Introduction to Robotics Lab 4

Karin Dyer, Saibernard Yogendran

1 Methods

The following methods were implemented to create a potential field planner for the Panda Robot Arm. The robot feels forces from an artificial potential field U defined across the configuration space. U is designed to attract the robot to the target configuration and repel it from the boundaries of obstacles. The potential field is written as a combination of attractive and repulsive fields.

$$U(q) = U_{attractive}(q) + U_{repulsive}(q) \quad \text{Eq. 1}$$

This can be viewed as an optimization problem, where we want to find the global minimum in U starting from qs, the starting configuration. We can use gradient descent to guide the robot to the target configuration (See Sect. 1.4):

$$\tau(q) = -\nabla U(q) \quad \text{Eq. 2}$$

1.1 Attractive Force

The <u>conic well potential</u> attractive field for joint *i* is written as:

$$U_{attractive,i}(q) = \left\| o_i(q) - o_i(qf) \right\| \quad \text{Eq. 3}$$

Where q is the joint configuration, *qf* is the target joint configuration and o_i is the translation vector of joint *i* for configuration q and *qf*.

Differentiating this equation will give us the attractive force on joint *i*, $F_{attractive,\, i}$.

$$F_{attractive,\, i}(q) = -\frac{o_i(q) - o_i(qf)}{\left\| o_i(q) - o_i(qf) \right\|} \quad \text{Eq. 4}$$

The limitations of using conic well potential include discontinuity at the goal position. This is due to the step size being too large, causing the position to miss the goal and resulting in chatter of the robot arm. To address this problem, we can utilize the <u>parabolic well potential</u> in certain situations.

The <u>parabolic well potential</u> attractive field for joint *i* is written as:

$$U_{attractive,i}(q) = \frac{1}{2}\xi_i \left\| o_i(q) - o_i(qf) \right\|^2 \quad \text{Eq. 5}$$

Where ξ_i is the attractive field strength set by the user.

Differentiating this equation will give us the attractive force on joint *i*, $F_{attractive,\, i}$.

$$F_{attractive,\, i}(q) = -\xi_i(o_i(q) - o_i(qf)) \quad \text{Eq. 6}$$

The limitations of using parabolic well potential include the gradient being extremely large when the current position, q, is far from the target position, *qf*. To overcome these limitations, we can implement the conic well potential when q is far away from *qf*, and switch to parabolic well potential when q is within a certain distance threshold from *qf*.

1.2 Repulsive Force

The repulsive force is related to the obstacle position and the distance between the current configuration and the obstacle.
When the shortest distance between the current position of joint *i* and the obstacle is larger than the radius of influence of the obstacle, $\rho_i(q) > \rho_0$, The potential field is zero, $U_{repulsive\,,i}(q) = 0$. When $\rho_i(q) \geq \rho_0$, the potential field on joint *i* is written as:

$$U_{repulsive\,,i}(q) = \frac{1}{2}\eta_i \left(\frac{1}{\rho_i(o_i(q))} - \frac{1}{\rho_0}\right)^2 \qquad \text{Eq. 7}$$

Where η_i is the repulsive field strength set by the user, and $\rho_i(o_i(q))$ is the shortest distance between the current position of joint *i* and the obstacle.

Differentiating this equation will give us the repulsive force on joint *i*, $F_{repulsive,\,i}$.

$$F_{repulsive,\,i}(q) = \eta_i \left(\frac{1}{\rho_i(o_i(q))} - \frac{1}{\rho_0}\right)\left(\frac{1}{\rho_i^2(o_i(q))}\right)\nabla\rho_i(o_i(q)) \qquad \text{Eq. 8}$$

Where $\nabla\rho_i(o_i(q)) = \frac{o_i(q)-b}{\|o_i(q)-b\|}$ and *b* is the point on the obstacle boundary closest to $o_i(q)$

1.3 Joint Efforts

The joint efforts are defined as the torque on a joint due to the potential field. This consists of both the attractive and repulsive joint efforts, which use the attractive and repulsive forces on each joint due to the potential field. The attractive joint efforts for joint *i* is written as:

$$\tau_{attractive,\,i} = J_{v,\,i}^{T} F_{attractive,\,i}(q) \qquad \text{Eq. 9}$$

Where $J_{v,\,i}^{T}$ is the linear velocity Jacobian for joint *i* transposed (see Lab 3)

Similarly, the repulsive joint efforts for joint *i* is written as:

$$\tau_{repulsive,\,i} = J_{v,\,i}^{T} F_{repulsive,\,i}(q) \qquad \text{Eq. 10}$$

Where $J_{v,\,i}^{T}$ is the linear velocity Jacobian for joint *i* transposed (see Lab 3)

We can then calculate the combined joint efforts:

$$\tau_i = \tau_{attractive,\,i} + \tau_{repulsive,\,i} = J_{v,\,i}^{T} F_{attractive,\,i}(q) + J_{v,\,i}^{T} F_{repulsive,\,i}(q) \qquad \text{Eq. 11}$$

1.4 Gradient Descent

To perform gradient descent, we have a starting configuration *qs*, and a target configuration *qf*. If the magnitude of *qs* - *qf* is larger than an error threshold (set around *qf*), we must take a step along the gradient. This step is defined as:

$$q_{next\ step} = q_{current} + \alpha\frac{\tau_i(q_{current})}{\|\tau_i(q_{current})\|} \qquad \text{Eq. 12}$$

Where α is a fixed step size in the joint space and $\frac{\tau_i(q_{current})}{\|\tau_i(q_{current})\|}$ is the normalized combined joint efforts.

$q_{current}$ is replaced with $q_{next\ step}$, and the descent is looped for another iteration. This loop will continue until the current configuration is within the threshold of *qf*. When the robot reaches the target configuration, a list of all the configurations generated by the gradient descent will be returned, representing a map of the robot's journey to the target.

There can be instances where the robot becomes stuck in a local minimum, where the attractive forces from the target configuration pull the robot but repulsive forces from the obstacle prevent the robot from moving closer to the goal (Figure 1).

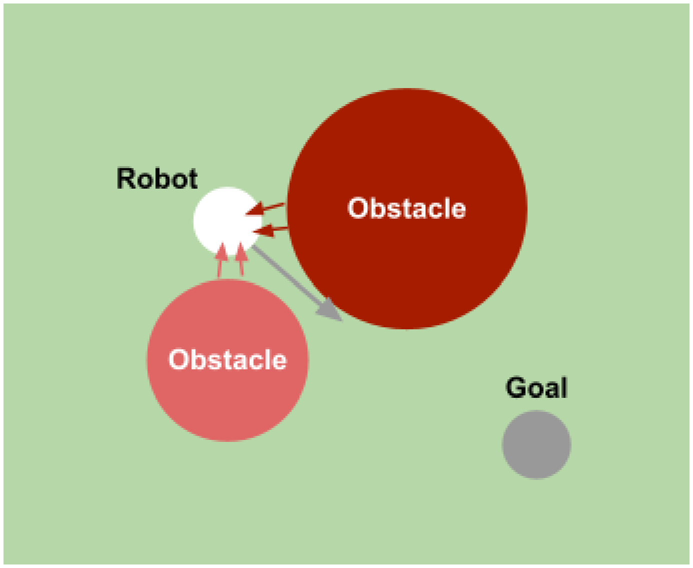


Figure 1: Robot stuck at local minima

To escape a local minima, we can detect a lack of motion in the q steps (recall we have a map of the configurations, q, of each step in the gradient descent) and generate a random configuration to move the configuration away from the obstacles.

1.5 Collision Detection

To successfully reach the target configuration, the robot must not collide with the obstacles. We can set a distance of influence around the obstacles and check that none of the joint positions overlap with the points included in the distance of influence. However, the robot links must also not collide with the obstacles. As the repulsive forces are only acting on the joints, we must add floating repulsion points along the links. To check for collisions along the links, we can generate lines that represent the links by joining successive pairs of joints positions i.e. $o_2(q)$ and $o_1(q)$, $o_3(q)$ and $o_2(q)$ etc. We can then check if these lines intersect with the points included in the distance of influence.

1.6 Code (Saibernard's)

- Initialize attractive field strength (ξ_i).
- If position is outside the set threshold for goal, calculate attractive force from Eq 4. If position is inside the threshold, calculate attractive force from Eq 6.

- Initialize repulsive field strength (η_i) and radius of region of repulsion (ρ_0).
- Obtain the shortest distance from the obstacle to the joint (ρ_i) and the unit vector in the direction from the obstacle to the joint using a predefined distance function.
- If $\rho_i = 0$, the repulsive force is set to 1000 to avoid it going to infinity, if the obstacle size is zero, the repulsive force is zero, else, calculate repulsive force from Eq 8.
- Compute the total forces for each joint by adding the attractive and repulsive forces and storing in a 3x7 matrix.
- Find the Jacobian of the current configuration q and slice the Jacobian for each joint by extracting columns up to the desired joint with the remaining columns set to zero.
- Calculate the individual joint torques by multiplying the joint Jacobian with the joint force for each joint.
- Calculate the combined joint torques by adding the individual joint torques.
- Calculate the normalized joint efforts/gradient (dq) by multiplying the predetermined step size (α) with the normalized joint torque.
- While the robot is not at the goal, perform gradient descent from Eq 12.
- Check for joint collisions by comparing joint positions with the obstacle boundaries. Raise error if collision detected.
- Check for link collisions by generating a line connecting sequential joints and checking points in the line with the obstacle boundaries. Raise error if collision detected.
- Check for local minima by comparing the difference between the last 3 steps. If no large distance changes occur within the last three steps, generate a random step to escape.
- Check for goal by checking if the norm of the current and goal position is smaller than the threshold for goal.

2 Evaluation

2.1 Testing Process

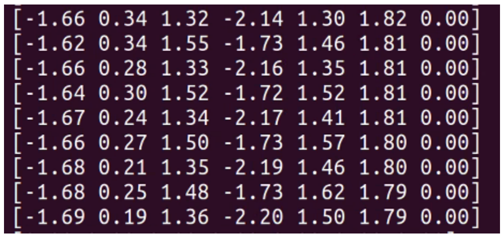
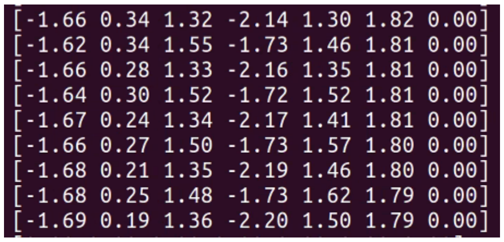
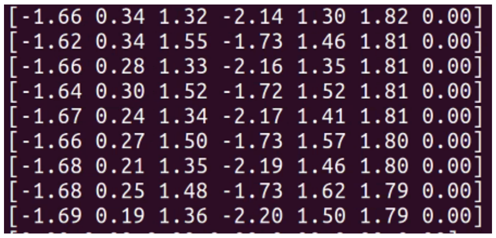
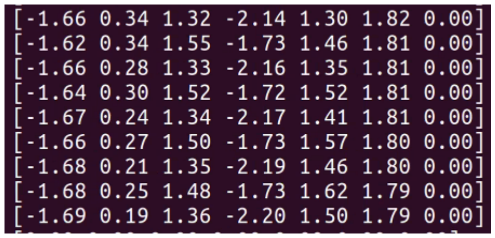
Our path planner succeeded in all the iterations of the same start and goal configurations with the following conditions (Table 1):

Table 1: Conditions for Potential planner runs

Variable	Value
Gradient descent step size α	0.5
Radius of influence around the obstacle ρ_0	[0.01, 0.01, 0.01]
Repulsive field strength η_i	0.01
Attractive field strength ξ_i	500
Start configuration	[0, -1, 0, -2, 1.57, 0]
Goal configuration	[-1.2, 1.57 , 1.57, -2.07, -1.57, 1.57, 0.7]

To evaluate the effectiveness of the planner, we can note how long it takes the planner to find a path. This is known as the runtime. Planners with a long runtime are less desirable as results are computationally costly and time consuming. Table 2 shows the runtime and path values of four trials under the same conditions.

Table 2: Trial results for runtime and path values

Trial	Runtime	Last n rows of Path
1	0.81	
2	1.83	
3	2.19	
4	1.77	

```
[-1.66 0.34 1.32 -2.14 1.30 1.82 0.00]
[-1.62 0.34 1.55 -1.73 1.46 1.81 0.00]
[-1.66 0.28 1.33 -2.16 1.35 1.81 0.00]
[-1.64 0.30 1.52 -1.72 1.52 1.81 0.00]
[-1.67 0.24 1.34 -2.17 1.41 1.81 0.00]
[-1.66 0.27 1.50 -1.73 1.57 1.80 0.00]
[-1.68 0.21 1.35 -2.19 1.46 1.80 0.00]
[-1.68 0.25 1.48 -1.73 1.62 1.79 0.00]
[-1.69 0.19 1.36 -2.20 1.50 1.79 0.00]
```

```
[-1.66 0.34 1.32 -2.14 1.30 1.82 0.00]
[-1.62 0.34 1.55 -1.73 1.46 1.81 0.00]
[-1.66 0.28 1.33 -2.16 1.35 1.81 0.00]
[-1.64 0.30 1.52 -1.72 1.52 1.81 0.00]
[-1.67 0.24 1.34 -2.17 1.41 1.81 0.00]
[-1.66 0.27 1.50 -1.73 1.57 1.80 0.00]
[-1.68 0.21 1.35 -2.19 1.46 1.80 0.00]
[-1.68 0.25 1.48 -1.73 1.62 1.79 0.00]
[-1.69 0.19 1.36 -2.20 1.50 1.79 0.00]
```

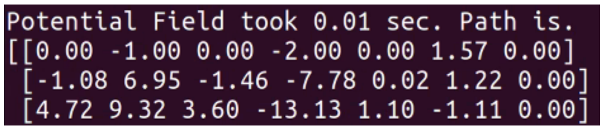
```
[-1.66 0.34 1.32 -2.14 1.30 1.82 0.00]
[-1.62 0.34 1.55 -1.73 1.46 1.81 0.00]
[-1.66 0.28 1.33 -2.16 1.35 1.81 0.00]
[-1.64 0.30 1.52 -1.72 1.52 1.81 0.00]
[-1.67 0.24 1.34 -2.17 1.41 1.81 0.00]
[-1.66 0.27 1.50 -1.73 1.57 1.80 0.00]
[-1.68 0.21 1.35 -2.19 1.46 1.80 0.00]
[-1.68 0.25 1.48 -1.73 1.62 1.79 0.00]
[-1.69 0.19 1.36 -2.20 1.50 1.79 0.00]
```

```
[-1.66 0.34 1.32 -2.14 1.30 1.82 0.00]
[-1.62 0.34 1.55 -1.73 1.46 1.81 0.00]
[-1.66 0.28 1.33 -2.16 1.35 1.81 0.00]
[-1.64 0.30 1.52 -1.72 1.52 1.81 0.00]
[-1.67 0.24 1.34 -2.17 1.41 1.81 0.00]
[-1.66 0.27 1.50 -1.73 1.57 1.80 0.00]
[-1.68 0.21 1.35 -2.19 1.46 1.80 0.00]
[-1.68 0.25 1.48 -1.73 1.62 1.79 0.00]
[-1.69 0.19 1.36 -2.20 1.50 1.79 0.00]
```

The potential field planner includes many variables that are set by the user. These include attractive field strength ξ_i, repulsive field strength η_i, radius of influence around the obstacle ρ_0, the threshold for switching from conic to parabolic well potential, and the gradient descent step size α. To investigate the relationship between these variables and the planner outcome and efficiency, we tested changing these variables and observing the resulting runtime, path success, and collisions.

2.2 Testing Results

By setting the gradient descent step size to 10, we obtained the result in Figure 2:



```
Potential Field took 0.01 sec. Path is.
[[0.00 -1.00 0.00 -2.00 0.00 1.57 0.00]
 [-1.08 6.95 -1.46 -7.78 0.02 1.22 0.00]
 [4.72 9.32 3.60 -13.13 1.10 -1.11 0.00]
```

Figure 2: Generated path of gradient descent step size = 10

The planner only ran for three iterations as the step size was so large. Further values of alpha were tested ranging from 0.001 to 10. The results are tabulated below (Table 3):

Table 3: Gradient descent values and resulting planner iterations

α value	**Planner Iterations**
0.001	Exceeded maximum iterations of 500
0.1	300
1	150
10	3

In simulating the planner, the robot was initially set to the starting configuration (Figure 3)

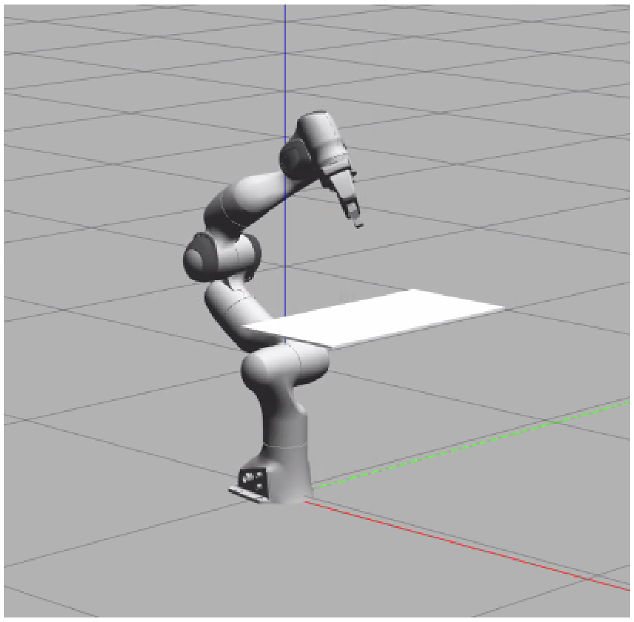
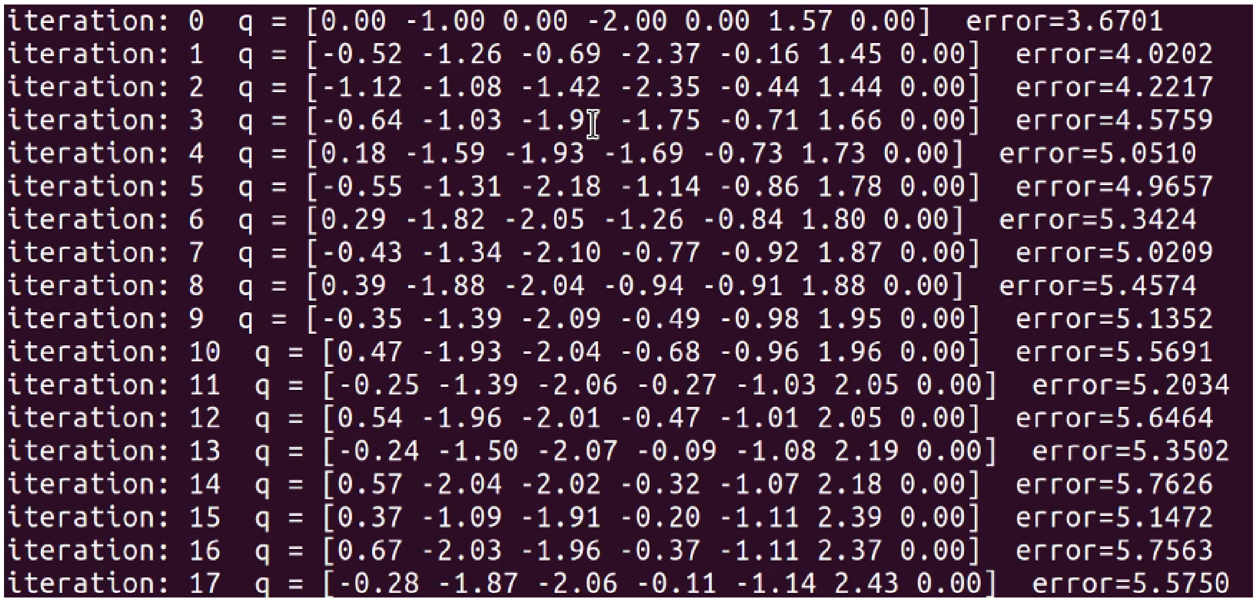


Figure 3: Robot in starting configuration

To test how the robot reaches the final configuration, we set the last iteration as the goal position manually, then ran the planner and observed the error and the computed final configuration (Figure 4, 5)



```
iteration: 0  q = [0.00 -1.00 0.00 -2.00 0.00 1.57 0.00]  error=3.6701
iteration: 1  q = [-0.52 -1.26 -0.69 -2.37 -0.16 1.45 0.00]  error=4.0202
iteration: 2  q = [-1.12 -1.08 -1.42 -2.35 -0.44 1.44 0.00]  error=4.2217
iteration: 3  q = [-0.64 -1.03 -1.97 -1.75 -0.71 1.66 0.00]  error=4.5759
iteration: 4  q = [0.18 -1.59 -1.93 -1.69 -0.73 1.73 0.00]  error=5.0510
iteration: 5  q = [-0.55 -1.31 -2.18 -1.14 -0.86 1.78 0.00]  error=4.9657
iteration: 6  q = [0.29 -1.82 -2.05 -1.26 -0.84 1.80 0.00]  error=5.3424
iteration: 7  q = [-0.43 -1.34 -2.10 -0.77 -0.92 1.87 0.00]  error=5.0209
iteration: 8  q = [0.39 -1.88 -2.04 -0.94 -0.91 1.88 0.00]  error=5.4574
iteration: 9  q = [-0.35 -1.39 -2.09 -0.49 -0.98 1.95 0.00]  error=5.1352
iteration: 10  q = [0.47 -1.93 -2.04 -0.68 -0.96 1.96 0.00]  error=5.5691
iteration: 11  q = [-0.25 -1.39 -2.06 -0.27 -1.03 2.05 0.00]  error=5.2034
iteration: 12  q = [0.54 -1.96 -2.01 -0.47 -1.01 2.05 0.00]  error=5.6464
iteration: 13  q = [-0.24 -1.50 -2.07 -0.09 -1.08 2.19 0.00]  error=5.3502
iteration: 14  q = [0.57 -2.04 -2.02 -0.32 -1.07 2.18 0.00]  error=5.7626
iteration: 15  q = [0.37 -1.09 -1.91 -0.20 -1.11 2.39 0.00]  error=5.1472
iteration: 16  q = [0.67 -2.03 -1.96 -0.37 -1.11 2.37 0.00]  error=5.7563
iteration: 17  q = [-0.28 -1.87 -2.06 -0.11 -1.14 2.43 0.00]  error=5.5750
```

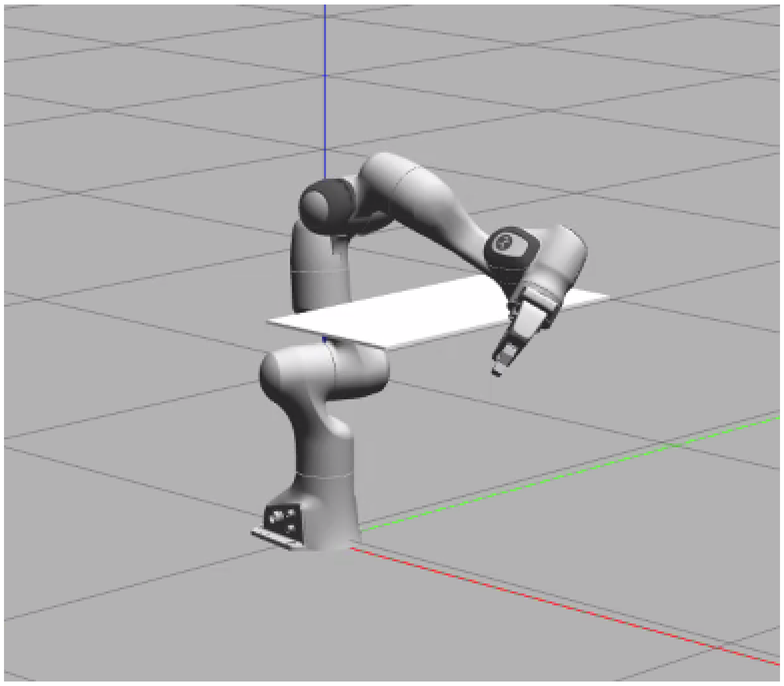


Figure 4: Planner output and robot arm in goal configuration

The goal position was removed from the last iteration and we checked the computed final configuration (Figure 4). We were able to observe errors in Joints 5 and 7, and further tests were planned to optimize the parameters to tune these values time permitting.

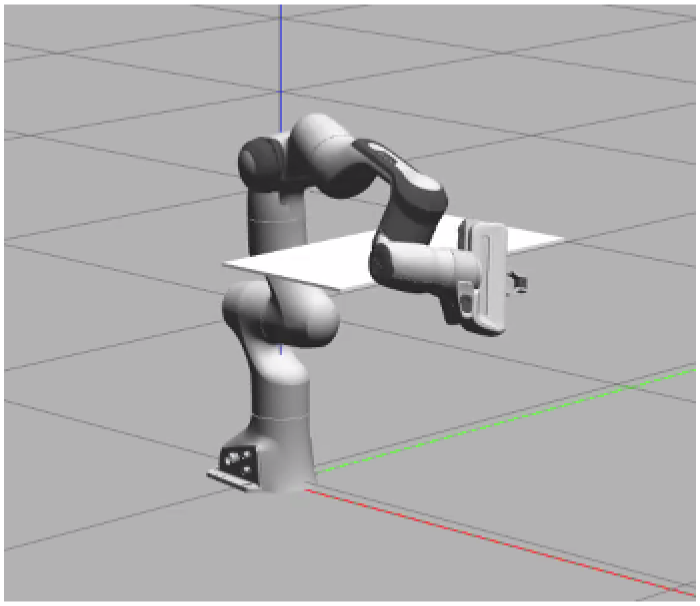


Figure 5: Robot in successful goal configuration without collisions

3 Analysis

In our analysis, we were able to find that the planner worked well for configurations mentioned in Table 1. The situations when the planner worked poorly was when the gradient descent step size (α) was increased to 0.7 and above as well as when the repulsive force (η) was changed to 0.5. This shows the importance of step size and how the influence of potential field parameters affects the controller. We simulated the planner in Gazebo to find the optimal parameters by observing the path traced and the robot visually. The parameters under which the path traced in minimal time and with minimal jitter were chosen as the optimal parameters for the controller.

After simulating Astar.py in gazebo, there were significant differences in situations where these two planning strategies could be applied. We observed that the A star planner took a longer time to plan the path compared to the potential field planner (max: 3 sec). From this we conclude that the A star planner could be used in situations where robustness and the shortest path is desired as the theory behind the A star planner is to find the path with minimum cost. However, there is a tradeoff between computational efficiency and the shortest path. Thus, depending on the needs of the situation the robot is operating in, we can select any one of the planners to work.

Situation 1: If precisional accuracy is required for picking up objects and traversing different obstacles on the way, A star could be preferred, since the planner is robust and saves time to pick up and place the object.
Situation 2: If the object must be picked up and placed from one location to another without many obstacles interfering, then a time saving path planner like Potential field is more useful.

If we had more time, we would optimize our planner to traverse the obstacle provided in map2, and adjust the parameters in such a way that the same control parameters could be chosen for both the obstacles. The parameter we would focus on changing for implementation is the gradient descent step size α, repulsive field strength (η_i) , radius of region of repulsion (ρ_0).