SM01-144: A Voronoi-Based Framework for Motion Planning and Maintainability Applications

Mark Foskey Maxim Garber Ming C. Lin Dinesh Manocha
Department of Computer Science
University of North Carolina
Chapel Hill, NC 27599-3175
{foskey,garber,lin,dm}@cs.unc.edu
http://www.cs.unc.edu/~geom/voronoi/fplan

Abstract

We present a new approach for motion planning and maintainability studies in complex 3D CAD environments. It automatically computes a collision-free path for moving objects among stationary and completely known obstacles. Our framework uses a combination of analytic, criticality-based and probabilistic techniques to compute the path. It starts with an global strategy, which uses an approximate Voronoi diagram of the workspace, computed using graphics hardware, and combines it with bridges computed by a localized path planner. The resulting system has been used for planning the path of free-flying rigid as well as articulated bodies. Applications of our system include robot motion planning, part removal, and assembly planning. We have implemented a simple planner within this framework and tested it on a number of benchmarks.

1 Introduction

The problem of autonomous motion planning deals with computing collision-free paths for moving objects in an environment composed of obstacles. Besides robotics, this problem arises in solid modeling, molecular modeling, computer animation, maintainability studies, assembly planning, radiotherapy planning etc. [Lat91, Lat99]. For example, in assembly maintainability studies, motion planning is used to find whether it is possible to remove a particular part from an assembly, and if so, to find one possible removal path [CL95]. Earlier, such tasks were performed by using a physical mock-up of the assembly being designed. The paths were computed by physically moving the mock-up part. As more and more assemblies are being designed using CAD systems, the trend is moving away from physical mock-ups and towards the use of motion planning tools for automatic path computation and refinement. Many of the commonly used computer-aided manufacturing and simulation packages like IGRIP from Delmia Inc., CimStation from Adept Technologies, THOR Arc Weld from Amrose Inc., PDMS from CADCENTER, ProductVision from GE Corporate R & D, etc. include support for motion planning. These packages are widely used for pipe routing for plant design, arc welding of complex assemblies, spot welding of car bodies as well as using robots to automate the manufacturing processes.

Motion planning has been extensively studied in robotics, computational geometry, solid modeling and computer-aided manufacturing for more than three decades. However, it is still considered difficult to solve the problem in its most basic form, e.g., to find a collision-free path for a rigid or articulated object among static obstacles. The best known complete algorithm for computing a collision-free path has complexity exponential in the number of degrees of freedom (dof) of the robot or the moving object [Can88]. Such planners, also called *criticality-based* planners, rely on an explicit, global geometric analysis to generate a provably complete representation of the configuration space for the robot so that it can be effectively searched for a path. In practice, these planners are challenging to implement and slow during execution. The theoretical and practical complexity of complete planners has motivated the development of planners that rely on approximate or heuristic methods [GdPe88, Lat91]. Such planners are relatively simple to implement. However, they are not always guaranteed to find a path, if one exists. Examples of such planners include those based on cell decomposition approaches or potential field methods. In practice, these planners can work well for simple motion planning problems or for robots with less than four or five degrees of freedom. Over the last decade, a more promising approach based on randomized sampling of robot's configuration space has been proposed [KSLO96]. It builds a *probabilistic* *roadmap* (PRM) in the configuration space of the robot. The resulting planner has been shown to work well in many scenarios, including high-dof robots. However, these planners may not work well when the robot's free configuration space has *narrow corridors* or *narrow passages*. Such cases frequently arise in maintainability studies or assembly planning [CL95, VJA00]. Most planners used in commercial systems are based on cell decomposition or randomized approaches.

Main Contributions: We present a novel framework for motion planning that is based on a Voronoi diagram of the workspace of an environment. The framework uses a hybrid approach that combines some aspects of criticality based methods along with randomized sampling. In a preprocessing step, it computes a discrete approximation of the generalized Voronoi diagram (GVD) of the workspace using graphics hardware. Intuitively speaking, the GVD corresponds to the set of points that are farthest from the obstacle boundaries and have maximum clearance. We use it to generate an estimated path for the robot through the environment. In our framework, we identify *invalid segments* of this path, which correspond to configurations that do not lie in the free space. We use randomized sampling or other localized approaches to compute a path in the regions corresponding to invalid segments. Moreover, where possible we use geometric information from the GVD to guide the localized planner, for example to bias the sampling in a randomized planner.

Our framework has been applied to different environments and maintainability applications, including ones with narrow passages or large CAD environments, for both articulated and rigid robots.

Organization: The rest of the paper is organized in the following manner. We briefly survey previous work on motion planning in Section 2. Section 3 presents an overview of our approach. We describe the preprocessing steps and algorithms for Voronoi computation in Section 4. The planning algorithm is presented in Sections 5 and 6 and we describe its implementation and performance in Section 7.

2 Background and Related Work

In this section, we provide some background information related to motion planning and provide a brief survey of related work in this area.

2.1 Notation and Representation

We briefly introduce the notation used in the rest of the paper. We will denote the robot or the moving object as $\mathcal{R}$ and the union of the set of obstacles as $\mathcal{O}$. We assume that the robot and each obstacle is a closed and bounded set whose geometry and location are both accurately known. We also assume that the obstacles remain fixed. The robot is moving in a Euclidean space $\mathcal{W}$, called the *workspace*. The robot may be a free-flying rigid object or an articulated object. A free-flying object has no kinematic constraints that limit its motion. On the other hand, an articulated object $\mathcal{R}$ is made of several moving rigid objects $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n$, called links, connected by joints. Each joint constrains the relative movements of the two objects it connects. The basic motion planning problem that we address in this paper is: Given an initial position and orientation and a goal position and orientation of $\mathcal{R}$ in $\mathcal{W}$, generate a path P from the initial configuration to the goal configuration. The path P is a continuous sequence of positions and orientations of $\mathcal{R}$ avoiding contact with $\mathcal{O}$.

We use the *configuration space* formulation [LPW79] in this paper to solve the planning problem. In this formulation, the robot is represented as a point in a higher dimensional space, called the configuration space of the robot. In this way any planning task can be interpreted in terms of a point robot, so that the problem of characterizing the constraints on the robot's motion is isolated from that of finding a path that satisfies those constraints. For a rigid body, the configuration, denoted by $\mathbf{c}$, is specified by six coordinates, three determining the position of some fixed reference point on $\mathcal{R}$ (its *origin*), and three determining the robot's orientation. An articulated object can be interpreted as a set of n moving rigid objects connected by joints. By convention, each joint affords a single degree of freedom, and a physical joint that allows more than one degree of freedom is represented by multiple joints at a single location, so that there can be more joints than links. With this convention, a configuration $\mathbf{c}$ for an articulated body is specified by six coordinates determining the position and orientation of a given link, the *base link*, along with an additional coordinate for each joint. The configuration space $\mathcal{C}$ is the set of all possible configurations $\mathbf{c}$. Furthermore, $\mathcal{C}$ can be partitioned into the *free space* $\mathcal{F}$ and the *blocked space* $\mathcal{B}$. The blocked space is the set of configurations for which the robot $\mathcal{R}$ collides with at least one of the obstacles. The rest of the configuration space is the free space. In other words, $\mathcal{F} = \mathcal{C} \backslash \mathcal{B}$.

2.2 Voronoi Diagrams in Motion Planning

Generalized Voronoi diagrams have long been used as a basis for motion planning algorithms [ÓSY83, CD88, CB95, CB96, WAS99a]. The GVD represents the connectivity of a space but has a dimension lower by one, and (in three dimensions) it is composed of surfaces of maximal clearance. Unfortunately, no good and practical algorithms are known for computing the Voronoi diagrams of large environments. In the worst case the complexity of Voronoi diagram is $O(n^2)$, where n is the number of polygons in the environment. Moreover, the Voronoi diagram of a polygonal environment is composed of quadric surfaces and degree four curves that meet at junctions whose algebraic degree is eight. It is hard to accurately compute an arrangement of these curves and surfaces using fixed precision arithmetic.

2.3 Randomized Planning

Recently a class of randomized algorithms known as *Probabilistic Roadmap Methods* (PRMs) have been shown to be very successful for many planning scenarios [KL94, HLM99]. The original PRM planner [KL94] generates samples at random in configuration space and tries to connect them by collision-free local paths. By using a collision checker, the planner avoids the prohibitive computation of an explicit representation of the free space. Ultimately the graph produced by a PRM planner will tend to represent the connectivity of the C-space reasonably well, and a query can be rapidly performed by linking the search points to the graph and then searching the graph. Under reasonable geometric assumptions on the free space, it can be shown that the probability that a PRM planner fails to find a path while one exists decreases exponentially towards zero with the number of samples. Experiments with PRM have been quite successful in many situations and they have been shown to handle robots with many degrees of freedom.

2.3.1 Narrow Passages

The PRM planners work well in general, except when there are narrow passages in the configuration space. This can happen when the environment is cluttered or the obstacles are close to each other or have small tolerances between them. The performance of PRM planners degrades in such cases as the sampling algorithms are unlikely to generate configurations in the narrow regions of the C-space. One such example is shown in Fig. 1. Many modifications to the basic PRM algorithm have been proposed to handle narrow passages. 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. However, dilation can alter the topology of the free space. Furthermore, no good practical algorithms are known for penetration depth computation between polyhedral models.

- **Sampling near Obstacle Boundaries** [ABD+98]: This approach 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.

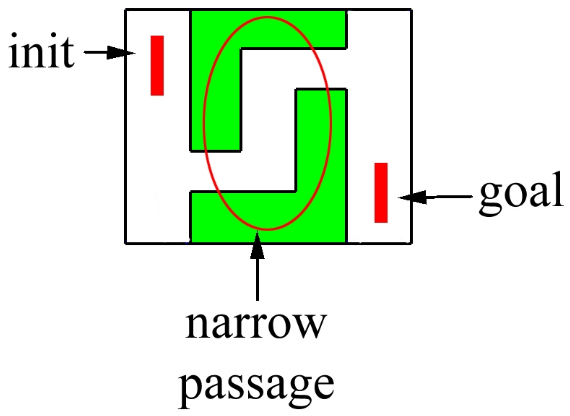

Figure 1. The rectangular robot must traverse the narrow passage in order to move from the initial position to the goal position.

- **Analysis of 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, PHLM00]: The main idea is to generate nodes that lie on the medial axis of the workspace or the free space. Wilmarth et al. [WAS99b, WAS99a] generate random configurations and retract them onto the medial axis without explicitly computing the medial axis. Guibas et al. [GHK99] compute Voronoi diagrams of points on the boundary of the obstacles used them for generating the samples. Pisula et al. [PHLM00] used the approximate medial axis, computed using graphics hardware, to bias the sampling.

2.4 Other Motion Planning Algorithms

Many other motion planning algorithms have been developed for different applications. These include part orientation and positioning, assembly sequencing and maintainability studies.

2.4.1 Part Orientation and Positioning

Different motion planning algorithms have been developed for part feeding. These include algorithms that plan the tiling motions of a tray containing a planar part of known shape to orient it to a desired position [EM88] or that compute a sequence of squeezes by a frictionless, sensorless gripper to achieve a single orientation of a polygonal part [Gol93].

2.4.2 Assembly Sequencing and Planning

Assembly planning is the problem of finding a sequence of motions to assemble a product from its parts. An assembly planner must compute both an order in which the parts can be assembled and the motions of the individual parts. Planning algorithms include those used to compute the cone of possible motions of a part in contact with another [dMS91]. Other techniques focus on capturing the (de)assembly sequencing information that is implied by existing surficial contact information [LPW93]. More recently, Halperin et al. [HLW99] have presented a framework based on the motion space approach, which is similar in spirit to the configuration space framework for motion planning.

2.4.3 Maintainability Studies

The main goal in a maintainability study is to determine whether a part or the line replaceable unit (LRU) can be removed form an assembly, and, if so, by what sequence of movements [CL95]. Some of the commonly used systems, like ProductVision from GE, utilize randomized motion planning techniques.

3 Overview

We propose a general framework for free-flying rigid and articulated models. It is a hybrid approach that incorporates some features of criticality based approaches along with randomized and other methods. The framework decomposes the planning problem into a global analysis based on the Voronoi diagram of the workspace, followed by a series of localized planning queries, as shown in Figure 2 The overall algorithm proceeds as follows:

1. Construct a discrete approximation to the generalized Voronoi diagram (GVD) of the workspace.

2. Use the geometric information encoded in the GVD to generate an *estimated path* for the robot in the configuration space $\mathcal{C}$. Points on the GVD are used to place the origin of the robot, and the shape or local topology of the GVD is used to orient the robot along the path.

3. Determine portions of the estimated path for which the robot is colliding with any of the obstacles in $\mathcal{O}$. We call these portions *invalid segments* of the estimated path.

4. Use randomized and localized approaches to replace each invalid segment with a path in the free space $\mathcal{F}$.

The idea of using randomized planning method to correct an estimated path is the key to our hybrid approach. The use of randomized planners to correct invalid segments allows us to use simple methods for global analysis, since we need not ensure that our global analysis initially gives us

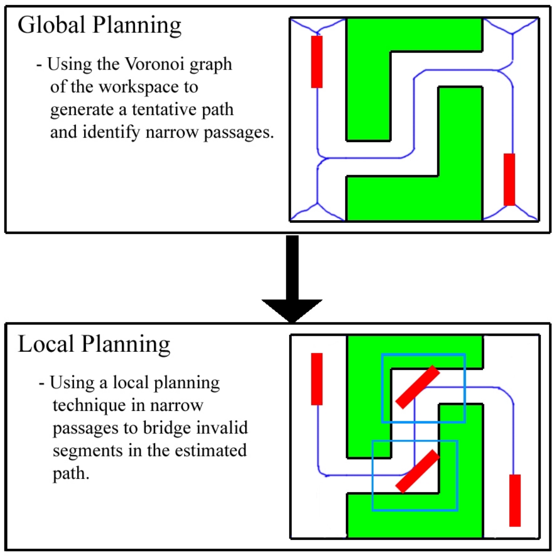
Global Planning
- Using the Voronoi graph of the workspace to generate a tentative path and identify narrow passages.
Local Planning
- Using a local planning technique in narrow passages to bridge invalid segments in the estimated path.

Figure 2. A block diagram showing how our approach breaks the motion planning problem into a global planning phase, which generates an estimated path, and a series local planning steps with domains restricted to areas where the estimated path leads to collision (shown in the two marked rectangular areas).

a complete and correct path. Conversely, the global analysis and the estimated path helps us in identifying potential narrow passages in the configuration space. As a result, the randomized approaches can restrict the sampling to localized regions, as opposed to searching and sampling in the entire configuration space.

4 Preprocessing

In this section, we describe the algorithm used to compute the discretized Voronoi diagram. The resulting graph is used in computing the initial estimated path for the planner.

4.1 Generalized Voronoi Diagram

The generalized Voronoi diagram [OBS92] is defined in terms of a set $\{s_1, \ldots, s_n\}$ of geometric objects, called *sites*. For each site s_i we can define a distance function d_i on $\mathcal{R}^3$ by $d_i(\mathbf{x}) = \mathrm{dist}(s_i, \mathbf{x})$. The *Voronoi region of* s_i is the set is the set V_i of points at least as close to s_i as to any other site. That is, $V_i = \{\mathbf{x} \mid d_i(\mathbf{x}) \leq d_j(\mathbf{x}) \forall j \neq i\}$. In three dimensions, the intersection of two Voronoi regions is called a *Voronoi face*. Similarly, an intersection between Voronoi faces is called a *Voronoi edge*, and an intersection between Voronoi edges is a *Voronoi vertex*. The collection

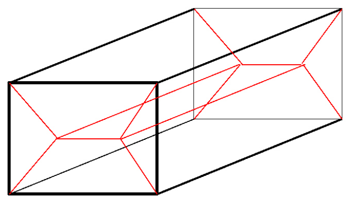

Figure 3. The Voronoi edges for a simple rectangular box. The interior of the box is divided into six regions, each of which is closest to one of the faces.

of Voronoi regions $\{V_i\}$ is the GVD of the sites S_i. Often we will also use the term "GVD" to refer to the union of the Voronoi faces. The Voronoi vertices and edges form the generalized Voronoi *graph* (GVG) of the sites. See Figure 3 for a simple example. The GVD is a *deformation retract* of the workspace [CD88] which ensures that it reflects the topology of the workspace. The GVG, however, can be disconnected even if the workspace is connected [CB96].

4.2 Computation Using Graphics Hardware

Our method for computing the generalized Voronoi diagram is based on the algorithm presented by Hoff et al. [HCK+99], and it uses standard polygon rasterization hardware. We compute the discrete Voronoi diagram in slices. For each slice L_z determined by a given z value, and each site s_i, there is a real-valued distance function d_i^z given by $d_i^z(x, y) = d_i(x, y, z)$. In words, the value of $d_i^z(x, y)$ is the distance in 3-space from (x, y, z) to s_i. As an example, consider the case of a Voronoi site s_i that is the single point $(0, 0, 1)$, and the slice L_0. In this case, $d_i^0(x, y) = \sqrt{x^2 + y^2 + 1}$.

The graph of d_i^z is a surface for which we generate a triangular approximation called a *distance mesh*. We assign each site s_i a unique identifying color, and we render its distance mesh d_i^z in that color by using a parallel projection. After all the distance meshes are rendered, we have, for each pixel, the identity of the nearest site, determined by the color, and the distance to that site recorded in the depth buffer. The algorithm reads back the color buffer and the depth buffer. The depth buffer contains the distance field, i.e. closest distance to one of the obstacle for each pixel in the slice.

In our discrete representation of the GVD, we regard the Voronoi boundaries as lying between neighboring pixels. A discrete Voronoi edge consists of a sequence of *pixel edges*, with each pixel edge bounded by four pixels. The endpoints of pixel edges are *pixel vertices*. If each pixel is regarded as filling a small solid cube, then the pixel edges and vertices are the edges and vertices of the cube. We compute the GVD by scanning the 3-D pixel map, two slices at a time, seeking pixel edges whose neighboring pixels are not all the same color. A pixel edge whose neighboring pixels exhibit exactly two different colors represents a portion of a Voronoi face, and one whose neighboring pixels have at least three colors lies on a Voronoi edge. A pixel vertex lying on at least three such edges represents a Voronoi vertex, and a chain of pixel edges linking two such vertices represents a Voronoi edge. Together, all of these pixel edges and vertices make up a dense graph, which fairly closely represents the structure of the GVD, and which can be searched for a path through the workspace. The complexity of this graph can be reduced by choosing a subset of the pixel vertices, and linking them by graph edges to retain the essential topological information. In a further reduction, we may consider only the Voronoi graph, with a small number of additional edges added to preserve the proper connectivity. We will refer to the particular graph that we construct as the *Voronoi roadmap*.

One possiblity to compute the GVD is to use an incremental marching algorithm. Rather, we scan through each pixel on a slice-by-slice basis. In practice, we have found that this approach works better because of its superior memory coherence.

As we construct the diagram, we construct a list for each Voronoi site recording all the Voronoi vertices to which that site is a nearest neighbor. When a Voronoi vertex is found, its nearest neighbors are determined by the colors in the adjacent pixels. We add a pointer to that vertex to the lists associated with those neighboring sites.

5 Estimation Phase

In this section, we present our algorithm to compute the estimated path in the configuration space. After computing the discretized GVG, we use it to generate an estimated path for the robot in the configuration space. This path is not guaranteed to lie in the free space of the robot and is only used as an initial approximation to the final path computed by our framework.

5.1 Generating a Path in the Workspace

The first step in generating the estimated path is to find a path through the workspace suitable for a single point robot. The origin of the robot $\mathcal{R}$ will follow this *workspace path*, and we will use other techniques to determine the orientation of the robot and the position of its joints as it moves.

Define a *query configuration* to be an initial or goal configuration, and a *query location* to be the location in $\mathcal{W} \subset \mathbf{R}^3$ determined by a query configuration. Then the workspace path links the initial and goal query locations.

To compute the workspace path, we first modify the

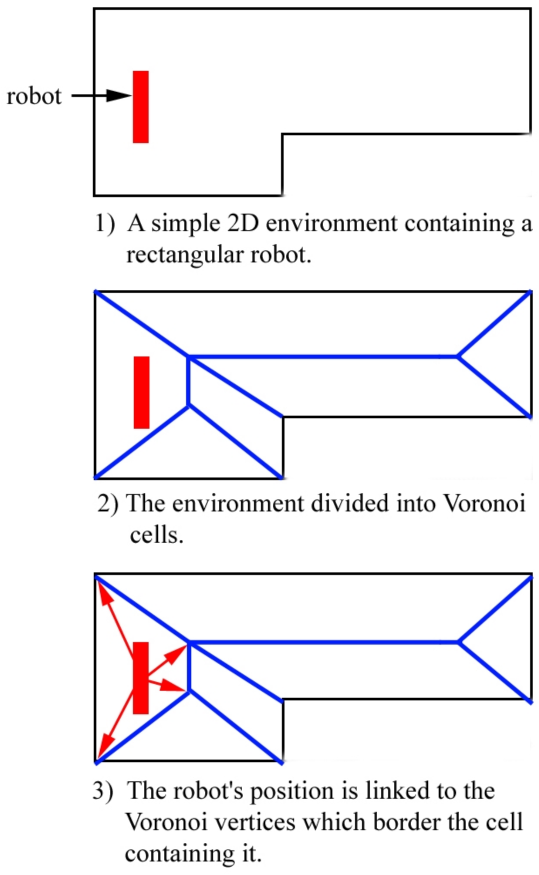
robot
1) A simple 2D environment containing a rectangular robot.
2) The environment divided into Voronoi cells.
3) The robot's position is linked to the Voronoi vertices which border the cell containing it.

Figure 4. A simple 2D example showing how a query location is linked to the GVD.

roadmap by adding the query locations as nodes in the graph, and connecting them to the rest of the graph by simple edges.

To link a query location p, we first determine the Voronoi region containing p, and then we compute line segments from p to each Voronoi vertex of the region as shown in Fig. 4. These vertices are read off from the list made during construction of the roadmap. Using a collision detection package, we check these segments for intersection with the obstacles, and discard those that intersect. The remaining edges we add to the graph. Each newly added edge contains a list of points and the value of the minimum distance to the environment.

After linking the query locations to the GVG, we use a generalized single-source shortest paths algorithm. The path "length" metric we use is chosen so that a path will always be shorter than another if it has a greater clearance (measured to the nearest obstacle) at its narrowest point. If two paths have equal clearance (as will happen if they both pass through the same bottleneck), then paths are compared by a weighted length that takes into account the actual length of the path along with the reciprocals of path width measured at all points along the path.

The rationale for the choice of this metric is that the narrowest point on a path through $\mathcal{W}$ is likely to determine the difficulty of finding a valid path for the robot through that approximate route. Since there will be many paths sharing the same narrowest point, it is essential that there be a method for breaking ties, and the appropriate criterion will reward paths that are both wide and short.

5.2 Orientation and Joint Positions

After finding the workspace path, we must choose an orientation for the robot at each point on the path. If it is an articulated robot, we must also choose joint positions for each joint. We propose two approaches to this problem.

The first approach is most effective for a rigid robot. We determine a major axis for the robot, and align it with the tangent vector of the path, as determined by a finite difference estimate. For a complex shape, there are many reasonable definitions of the "major axis." For our purposes, we want an axis around which the robot fits as tightly as possible. To determine such an axis, we use linear regression to fit a line to the vertices of the robot. This line is chosen to minimize the root mean square of the (Euclidean) distances of the vertices to the line. The origin of the robot is defined to be the center of gravity of the vertices. This method can be applied in a very elementary way to an articulated robot by aligning the major axis of the base link with the tangent to the workspace path, and simply interpolating the joint positions between the initial and goal configurations.

Once we have determined how to align the specified major axis of the robot, it is still free to rotate about that axis. The choice of orientation about the major axis (i.e., the "roll") is made arbitrarily. We simply make sure that the orientation varies continuously as the robot traverses the path.

A more sophisticated approach, that applies to articulated robots as well as rigid ones, is to allow the workspace path to define a *potential field* that behaves as a force attracting points on the robot towards the path. One can use a simple physical simulation to find the orientation and joint positions that approximately minimize the potential for the robot. Similar approaches have been taken by Holleman and Kavraki [HK99] and Hoff et al. [ICK$^+$00].

6 The Localized Planning Phase

The estimated path computed for the robot may not lie in the free space. As a result, our framework uses a variety of local or randomized approaches to correct the path or locally recompute portions of that path to compute a collision-free path from the initial to the goal configuration.

6.1 Finding Invalid Segments

The first step in completing the path is to check the estimated path for validity. We attempt to connect each configuration to its successor by a straight line in configuration space, using an algorithm given in [HLM99]. Configurations for which the robot is in collision, or which cannot be connected to a neighbor, are marked "invalid". We define a sequence of consecutive invalid configurations between two valid configurations as an *invalid segment*, and similarly a maximal consecutive sequence of valid configurations is a *valid segment*. By *endpoints* of an invalid segment, we mean the immediately adjacent valid configurations.

6.2 Bridging Invalid Segments

The estimated path has now been decomposed into valid and invalid segments. For each invalid segment, we perform a query with a different planner, which might be, for instance, a randomized planner or a potential field planner. The initial and goal configurations for this query are simply the valid endpoints bounding the invalid segment. The choice of local planner can be made at run time, similarly to the approach of Vallejo, Jones, and Amato [VJA00].

We have found the planner of Hsu, Latombe, and Motwani [HLM99] to be useful here. This planner is a PRM planner designed for single-shot queries. It maintains trees of free configurations rooted at the start and finish. At each iteration (called an *expansion iteration*), it chooses a configuration c from one of the trees, generates new configurations in a neighborhood of c, and retains those which can be linked to c by a straight-line path in $\mathcal{F}$. The local planner terminates when the two trees are connected. This algorithm automatically biases sampling towards configurations known to be free.

This approach applies equally well to articulated and rigid robots. In each case, a configuration is represented as a tuple of floating point numbers, representing a position, an orientation, and the joint positions if the robot is articulated. A neighborhood of a configuration is just a cartesian product of intervals.

The configuration space for the localized query is defined so that the orientation and any joint positions are unrestricted, and the position in 3-space is restricted to the tightest axis-aligned box that contains bounding balls for the robot at both query locations. See Figure 5.

This restriction of the configuration space limits the region that the randomized planner can explore, and hence the time taken.

It can happen that the restricted configuration space for the PRM query is too small, so that there is not enough room for the robot to traverse the invalid segment. To handle such situations we use a simple expedient: If, after a fixed number of expansion iterations, the planner has not linked the two ends of the invalid segment, the planner's configuration space is enlarged to the full original C-space, and randomized planning is resumed. If this is not successful after another predetermined number of iterations, then it is assumed that the heuristics guiding the initial path estimate have failed, and the planner simply uses other planning methods (i.e., PRM in our current implementation) to link the original start to the original finish.

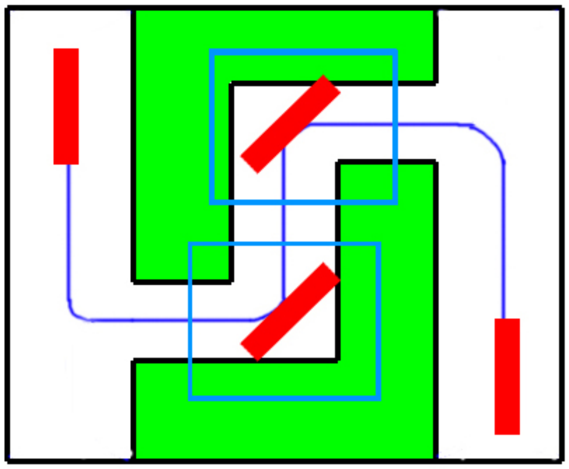
Figure 5. The two marked rectangular areas indicate the restricted configuration space for randomized sampling.

6.3 Narrow Passages

Narrow passages arise frequently in maintanability applications, and they cause particular problems for randomized planners. The difficulty is that the probability of generating a configuration at random in the narrow passage is low. Observe that the planner of [HLM99] improves this situation if either of the query configurations is in or near the narrow passage—since new configurations are generated near old ones, they are more likely to be in the narrow passage. However, if the query configurations are not near a narrow passage through which the robot must pass, then the planner will spend a great deal of time fruitlessly searching irrelevant parts of the free space.

Our planner alleviates this problem by calling the PRM planner only when needed. Whenever the PRM planner performs a query, the initial and goal configurations are adjacent to configurations for which the workspace was constricted enough so that the estimated path caused the robot to collide with the obstacles. Since the initial and goal configurations are near the entrance to a narrow passage or in the narrow passages, and the configuration space for sampling is restricted, the likelihood of finding samples in the narrow passages is much increased.

7 Implementation and Performance

7.1 Implementation

We have implemented an algorithm following our framework in a preliminary form. The program was written in C++, and runs on a PC running Windows as well as an SGI running Irix. We discuss each phase of the algorithm, focusing on details specific to our implementation.

7.1.1 Computing the Voronoi Roadmap

In our implementation, we simply use the GVG as our Voronoi-based roadmap because of its simplicity. If the GVG fails to connect the query locations then the planner can still find a path using the planning method that it applies to correct invalid paths.

7.1.2 Constructing the Estimated Path

We determine the orientation of the robot by aligning the major axis with the tangent to the workspace path, as described in Section 6.2. To determine the joint positions of an articulated robot, we currently simply retain the initial positions for all but the final configuration. We use Magic Software by D. Eberly to compute the major axis and center of the robot.

7.1.3 Localized Planning

We use the PRM planner of [HLM99], described in Section 6.2, as our only localized planner. We use PQP [LGLM99] for collision detection and distance computations during randomized planning

7.2 Performance

We have tested the performance of our framework on a number of benchmarks. These include:

- **Walls-Stick (Image 1):** A series of six walls, four of which have small holes through which the robot must pass. We require the robot to pass from one end of the maze to the other, through all four holes. This benchmark was designed at Texas A & M university [VJA00].

- **Crane (Image 2):** A CAD model of a crane complex composed of more than $128{,}000$ triangles. The model contains 143 separate polyhedral parts, which are rendered in distinct colors in the figure. We performed queries with two free-flying robots: a synthetic dart shape, and an articulated robot arm part with 10 degrees of freedom.

Scene	**Res**	**GVG**	**Query**
Walls	128	2.98	0.55
Crane (Rigid)	64	138.22	17.53
Crane (Art.)	64	138.65	334.71
Maze	128	5.73	59.66

Table 1. Benchmark timings in seconds. Res: Voronoi resolution. GVG: Voronoi graph computation. Query: query phase, after Voronoi computation.

- **Maze (Image 3):** A simple maze with an articulated, free-flying robot. The robot is a series of boxes with 9 degrees of freedom.

The results of the benchmarks are summarized in Table 7.2. **Res** gives the resolution, along the largest axis of the scene, at which the Voronoi graph was computed. **GVG** is the time (in seconds) for computing the Voronoi graph. **Query** represents the time (in seconds) for the query phase, including both the path estimation and localized planning, after the Voronoi graph was constructed.

7.3 Analysis

Our framework is intended to acquire as much information as possible from a global geometric analysis that can be performed relatively quickly. From our experience with our basic implementation, we have been able to determine the types of environments for which the Voronoi-based information is particularly helpful, and those environments for which it is not. In particular, we can make the following observations:

- This framework will perform well when the robot is small relative to the entire workspace. In such cases, a robot placed on the Voronoi graph has a high likelihood of being free, whatever its orientation or the position of its joints.

- If the robot is a rigid stick-like object, then our current heuristic does a good job of choosing orientations. In general, articulated chains or other shapes that can be made to fit tightly around the workspace path should do well under our framework.

- Our use of the Voronoi diagram provides relatively little benefit if the robot is large compared to the entire scene and has a highly complex shape.

We make an additional observation about narrow passages. Narrow passages provide two kinds of problems for a planner. The planner must determine that the robot needs to go through a particular narrow passage, and it must then find a path for the robot through the passage. These two challenges are different. When the robot is already in a narrow passage, the planner of Hsu et al. works reasonably well, because newly generated configurations, being near free configurations, are likely to be free themselves. Similarly, a potential field planner, which relies on simulated forces directing the robot away from obstacles and roughly towards the goal, can do well if there is essentially only one way for the robot to go.

However, both of those methods fare poorly if, for instance, the robot must pass from an open space, through a narrow passage, to another open space. In such a situation the PRM planner can spend much time fruitlessly exploring the open spaces, while in a potential field planner the robot will tend to be "repelled" from the narrow passage, because the value of the potential field will be higher near the mouth of the passage than in the middle of the open space. In this kind of situation the determination of a workspace path provided by our approach is very helpful in forcing the robot to enter the narrow passage.

8 Conclusion and Future Work

In conclusion, we have proposed a framework for motion planning that combines a large-scale geometric analysis with the use of local planners in constrained areas. We have implemented a planner within this framework for both rigid and articulated free-flying robots that performs well in a number of benchmarks.

There are a number of areas of potential future work:

- It would be worthwhile to study the effectiveness of localized planners other than the PRM planner we have used in our implementation. There are many planners that show promise of working well in this context, including potential field approaches and other randomized approaches. Vallejo, Jones, and Amato [VJA00] propose a framework for adaptively selecting different local planners based on inferred properties of the scene; it would make sense to adapt that approach to our setting.

- We have already noted that our current method of generating initial orientations and joint positions by aligning the major axis of the robot with the workspace path is of limited generality. We would like to implement and test more sophisticated approaches, perhaps using potential fields as described in Section 5.2.

- The GVD of the robot can provide useful information about the robot's geometry. We intend to investigate how that information can be used.

- We would like to apply our framework to more realistic benchmarks, including assembly maintainability studies.

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.

[Can88] J.F. Canny. *The Complexity of Robot Motion Planning*. ACM Doctoral Dissertation Award. MIT Press, 1988.

[CB95] 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.

[CD88] J. F. Canny and B. Donald. Simplified voronoi diagrams. *Discrete and Computational Geometry*, 3:219–236, 1988.

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

[dMS91] L. S. Homem de Mello and A. C. Sanderson. A correct and complete algorithm for the generation of mechanical assembly sequences. *IEEE Trans. on Robotics and Automation*, 7(2):228–240, 1991.

[EM88] M. Erdmann and M. Mason. An exploration of sensorless manipulation. *IEEE Tr. on Robotics and Automation*, 4:369–379, 1988.

[GdPe88] K. Gupta and A. del Pobil (eds.). *Practical Motion Planning in Robotics: Current Approaches and Future Directions*. John Wiley, West Sussex, England, 1988.

[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.

[Gol93] K. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10:201–225, 1993.

[HCK+99] 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.

[HK99] C. Holleman and L. Kavraki. A framework for using the workspace medial axis in PRM planners. *International Journal of Computational Geometry and Applications*, 9((4 & 5)):495–512, 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.

[HLM99] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *International Journal of Computational Geometry and Applications*, 9((4 & 5)):495–512, 1999.

[HLW99] D. Halperin, J. Latombe, and R. Wilson. A general framework for assembly planning: The motion space approach. *Algorithmica*, 1999. Special issue on Robot Algorithms.

[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.

[ICK+00] Kenneth Hoff III, Tim Culver, John Keyser, Ming C. Lin, and Dinesh Manocha. Interactive motion planning using hardware-accelerated computation of generalized voronoi diagrams. In *Proc. IEEE International Conference on Robotics and Automation*, 2000. To appear.

[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.

[Lat99] J.C. Latombe. Motion planning: A journey of robots, molecules, digital actors, and other artifacts. *International Journal of Robotics Research*, pages 1119–1128, 1999.

[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.

[LPW79] T. Lozano-Pérez and M. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Comm. ACM*, 22(10):560–570, 1979.

[LPW93] T. Lozano-Perez and R. Wilson. Assembly sequencing for arbitrary motions. *Proc. IEEE International Conference on Robotics and Automation*, 1993.

[OBS92] Atsuyuki Okabe, Barry Boots, and Kokichi Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, Chichester, UK, 1992.

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

[ÓSY83] C. Ó'Dúnlaing, Micha 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.

[VJA00] D. Vallejo, C. Jones, and N. Amato. An adaptive framework for single shot motion planning. In *Proc. of IROS*, 2000.

[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.