Roman Yakobnyuk (s4375374)

COMP7702 - Artificial Intelligence

Lecturer: Hanna Kurniawati

09 September 2015

ABSTRACT

This paper describes an approach to finding a collision-free path in a multidimensional configuration space of the N joint robot, where 2 + N would represent number of dimensions for the configuration space of the robot that is required to search for a valid solution. Method that is used to solve this task is a version of rapidly exploring random tree algorithm that is designed to efficiently search high-dimensional spaces by randomly building a space-filling tree. This tree is constructed incrementally from samples drawn randomly from the search space and is inherently biased to grow towards large unsearched areas of the problem. In the presented version of the algorithm, 1% probability bias towards to goal configuration is applied to make algorithm search more greedily towards the goal.

Works Cited

LaValle, Steven M. (2006). "Planning Algorithms" (Website).(http://planning.cs.uiuc.edu/node236.html)

PROBLEM FORMULATION

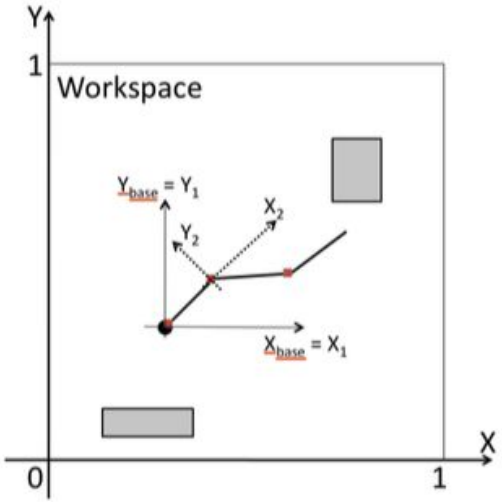


Figure 1. Joint robot in normalised 2D space.

The problem can be described as a movement path search for a joint robot in a 2D space as shown by figure 1, which is normalised to be in range $[0, 1] \times [0, 1] \subset R2$ for x and y. Every joint has local coordinate system attached and can move its link between -150º and 150º degrees in that coordinate system that coincides previous link by X axis. This space contains obstacles, forbidden region for a robot, and no configuration in his path from initial to the goal configuration should collide with this forbidden region. Input information for search algorithm is an initial and goal configurations of this robot, as well as configuration of obstacles in its workspace. Path from initial to goal path is considered valid if maximum distance from one configuration to next one does not exceed 0.001 unit of distance and every line does not move more than 0.1º .

METHOD DESCRIPTION

Method used for solving that problem is an algorithm called rapidly exploring random tree algorithms. In order to represent configuration space of N-joint robot, we use N+2 values of type Double. Distance between two configurations is calculated using Euclidian Distance formula.

Rapidly Exploring Random Tree: Initial configuration of the robot becomes first node in the space-filling tree built by the algorithm, after that following steps repeated in order to find a path to goal configuration:

(1) Pick random point from the C-space, using uniform randomisation;

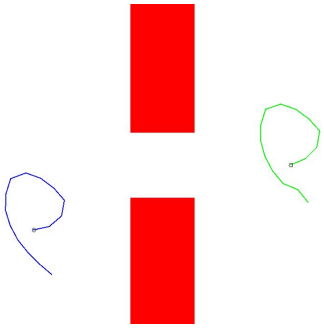
(2) Find nearest node to that random point from existing tree;

(3) If collision-free path is available from nearest tree node to random node - add it to the tree; or (4) find closest collision-free point on the line between nearest and random configurations, then add it to the tree.

(5) If collision-free path is available from new tree node to the goal node - search is done, collision-free path from initial to goal configuration is found.

With uniform sampling of the search space, the probability of expanding an existing state is proportional to the size of its Voronoi region. As the largest Voronoi regions belong to the states on the frontier of the search, this means that the tree preferentially expands towards large unsearched areas. RRT growth can be biased by increasing the probability of sampling states from a specific area. Presented implementation uses 1% probability of sampling goal configuration instead of random one in order to make algorithm search more greedily towards the goal. This particular probability was chosen because it is claimed to perform well experimentally (Steven M. LaValle, 2006).

EXPERIMENTAL ARGUMENTS



Due to its random sampling nature, performance of the algorithm can vary greatly for the multiple executions on the same input, which makes it difficult to properly measure and compare performance of the algorithm on different problems. However, by taking multiple executions information some important assumptions on the algorithm performance can be made.

Figure 2. 15_joints_2.txt example.

This two examples show how algorithms behaves with different amount of different valid path available. In practice, it takes RTT implementation around 3-5 seconds to solve 17-dimensional problem with a wide free area, comparing to 40-50 seconds for 12-dimensional problem with narrow path available for sampling.

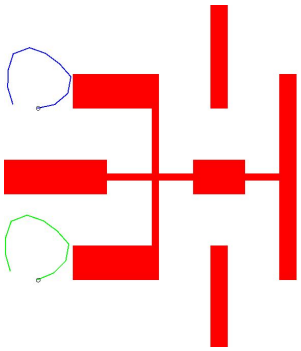


Figure 3. 10_joints_11 example

WHY THIS SOLUTION

Rapidly exploring random tree algorithm performs well when it is required to search through multidimensional space, because it is not required to discretize space before running it, nor it requires any complicated heuristics to achieve desired result. It is especially useful because it does not require to compute geometry of the obstacles in the workspace, if simple collision checking can be made. That allows for a better performance comparing to other solutions.