MEAM 520 Lab 4

Raima Sen, Renu Reddy Kasala

November 19, 2022

1 Introduction

This report presents the complete derivation of trajectory planning using the Artificial Potential Fields for the FRANKA EMIKA PANDA arm. Using gradient descent, the joint angles are iteratively changed to reach the goal configuration and at the same time measures are taken to avoid obstacles, detect collision and escape from local minima. These steps are detailed in the subsequent sections.

2 Methodology

For a given map, the attractive forces to the goal location and the repulsive forces from each obstacle in the map was calculated for each joint angle in the current configuration of the FRANKA arm. These effective forces were converted to torques using the linear velocity Jacobian (J_v). Finally the current configuration is updated to a new configuration until the goal configuration is reached. These steps are detailed as follows :

2.1 Calculating Attractive Forces (F_{attr})

For finding the attractive force on each joint of the arm, we consider a combination of conical well potential and parabolic well potential around each joint of the target configuration. The conical potential is used when the current joint configuration is far from the goal. A parabolic potential is used closer to the goal in order to prevent chatter (discontinuity) at the goal. A threshold was defined to switch between the 2 potentials and this parameter was tuned in the experiments. We also tuned the attractive field strength as a hyperparameter. Increasing the attractive field strength should make the robot reach the target quicker, but it also makes the robot joints prone to hitting obstacles. We used the following equations to calculate the attractive force for each joint i as per the conical well potential :

$$U_{\mathrm{att},i}(q) = \|o_i(q) - o_i(q_f)\| \tag{1}$$

$$F_{\mathrm{att},i}(q) = -\nabla U_{\mathrm{att},i}(q) \tag{2}$$

$$F_{\mathrm{att}\ ,i}(q) = -\frac{(o_i(q) - o_i(q_f))}{\|(o_i(q) - o_i(q_f))\|} \tag{3}$$

The following equations were used to calculate the attractive forces for each joint i as per the parabolic well potential:

$$U_{\mathrm{att},i}(q) = \frac{1}{2}\zeta_i \|o_i(q) - o_i(q_f)\|^2 \tag{4}$$

$$F_{\mathrm{att},i}(q) = -\nabla U_{\mathrm{att},i}(q) \tag{5}$$

$$F_{\mathrm{att},i}(q) = -\zeta_i (o_i(q) - o_i(q_f)) \tag{6}$$

Where $o_i(q)$ is the current joint position in the world frame and $o_i(q_f)$ is the desired joint position in the world frame. ζ_i is the attractive field strength at joint i, which is a tunable hyperparameter. These joint positions are obtained from the forward kinematic implementation adopted in Lab1, where we had obtained the intermediate joint positions with respect to the world frame for a given configuration q. The last column of $T_1^0, T_2^0, T_3^0, T_4^0, T_5^0, T_6^0, T_7^0$ gives the homogeneous coordinates of the joint positions

in the world frame.
Parameters tuned : attractive threshold, attractive field strength

2.2 Calculating Repulsive Forces (F_{rep})

For calculating the repulsive forces caused by a single obstacle in the map, we define its region of influence ρ_o. If the shortest distance between the joint and the obstacle lies within the region of influence, then a repulsive force activates. If the joint position is outside the region of influence then the repulsive force is set to 0. This sort of thresholding is maintained since we do not want the repulsive force to activate unnecessarily even when the joint is far away from the obstacle. The following equations are used for finding the repulsive force for each joint i with respect to a single object:
$When\, \rho_i(q) > \rho_0,$

$$U_{\text{rep },i}(q) = 0 \tag{7}$$

$$F_{\text{rep },i}(q) = 0 \tag{8}$$

$When\, \rho_i(q) < \rho_0,$

$$U_{\text{rep},i}(q) = \frac{1}{2}\eta_i \left(\frac{1}{\rho\left(o_i(q)\right)} - \frac{1}{\rho_0}\right)^2 \tag{9}$$

$$F_{\text{rep },i}(q) = \eta_i \left(\frac{1}{\rho\left(o_i(q)\right)} - \frac{1}{\rho_0}\right) \frac{1}{\rho^2\left(o_i(q)\right)} \nabla\rho\left(o_i(q)\right) \tag{10}$$

Here if the object is convex, which was the case for all the maps tested,

$$\nabla\rho\left(o_i(q)\right) = \frac{o_i(q) - b}{\|o_i(q) - b\|} \tag{11}$$

Where $\rho_i(q)$ is the shortest distance between the joint position and the obstacle, ρ_0 is the region of influence and b is the point on the obstacle boundary closest to o_i. η_i is the repulsive field strength at joint i which is a tunable parameter.Increasing η_i or ρ_0 will push the robot away from obstacles but it will also lead to more local minima if the map is dense with several obstacles.
Parameters tuned : region of influence, repulsive field strength

2.3 Calculating Total Forces

In order to calculate the net effective force on each joint, we sum the attractive forces calculated from section 2.1 and repulsive forces calculated from section 2.2 over all the obstacles present in the map. For particularly repulsive force, each obstacle in the map influences the net repulsive force in each joint. So it is required to sum the repulsive influence of all the obstacles on each joint.

$$U(q) = U_{\text{att}}(q) + U_{\text{rep }}(q) \tag{12}$$

$$\vec{F}(q) = \vec{F}_{\text{att}}(q) + \vec{F}_{\text{rep }}(q) \tag{13}$$

2.4 Calculating Total Torques at Current Configuration

For each joint, the full Jacobian matrix is calculated at the current joint angles. Since the obstacle and the target are static, only the linear velocity Jacobian is relevant for this lab. Using the following equation we get the torque at each joint :

$$\vec{\tau} = J_v^T(\vec{q})\vec{F} \tag{14}$$

We use the calcJacobian function defined in Lab 3 to calculate the linear velocity Jacobian. Each joint prior to the current joint influences the torque of the current joint. The Jacobian columns of the joints after the current joint has no influence on the torque of the current joint. Once we calculate the torques resulting from all the forces applied to each joint, we can sum it up, joint by joint.

2.5 Computing Gradients

In this step we calculate the new joint angles using gradient descent approach. The correct joint angle configuration is at the global minima and to bring the current joint angles closer to the minima, we update each joint angle using an iterative approach with the following equation:

$$q^{i+1} = q^i + \alpha_i \frac{\tau_i(q)}{\|\tau_i(q)\|} \tag{15}$$

Where α is the learning rate. Having a very low learning rate leads to slow convergence to the ground truth joint angles. A high learning rate may cause overshoot from the minima.
Parameters tuned : learning rate

2.6 Planning

The complete planning algorithm checks the L2 norm between the desired joint angles for a configuration and the joint angles that the robot achieves at each step. The gradient descent continues until this L2 norm is less than the fixed tolerance that we set.

2.6.1 Converge to Goal

The primary goal of the algorithm is to reach the goal configuration within limited iterations. This means that the L2 error should be lesser than the set tolerance value in order to indicate the goal configuration has been achieved. All the intermediate joint angles are recorded in order to record the trajectory followed by the PANDA arm.

2.6.2 Escape Local Minima

This part of the algorithm is implemented by saving the previous joint angle configuration and comparing it with the current joint angle configuration. If the values are close to each other but far from the goal joint angle configuration, then it is considered that the robot has reached a local minima. To escape local minima the current joint angles are given random values. This is called random walk. The following checks are made to detect local minima :
if $\|q_i - q_{i+1}\| < \epsilon$ and $\|q_i - q_{i+2}\| < \epsilon$ and $\|q_i - q_{i+3}\| < \epsilon$ then q_i is near local minimum

2.6.3 Detect Collision

In order to detect collision with the obstacle, the interception of line joining 2 joint origins with the obstacle dimensions is checked. This is done in a pairwise manner between each joint origin, and if a collision is detected then the robot arm is expected to stop moving. A detect collision check is done in each iteration of the gradient descent planner.

2.6.4 Pseudocode

```
initialise $q = q_0 \quad i = 0$
    for each $q^i$ in $q$ :
      if $\|qi - q_f\| >$ tolerance & iter < max iter
        get $O_i(q)$ from $T_i^0$
        get $O_i\,(q_t)$ from $T_i^0$
        $F_{\text{atti}}(q) = -\zeta_i\,(0_i(q) - 0_i\,(q_f))$
        if $|\ \vec{p} - 0_i(q) -$ radius_obstacle $| > \rho_0$ :
            $F_{\text{repi}}\ (q) = 0$
        else:
            $F_{\text{rep}}(q) = \eta_i \left[\frac{1}{\rho(Oi(q))} - \frac{1}{\rho_0}\right] \frac{1.\nabla\rho(0i(q))}{\rho_0^2(0i(q))}$
            note if object is convex,
        $F_i(q) = F_{\text{att}\,i}(q) + F_{\text{rep}\,i}(q)$
        $\tau_i(q) = J_v^\top F_i(q)$
        $q^{i+1} \leftarrow q^i + \alpha^i \frac{\tau_i(q)}{\|\tau_i(q)\|}$
        $i \leftarrow i + 1$
    else:
        return $q_o$
```

2.7 Code Structure

The code written by Raima Sen was used for experiments. The potential field planner was implemented using various subroutines. Firstly, the attractive forces subroutine calculates the attractive forces between the goal and the current joint configuration for a single joint. The repulsive forces subroutine calculates the repulsive force between a single obstacle and the current joint configuration for a single joint. Subsequently in the compute_forces function, we calculate the attractive forces between the goal and all joints and repulsive forces between all the obstacles with all the joints. The compute_torque function was then used to calculate the torque at each joint due to the forces acting on other joints. This torque at each of the intermediate joint configurations was used to calculate the gradient in the compute_gradient function. Finally the planner function iteratively updates the joint angle configurations until the joint angles converge with the goal joint angles. This is is the main gradient descent algorithm. Along with this certain checks are also needed like collision detection and check and escape from local minima. These are also implemented within the loop which iterates up to a maximum limit.

3 Evaluation

The evaluation process was divided into 3 broad categories :

3.1 Analytical Evaluation

In this initial process of evaluation, the environment used was map 1. We started by tuning the learning rate and noted that $\alpha = 1e-3$ was a suitable choice. We also noted the shortest distances between the obstacle and each joint and using these numerical values, chose $\rho_0 = 0.3$. Next, we chose scalar attractive and repulsive field strengths for each joint and noted the error convergence. We even tested 2 other cases :

- When both the ζ and μ are increasing with every joint origin
- When both the ζ and μ belong to a Gaussian distribution between 0 and 1

It was noted that a Gaussian distribution of ζ and μ led to maximum convergence of the error value. However, scalar values were used in the simulation and hardware testing for simplicity.

3.2 Simulation Testing

We extensively tested out the code on simulation across all the provided maps and also modified the start and end goal positions in maps 3 and 4. All the simulation tests were done with a single ζ and μ for each joint.
For map 1, we used :
Test case 1 : $start = [0, -1, 0, -2, 0, 1.57, 0]$ $end = [-1.2, 1.57, 1.57, -2.07, 0, -1.57, 0.7]$
Test case 2 : $start = [0, 0, 0, 0, 0, 0, 0]$ $end = [-1, 0, 0, -2.07, 0, -1.57, 1.57, -2]$
Figure 1 and 2 indicate the final configuration of the robot for both test cases. Since map 1 had no dense set of obstacles or narrow passages, the path planner succeeded each time. The trajectory adopted over multiple runs was also similar especially for the second test case. The time taken for test case 1 was 0.83 seconds when we ran it the first time. In the next 2 runs and no other changes, the time taken to converge to goal was 1.46 seconds and 1.34 seconds.
For map 2, we used :
Test Case : $start = [0, 0.4, 0, -2.5, 0, 2.7, 0.707]$ $end = [1.9, 1.57, -1.57, -1.57, 1.57, 1.57, 0.707]$
Figure 3 shows the final configuration achieved by the robot when it reaches the goal configuration. In this case, our planner succeeded very occasionally. This is because the narrow passage would either lead to collision if we decreased the repulsive field strength and increased the attractive field strength or it would lead the robot to first get stuck at local minima due to high repulsive field strength. Although the robot would manage to get out of the local minima, it would move randomly and collide

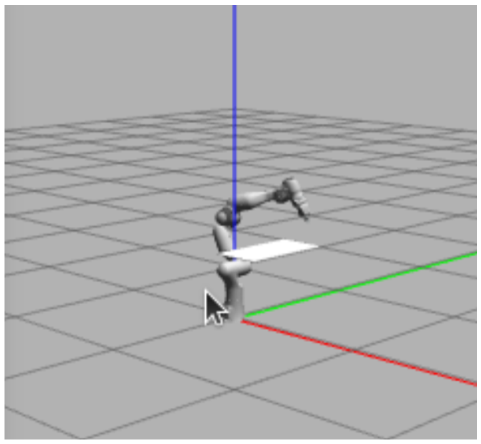
Figure 1: Test Case 1 on simulation in map1

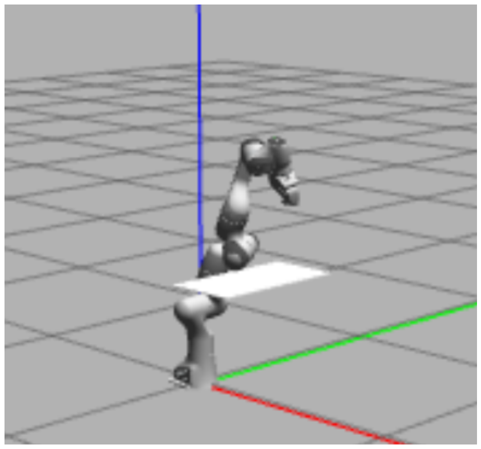
Figure 2: Test Case 2 on simulation in map1

with either of the walls and then stop motion. This simulation took longer than the tests conducted on map 1. It took about 1.54 seconds the first time and 2.07 seconds in the next test (no parameters were changed). The path was also not the same over these 2 runs.

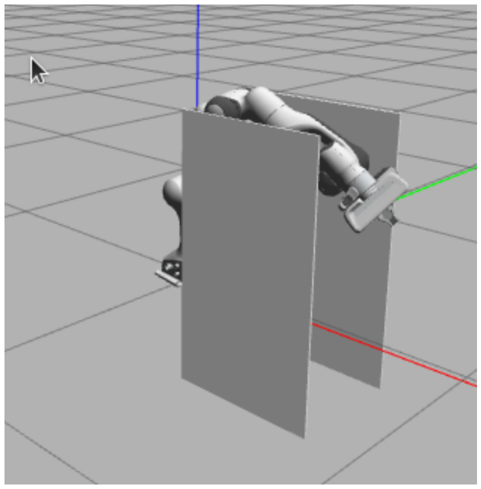
Figure 3: Test Case on simulation in map 2

For map 3, we used :

Test Case : $start = [0.1, 0.4, -1.25, -2.1, 0.4, -1, 0]$ $end = [-2, 1.5, 0.5, -2.1, 0.5, -1, -1.57]$
Figure 4 and 5 show the initial and final configurations achieved by the robot when it reaches the goal configuration. In this case, our planner succeeded always as the repulsive field around the obstacle was uniform. The running time was 1.54 seconds and the trajectory followed was almost the same over multiple runs and no changes in parameters.

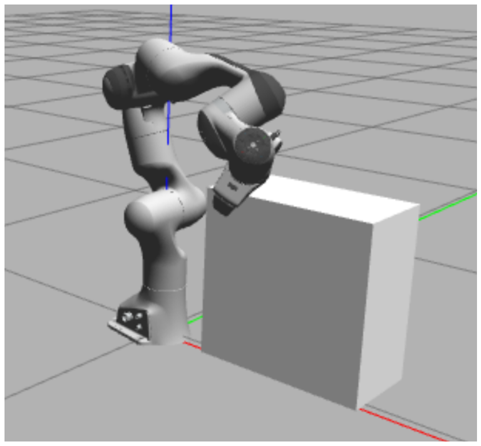

Figure 4: Start configuration on simulation in map 3

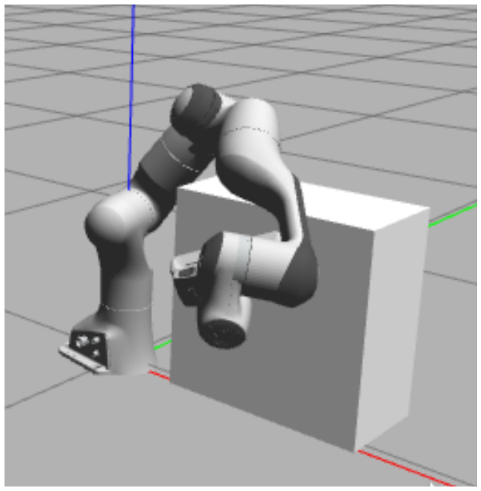

Figure 5: End configuration on simulation in map 3

For map 4, we used :
Test Case : $start = [0, 0.4, 0, -2.5, 0, 2.7, 0.707]$ $end = [0, -0.4, 0, 2.5, 0, -2.7, -0.707]$
Figure 6 and 7 show the initial and final configurations achieved by the robot when it reaches the goal configuration. In this case, our planner succeeded almost always as the tuned parameters worked well. The running time was 1.81 seconds in the first run and 2.01 seconds in the second run. We specifically chose a test case where the robot shall have to pass through the gap between the 2 blocks, which it was successfully able to do. The path would occasionally lead to collisions although this was rare. So we cannot say that the planner finds the same path over multiple runs.

3.3 Hardware Testing

We tested the code mainly on simulation and ran 1 test case on hardware. We primarily saw the robot follow the same trajectory as in the simulation. The lack of actual obstacles made it hard to gauge whether the attractive and repulsive potential fields were actually in action or not. A video to the hardware test can be found here : Link. It mainly checks the trajectory in Test Case 1 of Map 1.

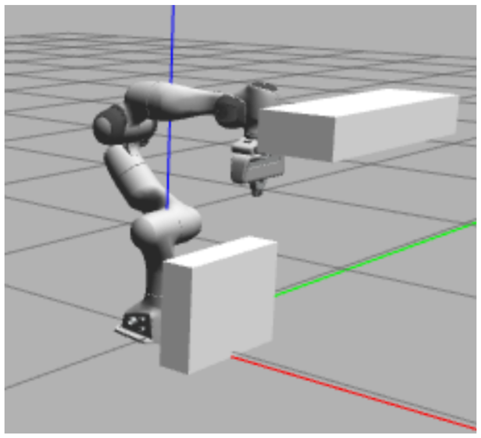

Figure 6: Start configuration on simulation in map 4

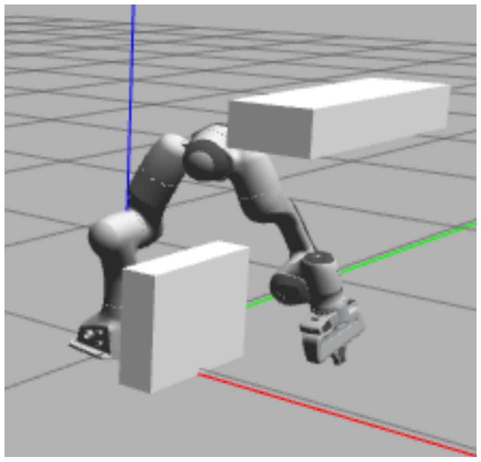

Figure 7: End configuration on simulation in map 4

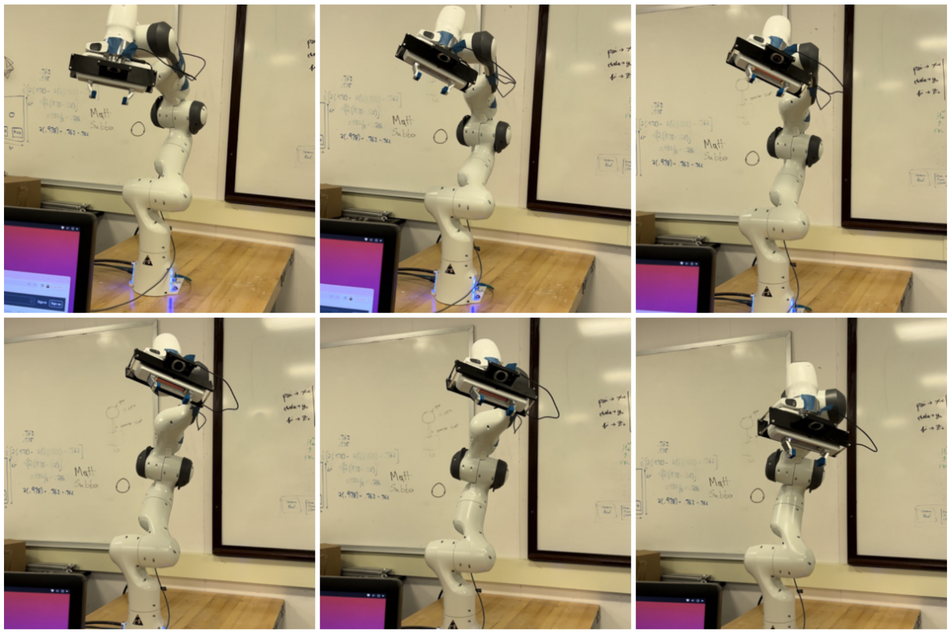

Figure 8: Hardware test on map1

4 Analysis

- Our planner worked well for map 1 and map 3. But did not work that well for map 2 and map 4. That is it worked well in cases where the environment had only one object. But when the environment was densely packed with obstacles it did not perform well. For example, our Franka panda arm passed very close by the obstacles in the map2. That is, it performed poorly in cases where it had to pass through narrow corridors, i.e., when obstacles were close to one another.

- Strategy for choosing parameters for the potential field:
  Initially, we randomly chose the attractive and repulsive field strengths. But their effect on the potential fields varies as we increase or decrease the field strengths.

  Increasing the attractive field strength will bring it to the goal faster but may cause it to get closer even to towards the obstacles. Increasing the repulsive field strength or increasing the distance of influence of the obstacle will push it more away from the obstacles, and if the environment is dense then it will generate more local minima.(If the strength of the repulsive forces is so high that it cancels put attractive force, then it generally leads to a local minima. In such cases a random walk is done in order to push the robot out of the local minima)

  It is also necessary to pick ρ_o, the distance of influence of the obstacle as it effects the repulsive force from obstacles. We chose $\rho_o = 0.3$. Also we used the attractive field threshold as 0.5. These thresholds were tuned based on the minimum distance of each joint from obstacle (which was less than 0.3). Also we selected the repulsive field strength to be higher than the attractive field strength as more importance was given to obstacle avoidance.

  In order to converge to the goal position, it is important to choose the learning rate. If learning rate is too large then the solution will over shoot and if its too small then it takes too long to converge to the target position. So, we settled on the value of learning rate to be 1e-3.

- Potential fields are computationally inexpensive and can be used for unknown environments. They can also be used for dynamically changing environments. But the precision is not that high. Whereas grid based over off-board planners like A* and RRT can be used when the environment is known and high precision planning is required.

- Some of the changes we would've made to our implementation if we had more time with the lab are: Rigorously test on more maps especially with cases where there obstacles where closer to one another. We would've optimised our code to perform path planning in cases where the obstacles are smaller than the size of the Franka Panda arm links. Further, we would've also added various links to have different attractive field strengths. This would optimise the attractive forces exerted by each joint. We would've further tested our potential field implementation against off board planner and compare the difference in precision.

[1] Hutchinson, Seth A. and Mathukumalli Vidyasagar. “Robot modeling and control / Mark W. Spong, Seth Hutchinson, M. Vidyasagar.” (2006).