Project for Computer Science 476
By Bing Yuan
(5/5/2000)

Note:
I did this project by JAVA because I did other homeworks with java.
But for the final project, I can't find a classmate to be grouped with using JAVA. So, I did this alone.

Project Description:

This project is to develop and implement a path planning system. In this system, the robot starts from an initial configuration, then it uses a path planning method to find the path to goal configuration. Finally, it will move along the path to the target configuration. The world can be a complicated geometric model. The path must be a collision-free path and it should be the fast and short one.

World Model:

The world would be 2D model. There are some obstacles in the world. The obstacles are union sets of the convex polygons. The initial configuration and the goal configuration of the robot are in the C_free configuration space.

Robot Model:

The robot's body will be a 2D rigid body, just like the car. The configuration is (x, y, θ).

Motion Model:

The motion model will be a nonholonomic model. The state transition functions are:

$$\begin{bmatrix} x_{new} = x_{old} \times \cos(\theta) \\ y_{new} = y_{old} \times \sin(\theta) \\ \theta_{new} = \theta_{old} \times step \times \tan(\phi) / Length \end{bmatrix}$$

(ϕ is the steering angle and Length is the axis length of the robot.)

When the ***step*** is [-1,1] , it means the robot can be forward and reverse. When the ***step*** is [0,1] , it means the robot can be forward only. When the robot reaches the goal, it must fit the configuration totally. There are also other constrains like the max steering angle and minimal steps. We assume the steering angle can change instantly.

Path Planning Model:

The path searching will use the method of Rapid-Exploring Random Tree (RRT). To improve the performance, it will construct goal-bias RRT to find the path. This means we can use the goal as the 10% random selection. Another improvement is that when the robot is near the goal, the random selection is selected in a range near the goal. This improves the performance when the robot is near the goal. We set a maximum-loop-number. After the number, if the robot still does not reach the goal, we say the planning is failure this time and we can try again. According to different complexities of world models, the performance will be difference.

Collision Detection:

I use the differential method to do the collision detection. For a segment, I divide it into some points. If every point is in C_free, then I say the whole segment is in C_free. Using this method, I test all the segments of the bound of the robot to see if the robot is in C_free.

Software Interface:

I used the java applet to demo the whole process. The map of the obstacles can be read in from the file that is created by XFIG. This is very convenient for users to construct different world models. The user can randomly initial the start point of the robot and select moving model (FWD and REV / FWD Only). To find a path from the start to the goal, the program generate the RRT, then it find the shortest path of the tree. At last the robot moves along the path. By this program, the user can see visually how the RRT path is constructed according variant situation.