Instructional Suite for Motion Planning of Articulated Robot with multiple links and polygonal obstacle

Rajiv Krishna Omar
rajivk@iitk.ac.in
CSE, IIT Kanpur

Lalit Kumar
lalitkr@iitk.ac.in
CSE, IIT Kanpur

Amitabha Mukerjee
amit@cse.iitk.ac.in
CSE, IIT Kanpur

18 April 2013

Abstract

In this project we worked on building an instructional suite for robot motion planning of an articulated robot with multiple links in a environment with multiple polygonal obstacles. We built a GUI interface to interactively specify the robot, obstacles and starting/ending and showed the path computed using probabilistic roadmap on the GUI.

Introduction

Robot motion planning is a fundamental problem in robotics. The path planning problem is formulated as given a start and end configuration of a robot and a environment containing some obstacles find a path which avoids collisions with obstacles. An articulated robot is a particularly important robot having many rotary joints.
Among the most efficient methods available today, probabilistic roadmap (PRM) is a very good planner for robots with many DOFs and stationary obstacles.
The report includes the usage of the GUI with instructions. Skip to the section "how to use the GUI?" to learn its usage.

Configuration Space

The parameters of robot that uniquely determine the configuration of a robot are called generalized coordinates, and the vector space defined by these coordinates is called the configuration space.

Probabilistic Roadmap Algorithm

PRM is a sampling-based algorithm. Instead of obtaining free space in configuration space according to geometry of obstacles, algorithm creates random configuration spaces and check if it is free or not. Algorithm creates a probabilistic roadmap in the form of a undirected graph of configurations, which can later be used for querying. At query time a start and end configurations are provided and an optimal path from start to end is found in the roadmap. Algorithm finds a path from initial to goal after creating enough samples from configuration space.

Roadmap Construction

- Main learning phase
- PRM constructs random configurations/nodes
- Removes configurations which collides with obstacles
- Finds n collision free nodes
- Local Planner: Checks if two nodes are connected by a straight line not colliding with obstacles
- Gets k nearest non colliding neighbours with their distances using local planner
- Constructs a undirected weighted graph of these nodes having weights as distances

Choice of n, k :
Choice of n should be large enough to solve the problem. If workspace is full with complex obstacles and narrow passages, n is chosen large. If problem is easy to solve, n can be reduced to minimize complexity of calculations. My implementation, produces uniformly random configurations all over the configuration space.

Query Phase

In query phase, PRM finds a path between start and end. Nodes closest to start and end are found and an optimal path is found between them using a graph search algorithm.

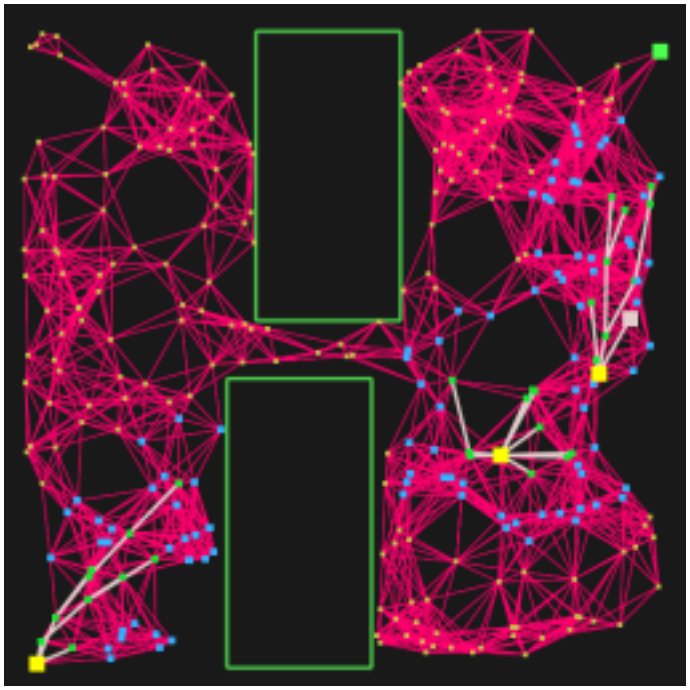

Figure 1: PRM at work

Motivation

Articulated robots are very useful in situations where humans cannot work. Articulated are used in maintenance of cooling pipes in a nuclear plant, point to point welding in car assembly(fig.2, ref [10]) and cleaning of airplane fuse lages. Since articulated robots are hyper-redundant, efficient motion planning is very difficult because there are infinite no of possible paths.

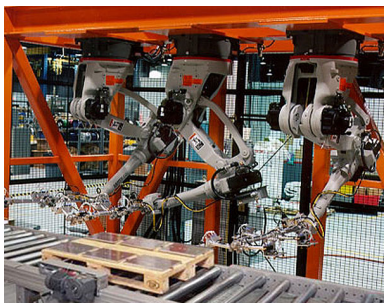

Figure 2: Articulated Assembly Robots

Previous Work

Reach Based Synthesis of Modern Hyper Reduntant Manipulaters by Kaushik Sinha, 2002
Hyper-reduntant Manipulators use large degrees of redundancy to improve the manipulator reachabilty performance in complex and cluttered workspace. In the synthesis problem, given n task space locations, the manipulator parameters need to be chosen such that it can reach all n goal locations without colliding with any obstacle and maintaining a reasonable quality of reach. In this work, he first define a modular design (r-SCARA) (see fig. below, ref [1]) for hyper-reduntant manipulators, where the degree of redundancy can be adjusted by adding or removing simple modules to the manipulators. The synthesis search space is then the set of all possible manipulator parameters, which is a function of the number of module variations.

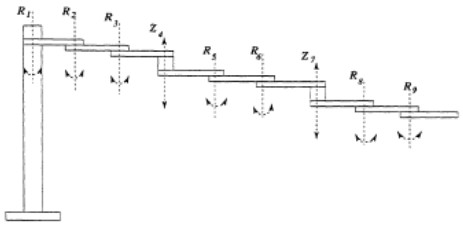

Figure 3: 7 link 9 *dof* ***r-SCARA*** arm

Figure 3: R-Scara robot model

Our Approach

There are two phases of this project

1. Front-end - GUI
2. Back-end - Path planning Algorithm

We have used C++ for both front-end as well as back-end phase. For the front-end we used Qt C++ API (Qt Creator IDE). The major classes that we used are QtCore, QtGui, QGraphicsScene and QGraphicsView. The ui was designed using Qt Designer which generates the C++ code from the interface prototype.

QGraphicsScene has no visual appearance of its own, it only manages the items. It is used together with QGraphicsView for visualizing graphical items, such as lines, rectangles, text, or even custom items, on a 2D surface The top left corner of QGraphicsView's viewport is always (0, 0), and the bottom right corner is always (viewport width, viewport height).

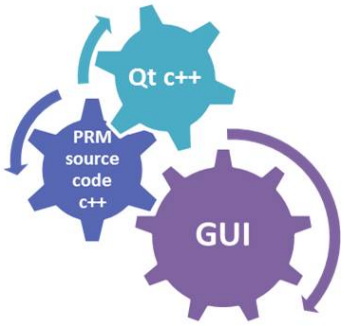

Figure 4: Modeling of Suite

Distance between two configurations

Since in our configuration space angles starts repeating after 360 i.e. angle 400 is same as 40 we cannot use Euclidean distance as a metric for distance computation. For our project we designed a Euclidean like distance metric. We used square root of sum squares of non-reflex angle distances between two angle vector as our distance in configuration space.

Collision Detection in PRM

Collision detection is the most expensive step in overall execution of probabilistic roadmap algorithm. For collision detection we divided the robot as well as obstacles into line segments and checked if any robot segment is intersecting with any obstacle by checking whether intersection point lies on both line segments.

Results

The lengths of robot links and sides of polygonal obstacles can be drawn using mouse. The start and end configurations can be specified in a separate file. A path is computed in CSPACE using PRM.This path is converted to coordinates using forward kinematics. The computed path is shown on GUI using push button to display successive paths. The problem as well as solution can be saved/loaded from a text file. A particular problem setup can be saved with its solutions and can be simply executed again without running the PRM. If the PRM is run again on the loaded setup the new solutions can be saved to a new file. The results are computed almost instantaneously so that live interaction is maintained. The animation is currently achieved by push buttons to display the next or previous intermediate configuration in the solution set of the robot.

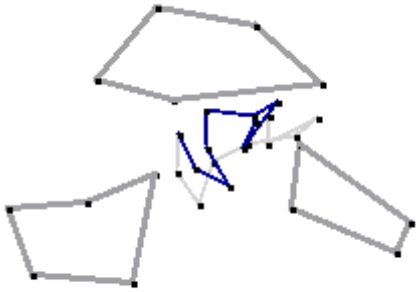

Figure 5: Intermediate configuration i of robot

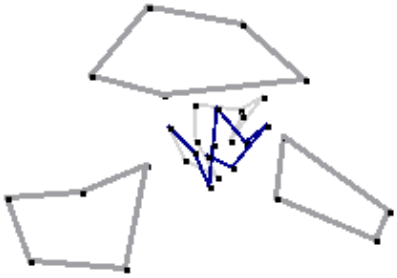

Figure 6: Intermediate configuration i+1 of robot

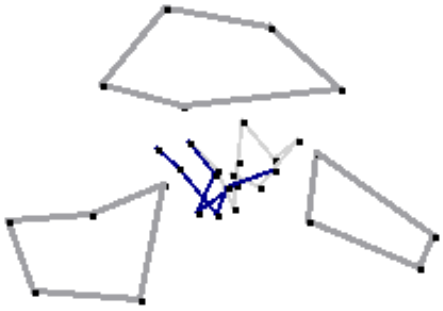

Figure 7: Intermediate configuration i+2 of robot

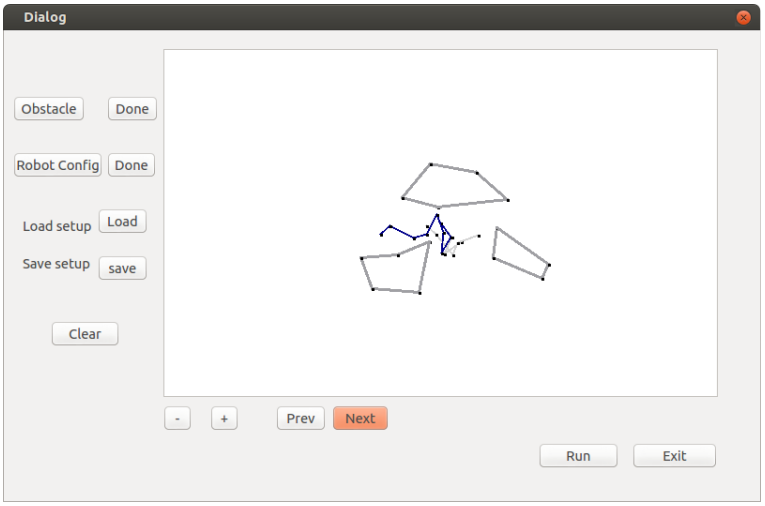

Figure 8: GUI showing solution to a problem

In the animation the two consecutive configurations are shown at a time The current configuration is highlighted with dark blue color where as the previous configuration is highlighted with light gray color. The output of the animation can be changed to the cluttering of configurations where all intermediate configurations are shown.

Conclusion

No. of Sample points needed for approximating optimal path increases exponentially with dimensions.
The GUI is available for use at
link: http://home.iitk.ac.in/ lalitkr/cs365/projects or
http://home.iitk.ac.in/ rajivk/cs365/projects.

Both windows and linux executables of the GUI can be downloaded from the site. All the essential dll files are zipped with the executable so you don't have to worry about the external support for compiler or other dlls. The zipped folder also contains a config file which is used to give the start and end configurations to the PRM. These start and end vectors' length is limited to 15 as the robot can have maximum of 15 dof. For a dof n<15 the input vectors n values can be given in Θ degrees or can be appended by 0's(vector length<=15). Edit the config file to give any start and end configuration and make sure the Θ are valid inputs.

How to use the GUI?

To use the GUI download the zipped files from the link mentioned in conclusion section. For Windows users download the Windows exe and for Linux users download the Linux exe. Save it to a location unzip the file. The directory contains the Line.exe (for windows users) or Line (for Linux users). The config file is in txt format which can be editted even during run time.
The GUI has pushbutton funtions.The graphics view contains a black dot in the middle which is used as the pivot for articulated robot. The coordintes of the pivot are (150,150) and the top left corner is the origin of the coordinate system (0,0).

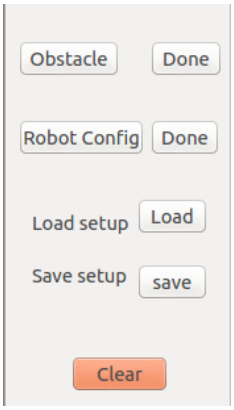

Figure 9: Control Panel with pushbuttons of GUI

In the above figure the side panel of GUI is shown which contains push buttons to enable drawing figures on the graphics view. The "Robot config" button enables the user to add a robot by clicking anywhere on the graphics view. The pivot is fixed to (150,150) on every mouse click an arm is added to the robot at its end link. The length and orientaion of the arm depends upon the distance and and angle of the end point of last arm of robot and the recent coordinate chosen on mouse click. When done with drawing robot the user is expected to click "Done" beside the "Robot cofig" pushbutton.
Similarly the obstacle can be added anywhere on the graphics view after first enabling the "Obstacle" pushbutton. The user is expected to draw n-1 sides of the polygon the nth side is completed when "Done" button is clicked.

Click "save" to save any problem setup in a new file or in a existing file. If the "save" is clicked after "run" then the solutions will also be saved in the file. To load an existing problem click "load". The setup will be shown on the Graphics View. If the setup has solutions saved with it you can see the solution by clicking "next", or you can again run PRM on it to get new set of solutions. You can even add new obstacle to an existing problem and open config file and change output at runtime. The "clear" pushbutton clears the screen.

Future Work

- One obvious extension would be incorporating other motion planning algorithms such as RRT using same interface. Since our code is modular, well commented and easy to understand adding RRT will involve adding a separate class.
- The GUI can be improved by incorporating features like dragging and copying of obstacles and Robots having finite width of segments
- We implemented an extra feature of generating random obstacles on GUI (by simply clicking a pushbutton) it generated regular polygons (no. of sides=4), which we removed from it as it was too much random. This feature can be re-incorporated with some care.
- Another dimension one can take is having multiple robots in the scene. This scenario will come with additional complexity of robot-robot interaction.

Acknowledgements

We thank Prof Amitabh Mukerjee for mentoring this project. We would like to thank Saurabh Saxena for helping us in selecting the project and planning it. We would like to thank M.S. Ram for guiding with layout of GUI and possible ways to implement PRM and GUI. We would also like to thank Bhanu, Farid and Nikhil for helping us with difficulties faced during designing the GUI.

References

[1] Reach Based Synthesis of Modern Hyper Reduntant Manipulaters by Kaushik Sinha, 2002

[2] Probabilistic roadmaps for path planning in high-dimensional configuration spaces by LE Kavraki, P Svestka, JC Latombe

[3] Principles of Robot Motion-Theory, Algorithms and Implementation by Howie choset

[4] Lecture notes of CS365- by Dr. Amitabha Mukerjee

[5] Notes by Choset http://www.cs.cmu.edu/ motionplanning/lecture/

[6] Notes given at https://sites.google.com/site/ahtakoru/robot-motion-planning-and-control-oedevleri/probabilistic-roadmap-algorithm

[7] http://en.wikipedia.org/wiki/Configuration_space

[8] Qt documentation : http://qt-project.org/doc/qt-4.8/qgraphicsscene.html

[9] Video lectures on Qt by http://www.voidrealms.com/tutorials.aspx?filter=qt

[10] Fig.2 refrence http://www.directindustry.com/prod/minster/articulated-assembly-robots-24594-539718.html