Obstacle Avoidance

Can Eldem

September 22, 2013

Abstract

This project aims to create a robot simulation which can reach given target point in space with obstacles.

Contents

Part I

Free Space Travelling

1 Description of theory and program

This part of the project aims to create a robot simulation which can reach a given target point in free space (without any obstacle) according to user input. The user is able to select two different methods:

1.) Potential Field Method

2.) RRT (A Rapidly-exploring Random Tree)

Potential Field Method: The idea of potential field method is to fill the robot's workspace (sensors) with potential based on the target and to make the robot decide to move according to potential. In a way target points act like a magnet to attract robot to itself in free space environment .[3]Points which are closer to Robot is more attractive than other points around target which are further to robot. Other points are unattractive for the robot to move to. The value of potential at the goal point is set to be 0 and the value of the potential at all other points is positive.

RRT: RRT is a planning algorithm quick search in high dimensional spaces. The idea of an algorithm is putting random points after every iteration and trying to expand the tree V unit towards that sample point. This step is repeated until reaching target point or area.[2]

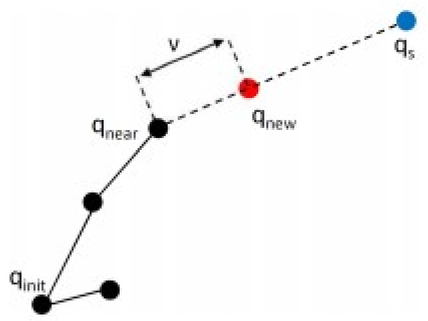


Figure 1: qs(sample point), qnew (expanded point) qnear closes root to qnew and qinit is initial point

Variables like size of robot, move range, sensor range, view angle and number of sensors are designed to be changeable. Like in Figure 3, blue and red dot positions are calculated according to range settings. Direction of this semi circle might change according to initial point of a target. For example, the target might be to the left side of robot for this initial statement sensorTowards() method draws first semi circle towards target point and rest is completed by loops In order to hold and evaluate information about every hit point, “Points” object is designed. “Points” object also holds the location of hit point and angle between the centre point of the robot.

Part II

Implementation

2 Robot Implementation for Potential Fields

For the potential field algorithm, the robot is designed as in figure 2.

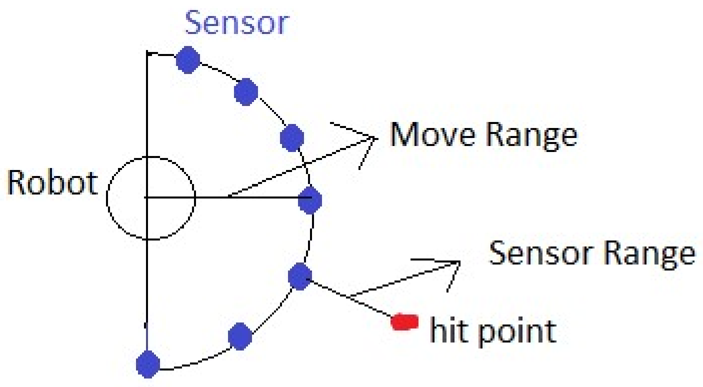


Figure 2: potential field algorithm design

Variables like size of robot, move range, sensor range, view angle and number of sensors are designed to be changeable. Like in Figure2, blue and red dot positions are calculated according to range settings. Direction of this semi circle might change according to initial point of a target. For example, the target might be to the left side of robot for this initial statement sensorTowards() method draws first semi circle towards target point and rest is completed by loops In order to hold and evaluate information about every hit point, “Points” object is designed. “Points” object also holds the location of hit point and angle between the centre point of the robot.

In every iteration the robot calculates potential field via its sensors and set potential value to created “points” object. Potential in non-obstacle environment is distance between target and hit point. The robot has the coordinates of the target point from an initial statement.

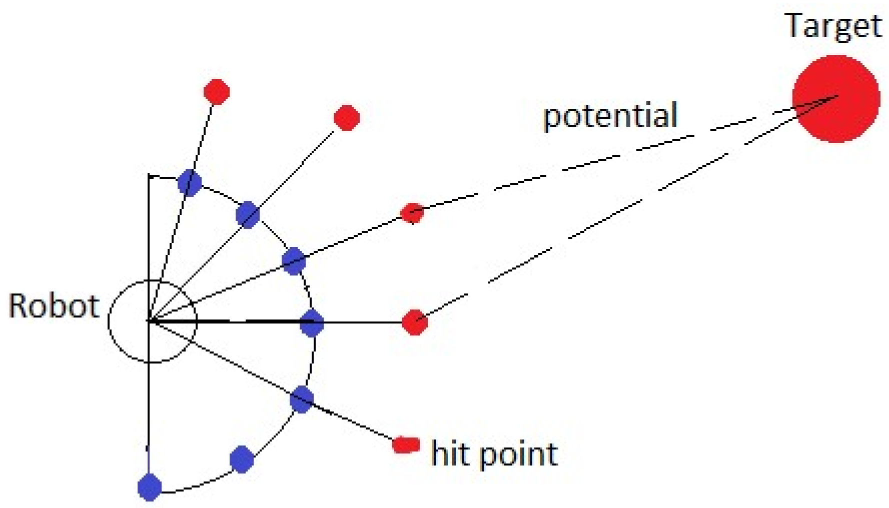


Figure 3: Calculating potential after every iteration

Figure 3 shows every hit point determined and calculates its potential with getNPointsOnCircle method. Afterwards "points" object with lowest potential field is chosen (selectBest method) and with the chosen point angle information another abstract semi-circle will be drawn.Thus, chosen points will be stored (line array) in order to draw a path in GUI. This iteration will continue until insideArea method detects path is in goal area .

3 Robot Implementation for The Rapidly-Exploring Random Tree

The RRT is designed to work like figure 1. CreateRandomPoint() runs in order to create random sample points on map . However, since this design also knows about location of target It aims to create random points up to %10 percent more further of target point otherwise it won't able to reach the target. Shown in figure 4 Thus, sample points created towards that point. In other words goal bias has implemented in this part. Effectiveness of this method will be discussed at further section.

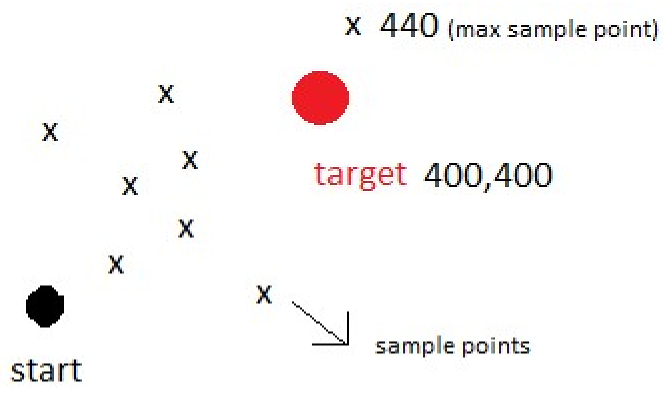


Figure 4: RRT creates sampling points and move towards to them

FindDrectionPoint method runs and creates step size point towards that sample point. Then, this new point added to closes node of tree and iteration continues until insideArea function detects it is desired goal area. Once tree reaches target area provided getPathFromRootTo(nearest) method runs and back trace is shown with red lines via provided coordinates.

Part III

I/O (testing)

4 Potential Field Testing in Free space environment

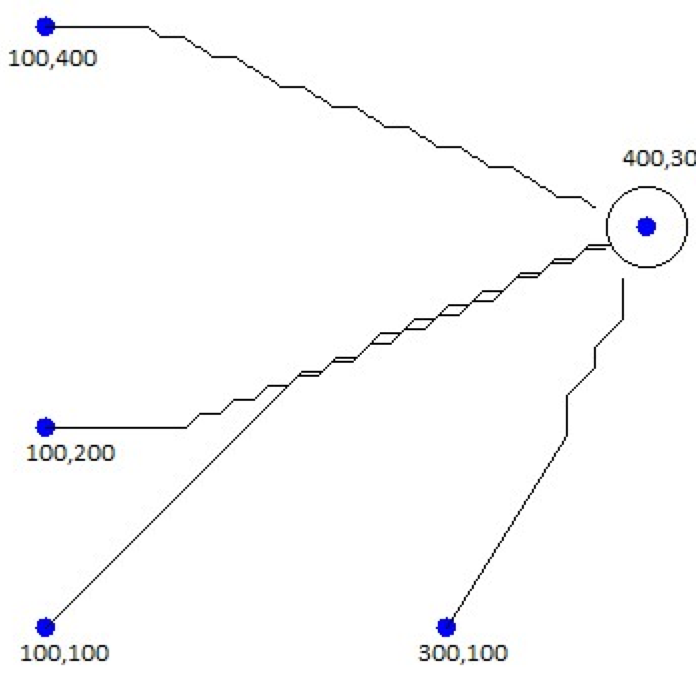


Figure 5:

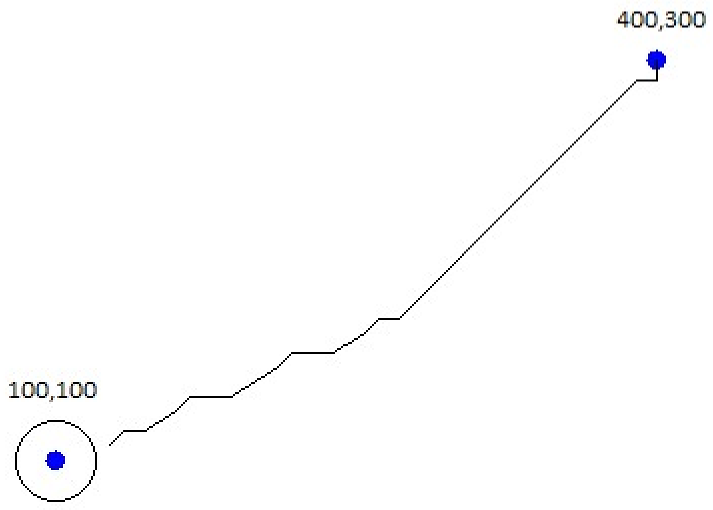


Figure 6:

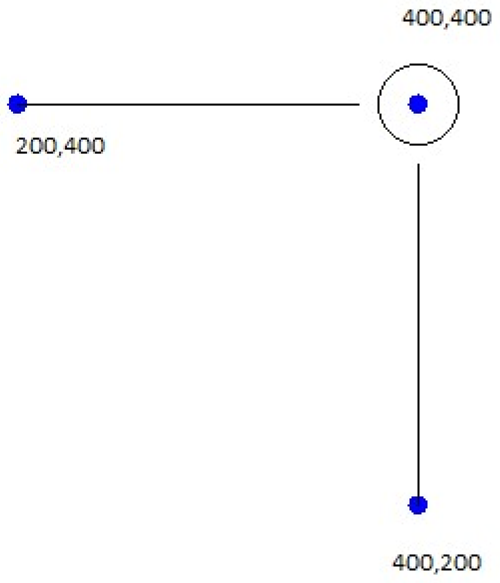


Figure 7:

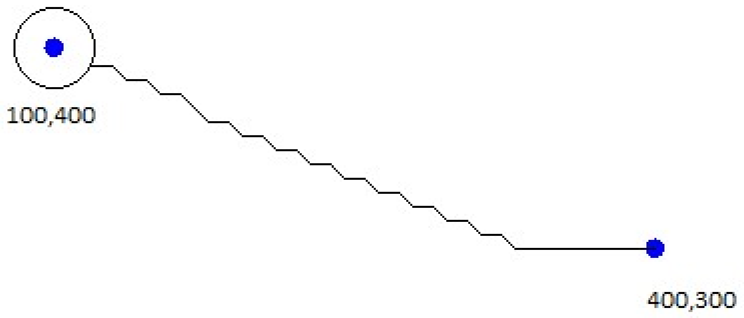


Figure 8: The robot is on the right side and the target is on left side but robot adjust its sensor semi-circle according to target direction .

5 RRT Testing in Free space environment. (with goal bias)

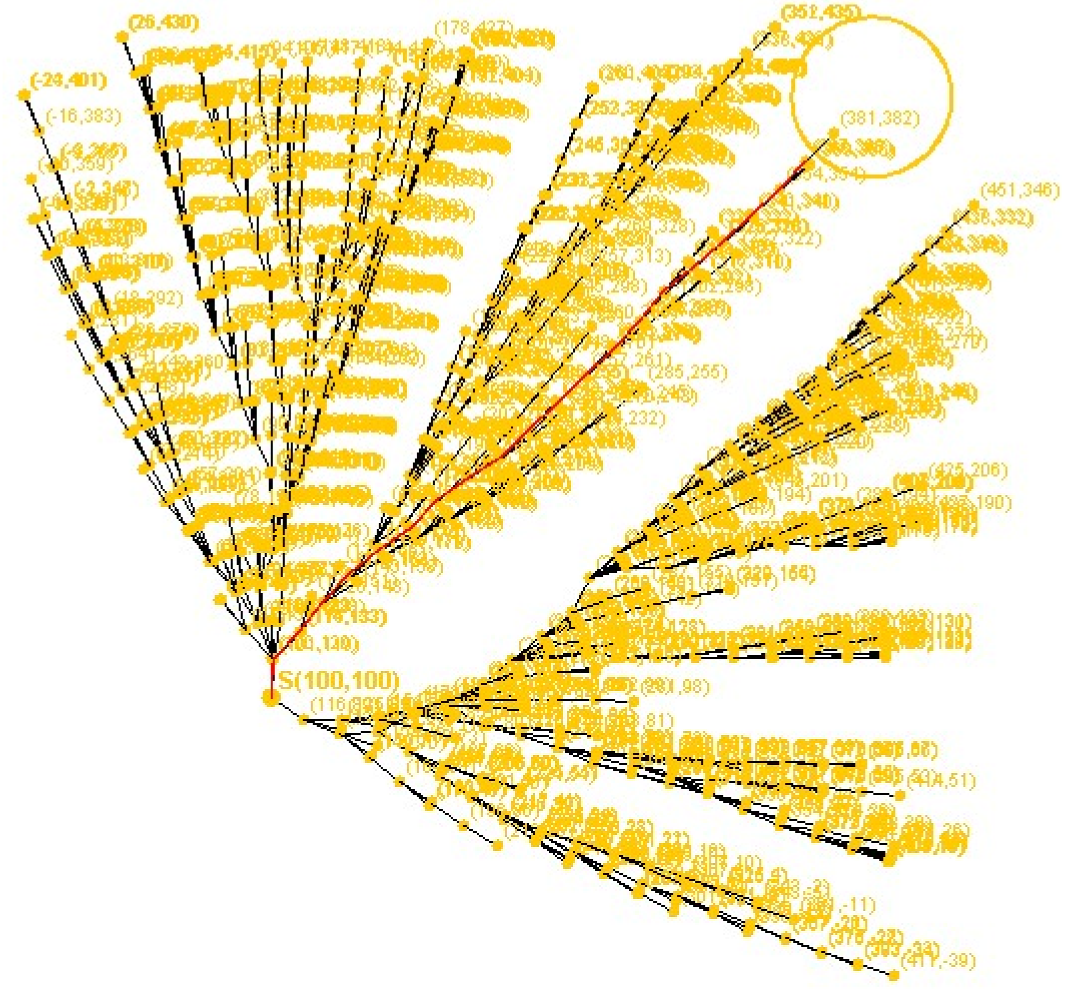


Figure 9: tree develops towards target direction with goal bias technique

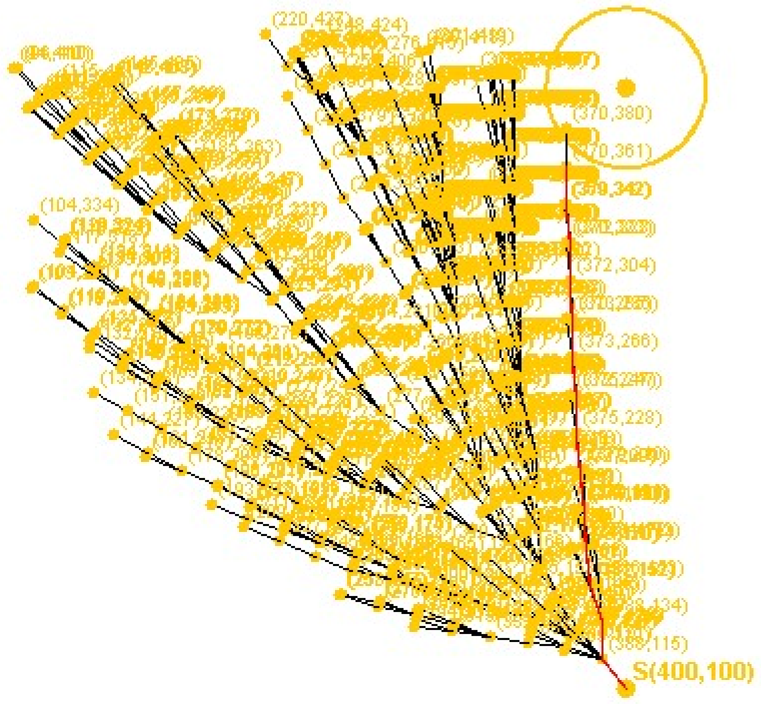


Figure 10:

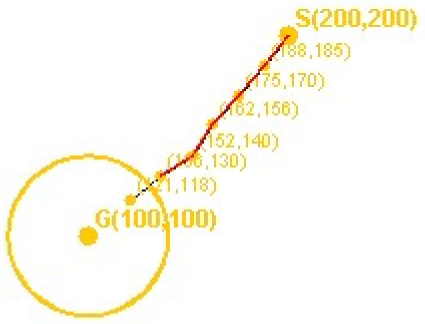

S(200,200)
(188,185)
(175,170)
(162,156)
(152,140)
(136,130)
(121,118)
G(100,100)

Figure 11:

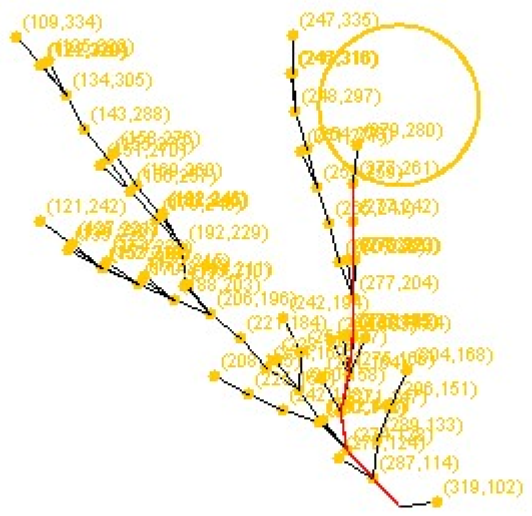
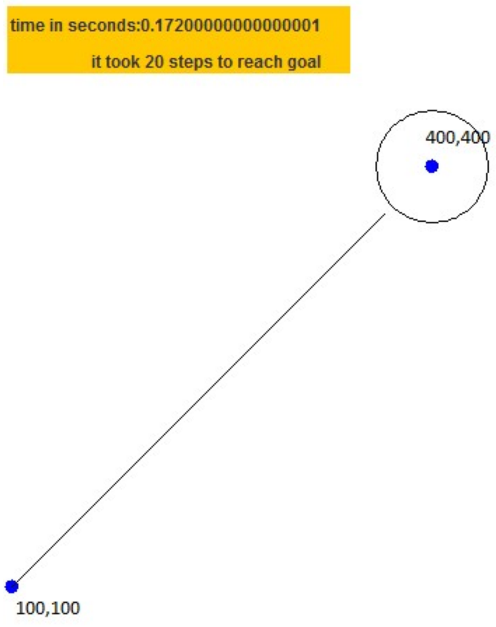


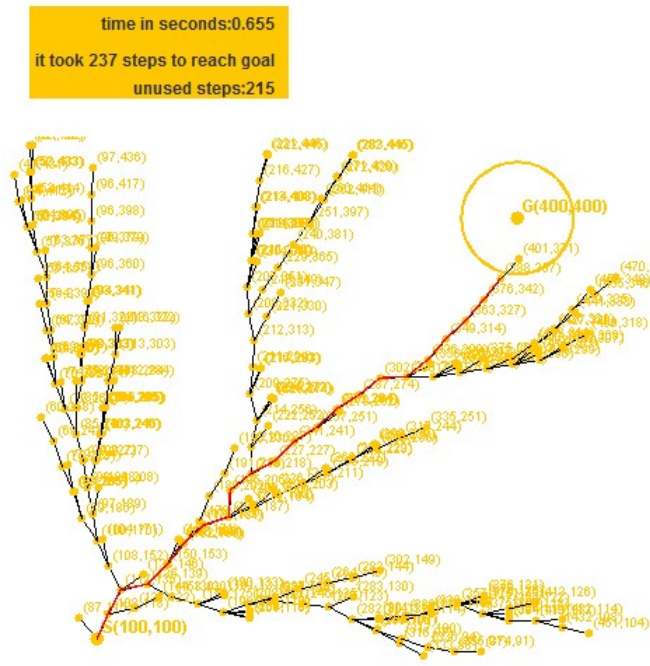
Figure 12:

6 How efficient are the two planners in Free Space?

In order to measure the effectiveness of the two algorithms, time and step size measures are taken into consideration.


time in seconds:0.17200000000000001

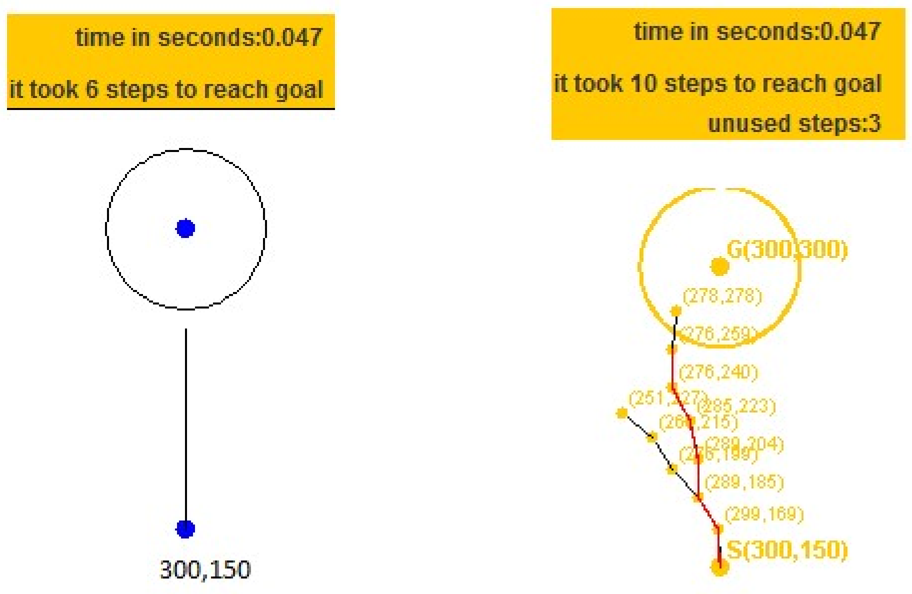
it took 20 steps to reach goal


time in seconds:0.655

it took 237 steps to reach goal

unused steps:215

Figure 13:

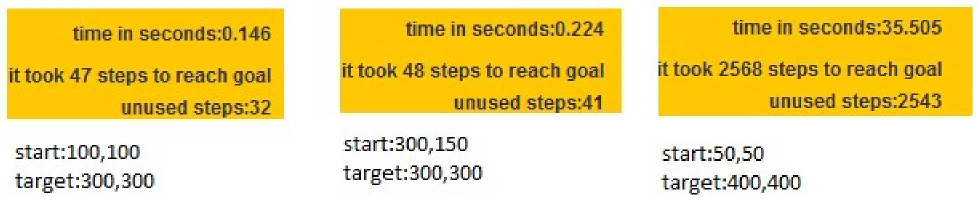

time in seconds:0.047
it took 6 steps to reach goal

time in seconds:0.047
it took 10 steps to reach goal
unused steps:3

Figure 14: step size set as 20 and all goals and target points are same.

It shows that potential field has a better performance than RRT. However, in close distance targets like in Figure 14, the amount of time elapsed for both algorithms have close or same values, but potential field used fewer steps than RRT.

7 Node Performance in RRT


time in seconds:0.146
it took 47 steps to reach goal
unused steps:32
start:100,100
target:300,300

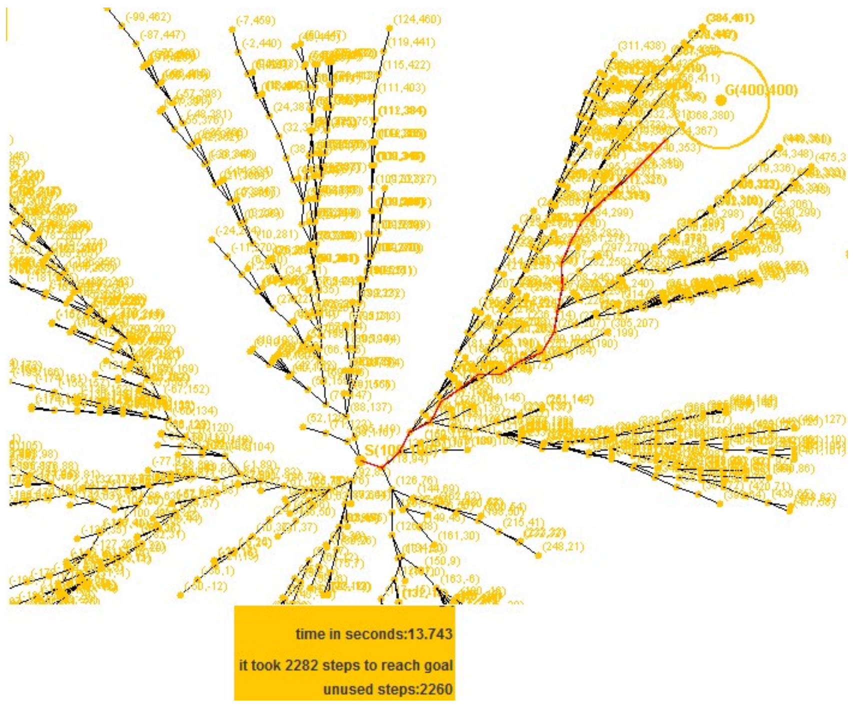
time in seconds:0.224
it took 48 steps to reach goal
unused steps:41
start:300,150
target:300,300

time in seconds:35.505
it took 2568 steps to reach goal
unused steps:2543
start:50,50
target:400,400

Figure 15: performance of three different cases

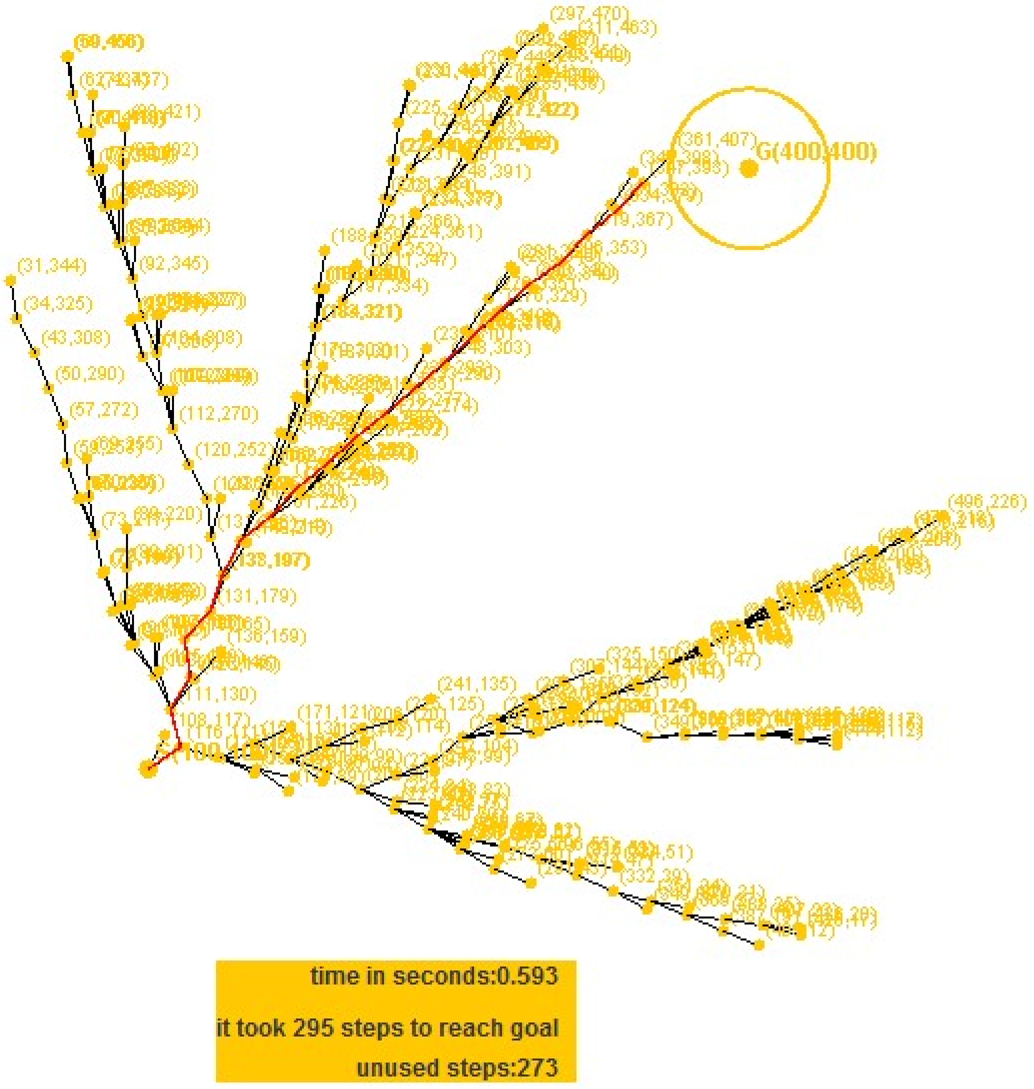
Figure 15 examines three different cases in order to indicate node performance of RRT. With small distances between starting point and target, unused nodes are 68%, and 85% respectively. However, in long distances between starting point and target point, the unused node rate is 99%. This shows that the percentage of unused node rate increases with distance between starting point and targeting point .

8 Effect of adding goal bias to RRT



time in seconds:13.743
it took 2282 steps to reach goal
unused steps:2260

Figure 16: without goal bias



```
time in seconds:0.593
it took 295 steps to reach goal
unused steps:273
```

Figure 17: with goal bias

Effect of goal biasing clearly shows itself comparing performance between Figure 15 and Figure 17. In Figure 15, tree expands on the other side of free space as well, since it does not use target information, whereas in figure 17, the tree expands towards the goal.

Part IV
Conclusion

Comparing with these two algorithms. I found potential field algorithm more useful than RRT for free space travelling purposes. The reasons for this are: it is really easy to implement potential field compared with RRT. You are not required to deal with complicated structures like tree. Secondly, potential field algorithm focuses on target directly unlike RRT tree. Therefore, it increases its performance which has been proven by tests.

Part V
Obstacle Navigation

9 Description of theory and program

This part of the project aims to create a robot simulation which can reach given target point in space with obstacles according to user input.

Potential field algorithm with obstacle avoidance method: Same logic applies, but in addition to goal force, obstacle force will be considered in order to move towards target. Obstacle force increases in inverse proportion to the distance to the nearest obstacle boundary.

RRT algorithm with obstacle avoidance method: This method will also apply same logic in Figure-1. However, the algorithm will check that whether an expansion of a new node is going to touch the obstacle or not. If it is touching the obstacle, iteration will repeat until it finds a free space on map.

10 Implementation Robot Implementation for Potential Fields

Same design structure has been used in Figure-2. However, additionally, the robot gets coordinate of obstacles from map environment and checks them in every step that whether hit points touch obstacle or not with isIntersect method, if one of the hit points touches the obstacle, it calculates the obstacle point based on distance between obstacle center coordinate and hit point coordinate then it updates obstacle potential attribute of "points" object for this specific hit point .

After calculating potentials selectBest method gets best point to move considering lowest obstacle potential and lowest potential to target. Algorithm also considers size of robot while crossing obstacles using isIntersect method.Calculation of Obstacle potential could be explanied with the figure below;

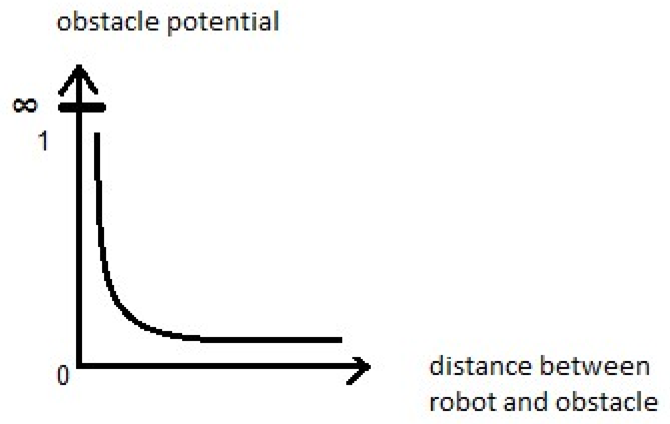


Figure 18:

11 Robot Implementation for RRT

In addition to the same design shown in Figure-2, additional check statement is added to RRT. In every expansion, qnew is checked whether it is touching an obstacle or not with insideArea method. The obstacle information is provided by the Map environment. If qnew encounters an obstacle, the loop starts again until it finds another free point to expand. Thus, goal bias range is increased %10 per cent based on initial point distance because if robot encounter with big object close to itself robot can't go around obstacle because sample points will be towards the goal .

Part VI
I/O

12 Potential Field Testing in obstacle space environment.

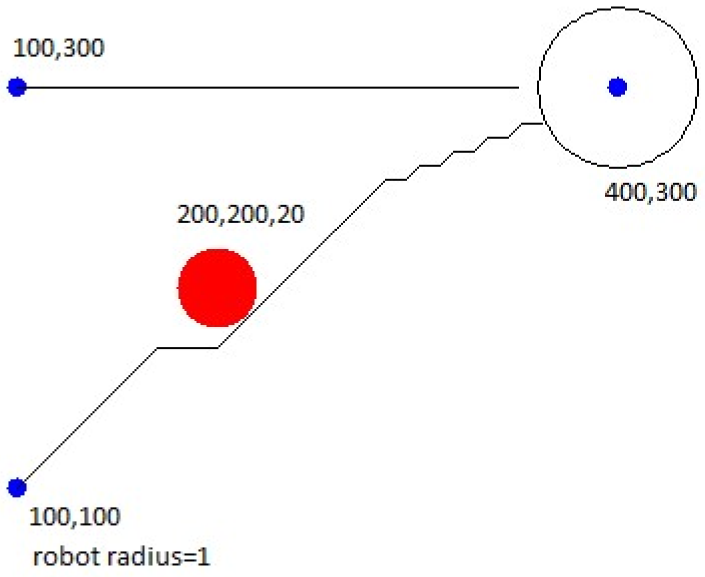


Figure 19:

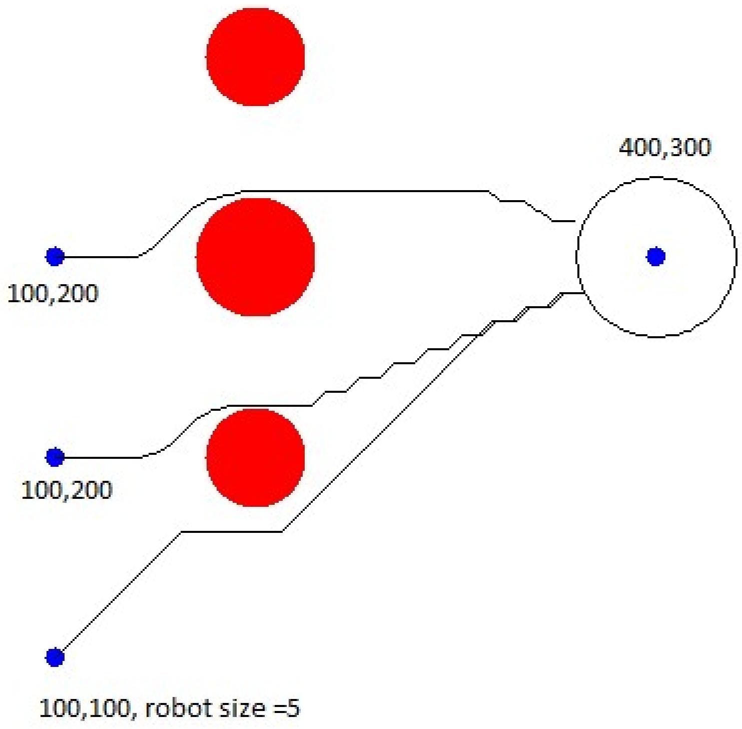


Figure 20:

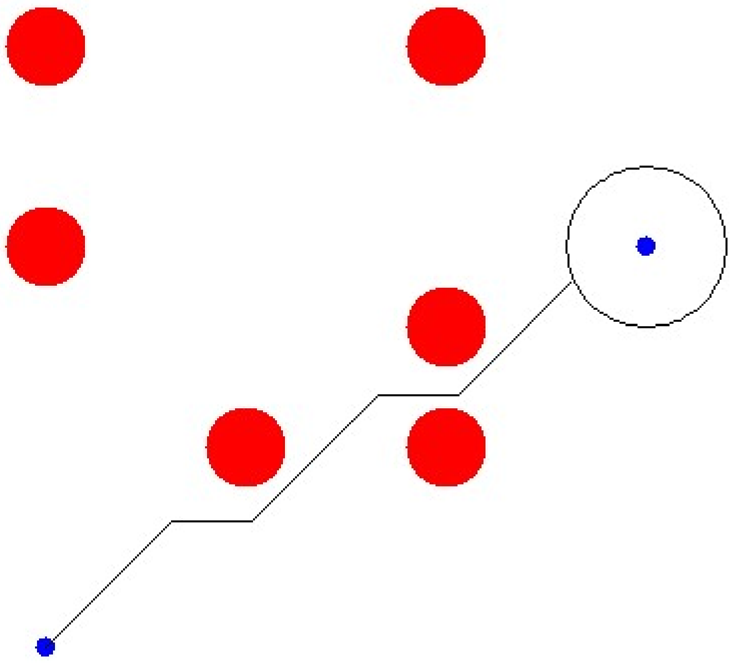


Figure 21:

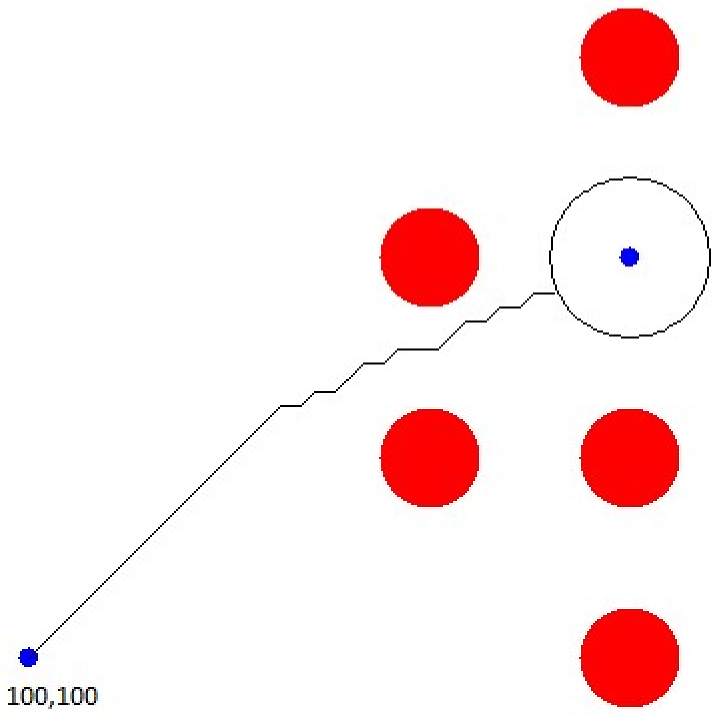


Figure 22:

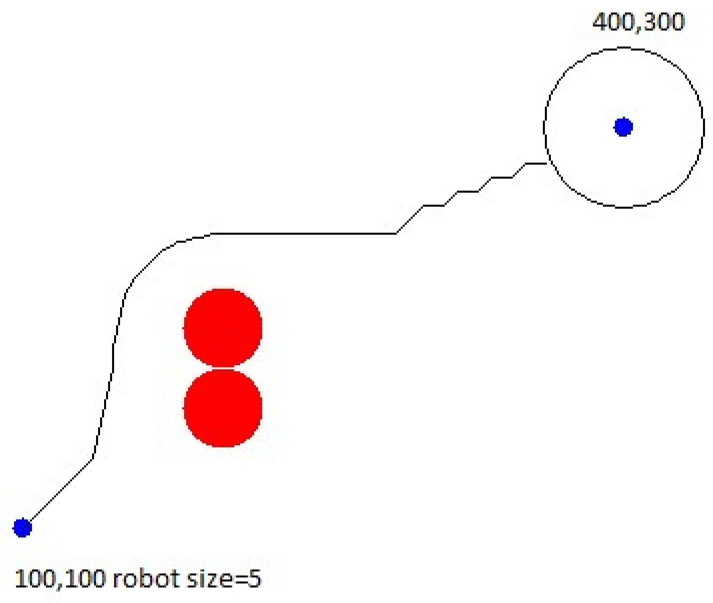


Figure 23:

13 RRT Testing in Free space environment.(with goal bias)

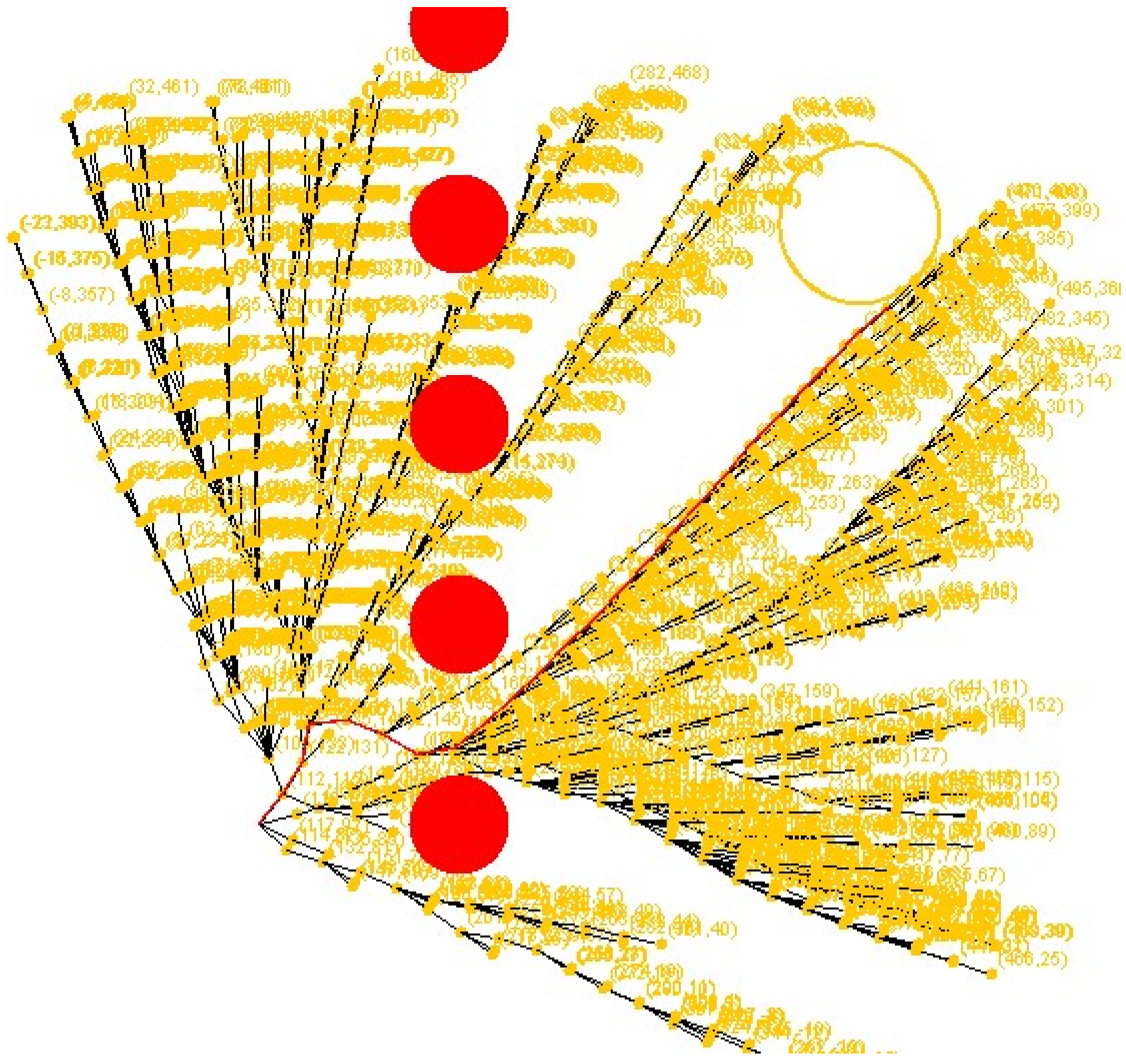


Figure 24:

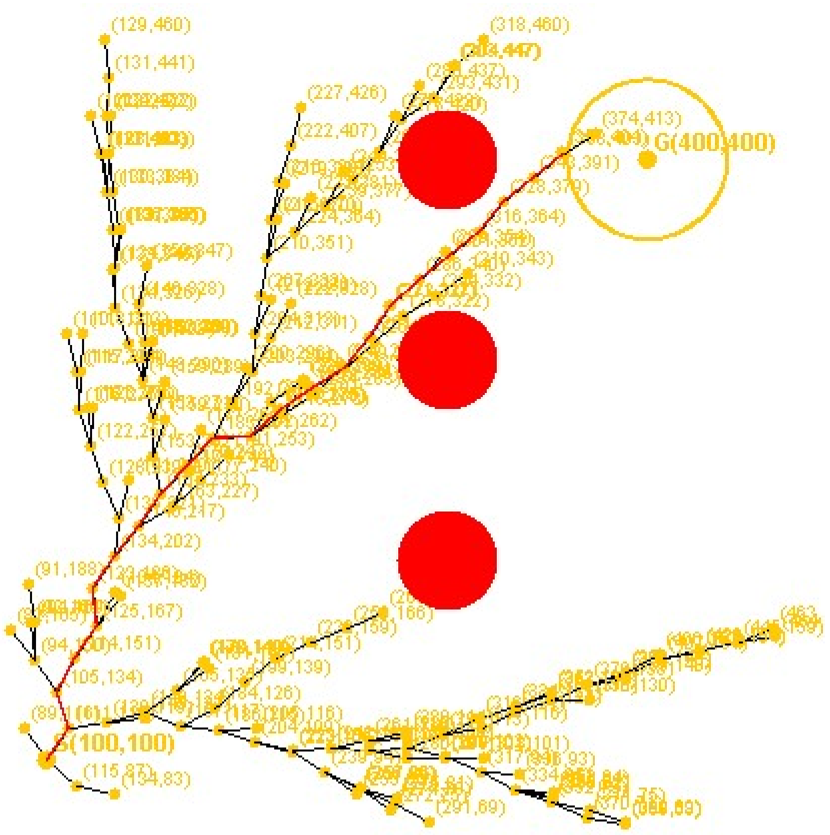


Figure 25:

14 How would a global map of obstacles help if it was provided beforehand or built up during travel?

Having a global map presents some advantages and disadvantages for the robot .The advantage is that the robot will be able to plan its path before it moves because the robot has the map. This may also save cost in terms of calculations because the robot is not required do sense and evaluate its environment. Thus, the robot will able to find real shortcuts to target. However, having global map does not meet real life requirements for design. For instance, the robot won't able to sense the changes in map area if objects start to move or when places of objects are updated. On the other hand, having a sensor based map costs more than global map because it requires robot to check obstacles around itself with every step. This project has designed to have dynamic map rather than static global map.

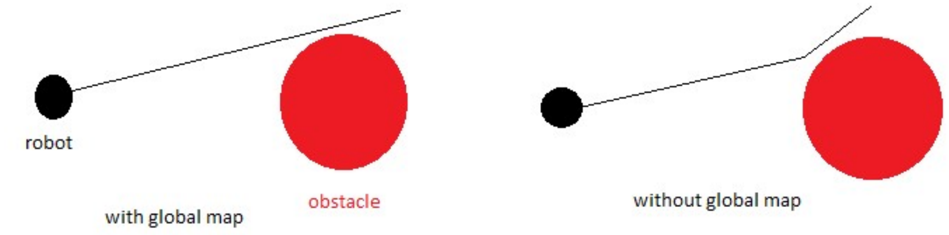


Figure 26: show possible attitude difference between having global and non-global map.

15 How efficient the two planners are in obstacle Space ?

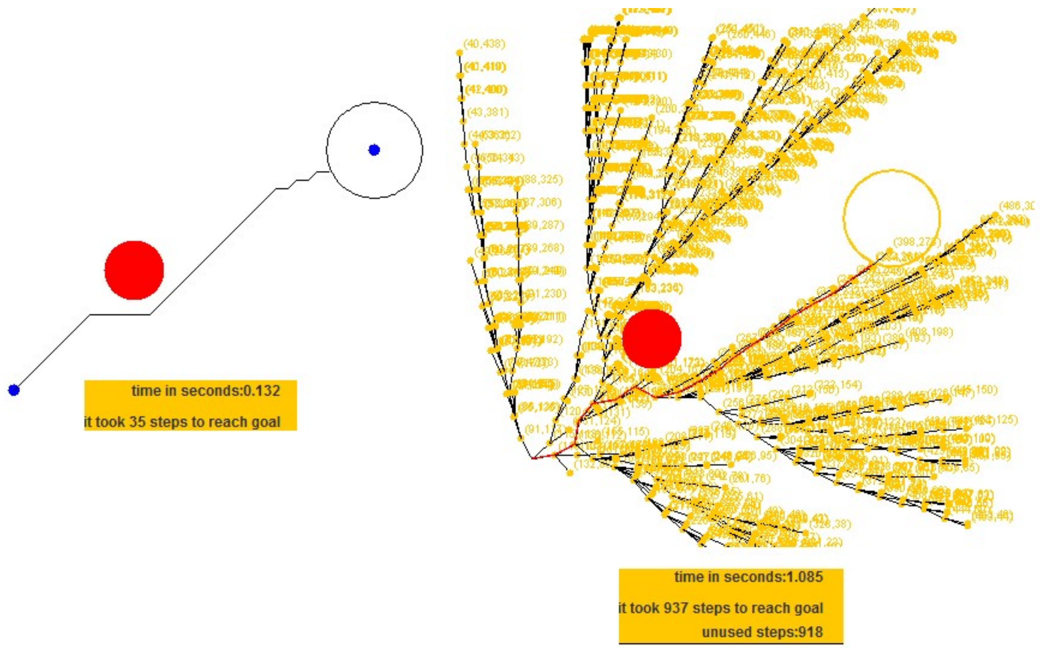
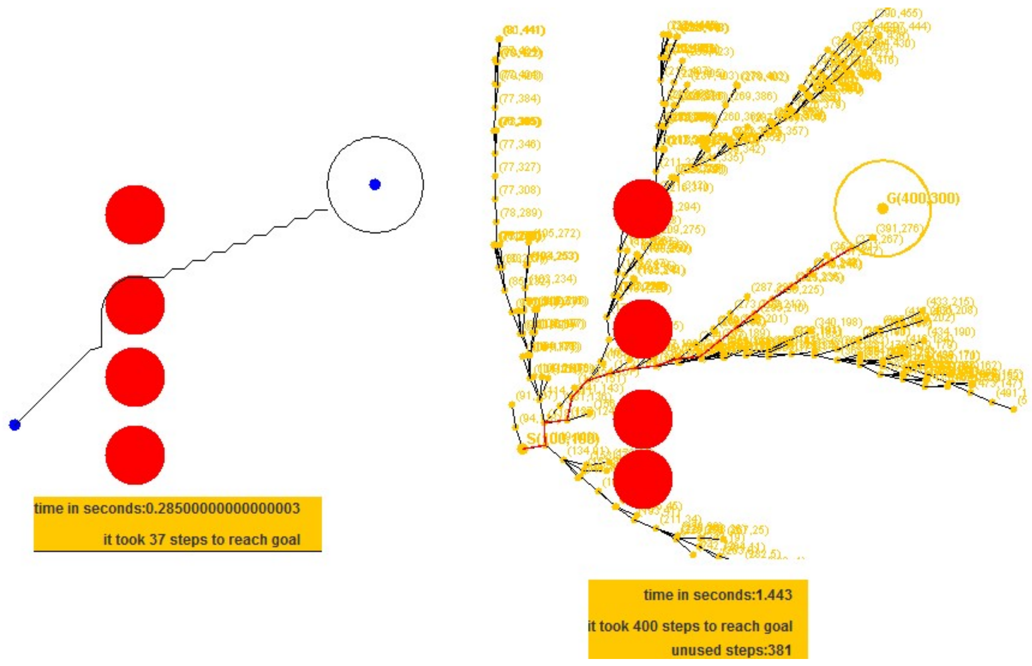

time in seconds:0.132
it took 35 steps to reach goal
time in seconds:1.085
it took 937 steps to reach goal
unused steps:918

Figure 27: starting, target and obstacle points are the same.



time in seconds:0.28500000000000003

it took 37 steps to reach goal

time in seconds:1.443

it took 400 steps to reach goal

unused steps:381

Figure 28: starting, target and obstacle points are the same.

As it is seen in figure-27 and Figure-28 potential field algorithm works faster in order to reach the target area. Thus, potential field algorithm is able to complete process with fewer steps.

16 Conclusion

I prefer RRT in obstacle space environment because although potential field has better performance compared with RRT sometimes potential field can't detect obstacles if obstacle is small or and there is a blind point between sensors. Potential field could fail in lune shape obstacles if sensor range is not big enough. Once a robot using potential field algorithm gets into a luna shape, the robot can't turn back.The structure of holding potential and calculating obstacle potential should be altered. The current way of calculating obstacle potential is based on the distance between obstacle's centre and robot's hit point. Instead of doing this, every obstacle could create negative potential around itself; this new design enables us also to use different range of obstacles in map regardless of the shape of the obstacle. For potential field, obstacles with different shapes such as lune could cause robot to be stuck in a way. In order to solve this problem, potential field method could be used with bug method. When the robot gets into the lune shape obstacle, it can hit the wall using the bug algorithm. The robot would then be able to get rid of the obstacle. In RRT using goal bias towards goal point might cause problems when obstacle is big because goal bias creates sample points towards goal therefore it might stuck to obstacle and can't move to other place . To solve this problem sample points should also appear other points far from goal with limitation .

References

[1] Akgun, & Stilman (2011). Sampling Heuristics for Optimal Motion Planning in High Dimensions. 6.

[2] Almahairi (2010). Rapidly-Exploring Random Trees in Highly Constrained Environments. 13.

[3] GE, & CUI (2000). dynamic motion planning for mobile robots using potential field method. 6.