CS 4649/7649: Robot Intelligence Planning (Motion Planning)

Project 2: Motion Planning Assignment

Bharadwaj Tanikella, Claire Bergman, Jose Hernandez Anton, Kenneth Marino, Pujun Bhatnagar

<u>**Group Assignment 1: Motion Planning**</u>

Bharadwaj Tanikella

- Wrote Task Constrained RRT's
- Write-up for Part 5.
- Compiled the document from all write-ups.

Claire Bergman

- Wrote Manipulation Planning RRT's
- Write-up for part 4.

Jose Hernandez Anton

- Wrote Manipulation Planning - Differential Kinematics
- Write-up for Part 3.

Kenneth Marino

- Wrote Part 1 and Part 2 of the Assignment.
- Write-up for Part 1 and Part 2.

Pujun Bhatnagar

- Wrote Manipulation Planning - Differential Kinematics
- Write-up for Part 3.

<u>**Note**</u>

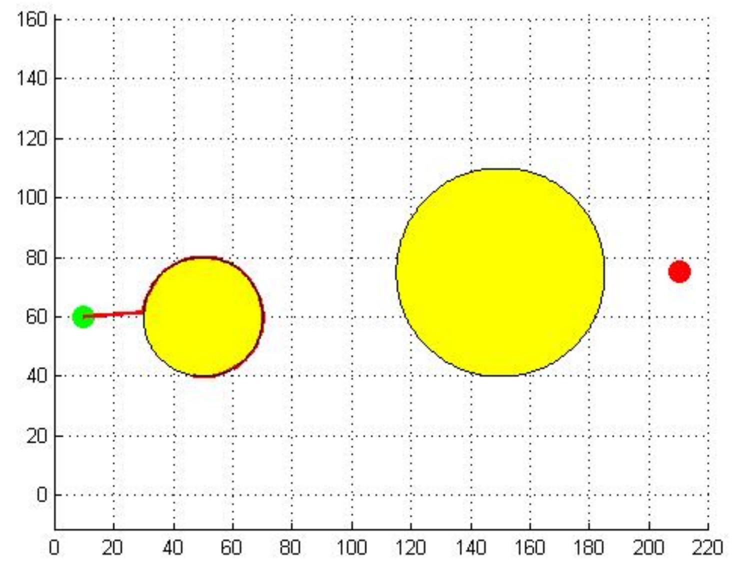
This report comprises of all the solutions and explanations to the problem set provided to us. The supporting files are provided in the repository, as they are large to hold in a zip file. The code/videos, which are generated, are provided in the respective directory in the repository. For instance the results for part 4 is provided in the /part4 directory.

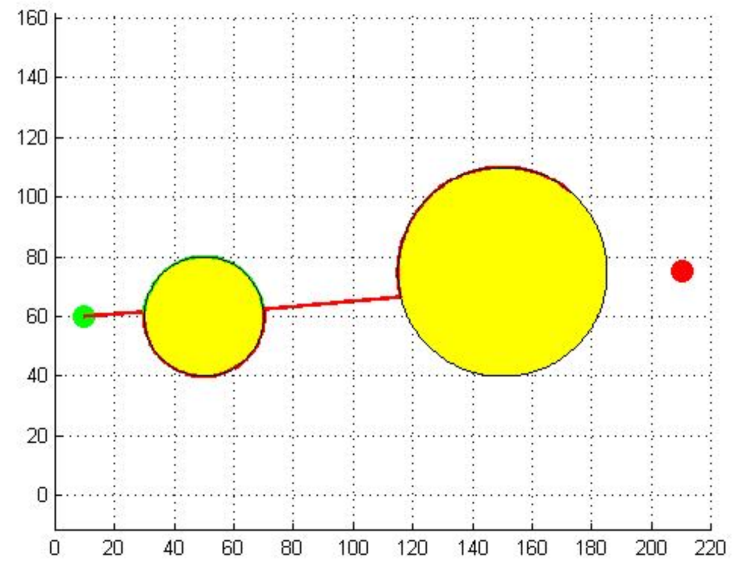
Github: https://github.com/pbhatnagar3/RIP2014P2

1. Navigation Planning - Bug Algorithms for Point Robot

a) Bug 1 algorithm:

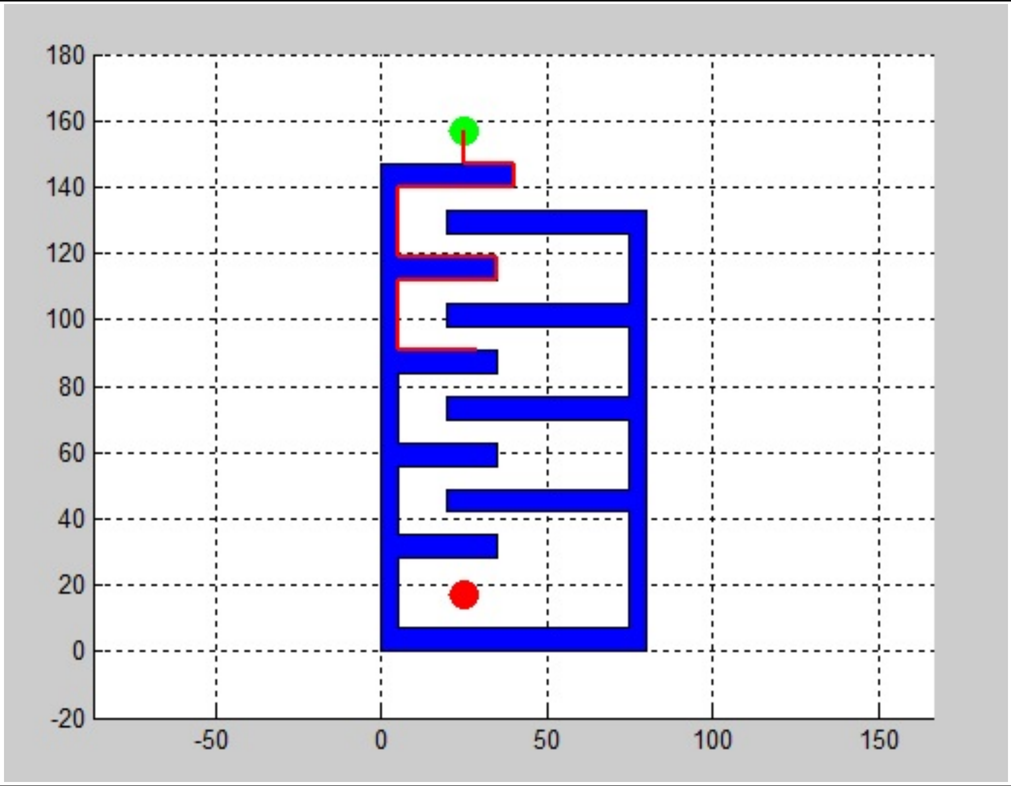
→ Domain 1

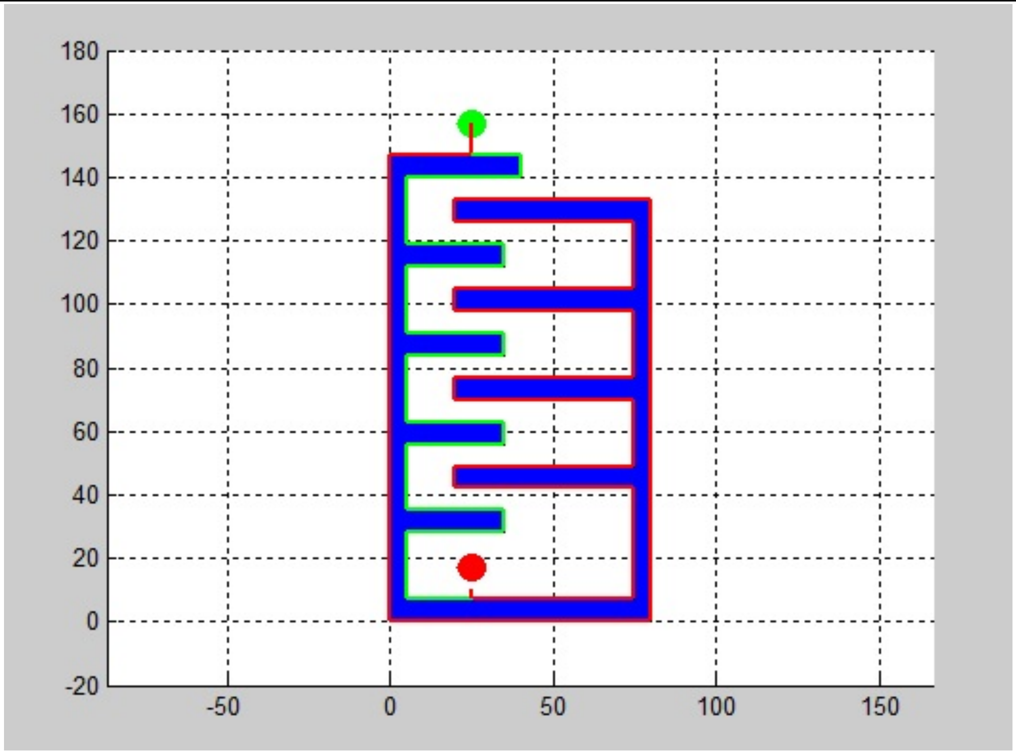




Actual Distance Traveled - 614.1994
Euclidian Distance to Goal - 200.5617
CR = 3.0625

→ Domain 2

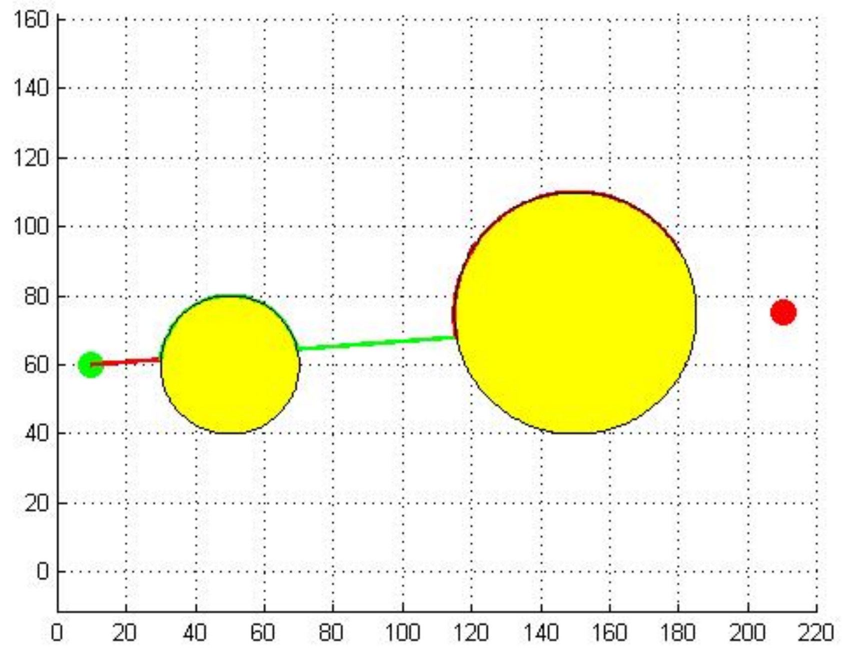


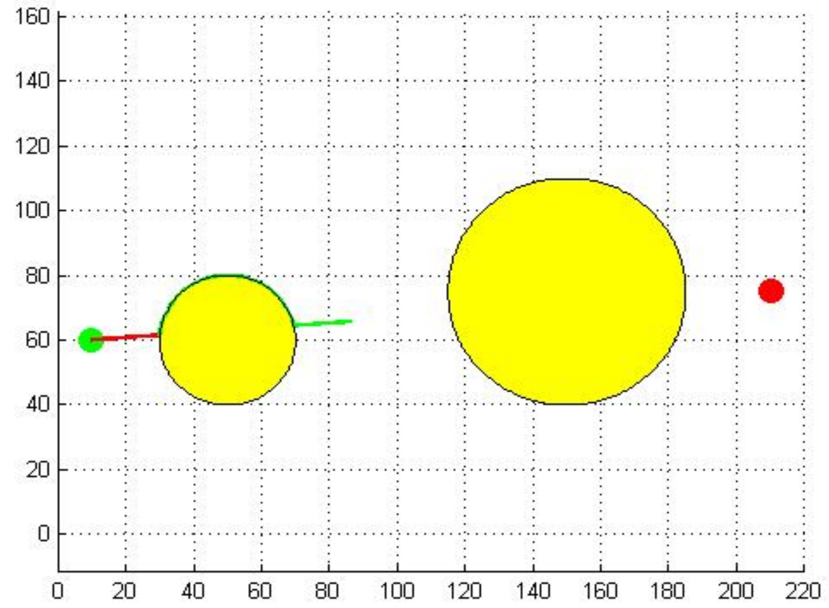


Actual Distance Traveled - 1926
Euclidian Distance to Goal - 140
CR = 13.7571

b) Bug 2 algorithm

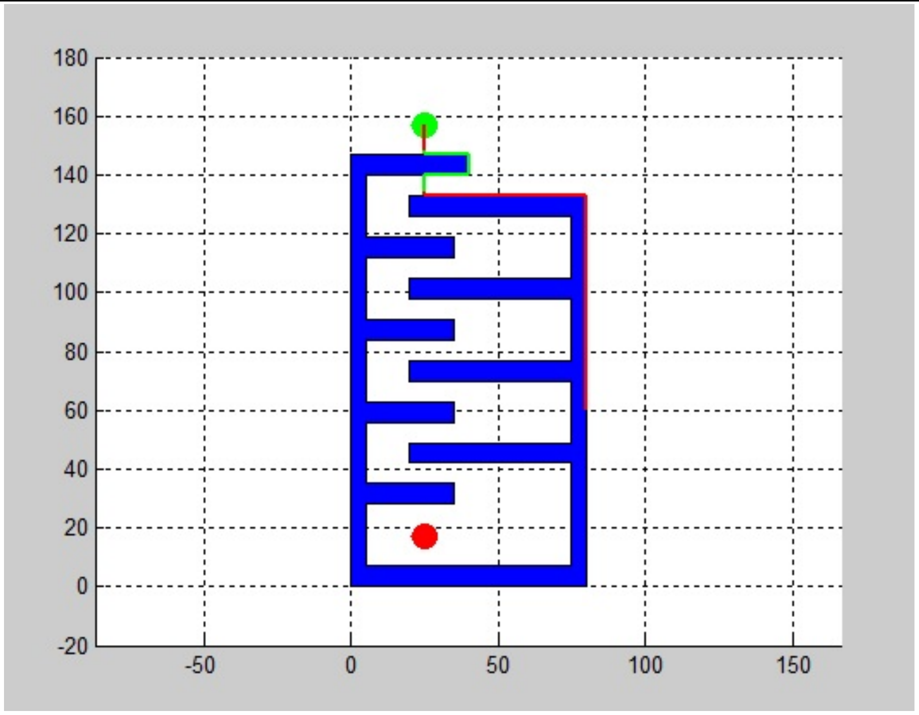
→ Domain 1

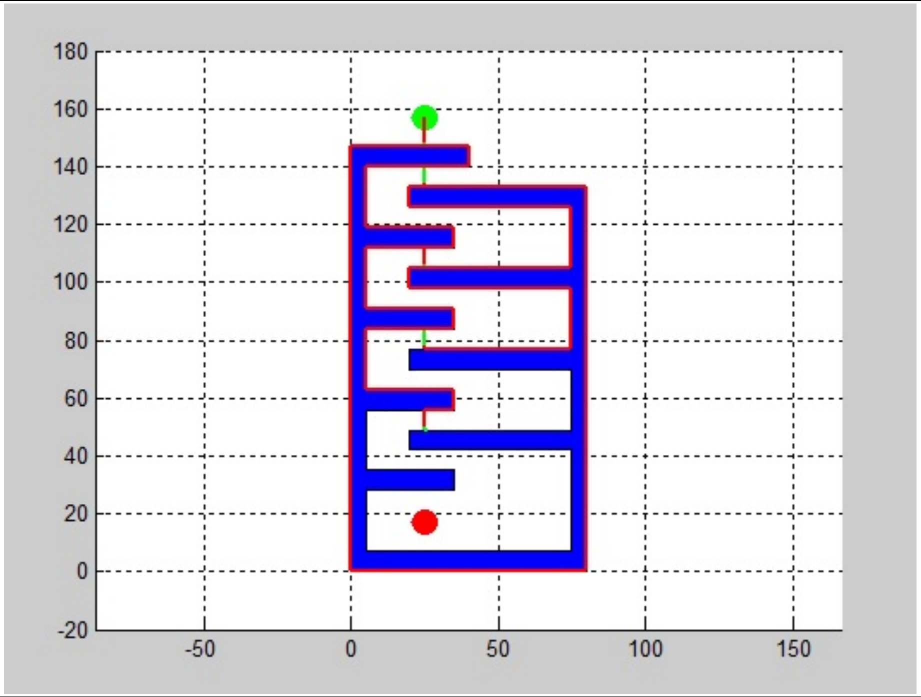




Actual Distance Traveled - 269.4318
Euclidian Distance to Goal - 200.5617
CR = 1.3434

→ Domain 2



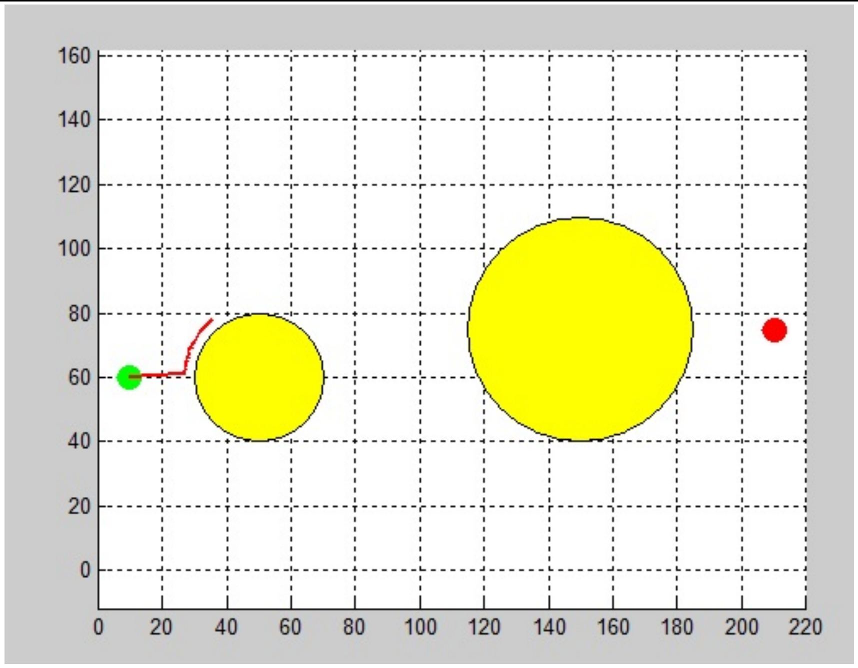


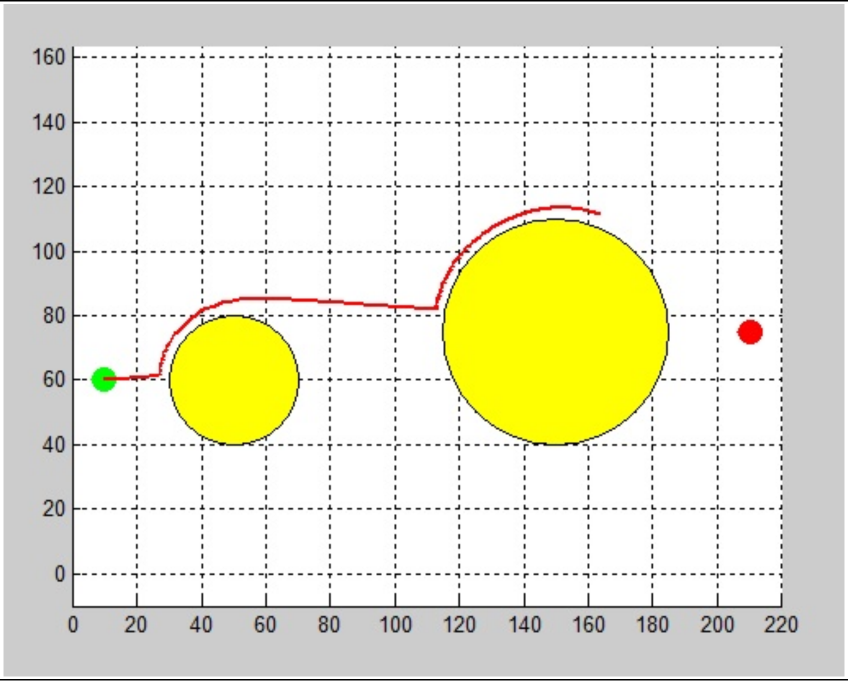
Actual Distance Traveled - 3702
Euclidian Distance to Goal - 140
CR = 26.4429

2 Navigation Planning - Potential Field Navigation for Point Robot

a) Potential field navigator:

→ Domain 1

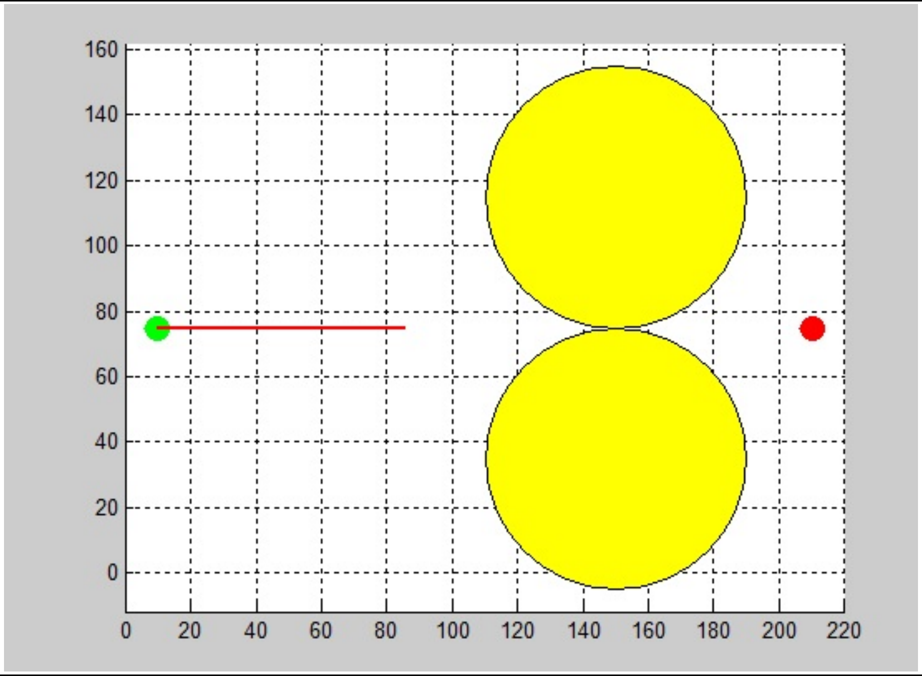


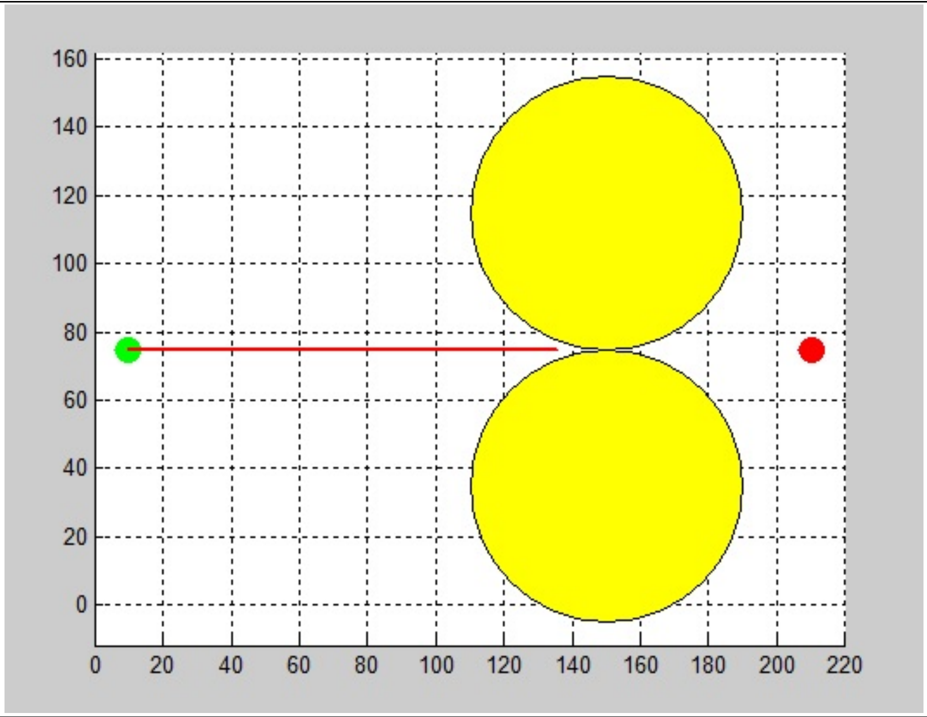


Actual Distance Traveled - 259.5066
Euclidian Distance to Goal - 200.5617
CR = 1.2939

b) Local Minimum Example

Circles of radius 40 at (150, 35) and (150, 115)





Robot has movement 0 at (35.7141, 75) and stays there. See video and run code for proof.

3 Manipulation Planning - Differential Kinematics

a) The video of the simulation is uploaded in the repository (/part3/) and is called 3a. Here are the screenshots (the starting position is the blue point and ending position is green dot)

Fig. Starting configuration of the robot arm

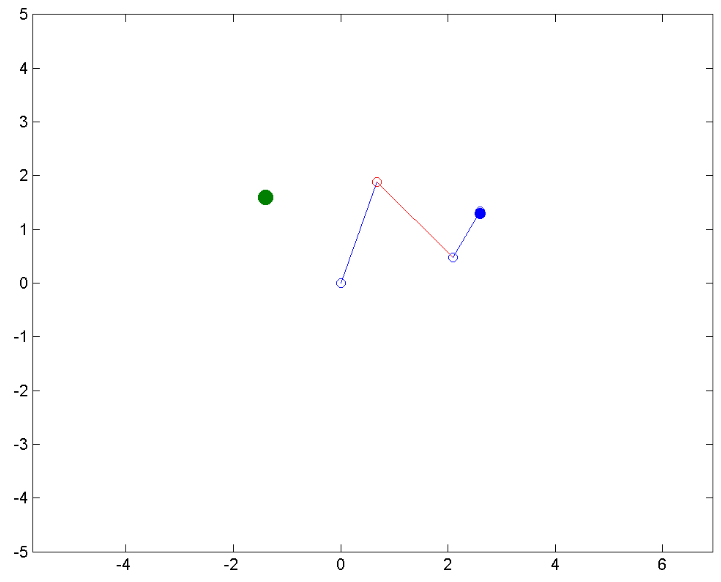
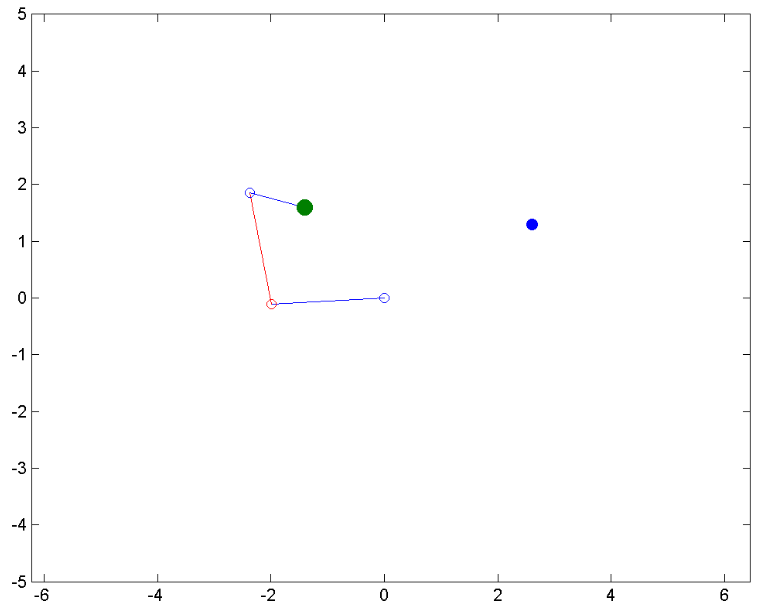
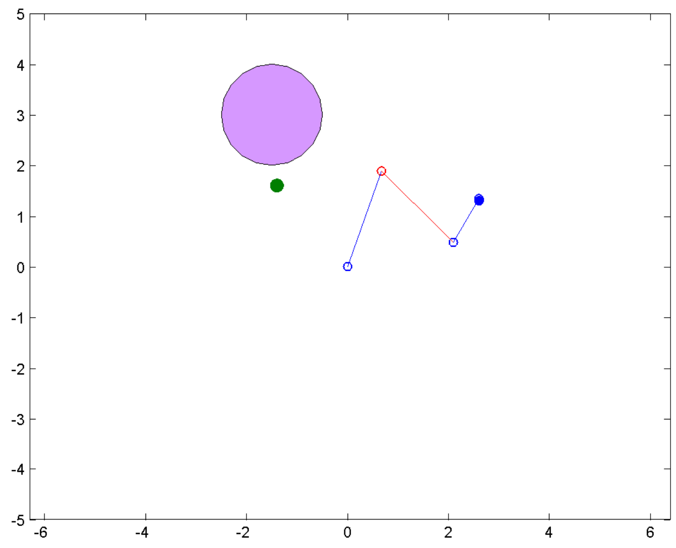


Fig. Ending configuration of the robot arm



b) To take the joint limit and collisions into consideration, potential field method was used. A circular object was created and placed in the 'regular' path (the path found between the start and ending point in part a). During the planning process, as soon as any part of the robot's arm came near the object, the algorithm started changing the value of thetas to avoid the collision and the Jacobean is recalculated. The video of the simulation is uploaded on the repository (/part3/) and here are the snapshots (the starting position is the blue dot and the goal is the green dot):

Fig. Showing the starting configuration of the robot arm



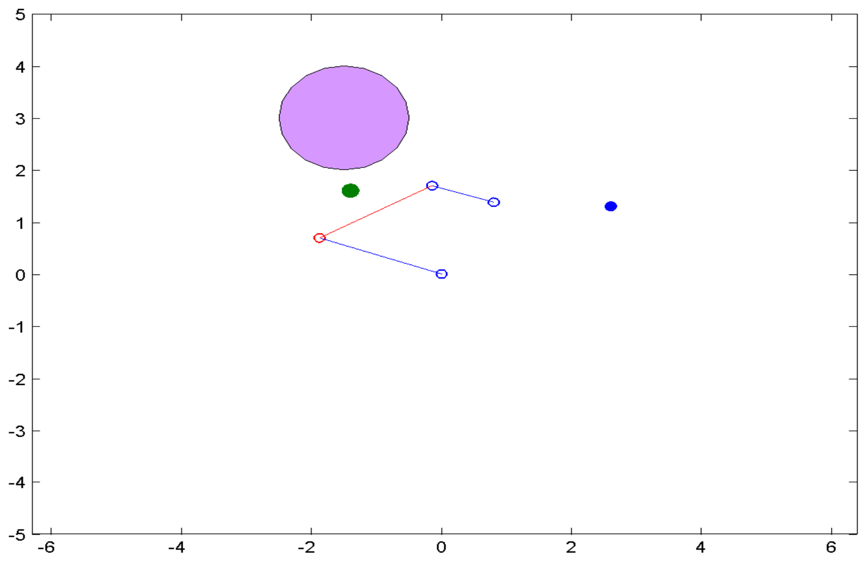


Fig. Showing the robot arm moving from the start to the goal position

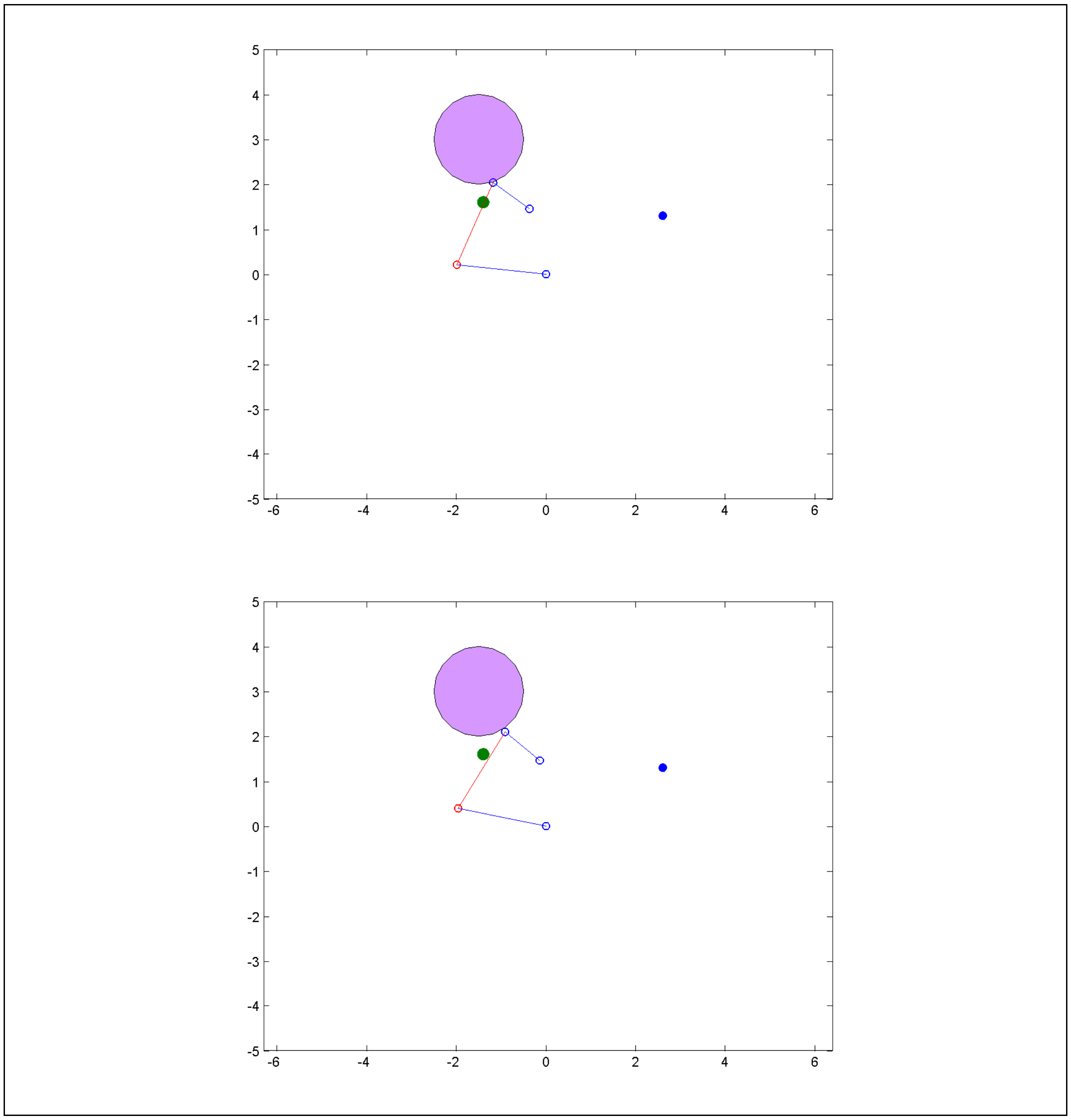


Fig. One of the parts comes close to the object

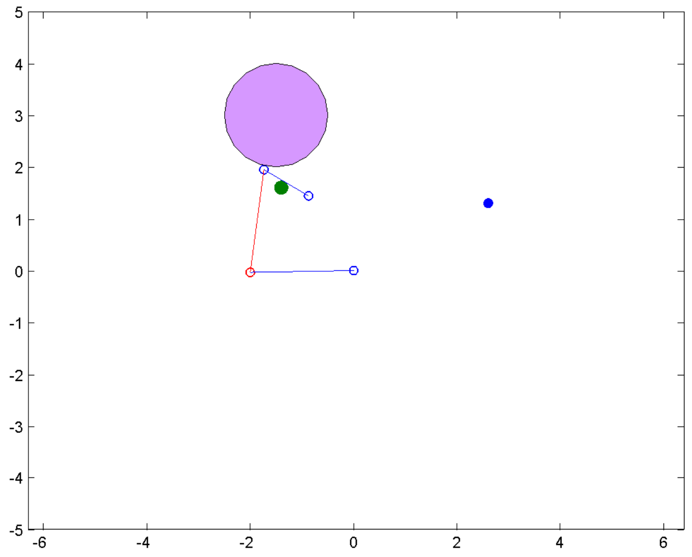
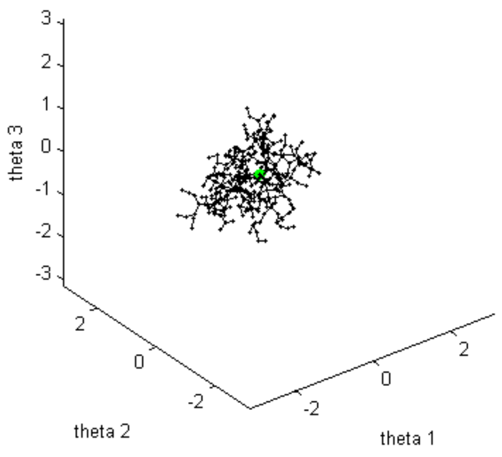


Fig. The values of theta changes to prevent collision

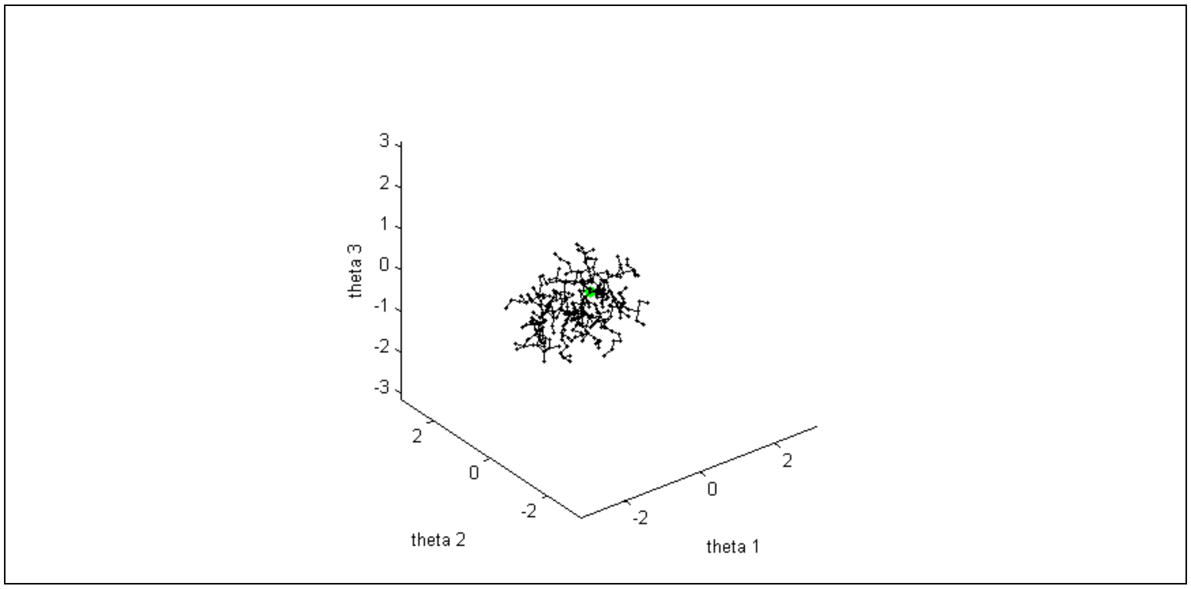
As shown in the above figures, as soon as joint 2 of the robot arm came near the circular obstacle, the value of theta 1 (angle between arm 1 and arm 2) changes to adjust the joint's path and avoid the collision.

4 Manipulation Planning - RRTs

a) Baseline:

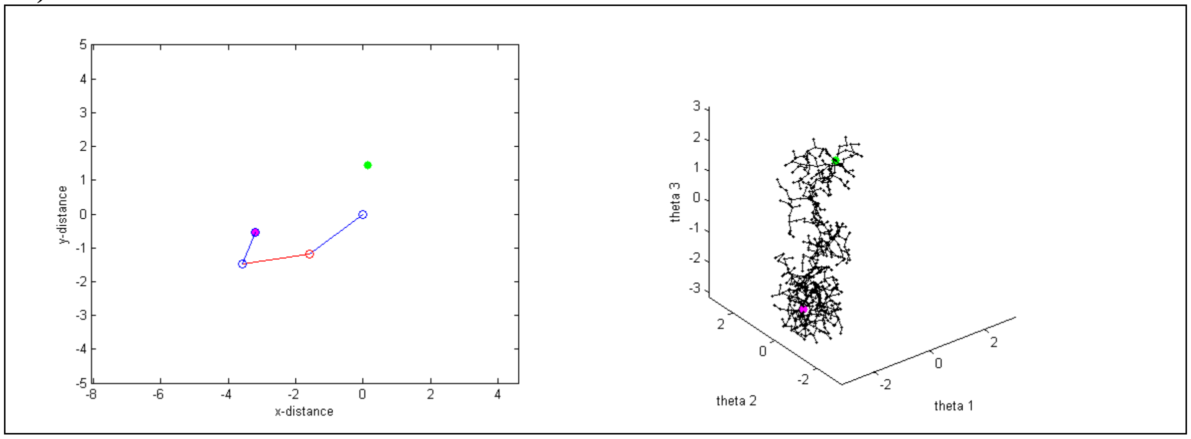


No Obstacle

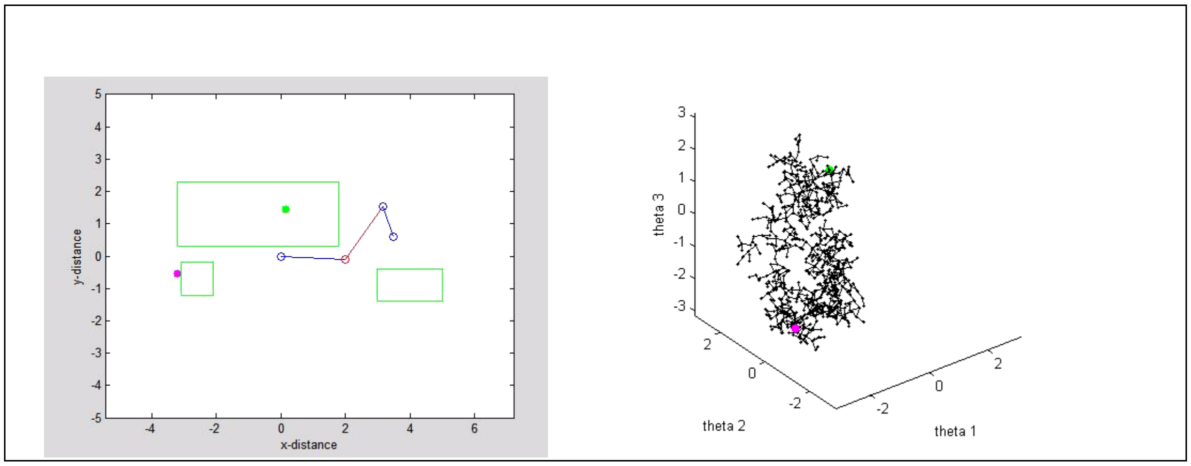


Obstacle

b) Goal-directed:

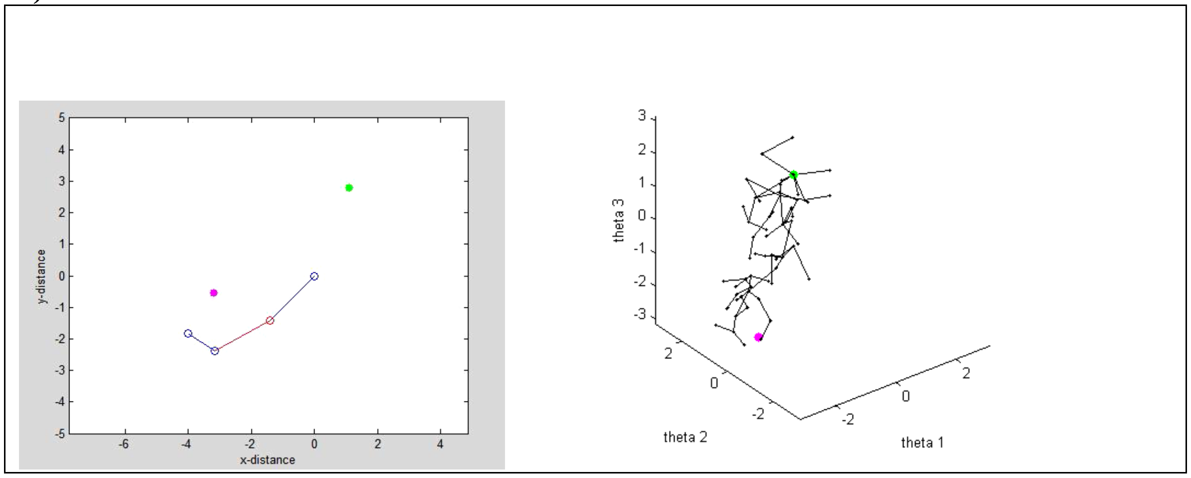


No Obstacles

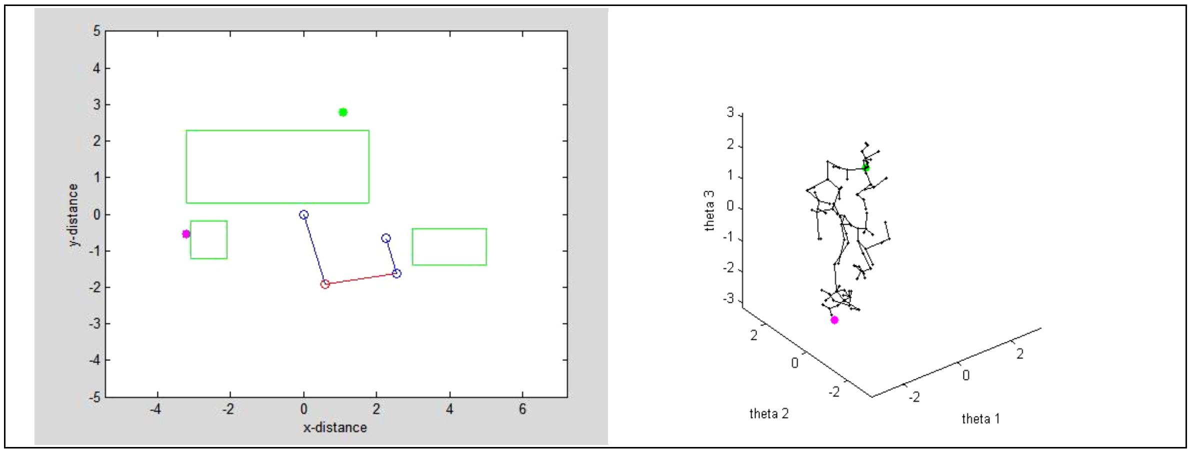


Obstacles

c) Connect

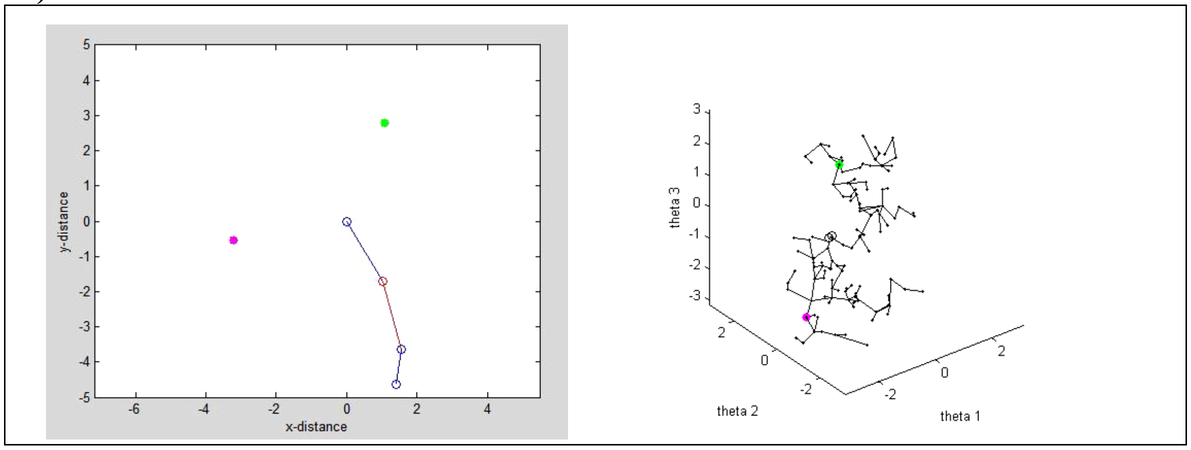


No Obstacles

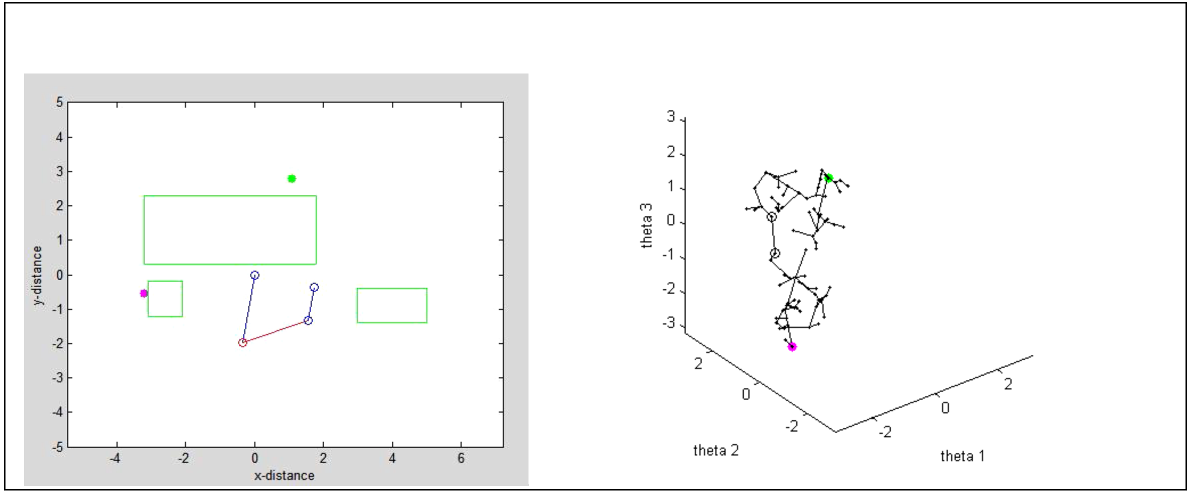


Obstacles

d) Bidirectional



No Obstacles



Obstacles

e) Analysis

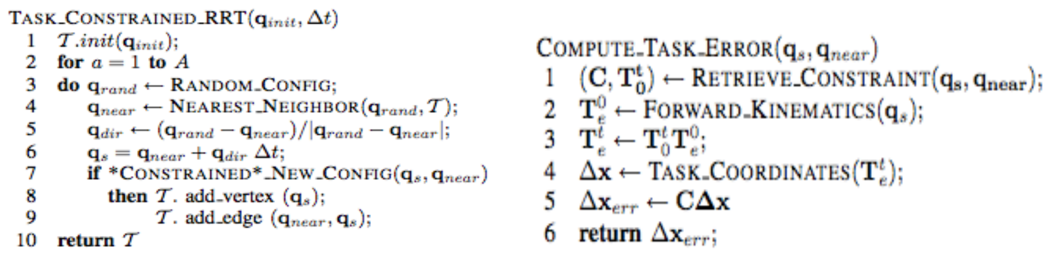
The Table 1 below shows the time, number of iterations, and path length in nodes for an average of 5 different RRTs for each condition. Based on this table, Connect RRTs are able to save some time, iterations, and path length when compared to a goal-directed RRT. Bidirectional RRTs have the advantage of time and number of iterations, however they also tend to generate a somewhat more roundabout path. That could be avoided if a better algorithm than just selecting a random point generated the goal node the bidirectional trees were growing toward. Additionally, the number of nodes generated by the goal-directed RRT could be reduced if the step size was increased. However, this causes the time to greatly increase for the case with the obstacle. Overall, as long as the number of iterations was above a certain threshold, it didn't greatly impact the performance of the RRT. It was the step size that, when varied, greatly affected the time, number of iterations, and overall "good fit" of the tree.

xType	Time (s)	Number of Iterations	Path Length in nodes
Goal	1.43198	700	28
Goal with Obstacle	4.36564	700	41.2
Connect	2.60318	203.2	14.8
Connect with Obstacle	3.30002	528.4	14
Bidirectional	0.86564	112.2	18.2
Bidirectional with Obstacle	2.32502	310	19

Table 1: Mean values for 5 iterations of each algorithm.

5 Manipulation Planning - RRT with Task Constraints

For a Task Constrained RRT the most important aspect is to specify concrete constraints. The problem was given to us with constraints of start and goal configuration as well as a value of y < 3. To implement a RRT with task constraints the paper by Dr. Stillman has been heavily utilized [1]. The paper provides a series of algorithms for task constrained RRT's with addition to computing an Error for the Tasks. The algorithm we implemented has trivially extended from the basic task constrained RRT algorithm to an algorithm with error prone Tasks for an efficient Tree.



```
TASK_CONSTRAINED_RRT($\mathbf{q}_{init}$, $\Delta t$)
    $\mathcal{T}.init(\mathbf{q}_{init})$;
    for $a = 1$ to $A$
    do $\mathbf{q}_{rand} \leftarrow$ RANDOM_CONFIG;
        $\mathbf{q}_{near} \leftarrow$ NEAREST_NEIGHBOR($\mathbf{q}_{rand}, \mathcal{T}$);
        $\mathbf{q}_{dir} \leftarrow (\mathbf{q}_{rand} - \mathbf{q}_{near})/|\mathbf{q}_{rand} - \mathbf{q}_{near}|$;
        $\mathbf{q}_s = \mathbf{q}_{near} + \mathbf{q}_{dir}\,\Delta t$;
        if *CONSTRAINED*_NEW_CONFIG($\mathbf{q}_s, \mathbf{q}_{near}$)
          then $\mathcal{T}$. add_vertex ($\mathbf{q}_s$);
                $\mathcal{T}$. add_edge ($\mathbf{q}_{near}, \mathbf{q}_s$);
    return $\mathcal{T}$
```

```
COMPUTE_TASK_ERROR($\mathbf{q}_s, \mathbf{q}_{near}$)
    $(\mathbf{C}, \mathbf{T}^t_0) \leftarrow$ RETRIEVE_CONSTRAINT($\mathbf{q_s}, \mathbf{q_{near}}$);
    $\mathbf{T}^0_e \leftarrow$ FORWARD_KINEMATICS($\mathbf{q}_s$);
    $\mathbf{T}^t_e \leftarrow \mathbf{T}^t_0\mathbf{T}^0_e$;
    $\Delta\mathbf{x} \leftarrow$ TASK_COORDINATES($\mathbf{T}^t_e$);
    $\Delta\mathbf{x}_{err} \leftarrow \mathbf{C}\Delta\mathbf{x}$
    return $\Delta\mathbf{x}_{err}$;
```

[1]

The implementation of the algorithm is done as follows: The Task Constrained RRT takes in the initial q value of the startingConfiguration and the initializes to the Tau value. It utilizes the randomization factor to produce a random configuration vector q_Rand between values of 3 to -3, further used to create a RRT tree with nearest paths. Further normalizing different q vector values to distances to find the optimal path to add to the vertex and the edge of the tree. This normalizing and task error retrieval has been done through Forward Kinematics function. Overall our implementation has been successful to provide an outlook of the Task Constrained RRT given a certain iteration count. The implementation produces two videos, which include an RRT tree and the arm.

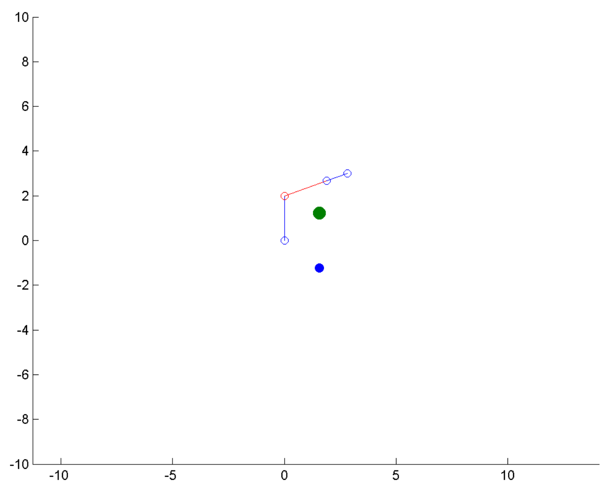


Fig 1. Arm Graphical Representation.

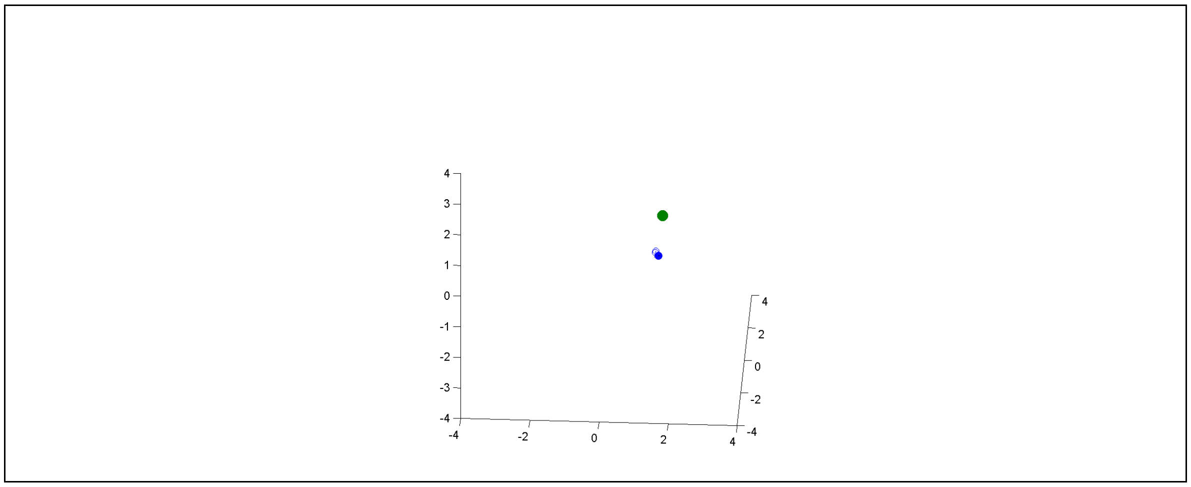


Fig 2. 3D map of the RRT Tree

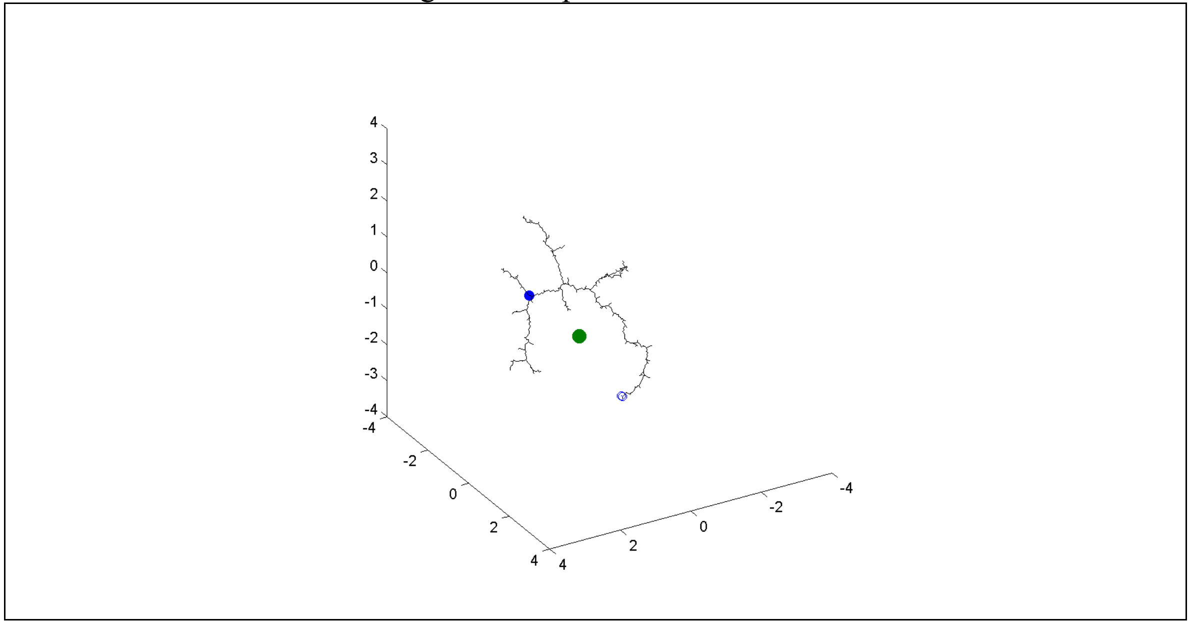


Fig 3. Growing RRT Tree.

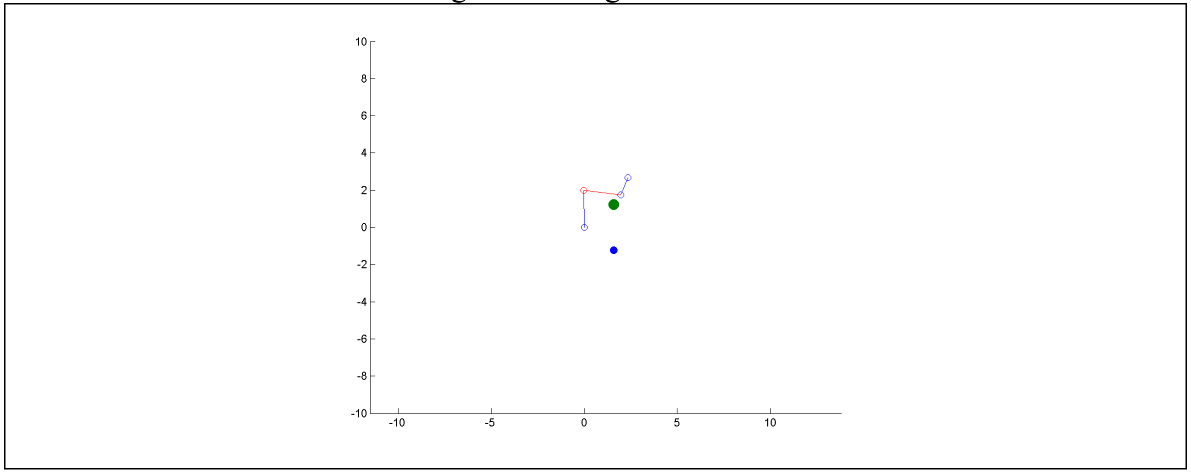
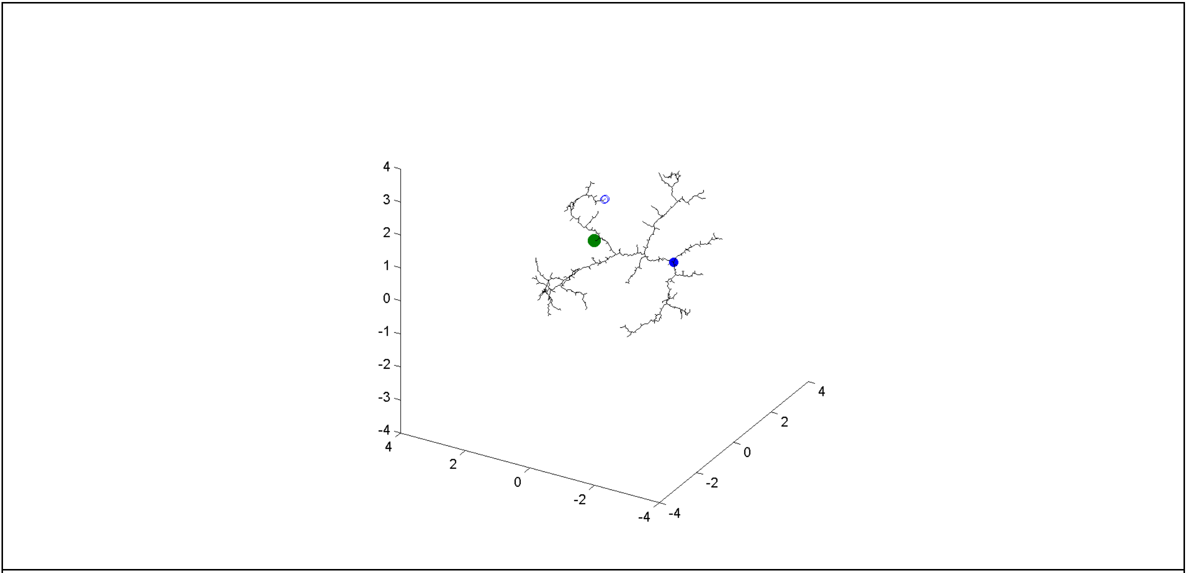


Fig 4. Arm movement, halfway through the iteration.



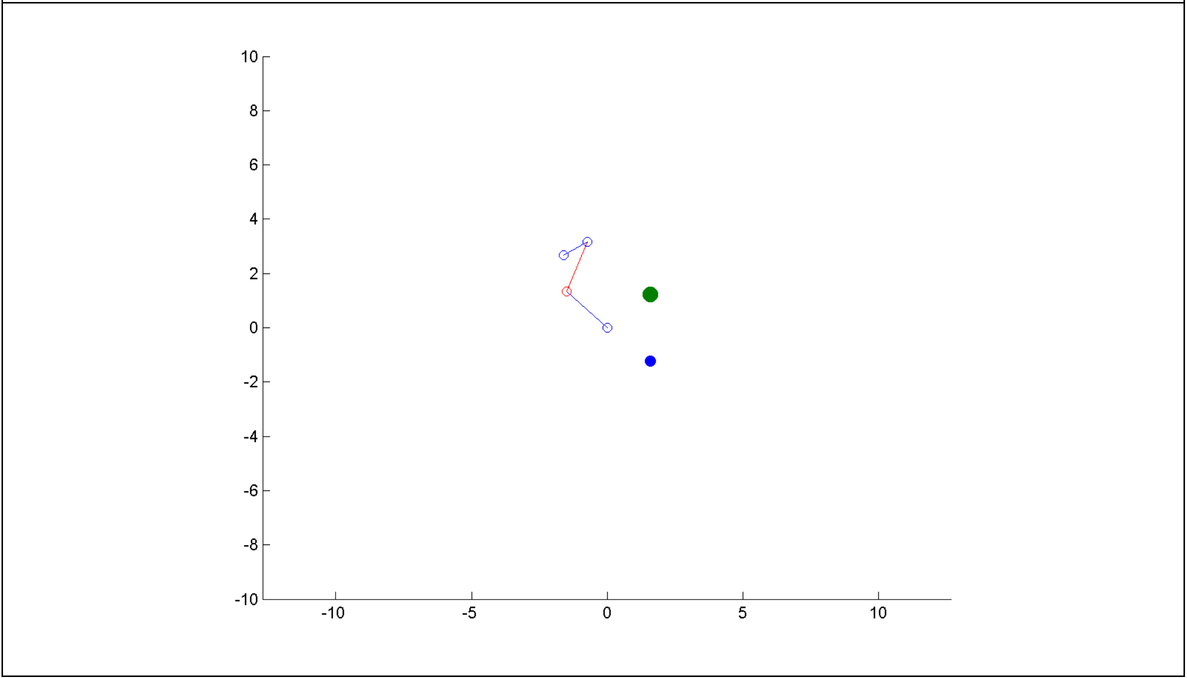


Fig 5, 6. Final Graphs.

References

[1] Stilman, Mike. ”Task constrained motion planning in robot joint space.” Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on. IEEE, 2007.