Randomized Path Planning for a Rigid Body Based on Hardware Accelerated Voronoi Sampling[*]

Charles Pisula, Kenneth Hoff III, Ming C. Lin, Dinesh Manocha
Department of Computer Science
University of North Carolina, Chapel Hill, NC, USA
{pisula,hoff,lin,dm}cs.unc.edu
http://www.cs.unc.edu/~geom/Planner

Abstract

Probabilistic roadmap methods have recently received considerable attention as a practical approach for motion planning in complex environments. These algorithms sample a number of configurations in the free space and build a roadmap. Their performance varies as a function of the sampling strategies and relative configurations of the obstacles. To improve the performance when the path of a robot has to pass through narrow passages, some researchers have proposed algorithms for sampling along or near the medial axis of the free space. However, their usage has been limited because of the practical complexity of computing the medial axis or the cost of computing such samples.

In this paper, we present efficient algorithms for sampling near the medial axis and building roadmap graphs for a free-flying rigid body. We use a recent algorithm for fast computation of discrete generalized Voronoi diagrams using graphics hardware [HCK$^+$99a]. We initially compute a bounded error discretized Voronoi diagram of the obstacles in the workspace and use it to generate samples in the free space. We use multi-level connection strategies and local planning algorithms to generate roadmap graphs. We also utilize the distance information provided by our Voronoi algorithm for fast proximity queries and sampling the configurations. The resulting planner has been applied to a number of free flying rigid bodies in 2D (with 3-dof) and 3D (with 6-dof) and compared with the performance of earlier planners using a uniform sampling of the configuration space. Its performance varies with different environments and we obtain 25% to over 1000% speed-up.

1 Introduction

Motion planning is one of the important, classical problems in algorithmic robotics [Lat91]. Besides robotics, it has applications in many areas, including animation of digital actors or autonomous agents [KKKL94], maintainability studies [CL95], drug design [FKL$^+$97] and robot-assisted medical surgery [STK$^+$94, TAL99].

This problem has been extensively studied over the last three decades. A number of analytical methods and approximate techniques have been proposed. However, due to the computational complexity of complete motion planning algorithms, most practical algorithms are based on randomized motion planning algorithms, such as randomized potential field methods (RPP) or probablistic roadmap methods (PRMs).

The simplest PRM algorithms generate a set of configuration in the free space. The planning algorithm involves three main steps [WAS99a, KL94, KSLO96, OŠ95]:

[*]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.

1. Generate a list of samples in the configuration space. The simplest algorithms use uniform sampling techniques.
2. Use collision detection or distance computation algorithms to select the samples lying in the free space.
3. Try connecting the samples in the free space by local planning methods and build a roadmap graph.

The roadmap graph is used to generate a path between the start and goal configurations. However, the efficiency of these planners can degrade in configurations containing narrow passages or cluttered environments. In such cases, a significant fraction of randomly generated configurations are not in the free space. Moreover, it is hard to generate a sufficient number of samples in the narrow passages or connect all the nodes in the free space using local planning methods. Many approaches have been proposed in the literature to overcome these problems. These include:

Dilating the Free Space[HKL$^+$98]: The main idea is to dilate the free space by allowing the rigid body to penetrate the obstacles by a small amount. The areas near these nodes are resampled to find collision free configurations. However, dilation can alter the topology of the free space. Furthermore, no good practical (in terms of efficiency and robustness) algorithms are known for penetration depth computation between polyhedral models. Some of the public domain implementations like I-COLLIDE and V-Clip for convex polytopes, based on Lin-Canny distance computation algorithm, provide only an approximation to the penetration depth.

Sampling near the Obstacle Boundaries[ABD$^+$98]: It samples the nodes from the contact space, the configurations where the robot just touches one of the obstacles. It works well in many cases, but its performance is difficult to analyze.

Information about the Environment[HST94, OŠ95]: These algorithms make use of information known about the environment. These include executing random reflections at the C-obstacle surfaces [HST94] and adding "geometric" nodes for non-articulated robots near the boundaries of the obstacles in the workspace [OŠ95].

Sampling Based on the Medial Axis [WAS99b, WAS99a, GHK99]: The main idea is to generate nodes that lie on the medial axis of the workspace or the free space. Intuitively speaking, the medial axis corresponds to a set of points that are furthest from the obstacle boundaries and have maximum clearance. It has been used for a number of motion planning algorithms [ÓSY83, CD87, Lat91, CKR97, Cho97]. However, its practical use has been limited because of the difficulties in accurately computing the medial axis or generalized Voronoi diagrams. Wilmart et al. [WAS99b, WAS99a] generate random configurations and retract them onto the medial axis without explicitly computing the medial axis. They use a search algorithm based on distance to the obstacles for the retraction. They have applied the resulting algorithm to a few configurations with narrow passages and obtained considerable speedups over uniform sampling. Guibas et al. [GHK99] use point approximations on the boundary, compute their Voronoi diagrams to approximate the medial axis and use it for motion planning of flexible objects.

1.1 Our Results

In this paper, we present improved algorithms for sampling based on medial axis and use them for efficiently constructing the Voronoi roadmap. We make use of bounded error discretized Voronoi diagrams, computed using graphics rasterization hardware [HCK+99a]. More specifically, we render distance functions for each primitive of the obstacle and use the frame buffer output to identify the Voronoi boundaries upto a given resolution. Furthermore, the depth buffer provides the distance information which is used to speed up the proximity queries and sampling the configuration space. This computational framework enables insightful analysis of the workspace to identify different types of narrow passages and efficient generation of sampling points with guaranteed distribution on the medial axis. We use multi-stage connection strategies along with local planning algorithms to build the roadmap graphs. The resulting PRM has been applied to a number of complex environments composed of 3-dof (in 2D) and 6-dof rigid bodies (in 3D). As compared to uniform sampling of the configuration space, it can considerably improve the performance of the planner. Our initial experiments demonstrate 25% to over 1000% speed-up in running times. Since the cost of generating the samples is relatively small, the planner never underperforms as compared to uniform sampling.

Some of the main advantages of our approach include:

- **Efficiency**: The resulting algorithms for generating the bounded error approximation of the Voronoi diagram and samples on the medial axis run relatively fast. For environments composed of thousands of polygons, we can generate Voronoi diagrams at $128 \times 128 \times 128$ resolution at in a few seconds or minutes (depending on the size of the environment) on a SGI workstation. Furthermore, the distance buffer helps us speed up the collision queries by a factor of two. Based on our sampling strategies, a high fraction of the nodes generated are in the free space.

- **Simplicity**: The resulting algorithm is relatively simple to implement and doesn't suffer from robustness or degeneracies. The basic PRM planner is a rather simple approach and our sampling algorithm doesn't introduce any complications.

- **Identification and Characterization of Narrow Passages**: The distance buffer information gives us information about narrow passages in the workspace. Using a combination of distance metric for dimension analysis and retraction methods, it can also be used to identify narrow passages in many cases.

- **Global connectivity information based on Voronoi Roadmaps**: The connectivity information provided by the discretized generalized Voronoi diagram is used in constructing the roadmap and accelerating performance of the local planner.

The rest of the paper is organized as follows: In Section 2, we highlight our notation, provide a brief overview of the medial axis and the fast approximate to compute a discretized approximation using graphics hardware. We describe the planner in Section 3 and present the multi-stage connection strategies to build the roadmap. Section 4 presents our sampling strategies based on medial axis of the workspace. Section 5 discusses various implementation issues, including collision culling, computations of Voronoi vertices and graphs, and other data structures. We also highlight the performance of our planner on different benchmarks. Finally we compare it with related approaches in Section 6.

2 Background

In this section, we highlight our notation and provide a quick overview of medial axis and fast computation of discretized Voronoi diagrams using graphics hardware.

2.1 Notation and Representation

In this paper, we restrict ourselves to dealing with non-articulated rigid bodies in 2D or 3D. We use the symbol W to represent the workspace. We will denote the robot as R and the set of obstacles as O. We assume that the robot and each obstacle is a closed and bounded set. Our current implementation assumes that each obstacle is represented as a collection of triangles. It is relatively simple to extend them to other primitives (e.g. spline models) based on our current algorithmic framework. The configuration space (denoted by C) of R is given by the set of all positions and orientations. We use (x, y, z) coordinates to represent the position and a unit quaternion to represent the orientation. The (x, y, z) coordinates will be referred to as the *translation component* of a configuration and the quaternion is the *rotation component* of a configuration. Furthermore, C can be partitioned into *free space,* F and *blocked space,* B. The blocked space corresponds to configurations, where the robot R collides with at least one of the obstacle. The rest of the configuration space corresponds to the free space. In other words, $F = C \backslash B$. Furthermore, we denote the boundary of the free space as ∂F and ∂O represents the union of boundaries of all the obstacles.

2.2 Medial Axis and Voronoi Diagrams

The medial axis of a solid object provides shape analysis in terms of its boundary elements. It is a skeletal representation that can be formulated as the locus of the center of a maximal sphere as it rolls around the object interior. It is closely related to the Voronoi diagram of a solid and for a suitably defined boundary, it can be computed from the Voronoi diagram and vice versa. The concept of medial axis was first proposed by Blum [Blu67] for biological shape measurement and since been used for mesh generation, feature recognition and molding simulation. Medial axis and Voronoi diagrams have been long used for robot motion planning [ÓSY83, CD87, Lat91, CKR97]. The medial axis of the free space F, denoted by MA(F) has lower dimension than F but is still a complete representation for planning the motion. Strictly speaking, MA(F) is a strong deformation retract of F, implying that F can be continuously deformed into MA(F) and maintain its topology structure [ÓSY83, CD87, CKR97, WAS99b].

Algorithms to compute Voronoi diagrams and medial axis have been extensively studied in computational geometry and solid modeling. Good theoretical and practical algorithms are known for point primitives. However, the boundaries of the Voronoi regions for higher order primitives (e.g. lines, triangles) correspond to high-degree algebraic curves and surfaces. No good practical algorithms are known for computing them efficiently and robustly. As a result, their applications have been limited.

Given the practical complexity of computing Voronoi diagrams, many researchers have proposed approximate approaches. Some common approximations include generating point approximations of the boundaries and computing their Voronoi diagrams [SAR95]. However, it is hard to give any guarantees on the accuracy of the resulting Voronoi diagram. Other approaches use spatial subdivision techniques [VO97, LBD$^+$92]. While these algorithms can be used to generate bounded error approximations of Voronoi diagrams, they can be rather slow for large environments.

2.3 Bounded Error Approximation of Voronoi Diagrams using Graphics Hardware

We compute a bounded error approximation of the Voronoi diagram of the obstacles in the workspace, W. If any of the obstacles is a closed and bounded solid, we do not compute the Voronoi diagram inside that solid. Each boundary triangle, edge and vertex of an obstacle is treated as a separate site. We compute a discrete generalized Voronoi diagram by rendering a three-dimensional distance mesh for each site. 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 color, and the corresponding distance mesh is rendered in that color using 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. The error in the mesh is bounded to be smaller than the distance between two pixels, in order to maintain an accurate Voronoi diagram. More details are given in [HCK$^+$99a]. This algorithm runs very fast in practice.

2.3.1 Motion Planning Using Discretized Voronoi Diagrams

Many motion planning algorithms have been proposed based on discretized Voronoi diagrams [VO97, HCK$^+$99a, HCK$^+$99b]. Based on the color and distance buffer information, [HCK$^+$99a, HCK$^+$99b] define a potential field and use it to navigate the robot. Since the distance buffer information is computed at interactive rates, this planner has also been applied to dynamic 2D environments (3-dof robots).

3 Path Planning Based on Discretized Voronoi Diagrams

In this section, we describe our path planning algorithm. Initially we present algorithms for computing the discretized Voronoi diagrams, followed by use of boundary finding algorithms that extract the Voronoi graph. We use a multi-stage strategy to build the roadmap. It generates configurations in the free space and connect them using local planning algorithms. The first stage selects nodes using the medial axis of the workspace and connects them using simple local planning algorithms. Next it attempts to connect different components. Finally, it tries to estimate narrow passages based on local characteristics of the Voronoi diagram and generates more samples in those regions. More details on improved sampling algorithms are given in Section 4.

3.1 Discretized Voronoi Diagram of the Workspace

Given a bound on the discretization error, the algorithm computes the Voronoi diagram of the obstacles O slice-by-slice (along the z-axis). It renders the distance function for each vertex, edge and triangular face of each obstacle. Each slice is generated using the graphics rasterization hardware. We read back the color buffer and the depth-buffer for each slice. The color buffer gives the index of the nearest obstacle to each sample point in the slice. The distance buffer gives the distance to that obstacle. It generates a 3D voxel grid that corresponds to a uniform sampling of the space containing the geometric primitives. This 3D image gives a volumetric representation of the generalized Voronoi diagram of the primitives. The resolution of the Voronoi graph can have significant impact on the performance of the planner (as highlighted in Section 5). In our current implementation, we take uniform samples along the z-axis. In theory, it is possible to vary the step size adaptively using bisection. For PRM, we are only interested in computing an accurate approximation of the Voronoi boundaries.

3.2 Extracting the Voronoi Graph

The Voronoi diagram is in the form of two 3D images: a color image corresponding to the IDs of the closest site to each sample point and a depth image giving the related distances to the closest sites. Since we are using a volumetric representation, the actual continuous boundaries of the Voronoi graph are described implicitly as lying between sample points of different colors (sample points that have different closest primitives).

Our goal is to extract the boundary graph structure in order to bias the randomized motion planning sampling. In 3D workspace, the Voronoi graph structure is composed of Voronoi vertices, edges, and faces. However, for our applications, we only need the vertices and the edges. This will give us a graph structure forming maximally clear paths from the obstacles in the workspace. Voronoi vertices are the set of points equidistant among four or more primitives, and Voronoi edges are the set of points equidistant among three primitives. To extract these features we use continuation methods, which are very similar to common iso-surface extraction techniques commonly used in volume rendering. Our goal is to continuously "bracket" the boundary curves in a $2 \times 2 \times 2$ region of sample points and then walk along the boundary one voxel at a time. We only step to the next $2 \times 2 \times 2$ region if it is part of the same boundary. In this manner, we only touch 3D sample points that are close to the boundary. Since we are effectively growing the entire boundary from some starting "seed" point, we are able to form a correctly connected graph structure easily and efficiently. The seed point is found along the boundaries of the voxel grid since the boundary graph must typically intersect the bounding volume of the workspace.

3.3 Multi-Stage Roadmap Construction

Given the Voronoi graph, we present a multi-stage algorithm to build the roadmaps. The goal is to initially generate portions of the roadmap using the medial axis and simple local planning algorithms. Next we connect different roadmaps generated using more sophisticated techniques. Finally, we make use of the Voronoi graph and the distance buffer to estimate narrow passages and generate additional samples along these narrow passages and try to connect them with the other sampled nodes. Our overall strategy in using a multi-stage algorithm is similar to that highlighted in [ABD$^+$98]. However, we choose samples based on the medial axis and estimate narrow passages, as opposed to generating samples in the contact space. Our roadmap construction algorithm proceeds in three stages.

1. **Preprocessing:** As part of a pre-process, we generate a number of nodes using points near the medial axis of the workspace. We assign configurations where the position of the robot corresponds to a point lying on or near the medial axis. The orientation is assigned either randomly or based on the local characteristics of the medial axis (see Section 4). Since the points on the medial axis have the maximal clearance from the obstacles, these nodes will bias the robot to plan a path near the medial axis. We use an adaptive strategy to select the nodes. Initially, we select nodes corresponding to the vertex locations in the Voronoi graph. We use a local planning algorithm to check whether we can generate an edge of the roadmap between those nodes. If not, we select a midpoint location along the Voronoi edge and repeat the process for each Voronoi edge in the roadmap graph. This process is repeated until both nodes are connected. At the end of this phase, the roadmap may consist of one or more connected components and its topology is similar to that of the Voronoi graph. The local planning algorithm is similar to *rotate-at-s* algorithm highlighted in [ABD$^+$98].

2. **Connecting Multiple Connected Components:** If the roadmap has more than one connected component, we try to connect them. This requires trying different set of nodes between each component and trying to find a path using the local planner. Or we generate more nodes that retain the same positions close to the medial axis of the workspace, but with different orientations for the robot. Then, we try to connect these nodes to different components of the roadmap.

3. **Sampling along Narrow Passages:** We estimate narrow passages based on the medial axis and the distance buffer. More details are given in Section 4. We generate nodes near these narrow passages by using a combination of uniform sampling, Gaussian sampling and biased angle sampling to connect them with the roadmap.

4 Sampling Strategies Based on Voronoi Graphs

In this section, we present strategies to estimate narrow passages and improved sampling algorithms based on the medial axis of the workspace.

4.1 Estimation of Narrow Passages

Our algorithm computes a discretized Voronoi diagram of the workspace. With each voxel, we associate a color that corresponds to an ID of the closest obstacle and a distance value, which stores the distance to that obstacle. By comparing this value against the dimensions of the robot, we can estimate the narrow passage in many cases. Given a robot R, let r_{out} be the radius of minimum enclosing sphere and r_{in} be the radius of the maximum inscribed sphere. If we are given a subset of voxels on the Voronoi boundary, whose distance value is less than r_{out}, then all the configurations whose corresponding positions in workspace (translation components) are close the location of these voxels may contain "narrow passages". In such cases, it may be difficult to plan the motions of the robot through these narrow passages, even when only *translational displacements* are required to navigate through the narrow corridors. Furthermore, if any collection of voxels on the Voronoi boundary have a distance value less than r_{in}, then all configurations whose translation components (or the corresponding positions in workspace) are close to the location cannot be contained in the free space.

Furthermore, the degrees and types of "tightness" in a cluttered environment are determined by the dimensions of the robot with respect to the dimensions of narrow corridor in the workspace (defined based on some type of distance metric). Normally, the medial representation computed is only the generalized Voronoi diagram or medial axis of the "workspace". Ideally we wish to compute the medial axis of the free space and identify the narrow passages more precisely by considering the rotational components of robot configurations. Unfortunately, no fast and practical algorithm is known for computing the configuration space.

Our algorithm computes an approximate medial axis of the workspace. Therefore, the scheme we mention to estimate the location of "narrow passages" is only an approximation technique. In the rest of this section, we describe some techniques that perform dimension analysis by comparing the dimensions of the robot with the distance values of the voxels that represent the discretized medial axis of the workspace.

Let us assume that the tightest-fitting arbitrarily oriented bounding box (OBB) is given for the robot. Moreover, the box's local coordinate frame and its dimensions with respect to the global coordinate frame are known. Algorithms based on covariance matrix to approximate the "principle component directions" of a given collection of polygons can be used to compute a tight fitting OBB [GLM96]. Let us further assume that the exact dimensions of the "potential" narrow corridors are given. Based on this information, we define, the following types of narrow passages:

Type-T Narrow Passages: Robot can move through narrow passages by translational motion. Rotations are *not* required for the robot to navigate through the narrow corridors. This occurs when all the dimensions of the robot's OBB are equal or slightly smaller than the dimensions of the narrow corridors. Images 1-3 are examples of such narrow passages where merely careful translational movements are sufficient to navigate the robot through the narrow corridors.

Type-R Narrow Passages: Rotations are required for the robot to navigate through the corridors. This occurs often when the largest dimension of the robot's OBB exceeds those of the narrow corridors. Images 3 and 6 highlight examples of two narrow passages that require rotations to plan the path for the robot and the piano.

4.2 Random Sampling Near the Medial Axis

Once we can identify different types of narrow passages in the configuration space, we can design better sampling strategies for handling these scenarios.

In general, if the robot encounters a type-T narrow passage, the robot can easily move through the narrow corridors by placing the geometric center of the robot's OBB along the edges of the Voronoi graph. In such cases taking samples on the medial axis (the first step in our multi-level strategy) has considerable benefits.

For handling type-R narrow passages, sampling *along* the Voronoi graph with random sampling of angles can lead to poorer performance. Instead, we propose to sample *near* the Voronoi graph using a *combination* of the following strategies:

- **Simple Uniform Sampling:** Place more sampling points uniformly inside the narrow passages, once they have been identified. This technique basically increases the density of sampled nodes inside the narrow passages everywhere.
- **Gaussian distribution:** Sample near the medial axis with a Gaussian distribution to randomly position the points with higher distribution density *near* the medial axis while uniformly sampling in the angular space.
- **Angular Bias:** Biasing the angular sampling by positioning the local coordinate axis along which the robot's OBB has its smallest dimension so that the axis is perpendicular to the tangent direction of the Voronoi graph with Gaussian distribution. Intuitively, this sampling strategy attempts to orient the robot to intelligently adapt to the change in curvatures of the Voronoi graph, as it moves through the narrow passages.

We have experimented with a combination of these strategies on several different benchmark environments and have observed good performance improvement over uniform sampling. More details of our implementation results are given in the next section.

5 Implementation and Performance

We describe the implementation of our planner and its application to different benchmarks.

5.1 Implementation

We have implemented a preliminary version of the planning algorithm for 2D (3-dof) and 3D (6-dof) environments on an SGI IRIX workstation. The planner proceeds in distinct stages: pre-processing, roadmap construction, and path planning query.

In the pre-processing phase, the planner computes information about the environment. First, it computes the discretized Voronoi diagram. With each voxel of the 3D image, we store information about the closest obstacle and the distance computed using the depth buffer. The performance of the Voronoi algorithm on different scenarios has been highlighted in Table 1. It is a linear function of the number of voxels in the grid and increases as we subdivide a voxel into 8 sub-voxels. We expect this performance can be improved by using adaptive grid size selection. We are currently investigating the implementation issues related to adaptive grid size.

We perform collision and distance queries between the robot and the obstacles using PQP [LGLM99]. PQP uses a hierarchy of swept sphere volumes and works efficiently on general polygonal models. A typical planner spends a high percentage of its time in performing collision and distance queries. To speed up these queries, we enclose the robot R with a bounding sphere. Let its radius be r_{out}. We compare the radius with the distance value associated with the voxel that contains the

Model Size (polygons)	16	32	64	128	256	512
46 (Benchmark 5)	.02	.05	.12	.48	3.22	25.33
2180 (Benchmark 6)	.16	.44	1.74	9.82	68.01	557.36
10900	.77	2.09	8.49	48.47	336.83	2897.43

Table 1: *The cost of computing the discretized Voronoi diagram in seconds as a function of the resolution. The table lists the number of triangles against Voronoi diagram resolution. All timings are computed on a SGI Infinite Reality (Onyx2). Each represents a different grid size (*$16 \times 16 \times 16$ *to* $512 \times 512 \times 512$*).*

position of the center of the enclosing sphere. If r_{out} is less than the distance value, it implies that the given configuration lies in the free space and we don't have to perform explicit collision check between the robot and all the obstacles. Otherwise, we use PQP routines for exact collision detection. We refer to this acceleration technique as *QR collision check* for quick rejections. Given the color buffer, we compute the boundary graph of the Voronoi diagram using a marching technique. We next compute an approximation to the medial axis, and identify regions to be considered as narrow passages in the workspace (as described in Section 4).

In the roadmap construction phase, the planner attempts to build a network of configuration nodes that can be used for the path planning queries. We use the multi-stage algorithm described in Section 3. We represent the roadmap as a graph with multiple connected components. Given an initial and goal configuration, we combine the roadmap and query phase into one step so that each query adds new connectivity information to the roadmap. The planner builds the roadmap by iteratively growing from initial configuration C_i and the goal configuration C_g. We use a growth strategy similar to the one highlighted in [KSLO96, HKL+98]. To grow a component c of the roadmap, the algorithm randomly selects a node N_e in c to expand. N_e is chosen from the set of previously generated samples in c, giving priority to those near the medial axis, and the ones in the low density areas. The main goal is to bias the planner towards the nodes whose translational component lies close to the medial axis and to select new configurations in free space.

To expand a node, we generate a number of random configurations N_r in the neighborhood of N_e using a Gaussian distribution. The size of the neighborhood is defined by the distance to the nearest obstacle and is computed using the distance information associated with each voxel. After that we check if N_e is in the free space by performing a collision query. We try to insert free configurations into the component c by checking for a valid path from N_e to N_e using the local planner. Finally, after each component has completed its expansion stage, an attempt is made to connect the unconnected components using local planning algorithms. This process is repeated until a path has been found between the initial and the goal configuration.

5.2 Benchmark Environments

We have tested the performance of the planner on several 2D and 3D benchmarks. We also compared its performance with Stanford PRM developed by David Hsu, J. Latombe et al. [HKL+98]. It uses uniform sampling in the configuration space to generate the configurations.

We have chosen a suite of environments and scenarios to test the effectiveness of our sampling scheme over basic uniform sampling. Most of them have either Type-T narrow passage or Type-R narrow passage. The set of benchmarks used include:

- **Benchmark 1:** A 2D environment requiring the robot to navigate a narrow passage to move from the open area on the left to the open area on the right (Image 1(a-b)). The environment consists of two wide open areas connected by one narrow channel.

  - Scenario (a): Type-T narrow passage, robot dimension of 2.5×14.0.
  - Scenario (b): Type-R narrow passage, robot dimension of 4.0×40.0.
  - Size of environment : 400×400
  - Width of narrow passage: 20
  - Number of line segments : 158

- **Benchmark 2:** A 2D environment requiring the robot to traverse a long Type-T narrow passage resembling a maze (Image 2).
  - Size of environment : 400×400
  - Width of narrow passage: 15
  - Number of triangles : 1044
  - Robot dimensions : 10×10

- **Benchmark 3:** A complex 2D environment consisting of chairs, pianos and a music stand, each projected into the XY plane (Image 3). The robot (a music stand) must navigate a Type-T narrow passage.
  - Size of environment : 15.5×17.5
  - Number of triangles : 12054
  - Robot dimensions : 1.0×1.0

- **Benchmark 4:** A 3D environment with two open areas connected by a single channel (Image 4(a-c)). The robot (a block) must move from the initial configuration to the goal by traversing a narrow tunnel.
  - Scenario (a): Type-T narrow passage, robot dimension of $.025 \times .025 \times .025$.
  - Scenario (b): Type-T narrow passage, robot dimension of $.08 \times .08 \times .08$.
  - Scenario (c): Type-R narrow passage, robot dimension of $.025 \times .025 \times .125$.
  - Size of environment: $1.0 \times 1.0 \times 1.0$
  - Width of narrow passage: $0.1 \times 0.1 \times 0.1$
  - Number of triangles: 28

- **Benchmark 5:** A 3D environment similar to benchmark 4, except that the passage is not a simple straight channel. The channel "spirals" through space. The tunnel itself provides a narrow passage in the workspace. The sharp corners further complicate planning by causing a Type-R narrow passage (Image 5).
  - Scenario (a): Type-R narrow passage, robot dimension of $.02 \times .02 \times .125$.
  - Scenario (b): Type-R narrow passage, robot dimension of $.07 \times .07 \times .07$.
  - Size of environment : $1.0 \times 1.0 \times 1.0$
  - Width of narrow passage: $0.1 \times 0.1 \times 0.1$
  - Number of triangles : 46

- **Benchmark 6:** A 3D environment consisting of chairs, a table, and a piano (Image 6). The goal is to move the piano through the window. The environment provides Type-R narrow passage, as the piano must rotate to fit through the window. This benchmark has been provided to us by Jean-Paul Laumond at LAAS, Toulouse.
  - Size of environment : $6000 \times 6000 \times 3000$
  - Dimension of narrow passage: 2000×1000 (W $\times$ H)
  - Robot dimension (with legs): $1000 \times 1210 \times 850$
  - Robot dimension (sans legs): $1000 \times 1200 \times 350$

5.3 Performance Data

We highlight the performance of the planner on different benchmarks. The different columns in the table used are:

- Sampling Type - We have compared our Voronoi Based sampling to Stanford's planner that uses Uniform sampling [HKL+98]. The basic structure of the implementations are the same. The main difference is in the sampling strategy.
- Time - The running time to service the benchmark query. The time to compute the Vornoi diagram is not included, but is listed in a separate table. But it includes the time to build the roadmap.
- C-Space Sample - The total number of random C-Space configuration generated by the planner. This number represents all configurations, even those in contact space, and those that are in free space that fail to connect to a node in the road map.
- Free Samples - The total number of samples which lie in free space and successfully become part of the roadmap.
- Full Coll. Checks - The number of times a full PQP collision query was performed.
- QR Coll. - The number of times a full collision query was avoided, by doing a quick rejection.
- Connect Comp. Calls - The number of times the planner attempted to connect two nodes from distinct components.

Benchmark	Sampling Type	Time	C-Space Samples	Free Samples	Full Coll. Checks	QR Coll.	Connect Comp. Calls
1(a)	Uniform	36.10	5062	3414	21740	0	2655585
1(a)	Voronoi Based	1.50	1239	1213	3202	9864	36447
1(b)	Uniform	80.22	14843	8636	20756	0	2360384
1(b)	Voronoi Based	20.47	8297	3365	8753	6201	201654

Table 2: *Benchmark 1, environment shown in Image 1. The size of the robot varies so that in (a) No rotation is required. (b) Rotation is required.*

Benchmark	Sampling Type	Time	C-Space Samples	Free Samples	Full Coll. Checks	QR Coll.	Connect Comp. Calls
2	Uniform	625.67	30692	3479	267045	0	2423896
2	Voronoi Based	213.89	10018	1393	86902	78251	308528

Table 3: *Benchmark 2, maze environment shown in Image 2. The narrow passage is long with respect to the robot dimensions.*

Benchmark	Sampling Type	Time	C-Space Samples	Free Samples	Full Coll. Checks	QR Coll.	Connect Comp. Calls
3	Uniform	540.44**	36631	17812	101087	0	104032
3	Voronoi Based	114.5	23680	8739	33646	24907	54986

Table 4: *Benchmark 3, house environment shown in Image 3. Notice that the Uniform based sampling does not even complete in the allocated time.*

6 Analysis and Comparison

Overall the performance of our planning algorithm varies with the environment. In all our current benchmarks, it significantly performs uniform sampling in Type-T narrow passages. It also improves the performance of the planner when there are Type-R narrow passages in the configuration space. In all our benchmark comparisons, we try to make minimal changes to the connection strategies. Based on the medial axis, we have also proposed a novel multi-stage strategy to build the roadmap. It leads to considerable improvement in the performance.

The idea of using the medial axis for sampling is not novel. Other authors have proposed using medial axis based sampling [WAS99b, WAS99a, GHK99]. The major benefit in our approach comes from the fact that we have bounded error approximation of the medial axis computed using graphics hardware. Wilmarth et al. [WAS99b, WAS99a] do not compute a medial axis. They take random configurations and retract them to the medial using iterative approaches. However, they either cannot guarantee sufficient number of samples in all the narrow passages, or it will take many more random configurations (followed by retraction) to generate sufficient samples in some of the challenging scenarios.

Guibas et al. [GHK99] take point samples on the boundary and compute their Voronoi diagram to estimate samples on the medial axis of the workspace. Again, in this case, it is difficult to guarantee any bounds on the medial axis or quality of the resulting samples.

Overall, having a bounded error approximation of the medial axis of the workspace, along with the distance information computed using the depth buffer, helps us in identifying the narrow passages and characterizing them in many cases. Clearly, there is no one universal sampling strategy that will work well in all cases. The ability to distinguish between different type of narrow passages enables us to intelligently choose a combination of sampling strategies to adapt to different regions of the environment.

7 Conclusion

We have presented techniques to improve the performance of PRM in configurations with narrow passages. We use a bounded error approximation of the generalized Voronoi diagram to improve the sampling and estimate narrow passages in the free space. We have applied it to a number of benchmarks and the preliminary results are promising.

Next, we would like to investigate the theoretical analysis of “Voronoi skeleton” or medial axis for Type-R narrow passages and develop more rigorous computational techniques using the graphics hardware. We would like to further extend our approach to motion planning with constraints and planning of articulated objects or manipulators. Other interesting directions include generalizing

Benchmark	Sampling Type	Total Time	C-Space Samples	Free Samples	Full Coll. Checks	QR Coll.	Connect Comp. Calls
4(a)	Uniform	8.28	4669	2714	10148	0	375736
4(a)	Voronoi Based	3.48	471	378	3098	2908	12685
4(b)	Uniform	7892.55	126189	15079	212859	0	55216398
4(b)	Voronoi Based	721.28	35328	2707	33344	21088	1803603
4(c)	Uniform	14179.06**	313014	54593	488821	0	57429630
4(c)	Voronoi Based	2956.83	93723	12718	93528	67087	4634587

Table 5: *Benchmark 4, a simple tunnel shown in Image 4. The size of the robot varies so that in (a) The robot is small, and no rotation is required. (b) Rotation is required. (c) The robot is large, but no rotation is required. Notice that in (c) the Uniform based sampling does not even complete in the allocated time.*

Benchmark	Sampling Type	Total Time	C-Space Samples	Free Samples	Full Coll. Checks	QR Coll.	Connect Comp. Calls
5(a)	Uniform	296.65	61675	10985	98704	0	2028713
5(a)	Voronoi Based	160.99	22021	6087	52956	32091	715593
5(b)	Uniform	19223.3	733663	13335	845172	0	14035450
5(b)	Voronoi Based	15434.1	543401	9874	618233	152987	10329874
5(c)	Uniform	296.65	61675	10985	98704	0	2028713
5(c)	Voronoi Based	12.45	5672	472	2732	982	Not Applicable

Table 6: *Benchmark 5, spiral tunnel shown in Image 5. In (a) and (b) the robot size varies. In (c) we compute the path with our boundary graph approach. (a) No rotation is required. (b) Rotation is required. (c) The same scenario as in (a) with the different connection scheme that takes advantage of medial representations. We observe substantial performance improvement by using a better connection strategy that exploits the Voronoi graph structures.*

this approach to planning of flexible bodies in 3D workspace [GHK99], designing smarter sampling and connection strategies, as well as adaptive hybrid approaches.

8 Acknowledgements

We are grateful to David Hsu and Jean-Claude Latombe for a version of their planner and many useful discussions related to this problem. We are also grateful to Jean-Paul Laumond for providing us with the model of Benchmark 6.

References

[ABD+98] N. Amato, O. Bayazit, L. Dale, C. Jones, and D. Vallejo. Obprm: An obstacle-based prm for 3d workspaces. *Proceedings of WAFR98*, pages 197–204, 1998.

[Blu67] H. Blum. A transformation for extracting new descriptors of shape. In W. Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, 1967.

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

Benchmark	Sampling Type	Total Time	C-Space Samples	Free Samples	Full Coll. Checks	QR Coll.	Connect Comp. Calls
6	Uniform	3948.86	91264	43542	250043	0	452459428
6	Voronoi Based	2983.24	58265	38098	220023	10821	351247399

Table 7: *Benchmark 6, piano environment shown in Image 6. A very challenging Type-R narrow passage still shows* 25% *speedup.*

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

[CKR97] 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.

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

[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.

[GHK99] L. Guibas, C. Holleman, and L. Kavraki. A probabilistic roadmap planner for flexible objects with a workspace medial-axis-based sampling approach. In *Proc. of IROS*, 1999.

[GLM96] 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.

[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. Interactive motion planning using hardware accelerated computation of generalized voronoi diagrams. Technical report, Department of Computer Science, University of North Carolina, 1999.

[HKL+98] D. Hsu, L. Kavraki, J. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. *Proc. of 3rd Workshop on Algorithmic Foundations of Robotics*, 1998.

[HST94] T. Horsch, F. Schwarz, and H. Tolle. Motion planning for many degrees of freedon - random reflections at c-space obstacles. *Proc. of IEEE International Conf. on Robotics and Automation*, pages 3318–3323, 1994.

[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.

[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.

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

[LBD+92] D. Lavender, A. Bowyer, J. Davenport, A. Wallis, and J. Woodwark. Voronoi diagrams of set-theoretic solid models. *IEEE Computer Graphics and Applications*, pages 69–77, September 1992.

[LGLM99] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha. Fast proximity queries with swept sphere volumes. Technical Report TR99-018, Department of Computer Science, University of North Carolina, 1999.

[OŠ95] M. H. Overmars and P. Švestka. A probabilistic learning approach to motion planning. In *Algorithmic Foundations of Robotics*, Boston, MA, 1995. A. K. Peters.

[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.

[SAR95] D. J. Sheehy, C. G. Armstrong, , and D. J. Robinson. Computing the medial surface of a solid from a domain delaunay triangulatio. In Chris Hoffman and Jarek Rossignac, editors, *Solid Modeling '95*, pages 201–212, May 1995.

[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.

[VO97] J. Vleugels and M. Overmars. Approximating voronoi diagrams of convex sites in any dimension. *International Journal of Computational Geometry and Applications*, 8:201–222, 1997.

[WAS99a] Steven A. Wilmarth, Nancy M. Amato, and Peter F. Stiller. Maprm: A probabilistic roadmap planner with sampling on the medial axis of the free space. *IEEE Conference on Robotics and Automation*, 1999.

[WAS99b] Steven A. Wilmarth, Nancy M. Amato, and Peter F. Stiller. 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.