# New Approach for PRMs Based on Geometric Features

ANTONIO BENITEZ AND DANIEL VALLEJO
Department of Computer Engineering
Universidad de las Américas - Puebla
Sta. Catarina Mártir, Cholula, Puebla. 72820
MEXICO
http://www.udlap.mx/~sc098381

*Abstract:* - This paper presents a new strategy for computing useful configurations; using a probabilistic roadmap method for free flying objects in a known indoor environment. We introduce two geometric features based on *straightness* and *volume* characteristics, which can be extracted of the geometry form of the objects into the environment. We describe a new method for sampling difficult configurations into narrow corridors taking advantage of these features. Simulation results using different benchmarks in motion planning show the potential of our approach.

*Key-Words:* - Probabilistic Roadmap Methods, geometric features, robotics.

## 1 Introduction

Designing a path planer is a central topic in robotics research. The ultimate goal of a path planner is to produce a valid path from an initial to a final configuration. However, the discrepancies between the geometric virtual world where the actions are planned, and the mechanical world where the robot operate, constrain the selection of actions which can be included in a solution path.

An important issue in PRM planners is the method for choosing the random configurations for the construction of the roadmaps. Recent works have considered many alternatives to a uniform random distribution of configurations as means for dealing with the narrow passage problem. A resampling step, creating additional nodes in the vicinity of nodes that are connected with few others, is shown in [10]. Nodes close to the surface of the obstacles are added in [3]. A dilation of the configuration space has been suggested in [8], as well as an in depth analysis of the narrow passage problem. In [17] a procedure for retracting configurations onto the free
space medial axis is presented. In [4] a probabilistic method for choosing configurations close to the obstacles is presented.

Probabilistic roadmap (PRM) have proven to be an effective tool to capture the connectivity of a robot's collision-free space and solve path-planning problems with many degrees of freedom (dofs) [1,3,9] and/or complex admissibility (e.g., nonholonomic, stability, dynamic, and visibility constrains) [12,13,16]. A PRM planner samples the configuration space at random and retains the collision-free point as free configurations. The free configurations and local paths form the *probabilistic-roadmap*. The motivation is that it is often impractical to explicitly compute the collision free subset (the free space) of a configuration space.

Several promising heuristics for path planning have been proposed, of which we mention a few here. The Randomized Path Planner (RPP) of Barraquand and Latombe [5] is a potential field method that uses random walks to attempt to escape local minima. Although this method has been shown to work well for many dof robots, there exist simple situations in which it performs poorly (i.e., does not find a solution) [7,9]. Researchers have proposed various potential functions (e.g., [6]) and other techniques. In general, potential field methods can be quite effective when the configuration space (C-space) [15] is relatively uncluttered. However, they have not been as successful for planning in crowed C-space.

## 2 Our Results

We propose a new algorithm which combines these two approaches by generating random networks whose nodes lie on the obstacles surface. Our central observation is that it is possible improve the connectivity of configuration space using geometric features on the workspace to find free configurations close to the obstacles. The main novelty in our approach is a new method for generating roadmap candidate points. In particular, we attempt to generate candidate points distributed around each obstacle on work-space taking advantage on their geometric features. Using this approach, high quality roadmaps can be obtained even when work-space is crowded. Experimental results with *free flying objects* with six degrees of freedom (dof) will be shown.

The approach extends fairly easily to dynamic environments. Our approach can be applied

to some important situations that have so far not been satisfactorily solved by heuristic methods (Paths through long, narrow passages in crowded Work-space can be found)..

# 3 Building the Roadmap

## 3.1 Problem Constrains

This work takes account several constrains on the motion planning problem, first the heuristic is focus to solve the planning problem for *free flying objects* into a three dimensional space. Second, the technique assumes the size of the bodies which conform the environment (including the robot) are relatively small than the environment size, besides we assume an obstacle can be formed by several small ones. This proposal does not compute the geometric features which it is using, the technique assumes that this characteristics can be extracted and are used to approximate the form of the bodies defined into the environment (given the constrains mentioned before). The last constrains allow to take advantage of the bodies form, because as we already said, an obstacle can be built by others small ones, therefore the *straightness* and *volume* features are defined for each one. The Figure 1 shows how this features are defined for each part of the obstacle (where the obstacles is a grid).

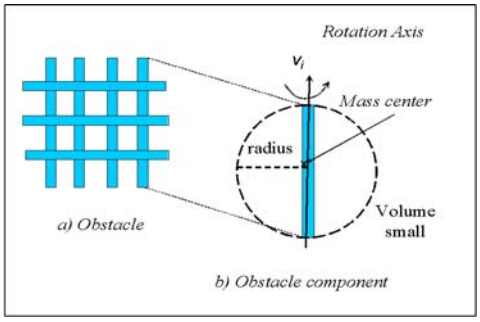


Figure 1. A big obstacle can be separeted into small ones, this is a good advantage on the present approach

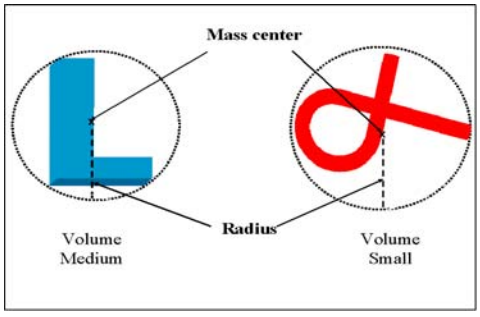


Figure 2. Defining important features on the geometry objects

## 3.2 Algorithm Desciption

The description of the algorithm is divided into three parts: first approximation of the configuration space, improving the connectivity using geometric features and planning. The following sections describe how the algorithm works and we give some details about its implementation. Some figures are included to show the main idea behind this new technique.

The algorithm is based on geometric features of the obstacles and the robot, as we said in previous section. We describe the method including a first sampling stage and an algorithm to improve the connectivity.

## 3.3 Definitions and Geometric Features

Let $V = V_0,...,V_n$ be a set of obstacles in the workspace $W$. Let $r_i$ be the radius associate to the sphere which involve each body in the workspace (including the robot).

Let $d_i$ be the distance between two configurations, one of them associate to the robot and the another one associate to the obstacle $V_i$.

Let $c(V_i)$ be the random configuration computed by the heuristic using the obstacle $V_i$.

The workspace is read from different files, these files contain the triangle meshes which represent the geometry of each body into the workspace. Using such representation, the algorithm computes several parameters which will be used by the heuristic and are defined as following:

**Mass Center**. This metric is calculated as the average of the $x_s$, $y_s$, and $z_s$ values of the vertex for a body into the environment. This parameter might not be placed into the object.

**The Body Radius**. This parameter is computed using the distance between the center mass and the farest vertex into the object. This metric is used to calculate the sphere which the object will be surrounded. The Figure 2 shows how radius is used to obtain the sphere to surround the body.

**Straightness**. Let $q_i$ be the vector which define the direction of the *"straightness"* feature for each $V_i \in B$, and $qr_i$ will define the same feature on the robot. This feature indicates the direction which the body presents its long side. The Figure 3 shows how this feature can be see for an object.

**Volume**. Let $vol_i$ be the *"volume"* of the obstacle $V_i$ with respect the volume of the sphere used to surround it. Both, the *"straightness"* and *"volume"* features are used to improve the connectivity of the roadmap and the Figure 2 shows a geometric representation of this parameter.

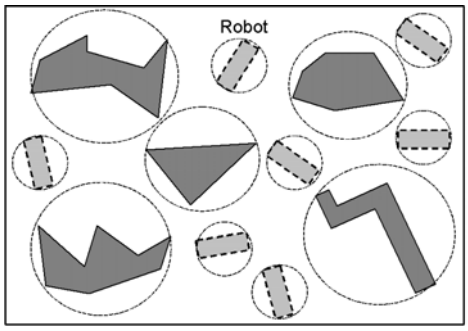


Figure 3. The rotation axis is defined in the same direction of the *straightness feature*.

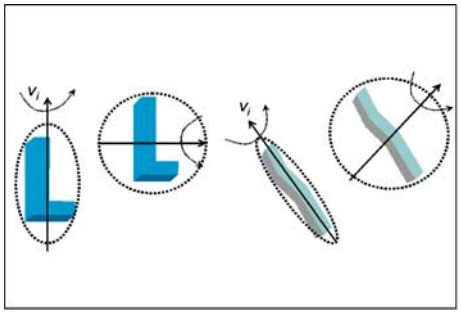


Figure 4. First approximation using spheres surrounding the robot and computing a reduced number of configurations.

### 3.4 Building the Roadmap

The roadmap is an undirected graph $R=(N,E)$. The nodes in $N$ are a set of configurations of the robot appropriately chosen over the free C-space. The edges in $E$ correspond to (simple) paths; an edge between two nodes corresponds to a feasible path connecting the relevant configurations. These paths are computed by an extremely fast, though not very powerful planner, called the local planner. During the query phase, the roadmap is used to solve different path planning problems in he input scene. Given a start configuration $q_{init}$ and a goal configuration $q_{goal}$, the method first tries to connect $q_{init}$ and $q_{goal}$ to some two nodes $q_{init}'$ and $q_{goal}' \in N$. If successful, it then searches $R$ for a sequence of edges in $E$ connecting $q_{init}'$ to $q_{goal}'$. Finally, it transforms this sequence into a feasible path for the robot by recomputing the corresponding local paths and concatenating them.

**First Approximation**. During this stage, the algorithm uses spheres to surround the robot. The Figure 4 presents the view of the first sampling (in all the figures we are presenting the explications in a two dimensional space). Using spheres (or circles into two dimensional space), we have two advantages, first the robot has the characteristic to rotate in any direction ($\alpha$ , $\beta$, $\lambda$), which will be used for the local planner in the improving stage, and second, the cost of collision detection is reduced, because the routine is limited to detect when two spheres are in collision (the sphere associated to each obstacle and the other one associated to the robot). In this process just a small number of configurations will be computed.

**Expanding the Roadmap**. If the number of nodes computed during the first approximation of the roadmap is large enough, the set $N$ gives a fairly uniform covering of $C_{free}$. In easy scenes $R$ is well connected. But in more constrained ones where $C_{free}$ is actually connected, $R$ often consists of a few large components and several small ones. It therefore does not effectively capture the connectivity of $C_{free}$ .

The purpose of the expansion is to add more nodes in a way that will facilitate the formation of the large components comprising as many of the nodes as possible and will also help cover the more difficult narrow parts of $C_{free}$. The identification of these difficult parts of $C_{free}$ is no a simple matter and the heuristic that we propose below goes only a certain distance in this direction.

In this phase, we generate a set $N$ of candidate roadmap nodes, each of which corresponds to a point in C-space. The general strategy of the node generation process is to construct a set $N_i$ of candidate nodes for each object $V_i$ such that each $c(V_i) \in N_i$ lies near to obstacle $V_i$ . The set of roadmap candidate nodes is the union of the candidate sets computed for each obstacle $V_i \in B$, i.e., $N = U_i N_i$. We now consider how to compute the candidate set for each obstacle. To obtain a high quality roadmap we would like the nodes to be uniformly distributed around the obstacles and close to them. This technique attempt to take advantage of some geometric features of the robot and the obstacles to obtain information that allows guide the search of useful configurations.

The first geometric feature used is called **"Straightness"**. This feature indicates the direction on which the object presents its long side and it is given by a vector $v_i$ , which we have used as the direction of the rotation axis. The Figure 3 shows the way which the rotation axis is represented on the robot and the obstacles. We can see that the object will sweep a minor area as result of defining the rotation axis in the same direction of the straightness feature.

The second geometric feature used is **"Volume"**. This feature give to the algorithm an approximation about how big is the object respect the volume of the sphere which is surrounded. After several empirical tests, we have computed the values

for the MAX and MIN parameters, which will be used to calculate a scale factor to be used in the next stept of the heuristic called for us **the elastic band process**. The values in the Figure 5 presents these results.

If the *volume* feature is small, that means that, its value is less than 25% of the volume sphere then we can think the object is very thin, therefore, many configurations can be computed near to its mass center.

Volume value and Scale parameter for the Elastic Band Algorithm

| Volume value % | MIN - value | MAX - value |
|---|---|---|
| Vol <= 0.25 | 0.050 | 1.0 |
| Vol > 0.25 && Vol <= 0.50 | 0.15 | 1.2 |
| Vol > 0.50 && Vol < 0.75 | 0.25 | 1.5 |
| Vol > 0.75 | 0.5 | 2.0 |

Figure 5. MAX and MIN values obtained using the volume feature after several empirical experiments.

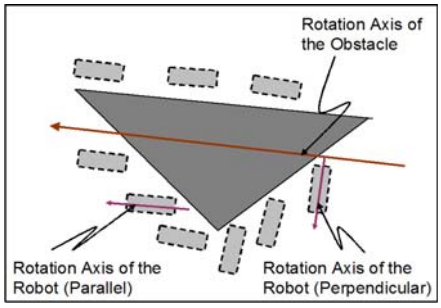


Figure 6. Parallel and perpendicular configurations around the obstacle.

**Finding points close to the obstacles**. We now consider how to generate $m$ points close to the $V_i$. For now, assume that we know the number of points we wish to obtain. Ideally, these *m* points should be uniformly distributed around the $V_i$. Using straightness feature, we propose the heuristic method outlined below to generate the points.

1. First, the algorithm search a collision configuration $c(V_i)$ on the $V_i$ obstacle, such $c(V_i)$ is calculated uniformly distributed around the sphere which the obstacle is surrounded (a vicinity for each $V_i$ is defined).

2. Next, the technique attempts rotate this configuration until it will be parallel to the obstacle (that is $v_{ri} \parallel v_i$), if the parallel configuration is not in collision then it is added to $N_i$, else the process called *elastic band* is applied searching to turn it in free configuration, which will lie close to the $V_i$ obstacle.

3. Once a parallel configuration has been processed, the algorithm computes the perpendicular configuration (that means that $v_{ri} \perp v_i$ ) taking the $c(V_i)$ calculated in step 1. In the same way like in step 2, if the new perpendicular configuration is not in collision then it is added to $N_i$, else the *elastic band* process is applied. The Figure 6, shows how the parallel and perpendicular configurations can be seen around the $V_i$.

**The Elastic Band Process** This process works as following, first it calculates the distance vector $d_i$ between the obstacle position and the $c(V_i)$ configuration, and attempt to approach and moving away the robot with respect to the obstacle. To compute this operation, the process scale the $d_i$ vector to calculate the next position where the $c(V_i)$ will be placed. The next algorithm describes this process.

*Elastic Band Heuristic*

1. *ri = radius of i-obstacle*
2. *ci_init = position of the i-obstacle*
3. *robot.get_parameters (MAX,MIN) Volume feature*
4. *k=0*
5. *di = distance vector between Bi- and the robot*
6. *do*
7. *scalar = random between ( MAX and MIN )*
8. *scale(di, escalar) the distance vector is scaled*
9. *c(Bi)=get_configuration_with_position (di)*
10. *rotate_robot_on_rotation_axis() Straightness feature*
11. *k=k+1*
12. *while ( c(Bi) is not free and k < CTE )*
13. *add_configuration c(Bi) to the roadmap*

The metric used to obtain the vector distance is the Euclidean distance in three dimensional space, and the scale factor is computed using the *"volume"* feature of the objects, see Figure 5. Thus, the scale factor will be able to be initialized since a low value (*scale*=MIN). This value has an important role, because we are interesting in configurations close to the objects, that mean that, the mass center of the objects will have to be near.

The elastic band process works with parallel and perpendicular configurations which are computed around the obstacle.

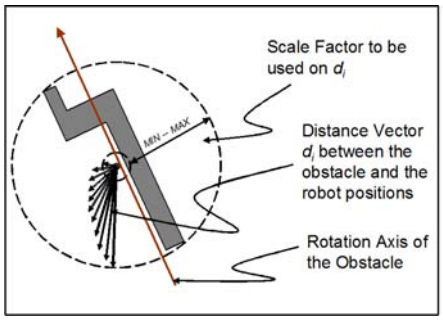


Figure 7. A distance vector between is scaled until reach a free configuration, approaching and moving the robot away from the obstacle.

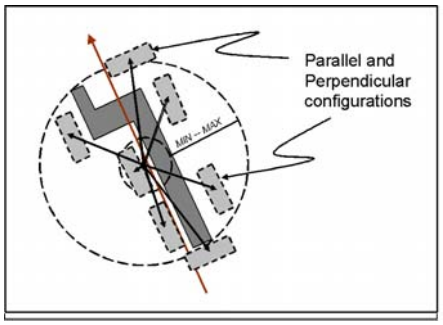


Figure 8. *Elastic Band* technique computes free configurations close to the obstacle surface.

While the vector distance is computing the next configuration to be tested, the robot is rotated on its rotation axis, searching to find a free configuration. Figure 7 and Figure 8 presents how the parallel and perpendicular configurations are computed around the obstacle and how the scalar vector is changing, approaching and moving the robot away from the obstacle.

As result of this method in Figure 9 presents the final connectivity, we can see that, many configurations around each obstacles are computed, consequently a better connectivity of configuration space is calculated, therefore, problems containing narrow corridors can be solved.

### 3.5 Connecting Roadmap Candidates

We now consider how to connect the candidate nodes $N = U_i N_i$ to create the roadmap. The basic idea is to use a simple, fast, local planner to connect pairs of roadmap candidate nodes. To save space, the paths found in this stage will not be recorded since they can be regenerated quickly. After the connections are made, the connected components in the roadmap are identified, e.g., by depth first search.

Ideally, the roadmap will include paths through all corridors in C-space. The degree to which this goal can be met depends upon a number of factors: the number and distribution of the candidate nodes, the effectiveness of the simple planner, and the number of connections attempted for each candidate node. Thus, a trade-off exists between the quality of the resulting roadmap and the resources (computation and space) one is willing to invest in building it.

Clearly, the method used to determine adjacencies in the roadmap will depend upon the intended use of the roadmap. In addition, as outlined below, the connection strategy (i.e., determining which connections to attempt) may vary from application to application. In the following, $k$ is a parameter, and distances are measured according the chosen metric in C-space.

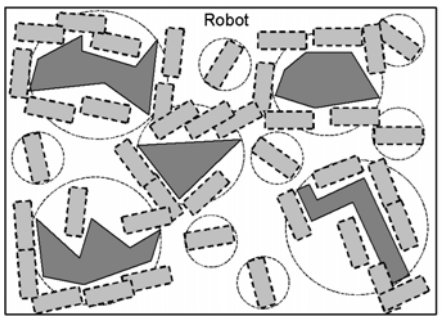


Figure 9. The connectivity computed after the *Elastic Band process*

Many different connection strategies could be used in path planning applications. For example, the method used in [9] is to try to connect each node $c(V_i) \in N$ to its $k$ nearest neighbors in $N$. A strategy that could be improved the chance of finding connections across wide work-space corridor is as follow. First, we define a *far configuration* which was computed during the first approximation of the roadmap, and we define a *near configuration* which was obtained in the expanding the roadmap stage.

Next, for each node $c \in N_i$ attempts to connect it to its $k$ nearest neighbors in $N$ and to its $k$ farthest neighbors in $N$. Note that both of these strategies require a non-trivial amount of computation to determine the $k$ closest and the $k$ farthest nodes. Thus, it might be desirable to use some heuristic method to identify the "close" and "far" nodes.

## 4 Planning

Planning is carried out as in any roadmap method: we attempt to connect the nodes $x_1$ and $x_2$, representing the start and goal configurations, respectively, to the same connected component of the roadmap, and then

find a path in the roadmap between these two connection points. The following approach, proposed in [9], is well suited for our roadmap.

First, the simple planner is used to try to connect the start a goal nodes to the roadmap; connections are attempted between $x_1$ and the $k$ - closets and farthest roadmap nodes. If no connections are made for $x_i$, then we execute a random walk and try to connect the initial or the end node to the roadmap. This can be repeated a few times if necessary. If we still can not to connect both nodes to the same connected component of the roadmap, then we declare failure. After both connections are made, we find a path in the roadmap between the two connection points using depth-first-search. Recall that we must regenerate the path between adjacent roadmap nodes since they are not stored with the roadmap. Finally, smoothing techniques can be applied to improve the resulting path.

## 5 Implementation Details

We implemented a path planner for a free flying objects with six degrees of freedom in a three dimensional workspace.

Generally, whenever there was a choice between implementation options, we choice the simplest method, which was often also the least efficient. This strategy was taken in order to produce a working prototype quickly and test the general concept of our approach.

Our implementation use this metric in both the roadmap connection and the planning phases to select the candidate nodes.

**Number of candidate nodes**. For simplicity, we attempt to generate the same number of candidate nodes for each obstacle. We did this by a uniform randomized sampling of $c$ nodes around the obstacle. In our experiments $c=25$. Even if as previously mentioned, for each of these nodes a number of 50 configurations are attempted during the elastic band process. Thus, the method is computing around 75 nodes for each obstacle, even few of them are free configurations.

**Collision detection**. The dominant operation in the creation of the roadmap is collision detection: it is heavily used both in the node generation and in the roadmap connection phases. Thus, its efficiency is of vital importance to the overall efficiency of the method. So, collision detection was performed with the *Rapid* library [14].

**Interconnection strategy**. In the interconnection phase, we would like to connect all the roadmap candidate nodes into a single connected component. Clearly, the interconnection strategy chosen can greatly effect how close we come to this goal. Recall that, we have a set $N_i$ of roadmap candidate nodes for each obstacle $V_i$ in the workspace. The basic approach we have taken is to attempt to connect each node $c \in N_i$ to its $k$ nearest neighbors in $N$ and to its $k$ farthest neighbors in $N$, where $k$ is a constant. Clearly, the larger $k$, the greater chance of connection, but also the more computation surrounded; for our experiments $k$=25.

Note that both of these strategies require a non-trivial amount of computation to determine the $k$ closest and the $k$ farthest nodes.

**Local Planner**. The local planner is used in both the roadmap connection phase of the preprocessing and in the planning. The efficiency of the local planner is of crucial importance since the roadmap connection phase required orders of magnitude more time than any other part of the preprocessing. On the other hand, the local planner most also be deterministic since we do not want to store the paths connecting nodes in the roadmap. Thus, great care should be taken in selecting this planner and different methods will be needed in different situations.

We used the following simple planner. Let $x$ and $y$ be two configurations we wish to connect. The local planner we tried was to move directly along the straight line segment connecting $x$ and $y$ in workspace, performing collision detection checks at uniform intervals on the line segment. The distance between subsequent points checked is determined by the resolution needed for the current problem instance.

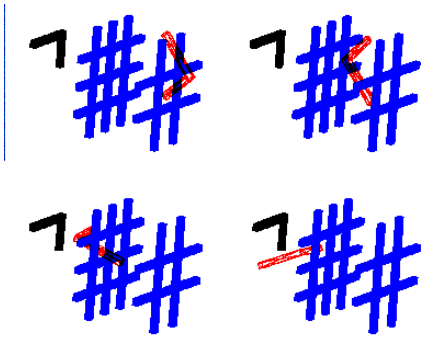

Figure 10. Env1: Reggiani's Benchmark. The robot is represented as “L”.

## 6 Experimental Results

We implemented a path planner for free flying objects with six degrees of freedom in a three dimensional workspace. The code was written in C++ on PC Intel Pentium 4, the CPU was a 2.4 Ghz with 512MB of RAM.

In the following, we analyze the performance of the method (this performance is seen since the capability of the method to solve the problems) on few scenes. In all cases we used a free-flying object robot with six dof. The various environments, and some representative configurations of the robot, are shown in Figures 10,11 and 12. Note that the roadmap size is influenced by the number of obstacles in the workspace since a set of roadmap nodes is generated for each obstacle, i.e., the size of the network is related to the complexity of the environment. The three samples shown are presented as result of the technique applied on the problems. We present three problems, they have different difficult level. The problems are labeled as Env1, Env2 and Env3. Below we discuss the environments in more detail.

**Env1:** This environment contains two grids, and the robot is represented by "L". The obstacles are placed in such a way that there are many narrow corridors between them. This roadmap is not easy to calculate, because the distance between the grids is small and the size of the robot is big. We can see in Figure 10 the solution computed by the algorithm. This sample is a Reggiani's Benchmark.

**Env2:** This scene is presented with two obstacles and we can see that the form of the robot is more complex. There is a narrow corridor which becomes difficult to solve, nevertheless, the heuristic is able to find a path which goes through the corridor. The Figure 11 presents this problem and some configurations on the environment. This sample is a Kavraki's Benchmark.

**Env3:** This problem is well known as the *"alpha puzzle"* problem. There exist several different versions of this problem [2]. The Figure 11 presents the solution for the 1.5 version of the problem. This problem have the difficult of having few configurations into the corridor, therefore its solutions is very complicated. Our method is able to find a path to solve it. In the Figure 12 some configurations into the corridor are shown.

The Figure 13 presents a table where we can see the performance of this new approach compared to our implementation of the PRM [11]. The numbers show how the new technique is able to solve the three samples. We test both algorithms using approximately the same number of configurations on the roadmap.

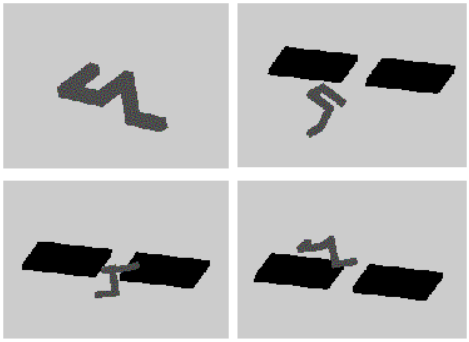

Figure 11. Env2: Kavraki's Benchmark. The form which the robot presented is more complex.

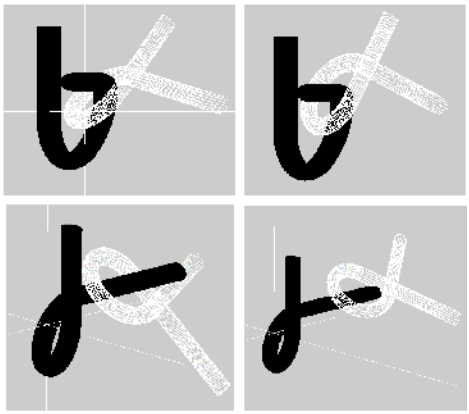

Figure 12. Env3: Amato's Benchmark. The *alpha puzzle* problem version 1.5

Results

| | Using PRM | | | Using Elastic Band Technique | | |
|---|---|---|---|---|---|---|
| Problems | Nodes in Roadmap | Time | Solution | Nodes in Roadmap | Time | Solution |
| Env 1 | 280 | 98.3 | Not Found | 260 | 240.0 | Found |
| Env 2 | 550 | 7.12 | Found | 540 | 9.30 | Found |
| Env 3 | 830 | 13.30 | Not Found | 860 | 15.30 | Found |

*The information in the table is the average after ten runnings for each sample, the time is showed in minutes.*

Figure 13. Table of results, comparing the Basic PRM and The Elastic Band Technique.

## 4 Conclusion

We have described a new randomized roadmap method for motion planning for collision free path planning. To test the concept, we implemented the method for path planning for *"free flying object"* in a

three-dimensional space. The method was shown to perform well. Currently, we keep on working on the free flying objects problems, and we are working to show the probabilistic completeness of this method.

*References:*


[1] J.M. Ahuactzin and K.Gupta. A motion planning based approach for inverse kinematics of redundant robots: The kinematic roadmap. *In Proc. IEEE Internat. Conf. Robot. Autom.*, pages 3609-3614, 1997.

[2] N. Amato. Motion Planning Puzzels Benchmarks. http://parasol.tamu.edu/~amato/

[3] N. Amato, B. Bayazit, L. Dale, C. Jones, and D. Vallejo. Obprm: An obstacle-based prm for 3D workspaces. *In P.K. Agarwal, L. Kavraki, and M. Mason, editors, Robotics: The Algorithm Perspective.* AK Peters, 1998.

[4] N. M. Amato and Y. Wu. A randomized roadmap method for path and manupulation palnning. *In Proc. IEEE Internat. Conf. Robot. Autoum,*. Pages 113-120, Mineapolis, MN, April 1996.

[5] J. Barraquand and J.C. Latombe. Robot motion planning: A distributed representation approach. *Internat. J. Robot. Res*., 10(6):628-649,1991.

[6] J. Barraquad, B. Langlois, and J.C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Trans. Sys., Man, Cybern., 22(2):224-241,1992*.

[7] D.J. Challou, M. Gini, and V.Kumar. Parallel search algorithms for robot motion planning. *In Proc. IEEE Internat. Conf. Robotics and Automation. (ICRA)*, volume 2, pages 46-51,1993.

[8] D. Halperin, L. E. Kavraki, and J.-C. Latombe. Robotics. *In J. Goodman and J. O'Rourke, editors, Discrete and Computational Geometry.,* pages 755-778. CRC Press, NY, 1997.

[9] L. Kavraki and J.-C. Latombe. Randomized preprocessing of configuration space for fast path planning. *In Proc. IEEE Internat. Conf. Robotics and Automation.,* pages 2138-2145, San Diego, CA, 1994.

[10] L. Kavraki and J. C. Latombe. Probabilistic roadmaps for robot path planning. *In K. G. and A. P. del Pobil, editor, Practical Motion Planning in Robotics: Current Approaches and Future Challenges*, pages 33-53. John Wiley, West Sussex, England, 1998.

[11] L. Kavraki, P. Svestka, J.C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *In Proc. IEEE Internat. Conf. Robot. Autoum,*. 12(4): 566-580, August 1996.

[12] J.J. Kuffner Jr. and S.M LaValle. RRT-connect: An efficient approachto single-query path planning. *In Proc. IEEE Internat. Conf. Robot. Autoum,*., 2000.

[13] D. Hsu, R. Kindel, J.C. Latombe, and S. Rock. Randomized Kynodinamic Motion Planning with Moving Obstacles. *In Algorithmic and Computational Robotic: New Directions*, B.R. Donald, K.K. Lynch, and D. Rus (eds.), AK Peters , Natik, MA, pages 246-254, 2001.

[14] M. Lin, D. Manocha, J. Cohem and S. Gottschalk. Collision detection: Algorithms and applications. *In Algorithms for Robotic Motion and Manipulation (WAFR96)*}, JP. Laumond and M. Overmars \(Eds\), AK Peters 1997.

[15] T. Lozano-Pérez. Spatial planning: a configuration space approach. *IEEE Tr. On Computers,* 32:108-120, 1983.

[16] M. Overmars and P. Svestka. A probabilistic learning approach to motion planning. *In Proc. Workshop on Algorithmic Foundations of Robotics*., pages 19-37 1994.

[17] S. A. Wilmarth, N. M. Amato, and P. F. Stiller. Maprm: A probabilistic roadmap planner with sampling on the medial axis of the freespace. *In Proc. IEEE Int. Conf. Robot. and Autom.*, Detroit, MI, 1999.