Sampling Based Planning Techniques

Edward Derek Lambert

March 27, 2019

1 Introduction

Geometric path planning or the piano mover's problem consists of finding a sequence of configurations between an origin and a destination within a field of obstacles. The problem is known to be PSPACE hard, that is to require at least a polynomial amount of memory to solve. It has been a topic of active research in robotics and animation for the last several decades and many practical solutions have been developed, especially for the case of a two dimensional workspace with polynomial obstacles. An overview is given in Handbook of Robotics, Part A: Robotic Foundations - Motion Planning [1].

Typically approaches make some approximation to make the problem tractable, at the cost of some aspect of global optimality of the solution. Approaches may be divided roughly into Mathematical Programming based and Sampling Methods, although there are others such as Potential Field methods and many combinations.

Sampling methods for motion planning can be divided into deterministic and randomized sampling. Deterministic samplers subdivide state space into a uniform lattice such as a grid or a more exotic symmetrical structure designed to span the search space effectively. Randomized sampling methods fall into two main categories: Rapidly Exploring Trees (RRTs) [2] and Probabilistic Roadmaps (PRM). Probabilistic roadmaps are typically used for static environments, where it is advantageous to make multiple queries on the same roadmap, for a robot performing different taks within the same environment. This is because it is time consuming to produce a new roadmap but quick to make additional queries on it. RRTs are more useful when the environment is changing rapidly such as a robot exploring new areas. The tree is quick to construct but there is little advantage to making subsequent queries.

2 State Lattice Planners

Sampling from the state space in a regular lattice to create a graph for efficient searching is one of the first practical approaches to path planning for mobile robots. This is because the computational complexity was low enough for real-time systems provided certain assumptions held. Typically the lattice took the form of a regular grid, either four connected or eight connected by straight lines. Each node represented the x-y state of a holonomic robot, able to move in any workspace direction freely. The holonomicity

Given:

1. A workspace $\mathbf{W}$ is either in R^2 or R^3
2. An obstacle region $\mathbf{O}$
3. A robot defined in $\mathbf{W}$ consisting of one or more rigid bodies
4. A configuration space $\mathbf{C}$ comprising $\mathbf{C_{obs}}$ and $\mathbf{C_{free}}$
5. An initial configuration $q_I \in \mathbf{C_{free}}$
6. A goal configuration $q_G \in \mathbf{C_{free}}$

Compute a continuous path $\tau : [0, 1] \rightarrow C_{free}$ such that $\tau(0) = \mathbf{q_I}$ and $\tau(1) = \mathbf{q_G}$

Figure 1: The piano mover's problem - geometric path planning.[1]

assumption allows many interesting path finding behaviours to be studied, but typically does not hold for practical systems like cars or fork lift trucks with Ackerman steering which are subject to differential constraints.

2.1 Smoothing

One way of incorporating differential constraints into the lattice planning approach is to fit a smooth curve which respects the constraints such as a cubic spline as closely as possible to use nodes returned by the planner. The difficulty here is that the smooth curve does not exactly follow the edges in the graph which were used to calculate the route cost and check for obstacle intersection. If the cost of the smooth path is different to that of the approximation used in the lattice, any optimality guarantees provided by the planning method such as Dikstras algorithm are compromised. Worse, the smooth path must be checked against the obstacle map for collisions, and it is not clear how the path should best be modified if any collisions are detected. For an example of smoothing with clothoid curves see [3] [4].

2.2 Differental Constraints

It is possible to carefully construct a lattice which captures differential constraints as detailed by Pivtoraiko et al [5].The trick is calculating the control inputs required to join a set of vertices which span state space (inverse kinematics) or sampling from the control space in order to generate a set of vertices (forward kinematics). Both methods have complications, for example the shape of the lattice must be known ahead of time to be reachable with a simple set of primitives for the inverse method. For the forward method the difficulty is in the choice of interval in control space that leads to complete and uniform coverage of state space. These difficulties are resolved for one example configuration in [5] with 16 discrete headings, 8 levels of curvature and a total of 192

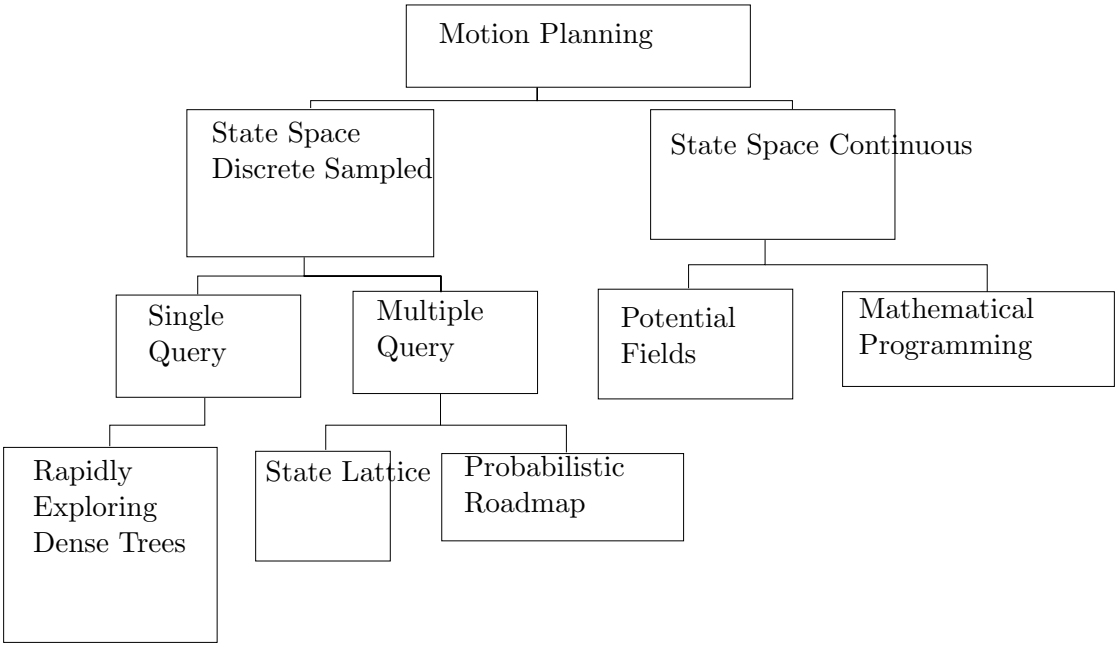

Figure 2: Hierarchy of planning approaches. See 'Handbook of Robotics, Part A: Robotic Foundations - Motion Planning' [1] for more details.

controls. The balance to be struck in the choice of resolution is that higher resolutions lead to a higher branching factor, increasing the memory required to represent a given area, but lower resolutions will give suboptimal paths. Imagine following a straight wall which does not align with one of the 16 cardinal directions in the lattice, but halfway between. The planner with return an optimal path which alternates between the two closest cardinal directions, rather than a straight line aligned with the wall. With a back of the envelope calculation, increasing the resolution to 10mrad to minimize artefacts increases the number of nodes required per square metre to exceed the number of atoms in the known universe. This trade off is the biggest obstacle to the use of lattice planners for general path planners and led to the development of randomized planners, which can give satisfactory uniformity and coverage of high dimensional state space without the direct link between the branching factor and the resolution when constructing a lattice.

3 Probabilistic Roadmaps

Probabilistic Roadmaps are an example of a multiple-query sampling based planner. They are probabilistic in the sense they sample randomly from $\mathbf{W}$ to build up a connectivity graph - the roadmap. They do not inherently cope with uncertainty in the obstacle field. One important component is a local planner, which is able to generate a path between two nearby configurations and test if it intersects with any obstacles. The algorithm proceeds by testing each sample by creating a local path from the nearest point

on the existing tree, and only adding the new point to the graph is the path is obstacle free. Depending on the local planner it is possible to incorporate smooth paths respecting vehicle dynamic limits, but other methods are more appropriate in cases where the obstacle field is dynamic because the simplifying assumption making roadmaps so fast is that of a static environment. Another important component is the graph method used to select the shortest path such as Dijkstra which is used to find the shortest path once the roadmap is constructed. There also must be some heuristic guiding the selection of new points.

The distinction between PRMs and lattice planners is only in the mechanism by which samples are drawn from state space to construct the roadmap [1]. They are both Roadmap planners. Either a repeating pattern or lattice is overlaid or states are selected randomly within an area of interest and discarded if they are not reachable according to the obstacle map and the local planner. This is based on the idea that randiom samples of high enough density will provide unbiased and complete coverage of the space. Particularly in the limit as the number of samples tends to infinity, a PRM may be 'probabilistically complete' so that as the number of samples tends to infinity the probability of finding a solution if one exists tends to one.

The modifications to both types of Roadmap planners which deal with a changing environment and differential constraints on motion are very similar. Differential constraints can be incorporated into the local planner as described in 2, althtough there are fewer constraints on PRM as there is no need to use kinmatics to ensure the samples form a regular lattice.

3.1 Uncertainty

Both types of roadmap planner are useful for dealing with uncertain representations of the world where the environment is represented as an occupancy grid, a grid of values covering the space, each value representing the occupancy probability of that cell. The best path can be found directly from this representation by minimising the sum of occupancy probability of every cell traversed by the path. This may be more useful than the 'hard constraints' offered by optimization methods, because although the solution will not violate the constraints, the constraints themselves are constructed by throwing away information about the uncertainty of the environment to create a binary representation where every position in space is either inside or outside polygonal obstacles [5].

4 Rapidly Exploring Random Trees

Rapidly Exploring Random Trees are an example of a single query planner introduced by LaValle and Kuffner [2]. Most simply, a tree is constructed by sampling from state space close to an existing node in the tree. A local planner is used to generate a trajectory from the existing node to the new sample avoiding obstacle regions if one exists. If an obstacle free trajectory is found, a new node is added at the sample state with an edge

from the existing node. Nearby nodes in the tree should also be checked and connected with edges if posible.

Starting with the initial position and the goal position, this process is repeated until sufficient coverage of the state space is obtained. When a new sample is reachable from both trees, origin and destination must be connected and a graph method such as Dijkstra can be used to identify the vertices through the graph which make up the shortest path between them. The tree growing phase of the algorithm can be terminated at this point but it may be possible to improve the quality of the solution by adding more reachable states to the connected graph in case a better solution can be found.

References

[1] Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*. Springer Handbooks. Springer, Berlin, 2 edition, 2016.

[2] Steven M LaValle. RRT-progress and prospects. *Algorithmic and Computational Robotics: New Directions*, 91:399–404, 2001.

[3] Marcus Lundberg. *Path planning for autonomous vehicles using clothoid based smoothing of A * generated paths and optimal control*. PhD thesis, KTH Royal Institute of Technology, 2017.

[4] D. J. Walton and D. S. Meek. A controlled clothoid spline. *Computers and Graphics (Pergamon)*, 29(3):353–363, 2005.

[5] Mikhail Pivtoraiko, Ross A. Knepper, and Alonzo Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009.