Lab 5: Trajectory Planning for the Lynx using Potential Fields

MEAM 520, University of Pennsylvania

November 20th, 2020

Venkata Gurrala, Sheil Sarda

Table of Contents

Introduction

The objective of this lab was to plan collision free trajectories through cluttered environments using the lynx robot. Artificially made obstacles within a boundary space were put into a txt file as a map for the python code to interpret before determining a feasible trajectory from a specified start and end position (q). The input q is a list of angles from theta 1 through 5. In lab 3, this was done using the RRT method. This lab aims to use the potential fields method to achieve the goal position.

Method

Unlike lab 3, this lab predominantly works in workspace rather than configuration space. The space that we are working with can be discretized to represent the volume of the robot and obstacles. From the map input given which represents the location and size of obstacles in addition to the boundaries of the discretized space, the obstacle collision space can be modelled. This method uses forces rather than a constant velocity to achieve its position. An analogy of the workings of this method can be described by understanding the behavior of magnets. The properties of magnets that allow it to attract and repel with other magnets can be simulated in determining an optimized path for a robot to achieve its end position. By virtually creating an attractive force between the start and goal configurations at each joint, we can determine the torque required to reach the goal. To avoid any obstacles, the obstacles themselves could be simulated to create repulsive forces. The net force can often lead the most straight forward path with little to no post processing required.

The potential field controller will consist of the two force equations the force of attraction and repulsion which will be strategically used to determine the general shortest path while considering the obstacles. The attractive force equation used is:

$$F_{att} = -\zeta^{*}\left(o_i(q) - o_i\left(q_f\right)\right)$$

Where the zeta represents the attractive field strength and $o_i(q) - o_i\left(q_f\right)$ represents the difference in the origin of the current and desired position. The equation considers the distance of the initial and final position of each joint. The force of attraction is proportional to the distance between the current and final position which helps the robot travel faster when the robot is initially following the trajectory. As the joint gets closer to the goal position, the velocity comes down which helps prevent overshooting.

The obstacles will emit a repulsive force which will help the robot's joints avoid them. The equation used is:

$$F_{rep,i} = \eta_i^{*}\left(\frac{1}{\rho\left(o_i(q)\right)} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2\left(o_i(q)\right)}\nabla\rho\left(o_i(q)\right)$$

$$\nabla\rho(o\_i \vdash (q \dashv) = \frac{o_i(q) - b}{\|o_i(q) - b\|}$$

η_i is a parameter used to describe the repulsive field strength. The ρ describes the shortest distance between the origin of the joint and the obstacle. ρ_0 describes the distance of influence of the obstacle. This equation is only used when the $\rho < \rho_0$ so that an attractive force is not generated from the obstacle to the joint if it is outside the sphere of influence.

Using the forward kinematics code, the current and final joint positions of each joint can be determined. Assuming that the parameters are also known, attractive forces between joints can be found and the repulsive forces at each joint by each obstacle is also found. The resultant force at each joint is calculated with the following equation:

$$F_{net} = F_{att} + F_{rep}$$

$$\tau = J_v^T F_{net}$$

The linear velocity Jacobian matrix can also be calculated before it is transposed and multiplied by the Fnet force calculated to determine the torque at each joint. The sum of all the forces can be calculated and its norm can be determined. When the robot is moving in the workspace, the normalized joint efforts can be added to the current position until it reaches within certain distance to the goal position. When it reaches a proximity to the goal position, a scaled effort can be added to prevent excessive oscillations. Using the given torques and joint positions, the following equation can be used in a while loop to direct the robot's current configuration to the next way point using a step size of α^i.

$$q^{i+1} = q^i + \alpha^{i\frac{\tau(q^i)}{\|\tau(q^i)\|}}$$

While updating q in each iteration, if the iteration count crosses over 50, it is considered to be stuck. In this case, an RRT planning method is used generate a single random point in the free configuration space. Once this random point is generated, the potential field planner attempts to converge again to a goal position. If it is unsuccessful, it will regenerate another random point and try to converge to the goal position again up to 99 randomly generated points.

Parameters to tune:

The parameters that have to be tuned would be the ζ (attractive field strength), η_i (repulsive field strength) and the obstacles distance of influence (ρ_0).

The η_i parameter has to be chosen carefully so that each obstacle's influence is considered carefully when planning a path. If the η_i is too small, the high attraction force can pull it very quickly against the relatively weaker repulsion caused by the obstacle, causing a collision. If the η_i is too large, it can reduce the number of paths available and can easily be stuck in a global minimum. η_i has to be set appropriately such that it has infinite repulsion on the surface of the obstacle so that the robot never collides with it.

If ζ is too high, the increased force of attraction could push the joint to move very quickly and run into obstacles. If ζ is too low, the robot may follow the path too slowly and with less certainty which can lead the force of repulsion from the obstacles to push it away from converging to the goal.

The $\rho_0$has to be chosen for the obstacle such that range of influence can prevent a joint from colliding with the obstacle. If ρ_0 is too large, it can result in fewer paths available. If the ρ_0 is too small, the joints can easily collide with the obstacle.

The step size, α was chosen to be large enough so that the joints would be able to converge to the local minima relatively quickly but small enough so that it will not easily collide within tight spaces. As the step size went up, the oscillations went up dramatically. To resolve this, the current and previous values in the path were averaged. Only two points were chosen to be averaged so that there would be minimum lag which would help prevent collisions.

In lab 3, if the obstacles in the space were purposely inflated to be able to plan paths that would avoid the volume of the robot to collide. This was a conservative approach to detecting and avoid obstacles. However, sometimes, it was too conservative, and it often could limit the number of paths available. In lab 5, we can avoid this potentially over-conservative method by tuning the ρ_0 parameter. This controls the distance from the obstacle that the force of repulsion is felt. If this is made higher, it would allow the robot to sense the negative torque that would oppose its motion towards an obstacle while assuming a zero-thickness robot. The ρ_0 has to be chosen such that it considers the volume of the link and joint with respect to the obstacle.

$$\rho_0 \geq Thickness\ of\ joint + adequate\ distance\ to\ repel\ from\ obstacle$$

Within this equation, the more arbitrary variable to tune is the distance added which will be adequate to repel the joint from the obstacle. Figure 1 shows the representation of the obstacle along with the ρ_0 parameter.

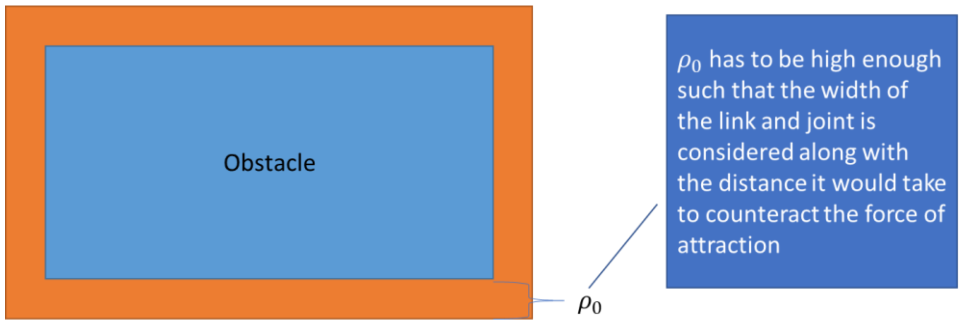

Obstacle
ρ_0
ρ_0 has to be high enough such that the width of the link and joint is considered along with the distance it would take to counteract the force of attraction

Figure 1. Representing the effects of changing rho0

The adequate distance to repel from obstacle will vary based on the input start and goal position within a map. If the start and goal positions are further away from each other, the force of attraction could be so high that joint will not have enough distance from the obstacle to feel the force of repulsion to slow down. The momentum caused by the high force of attraction will make the joint collide. If the distance between the start and goal positions were closer, the force of attraction would be low enough such that the a smaller ρ_0 could prevent collisions. If the ρ_0 is too high, the robot may not be able to find a path at all in certain situations. For example, map 2 has two obstacles that are very close together represented in figure 2. If each of the obstacle's ρ_0 is high, the potential field planner may not allow the joints to move within small spaces. Figure 2 shows an example of this where the goal position is enclosed between two obstacles with high ρ_0 so the start joint will never be able to reach the goal.

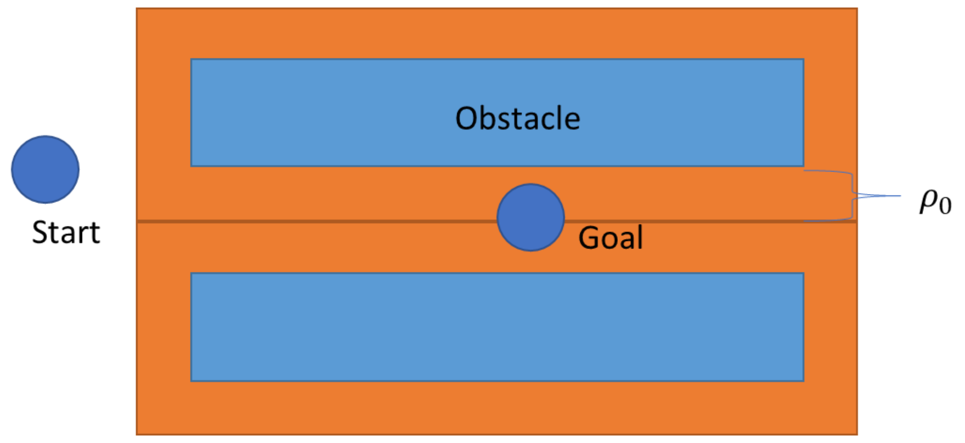


Figure 2. An instance where a high rho0 can negatively effect path planning

This is the reason why it is better to keep ρ_0 relatively small but with an η_i that is very high. This will prevent collisions from very close distances.

PARAMETERS CHOSEN (LINEAR SEARCH)

To obtain the optimal range of values and an evaluation of the robustness of the parameters, a linear search was performed for parameters η, ρ_0.

For the linear search of eta, we used 10e5 as our lower bound and 10e7 as our upper bound based on empirical findings based on the five maps used for testing. By uniformly sampling ten points in this range, the planner generated a feasible path on map 1 using the specified eta parameter. In the resulting scatter plot of path length vs. eta (included in the appendix) where a path length of zero signifies an unsuccessful run of the planner, we found that the planner had successful runs across the spectrum of input values. As a potential next step to find a narrower range of eta values while maintaining robust outcomes, we could repeat this exercise on all 5 of our test cases.

For the linear search of rho0, we used 10 as our lower bound and 100 as our upper bound, also based on empirical results. Using 50 samples, we generated a scatter plot of path length vs. eta as described above.

The figures are included in the appendix for reference.

Planning strategy when potential fields fail

An RRT planner was implemented in this lab which in principle, samples random points till a viable path is found from a specified start and end position. This was modified from the version created in lab 3. A common problem with the potential planning method is that the point it converges to may not be the goal position. This is shown in figure 3 as the start point reaches a local minimum and is unable to get out.

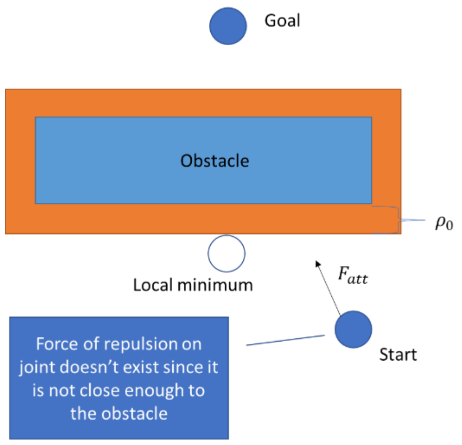

Goal
Obstacle
ρ_0
Local minimum
F_{att}
Force of repulsion on joint doesn't exist since it is not close enough to the obstacle
Start

Figure 3. An instance where a point can reach a local minimum which is not the goal position

When the potential field planner fails to converge close to the goal position, the RRT planner will attempt to create a random feasible point while considering the map of obstacles and boundaries. This is then added to the path array which the robot sequentially follows. Once it reaches this new position, it is then left to continue to use the potential field planner to reach the goal position. If the robot fails to converge again, the RRT planner will give another single random point in space and will leave the robot to follow the potential field planner to get to the goal. If this is unsuccessful, it will try again up to 99 points.

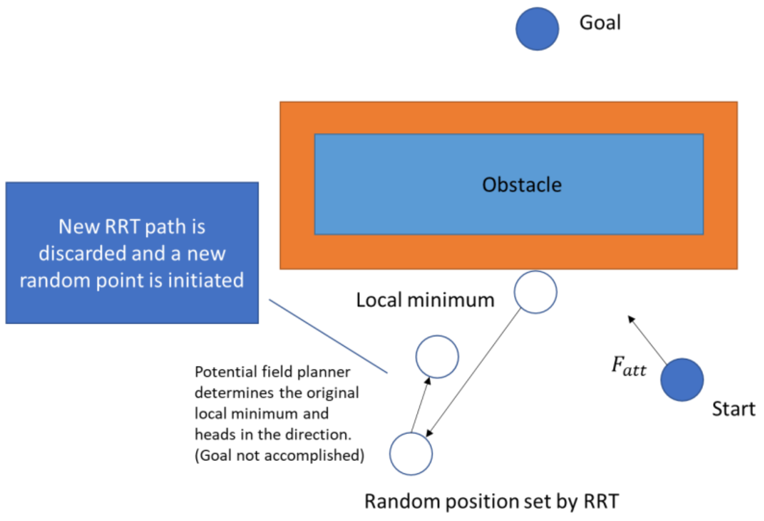

Goal
Obstacle
New RRT path is discarded and a new random point is initiated
Local minimum
F_{att}
Start
Potential field planner determines the original local minimum and heads in the direction. (Goal not accomplished)
Random position set by RRT

Figure 4. An unsuccessful attempt where the RRT method is used to generate a random point to converge to the goal

Figure 4 shows a random point generated using RRT from lab 3. After this point is given, the potential field planner will take over and attempt to converge to the goal position. If the planner still does not get to the goal, a new random point is generated again by RRT as shown in figure 5. In this case, once the potential field planner attempts to reach the goal, it is finally successful.

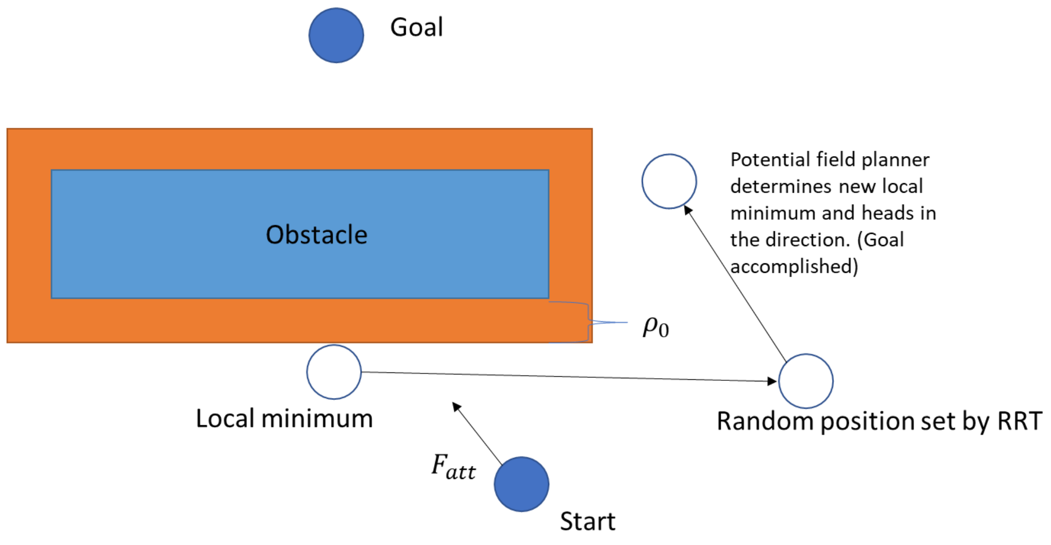

Goal
Obstacle
Potential field planner determines new local minimum and heads in the direction. (Goal accomplished)
ρ_0
Local minimum
Random position set by RRT
F_{att}
Start

Figure 5. A successful attempt to use RRT to generate a random point to converge to the local minimum

A common issue we experienced with Gazebo was that m any positions were not given enough time for the RRT planner to reach. While some positions can be reached relatively quickly, some positions took much longer before the next waypoint was given. This would often cause collisions. To solve this issue, the next waypoint was not given till each joint reached 0.01 rad of the position specified. This was successful in giving enough time for the robot to navigate to the desired position.

Evaluation

Empty Map:

With this map, the attraction forces were tested. From the zero position, each individual joint was given a non-zero configuration while the rest of the joints were kept at zero. It was checked whether there were any excessive oscillations near the goal or any overshoot. The step size was also analyzed to make sure it was able to reach the goal without taking over 50 iterations. We found that joint 4 did not always get close enough to the goal position unless the step size was decreased by an excessive amount. For this reason, the algorithm checks whether the first 3 joints are able to reach within the region of the goal before giving it the goal position.

Map 1:

Start: Zero position

Goal: [0,0,1.4,0,0]

From this start and end positions, there was little room given for the robot to reach the goal. The start position was located above the obstacle while the goal was located below the obstacle. Therefore, there was a local minimum right above the obstacle. The RRT code was called to determine 1 random walk that would find the goal to be the new local minimum. This was eventually accomplished after 3 minutes. Figure 6 shows the q value over each iteration for each joint. In a few cases, the robot's wrist still collided with the base of the robot.

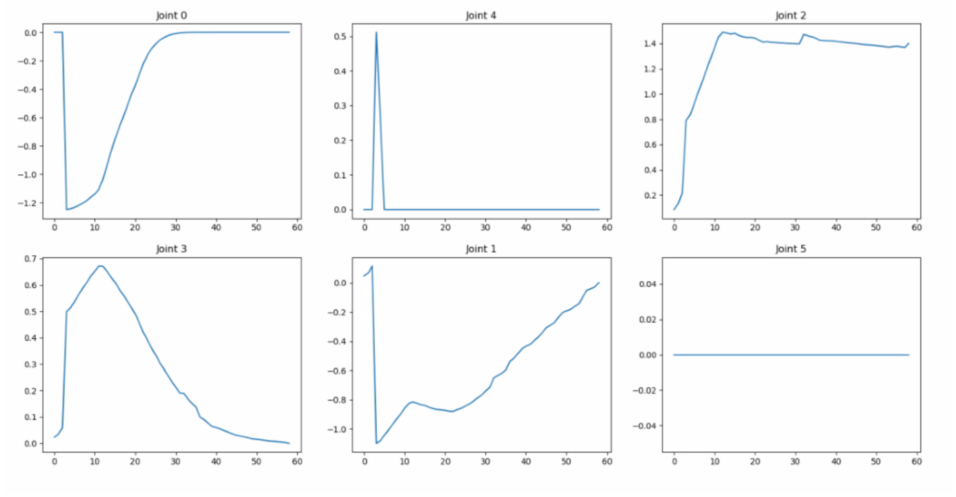


Figure 6. Trajectory charts for map 1 configuration

Map 2:

Start: [-1.2,0,0,0,0,0]

Goal: [1.4,0,0,0,0,0]

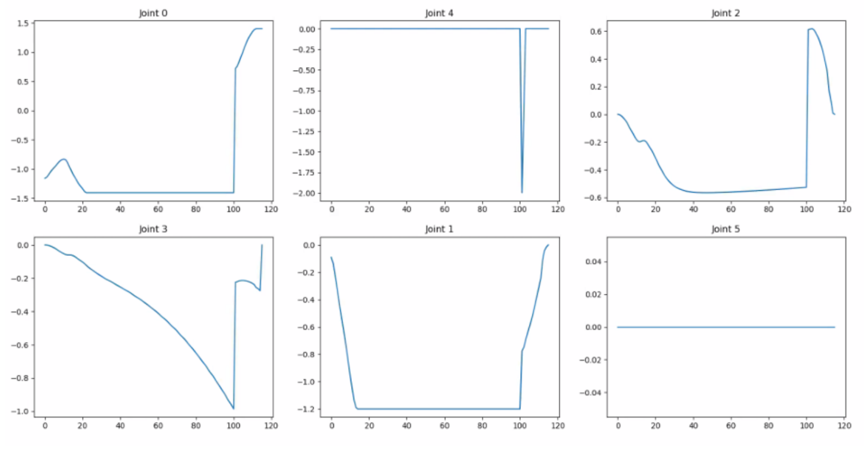


Figure 7. Trajectory charts for map 2 configuration

With the current map and initial conditions, there is a local minimum. This is easily detected and the RRT code determines a random configuration that will allow the robot to find the new local minimum across from the two parallel obstacles. The planning was done within 30 seconds and it did not collide in Gazebo. It took close to 120 iterations to get to the final goal. Figure 7 shows how RRT helped jolt the positions of all the joints such that it helped the joints 0-3 to converge to the goal position much more quickly.

Map 3:

Start: Zero position

Goal: (1.4,0,1,0,0,0)

The planner took less than 10 seconds to generate and was able to successfully reach the goal position. During planning, joint 2 converged to a value close to 1 but was supposed to converge to 1. This triggered the random RRT generator which helped it to successfully converge to the value of 1. Similarly, joint 3 was stuck in a local minima so RRT was able to help the potential field planner to converge to 0. To reach this tight space, it required different parameters: eta being 8e7 and rho0 being 60.

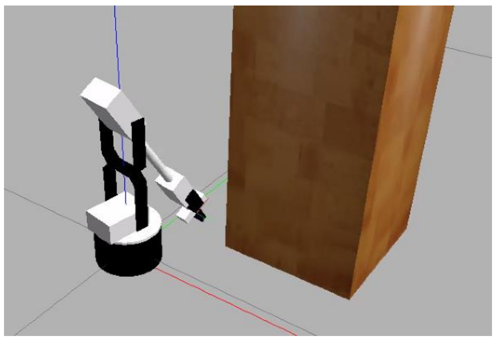


Figure 8. Robot successfully reaching goal position with the map 3 configuration

Start = Zero position

Goal = [0, 0, 1.4, 0, 0, 0]

This configuration was also successful with the planner taking less than 10 seconds to find a viable path. Figure 9 shows how joints 1, 2 and 3 was able to benefit from the RRT method as they quickly converged after the jolt provided. It was collision free in Gazebo as well.

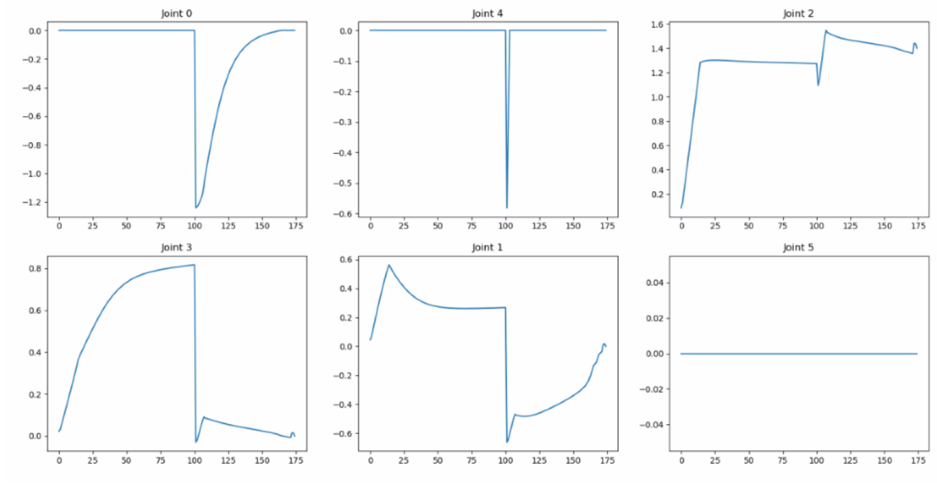


Figure 9. Trajectory charts for 2nd map 3 configuration

Map 4:

Start: Zero position

Goal: (0,0,-1.4,0,0,0)

This map was unique in that its enclosure made it difficult to reach the goal position. The RRT code was helpful it helping joint 2 and 3 to reach the goal position. This configuration was accomplished in less than 20 seconds without any collisions on Gazebo.

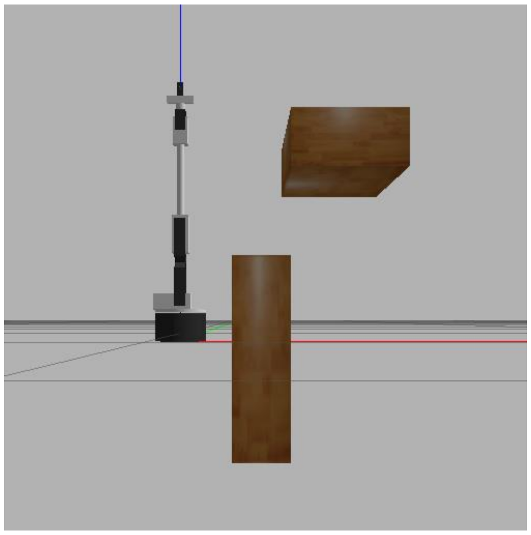


Figure 10. Successful Map 4 trajectory

Start=[1,0.5,0,0,0,0]

Goal: [0,-1.2,0,0,0,0]

This path initially had an issue with the convergence of joint 0 and 2. This triggered the RRT code which was able to give it the jolt necessary to find the new global minimum. This path was still formed in less than 20 seconds without any collisions in Gazebo.

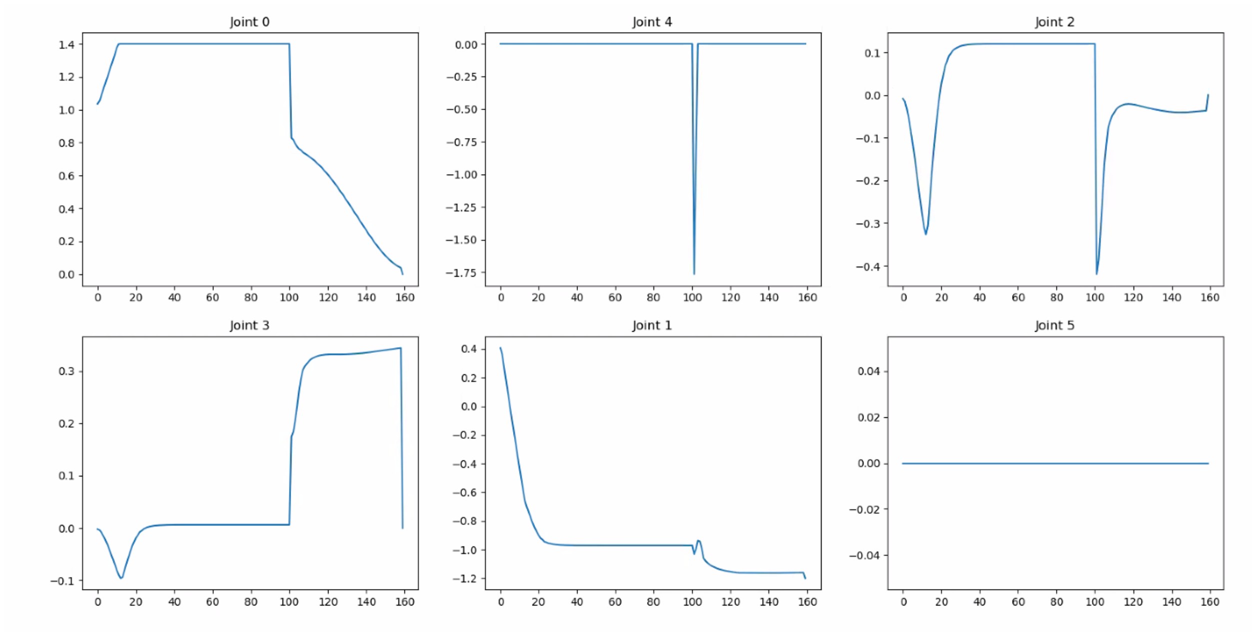


Figure 11. Map 4 configuration trajectory charts

Map 5:

Start: Zero Position

Goal: (0.5,-0.5,-pi/2,-1,0,0)

Map 5 was created to test the planner to use joints 3-e from one tight space to another. Previous maps predominantly involved the rotation of joints 1 and 2 to move the arm away from enclosed obstacles. This map and configuration combination tested the robot's dexterity in the enclosed area. The RRT planner which was also tested in this map and configuration took over 1 minute to compute a path before it was post-processed. The A* planner took even longer at 6-8 minutes at certain goal positions. The potential field planner was able to accomplish this task within 20 seconds successfully. There were no collisions with the Gazebo implementation. Figure 12 shows that one RRT point was given before it converged to the goal position.

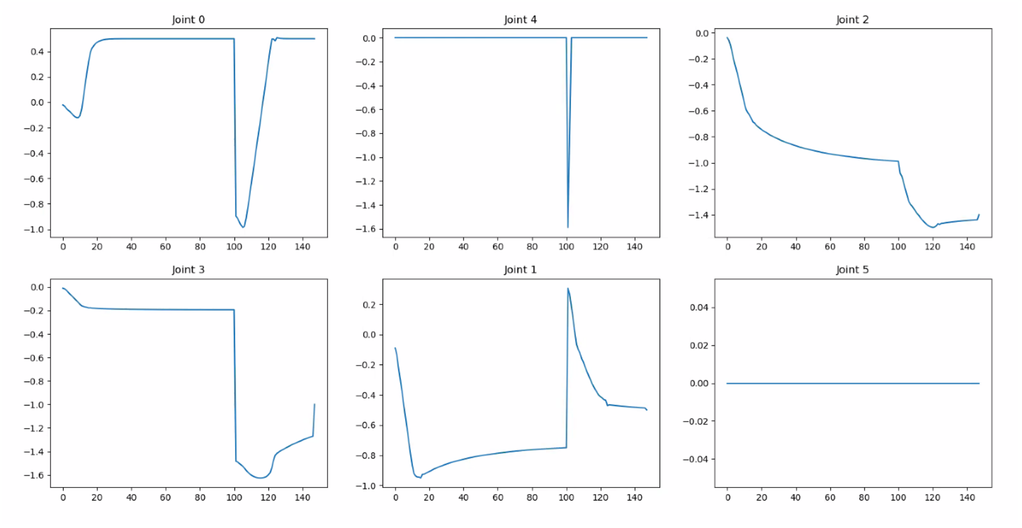


Figure 12. Map 5 trajectory charts

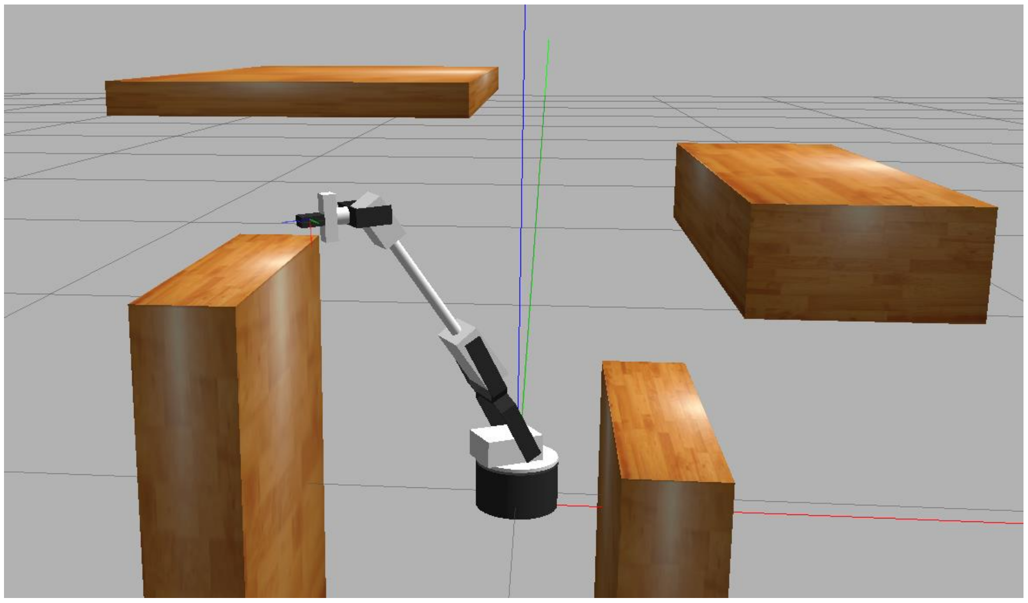


Figure 13. Correct positioning of robot to goal position in map 5

Analysis

Though the theory of potential field planners seems simple, their drawbacks were very clear to be seen in this lab. The potential field planner by itself didn't always guarantee to converge to the goal position. This was when the random point sampled by RRT usually helped it by giving it a "nudge" till it was able to successfully converge. The code was developed such that the RRT code only provided one feasible point in the free configuration space such that the robot can converge. This was done purposely to make sure that the potential field planner was still responsible to get it to the final goal position.

Common issues:

Since the forces in each joint are motivated by the differences in the current and goal joint positions, some movements can be difficult or even impossible to achieve. If a goal configuration is set such that the joint positions would not change, the robot will not be able to move to that new configuration. For example, if the start position was set to [0,0,-pi/2,0,0,0] and the goal was set to [0.3,0, -pi/2,0,0,0], the potential field planner will fail. This is a fundamental flaw with using the potential field path planning method. It is also the reason why the gripper position and the rotating wrist are manully set in the beginning rather than using the planner to set them. It is also the reason why almost all of the maps required the random point generated from the RRT code before converging to the goal position.

Another major flaw is the inability to have parameters that can be used for all maps. Some maps had more ideal parameters to prevent collisions than others. In our experiments, we found no general parameters that could be used for all maps with few to no collisions. However, the parameters mentioned in the methods section worked for most of the maps and configuration.

An advantage of using this method would be to be able to take very short paths to the goal configuration. Compared to the RRT implementation, the potential field path planner used approximately the same amount of time to determine a path while achieving a more optimized path most of the times it was run. This is the major appeal of the potential planning method. The jerk of the joints was also less predominant compared to the RRT method. Since we are able to vary the torques of each joint smoothly based on the proximity to attraction and repulsive forces in the map, there is little to no impulses in velocity. The jerk issue had previously caused unnecessary collisions with the RRT planner.

Potential improvements:

If we had more time with the lab, we would try to find a way to determine feasible paths by simulating walks around an obstacle boundary. This would help find more optimized configurations that would avoid obstacles to reach the goal position. Using the RRT method of generating one point worked but would not result in the most optimized path.

Another aspect that could be improved is the parameters used to achieve successful configurations. There were some parameters that were better tuned for certain maps over others. Therefore, we would try to set an adaptive parameter change if there are excessive iterations required to compute a viable path. For example, if there are excessive collisions, the rho0 could be slightly increased to prevent these collisions. The step size is the most important parameter that can become more adaptive to achieve a path with fewer waypoints in the path in the beginning. It could be made finer once a joint reaches a certain, close distance to the goal position.

All the paths given to Gazebo were excessively long for gazebo to process. There could have been more interpolated waypoints that would still avoid collisions. This could have helped the robot achieve complex paths in a much quicker way in gazebo.

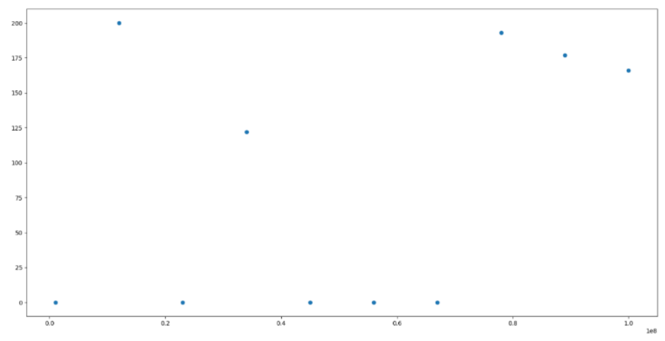
Appendix

Pseudocode for potential field path function

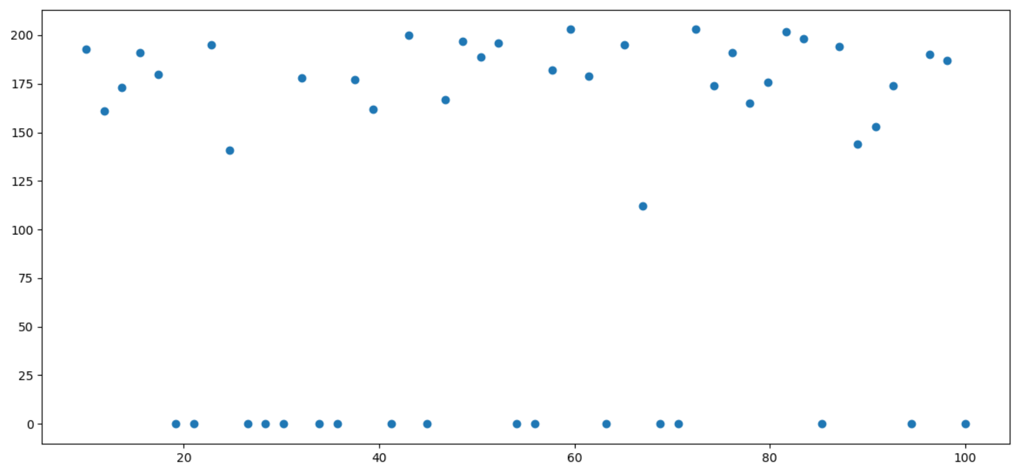
```
1. Find the path of the new point in space given the initial point
2. Append it to a variable named path
3. Start a while loop till the variable isDone=true where isDone represents if we successfully reach the goal
4. Within the while loop, a new configuration is calculated given the last configuration stored in the path array
5. The output of PotentialFieldStep is stored in the path array
6. The average of current and previous q is stored in the path array
7. The difference between the new path and the goal is determine for the first 3 joints. If all of them are within 0.04, the goal is accomplished and isDone is set to True
8. Assuming that the previous condition is not met, it checks whether the iteration count is at or over 100. At this point, it will assume that no solution has been determined or that it is stuck at a local minimum
9. If the previous condition is met, the RRT code runs to find a new random point in free configuration space. We append this to the path array and check if it converges. If it doesn't, it will redetermine a new random point.
```

Pseudocode for potential field step function

```
1. Parameters and arrays are initialized
2. The disttobox function is used to determine the unit vector that represents the negative gradient from the obstacles
3. The force of repulsion and attraction are determined and added together
4. The jacobians are calculated and multiplied with the net force to find the torque
5. Torque is made into a unit vector and added to the current q position
6. If the current q position array has Nan values, it will return delnan as true so that the path function can remove that configuration from the path array
```



Linear search on eta



Linear search on rho 0