

Robot Motion Planning

Ganesh Swaminathan
School of Engineering Science
Simon Fraser University, Canada

gswamina@sfu.ca

February 27, 2006

1 Introduction

Motion planning is a fundamental problem in robotics. The classic path planning problem is described as the following: given a three dimensional rigid body and a known set of obstacles, the task is to find a collision-free path from a start configuration to a goal configuration. Additionally, this task is to be completed in a reasonable amount of time. This is known as the *piano mover's problem*[1]. A more general scenario, known as the *generalized mover's problem* would have the robot as flexible polyhedra with moving polyhedral obstacles.

This report will review classical motion planning paradigms - potential field methods, roadmap methods such as visibility graph methods and probabilistic roadmaps. More emphasis will be placed on roadmaps as this technique has shown promising results for complicated robot planning problems.

Actions in the physical world are subject to physical laws, uncertainty and geometric constraints. The design and analysis of motion planning algorithms raises questions in a number of fields - mechanics, control theory, computational and differential geometry, and computer science. This report will only consider the simplified geometric problem and as such ignore issues such as incomplete information, nonholonomic constraints, moving and deformable objects, multiple robots and control related issues.

1.1 Mathematical definitions

Before we can go on with the review on motion planning, it is instructional to establish a uniform mathematic notation. The notation followed in this report is the same as in [2].

The configuration space \mathbb{Q} is defined as the set of all configurations a robot can achieve. This space is generally non-Euclidean. The dimension of this space is equal to the number of degrees of freedom in the robot. For example, a rigid body in three dimensions has six degrees of freedom - three for the position (x,y,z) and three for the orientation (roll-yaw-pitch). \mathbb{Q}_{free} is defined as the set of robot configurations that do not intersect an obstacle. A particular configuration from this space is defined as q .

2 Potential Field Method

Potential field methods were first described by Khatib[3] for on-line collision avoidance for a robot with proximity sensors. This approach reformulate the motion planning problem as a numerical one. This method makes use of a force field with parameters for the robot, the goal and obstacles. The robot

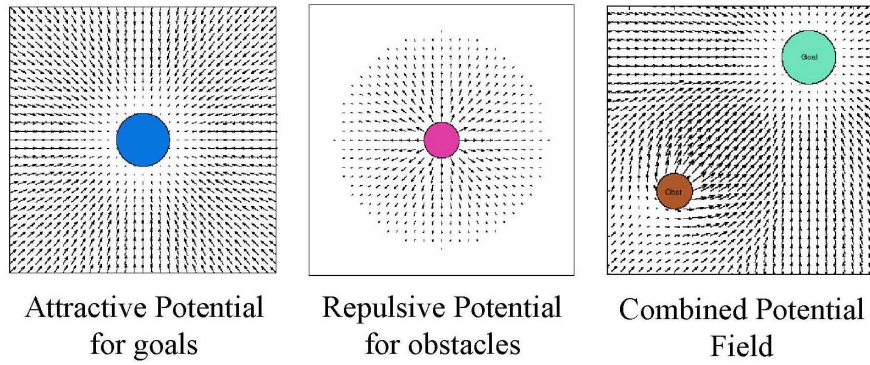


Figure 1: Typical potential fields.

and obstacles are parameterized as positive charges and the goal as a negative charge.

The planning can be thought of as the robot navigating this scalar, artificial potential field using repulsive interactions with obstacles (like charges) and attractive interactions with the goal. The total interaction is the sum of the two interactions.

$$U(q) = U_{attr}(q) + U_{repul}(q)$$

Each step is taken along the negative gradient of the potential. The robot terminates the motion when it reaches a point where the gradient is zero, signifying a minima. Gradients also become zero at saddle points, but these points are generally unstable and a slight perturbation will “release” it from this point.

Ideal force fields are smooth, have a single global minima at the goal, no local minimas and grows to infinity at the obstacles. It is also key that the force field is stable at the goal, otherwise it will lead to “rocking” motion at the goal.

Disadvantages are that failure is possible even when a valid path exists. These failure modes are visible to bystanders. Furthermore, this method stops when the gradient is zero and this could happen at local minimas and saddle points. It is hard to come up with a potential field with no local minima at all.

Randomized potential field method[4], proposed by Barraquand and Latombe for path planning can be applied to robots with many degrees of freedom. The method attempts to escape local minima by random walks.

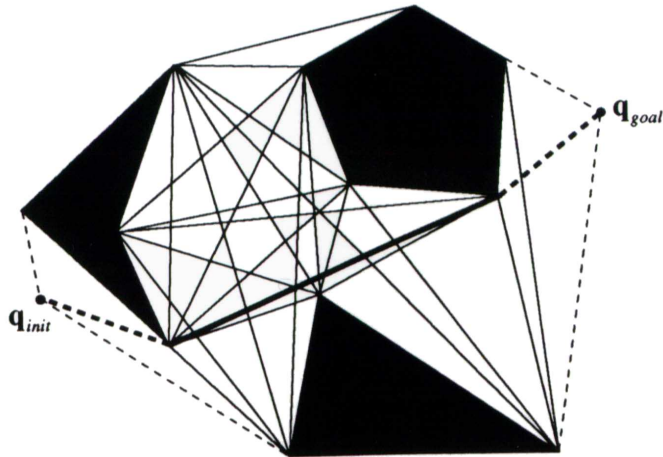


Figure 2: Visibility graph.

3 Roadmap Methods

A map is a data structure used to plan subsequent paths more quickly. The data structure tries to capture the connectivity and features of the configuration space of the robot. Using the map, a planner can find a path between any two configurations by first finding a collision free path from one of the configurations to the roadmap and likewise from the roadmap to the destination configuration.

3.1 Visibility Graphs

First explored by NJ Nilsson in 1969, visibility maps apply to C-space with polyhedral obstacles. Nodes of the map are the vertices of the polygon and two nodes for the visibility graph share an edge if their corresponding vertices are within line of sight of each other. The line of sight might come from a sonar reading, for example.

Visibility graph methods are complete. They are conceptually simple, but are really only suitable for two-dimensional C-space.

3.2 Probabilistic Roadmaps

When the number of degrees of freedom of the robot is moderately large, exact path planning approaches fail to work. Generalized mover's problem in which

the robot is a collection of polyhedra freely linked together at various vertices was proven to be PSPACE hard by Reif[5]. So, one has to resort to heuristic approaches to solving the problem. One such method is the probabilistic roadmap method (PRM). The premise behind this approach is that it is cheap to check if a single robot configuration q is in \mathbb{Q}_{free} or not.

The data structure used is a unidirectional graph $G(V, E)$, with vertices V and edges E . The basic PRM method can be broken down into two stages - a construction stage and a query stage.

3.2.1 Roadmap construction

Initially, the graph G is empty. The method begins by sampling the configuration space of the robot. If these sample points q are in \mathbb{Q}_{free} , they are added as nodes to G . Many sampling strategies are possible, but for simplicity assume that the configurations are sampled from a uniform distribution.

The nodes are connected by a local planner Δ . $\Delta(q, q')$ returns true if there is a collision-free path between q and q' , else it returns NIL. The simplest planner uses a straight line in C-space to create a path. Collision-free paths are added as edges to G .

It is not efficient to try to connect each node with every other node in the graph, so only the k closest nodes are sent to Δ . The closest nodes are computed by a distance metric $dist$, which for simplicity can be assumed to be the Euclidean distance. More complicated data structures can be used to speed this step up, including the use of kd-trees. It is also possible to make use of approximate distance computations to connect nodes.

The uniform distribution works quite well for a variety of problems, but when a path exists through narrow passages, more elaborate schemes such as OBPRM[6], medial axis bias methods[7] and bridge tests can be used[8].

Once this stage is completed, the graph G represents the connectivity of \mathbb{Q}_{free} .

3.2.2 Query phase

The roadmap constructed in the preceding section can be queried for a path between two arbitrary configurations q_{init} and q_{goal} . It might be possible that multiple paths exist between the two configurations - then, in that case, the shortest path might be selected according to Dijkstra's algorithm or the A^* algorithm. It is also likely that the graph G does not represent the connectivity of \mathbb{Q}_{free} that well or no path exists between the two configurations. In that case, we return to stage one and more nodes are added to the graph.

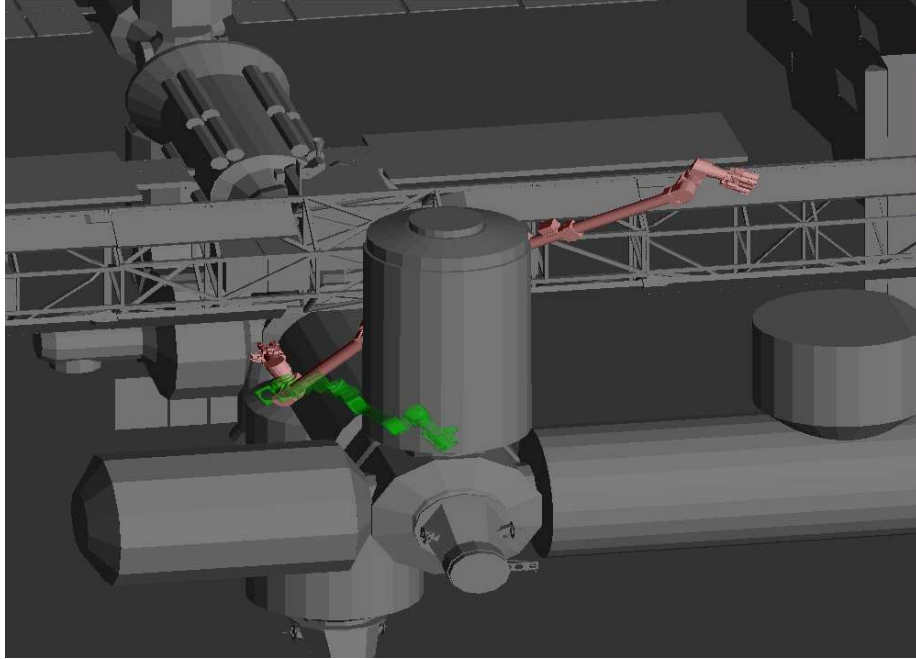


Figure 3: Robot at the International Space Station.

3.2.3 Analysis

Complete motion planning techniques such as cell decomposition that depend on explicit representation of \mathbb{Q}_{free} do not scale as the degrees of freedom of the robot increases. PRM on the other hand sacrifices completeness for speed and it can be described as probabilistically complete. The probability of finding a path approaches 1, if such a path exists, as the running time tends to infinity.

The probability of finding a path (assuming such a path exists) depends on three factors - the length of the known path, the closeness of the path to obstacles and the number of nodes in the graph.

$$Pr_{failure}[(q, q')] \sim e^{-N}$$

The book[2] has a proof of the above statement. In brief, the probability of failure drops exponentially to zero with the number of nodes.

In practice, PRM has successfully solved many complex problems. They are easy to implement and fast. The drawbacks of PRM are that the probabilistic nature of the algorithm leads to large standard deviations in planning time. They is also no termination criteria when a solution cannot be found.

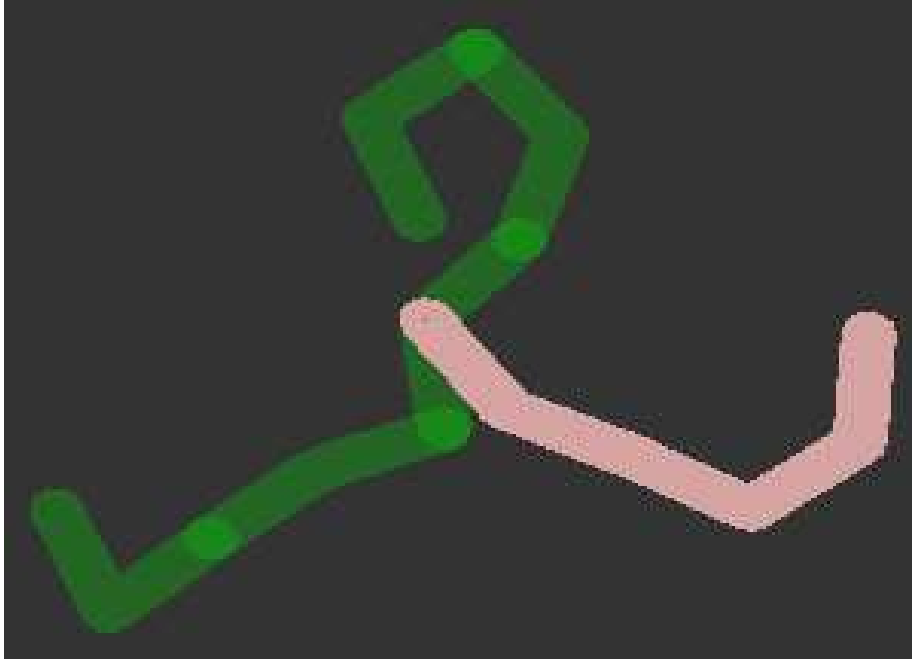


Figure 4: A 5-dof robot in MPK.

4 Motion Planning Kernel

Motion Planning Kernel (MPK) is a feature-rich software simulation system designed at the Computational Robotics Lab at Simon Fraser University[9]. It comes with a full suite of collision detection algorithms (V-collide[10] and SOLID[11] amongst others) and exact and probabilistically-complete motion planning schemes including ACA, RRT, PRM. Collision detection packages such as SWIFT++[12] also compute exact and approximate distances. It also comes with a wide variety of robots, including some sophisticated ones like the seven-DOF robot at the International Space Station in figure 3. MPK also allows the user to add arbitrary polygonal obstacles.

MPK was designed with extensibility in mind. The shortcoming with commercial packages is that the user has little, if not, no control over the underlying algorithms. MPK on the other hand allows the user to hand pick any collision or planning scheme of choice, or even extend existing routines to suit their needs.

The biggest advantage of MPK is the framework that it provides. This gives access to previously created libraries of robots and working environments. Under this framework, it is also possible to test new collision detection algorithms apart from planners. This simplifies the testing and benchmarking of any new

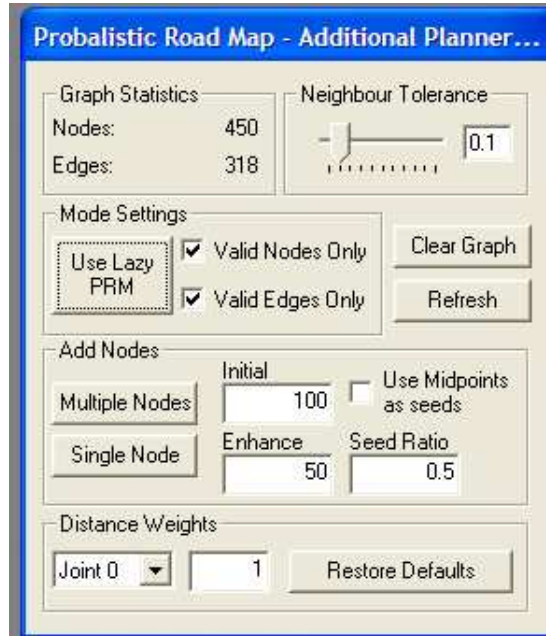


Figure 5: More control over PRM.

algorithms implemented. MPK is implemented in standard C++ with an object oriented design.

Figure 4 shows a 5-DOF robot that was used to play with the different modules available in MPK. The software allows the user to set the initial and goal configurations (shown in green), the planner, and the collision detection algorithm. Some planners open up a window for more control such as the one in figure 5 for PRM.

5 Acknowledgments

I would like to take this opportunity to thank my supervisor Dr. Kamal Gupta for his advice and discussions. I would also like to thank his graduate students in his robotics lab - specially Peng Peng Wang and Zhenwang Yao for some very interesting conversations in the hallway.

References

- [1] J. T. Schwartz and M. Sharir. On the piano mover's problem: II. General techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*, 4:298-351, 1983.
- [2] H. Choset, K.M. Lynch, S. Hutchinson, G.A. Kantor, W. Burgard, L.E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*, MIT Press, Cambridge, MA, June, 2005, p. 603.
- [3] Khatib, O., Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *1985 IEEE International Conference on Robotics and Automation*, 5(1), 90-8.
- [4] J. Barraquand and J.-C. Latombe. A Monte-Carlo algorithm for path planning with many degrees of freedom. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1712-1717, 1990.
- [5] J. Reif. Complexity of the mover's problem and generalization. In *Proceedings of 20th IEEE Symposium on Foundations of Computer Science*, pages 421-427, 1979.
- [6] OBPRM: An Obstacle-Based PRM for 3D Workspaces, Nancy M. Amato, O. Burchan Bayazit, Lucia K. Dale, Christopher Jones, Daniel Vallejo, In *Proc. Int. Wkshp. on Alg. Found. of Rob. (WAFR)*, pp. 155-168, Houston, TX, Mar 1998.
- [7] C. Holleman and L. E. Kavraki. A framework for using the workspace medial axis in PRM planners. In *Proceedings of the International Conference on Robotics and Automation*, pages 1408-1413, 2000.
- [8] D. Hsu, T. Jiang, J. Reif, Z. Sun. The Bridge Test for Sampling Narrow passages with Probabilistic Roadmap Planners. *IEEE International Conference on Robotics and Automation*, 2003.
- [9] I. Gipson, K. Gupta, and M. Greenspan. MPK: An open extensible motion planning kernel. *Journal of Robotic Systems*, 18(8):433-443, Aug. 2001.
- [10] Thomas C. Hudson, Ming C. Lin, Jonathan Cohen, Stefan Gottschalk, and Dinesh Manocha. V-COLLIDE: accelerated collision detection for VRML. In *VRML '97: Proc. of the Sec. Symposium on Virtual Reality Modeling Language*, pg. 117-123, 1997.
- [11] G. van den Bergen. Efficient Collision Detection of Complex Deformable Models using AABB Trees. *Journal of Graphics Tools*, 2(4):1-13 (1997)
- [12] S. Ehmann and M. Lin. Accurate and fast proximity queries between polyhedra using surface decomposition. *Computer Graphics Forum (Proceedings of EUROGRAPHS)*, 2001.