning. *IEEE Computer Graphics and Applications*, 9(3):20–28, 1989.

[HCK+99a] K. Hoff, T. Culver, J. Keyser, M. Lin, and D. Manocha. Fast computation of generalized voronoi diagrams using graphics hardware. *Proceedings of ACM SIGGRAPH 1999*, 1999.

[HCK+99b] K. Hoff, T. Culver, J. Keyser, M. Lin, and D. Manocha. Fast computation of generalized voronoi diagrams using graphics hardware. Technical Report TR99-011, Department of Computer Science, University of North Carolina, 1999.

[KC97] I. Konukseven and H. Choset. Mobile robot navigation: Implementing the gvg in the presence of sharp corners. *Proceedings of IROS*, 1997.

[KKKL94] Yoshihito Koga, Koichi Kondo, James Kuffner, and Jean-Claude Latombe. Planning motions with intentions. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 395–408. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[KL94] L. Kavraki and J. C. Latombe. Randomized preprocessing of configuration space for fast path planning. *IEEE Conference on Robotics and Automation*, pages 2138–2145, 1994.

[KLH98] L. Kavraki, F. Lamiraux, and C. Hollerman. Towards planning for elastic objects. *Proc. of 3rd Workshop on Algorithmic Foundations of Robotics*, 1998.

[KSLO96] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, pages 12(4):566–580, 1996.

[Kum96] S. Kumar. *Interactive Rendering of Parametric Spline Surfaces*. PhD thesis, Department of Computer Science, University of N. Carolina at Chapel Hill, 1996.

[Lat91] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[LRDG90] Jed Lengyel, Mark Reichert, Bruce R. Donald, and Donald P. Greenberg. Real-time robot motion planning using rasterizing computer graphics hardware. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 327–335, August 1990.

[ÓSY83] C. Ó'Dúnlaing, M. Sharir, and C. K. Yap. Retraction: A new approach to motion-planning. In *Proc. 15th Annu. ACM Sympos. Theory Comput.*, pages 207–220, 1983.

[PHLM00] C. Pisula, K. Hoff, M. Lin, and D. Manocha. Randomized path planning for a rigid body based on hardware accelerated voronoi sampling. In *Proc. of 4th International Workshop on Algorithmic Foundations of Robotics*, 2000.

[RMS92] J. Rossignac, A. Megahed, and B.D. Schneider. Interactive inspection of solids: cross-sections and interferences. In *Proceedings of ACM Siggraph*, pages 353–60, 1992.

[RR86] J. Rossignac and A. Requicha. Depth-buffering display techniques for constructive solid geometry. In *IEEE Computer Graphics and Applications*, pages 6(9):29–39, 1986.

[SAW99a] Peter F. Stiller Steven A. Wilmarth, Nancy M. Amato. Maprm: A probabilistic roadmap planner with sampling on the medial axis of the free space. *IEEE Conference on Robotics and Automation*, 1999.

[SAW99b] Peter F. Stiller Steven A. Wilmarth, Nancy M. Amato. Motion planning for a rigid body using random networks on the medial axis of the free space. *Proc. of the 15th Annual ACM Symposium on Computational Geometry (SoCG'99)*, 1999.

[STK+94] A. Schweikard, R. Tombropoulos, L.E Kavraki, J.R. Adler, and J.C. Latombe. Treatment planning for a radiosurgical system with general kinematics. *IEEE Conference on Robotics and Automation*, 1994.

[TAL99] R. Tombropoulos, J.R. Adler, and J.C. Latombe. Carabeamer: A treatment planner for a robotic radiosurgical system with general kinematics. *Medical Image Analysis*, pages 3(3):1–28, 1999.