”Indian Institute of Technology Kanpur”

Motion Planning for Redundant Manipulators

Devvarat Meena (Y9197)

April 14, 2012

CS365: Artificial Intelligence

Supervisor:
Prof. Amitabha Mukerjee
Department of Computer Science and Engineering
IIT Kanpur

ABSTRACT

Robotics is the most attractive area for scientists and engineers these days. Robotics has potential application in almost every area in human life such like manufacture automation, unknown area exploration, human search and rescue in catastrophe event and handicapped people assistance. In a robot system, manipulator is one of the most important compositions of a robot which can achieve many humanoid function, for example, object grasping and delivering. There are many kinds of manipulator. Classification on basis of DOF:- 1. Non-redundant manipulators, 2. Redundant manipulator Because the redundant manipulator has more DOF than necessary for position and orientation computation, for a specific position and orientation of end effector, multiple solution exist. For this reason, the redundant manipulator has much more advantages in joint limit constraint avoidance, obstacle avoidance and singularity avoidance compared to the non-redundant manipulator. Our work is to manipulate a redundant manipulator and analyze the computational defficulties in manipulating high DOF redundant manipulators. We are using Probablistic Roadmaps(PRM) to make a graph of obstacle space. We are using Dijkstra's Algorithm to calculate the shortest path from source to destination. We will be manipulating a 2 DOF arm robot in a 2-D space with obstacle.

Introduction

The inverse kinematic problem of a redundant manipulator a difficult problem in robotics. It attracted lots of attention in computer science and mechanical science. These problems are highly non-linear and there exists multiple solutions. Firstly redundant manipulators are the manipulators which have degree of freedom more than it need to perform the given task. So due to their high DOF they can perform complex motions. But that's also the reason that they are difficult to manipulate. In this paper we address the limitation of inverse kinematics and will implement probablistic roadmaps (PRM) from the start. In inverse kinematic approach we do know the final position of the end-effector and from that we calculate the possible joint angle configurations for the manipulator. Solutions obtained by inverse kinematics for 2 DOF robot arm 2 and for 3 and higher DOF robot arms infinite

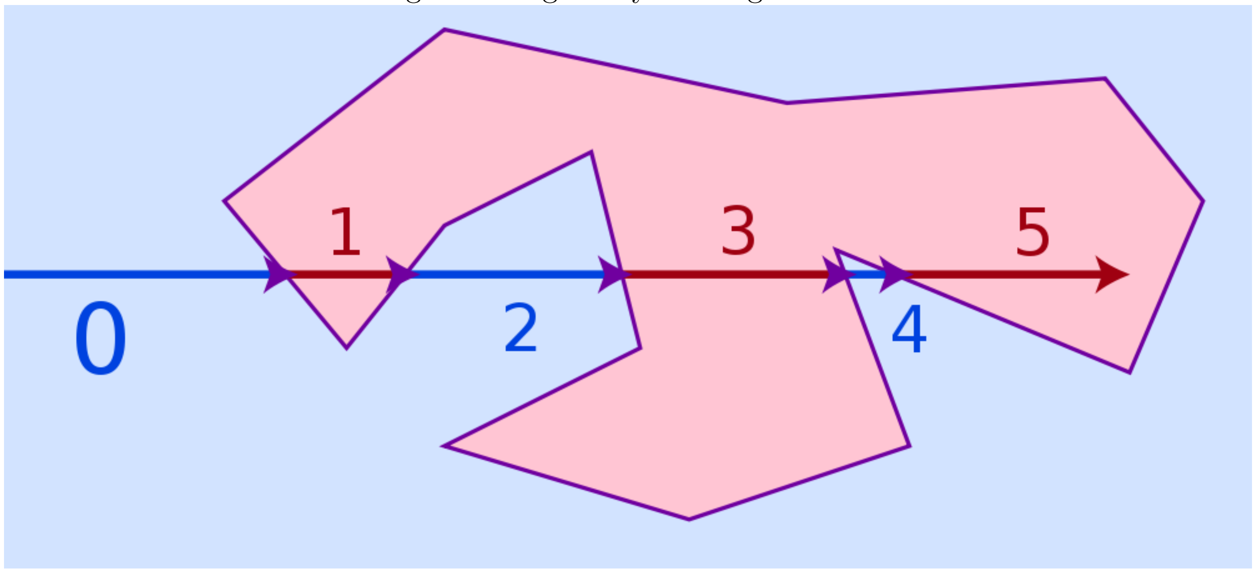
solutions exists in 2D space. The methods available for solving these problems may be broadly grouped into two major classesclassical techniques based on the differential kinematic relationship between the end-effector velocities and the corresponding joint angle velocities, and learning based techniques based on biologically inspired approaches like neural networks, fuzzy logic, machine learning etc.
So precisely we will manipulate a 2 DOF arm in a 2D space with obstacles from source point to destination point. To determine the path for end-effctor to move from start to end point probablistic roadmaps are used. In PRM points are generated randomly in robot free space then the graph is created by connecting those points. Applying Dijkstra on that graph we can determine shortest path form start to end point. This path will be used by end-effector to move. For every position of end-effector inverse kinematics is used to calculate the joint angle configurations for the robot. Since it is a 2 DOF robot arm we can use inverse kinematic. To avoid obstacles we are using a simple method. We are checking if for any joint angle configuration any arm segment is intersecting any obstacle line or not. We will be discussing these topics sections below.

Graph generation using Probablistic Roadmaps (PRM)

The probabilistic roadmap planner is a motion planning algorithm. It solves the problem of determining a path between a starting configuration of the robot and a goal configuration while avoiding collisions. The basic idea behind PRM is to take random samples from the configuration space of the robot, check them for whether they are in the free space, and use a local planner to attempt to connect these configurations to other neighbouring configurations. Than Starting and goal configurations are added to these configurations and than apply the path planning algorithm to the graph to determine the path between the start and goal configurations.
The robot is a arm of two links first of length 10 and second is of lenght 7. So maxreach of the arm robot is 17 and minreach is 3 so our arena is a circle of radius 17. We are generating random 5000 points in this area. Than using Point-in-a-polygon we check that the generated point is inside the obstacle or not. If the point is found to be inside the obstacle than we discard the point otherwise we save the point to make Graph. The algorithm we are using is

called *Ray Casting Method*. The algorithm is based on the fact that if a point is inside any close body and is moving in a direction than it will cross the boundary of that body odd times. And if the point is outside than it will cross the boundary even times first time going inside second coming outside. After getting the points we add the given start and end point in these points. In order to make a graph we sort these points according to x coordinates and than connect each point to its neighboring points within a constant distance. That gives us a graph. After getting a graph we apply Dijkstra's Algorithm to determine shortest path from start point to goal point. We get the path as a list of points.

Figure 1: Fig-1 Ray Casting Method



Inverse Kinematics

Inverse kinematics refers to the use of the kinematics equations of a robot to determine the joint parameters that provide a desired position of the end-effector. Inverse kinematics transforms the motion plan into joint actuator trajectories for the robot. We are using inverse kinematic to determine joint angles theta1 and theta2 of 2 DOF robot arm. For 2 DOF arm we get two solutions for one end-effector position than we compair the solutions with previous joint angle configuraions. We choose the nearer configuration as the solution for successive steps on the path. But as the DOF increases it get difficult to find the successive step due to infinite solutions. This happens for DOF 3 and higher that's why due to high redundancy these manipulators are called redundant manipulators. In many other approaches we have gone through they were using inverse kinematics with learning or other approaches to get the desired joint configurations. But these approaches take long time to compute the right solution. In our case using Dijkstra's we know the path so we manipulate the end effctor on that path using inverse kinematics. In next section we will talk about how to avoid the obstacles while following the path.

Obstacle Avoidance

Obstacle avoidance is the most important task for a robot manipulator. To go from source to destination it has to detect the robot boundaries and than it has to avoid them. We in our manipulator are using the line segmant intersection method to detect the collision of robot with the obstacle. It is a simple yet effective mathod. We already calculated the points of the obstacle, so to check the collision we check that if any link of the manipulator intersect with any side of the obstacle. Intersection of lines depicts the collision of manipulator arm with obstacle.

Observations

In our approach we saw some drawbacks of our approach. They are following:-

- During PRM calculation for the arena we are generating points randomly through out the arena. So during randompoint generation we found that on some places the density of points was very high and on another places it was very poor. Because we are connecting one point to its neighbours which are within a fix range distance. So some time it so happens that we end up gating few disconnected graph in the arena. In that case our shortest path algorithm fails in calculating the shortest path.

- To avoid the above problem we have to generate more points which increases the caculation time by a huge factor and it reduces the efficiency of the algorithm.

- The effects of random point generation can clearly be seen by observing the shortest path which we are calculation. The path seems to be somewhat distracted from its destination.
- In 2 DOF robot motion which strictly follow the path can not avoid the obstacle if it appears between the path and origin.

Results

We are able to implement the following approaches:-

- We have implemented Probablistic Roadmaps (PRM) from the scrach.
- We are able to implement a 2 DOF arm manipulator which follow the path generated using PRM and Dijkstra's Algorithm.
- It is able to detect the obstacles in its way from start point to the goal point in the arena.

References

[1] David Eppstein, updated by Matteo Dell'Amico, *Priority dict: a priority queue with updatable priorities (Python recipe)Link*. 2002.

[2] David Eppstein, *Dijkstra's algorithm for shortest paths. Link*
UC Irvine, 4 April 2002.

[3] Amitabha Mukerjee, *2 DOF Arm Manipulator.* Department of Computer Science and Engineering, IIT Kanpur, 2011.

[4] Rami Al-Hmouz, Tauseef Gulrez, Adel Al-Jumaily, *Probabilistic Road Maps with Obstacle Avoidance in Cluttered Dynamic Environment* . University of Technology Sydney, Australia, 2004.