Documentation

Author: Nikhil Podila

Assignment 1 ECSE 683

1 Robot

In this assignment, KUKA IIWA robot arm is chosen as the testing platform. The configuration space is the degree-of-freedom joints $\boldsymbol{q} \in \mathbb{R}^7$, and the task-space is the end-effector position $\boldsymbol{x} \in \mathbb{R}^3$.

The state is the position of the configuration space $x = [\boldsymbol{q}]$ and the action is the joint velocity, $u = \dot{\boldsymbol{q}}$. The trajectory is controlled through an inverse kinematic controller using Velocity Control method.

2 Test cases

To ensure that a solution exists, the start position and the target position of the end-effector are generated in the task space level, but with a few bounds for two reasons:

- To ensure that bounds on the position of the end-effector on the robot arm are not crossed.
- To ensure that Robot joints do not lead to singularity or cause its own links to collide.

Throughout the assignment, the value on x-axis for the end-effector start and target position were fixed to $x_x = -0.4$

The x_y and x_z values of the end-effector position for the start and target positions were randomly selected within its bounds while testing the algorithm on multiple experiments. Specifically, the submitted code randomly generates start and target positions for the end-effector using the following bounds:

- Start position:
  - $x_x^s = -0.4$
  - $x_y^s \in [-0.3, -0.2]$
  - $x_z^s \in [0.7, 0.8]$
- Target Position:

- $x_x^t = -0.4$
- $x_y^t \in [0.2, 0.3]$
- $x_z^t \in [0.2, 0.3]$

An obstacle has been placed half way between the start position and target position:

$$\boldsymbol{x}^o = \frac{1}{2}(\boldsymbol{x}^t - \boldsymbol{x}^s) + \boldsymbol{x}^s$$

The start and target positions of the joints are calculated using Inverse Kinematics:

$$\boldsymbol{q}^s = ik(\boldsymbol{x}^s),\ \boldsymbol{q}^t = ik(\boldsymbol{x}^t) \tag{1}$$

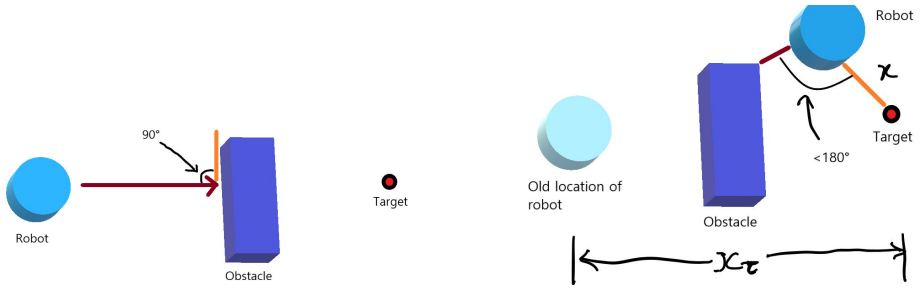
3 Approach

Obstacle avoidance is an important task in the field of Robotics. Many approaches have been identified to avoid obstacles, starting from papers such as [1]. However, every approach in literature since that period has some advantages and disadvantages, which make different algorithms applicable for different obstacle avoidance situations and environments. One such obstacle avoidance algorithm is the Repulsive Potential field algorithm. One of the drawbacks of using Repulsive Potential field algorithm is that when used with Attractive Potential field algorithm for reaching the target, it can have a few points of failures. For example, if the target, obstacle and robot are in a straight line where the obstacle occludes the target, the Attractive Potential and Repulsive Potential values cancel out each other when the robot is directly in front of the obstacle. This creates a net zero action by the planner, leading to failure to reach the target. This situation is known as the "Deadlock".

To avoid the deadlock created by Potential fields, we use a different method for avoiding obstacles. We propose to use a method described in Algorithm 2 of [2]. This method was described in a practical manner in the now-discontinued course [3].

The Boundary Following algorithm aims at moving along the boundary of the obstacle (or the minimum distance perimeter around the obstacle) to avoid contacting the obstacle. As it moves along the boundary, the robot estimates if there has been sufficient improvement and recalculates its aim towards the goal. The method is detailed using the Figure 1, where it is first described in 2 dimensions. As seen in Figure 1a, as the robot approaches the obstacle, it attempts to avoid the obstacle by moving along the boundary of the obstacle. If the Normal direction from the surface of the obstacle is known, the direction along the boundary is calculated by rotating by 90 degrees either Clockwise or Counter-clockwise. Figure 1b explains a scenario where the robot has been following the boundary. The robot continuously monitors two parameters:

- **Progress towards target**: Progress is successful if the current distance to Target (x in Figure 1b) is less than the distance to Target when the robot started following the boundary ($\boldsymbol{x}_\tau$ in Figure 1b).



(a) Before Obstacle (b) After Obstacle

Figure 1: Figures describe in 2D, the Boundary Following algorithm. In 1a, the planner moves orthogonal to the surface of the obstacle. In 1b, the planner (1) has progressed towards the target, and (2) has a clear shot of the target.

- **Clear Shot**: The robot has a clear shot at the target if the Normal direction to the surface of the obstacle and the direction from robot to Target are within 180 degrees. This ensures that the robot is not moving towards the obstacle once again.

Once both the parameters are succeeded, the Robot switches back from the Obstacle Avoidance algorithm to the Target (Go to Goal) algorithm, and continues until it reaches the goal. As a result of this switching, this algorithm is termed as a Hybrid algorithm. The Go to Goal algorithm used in this assignment continues to be Attractive Potential algorithm.

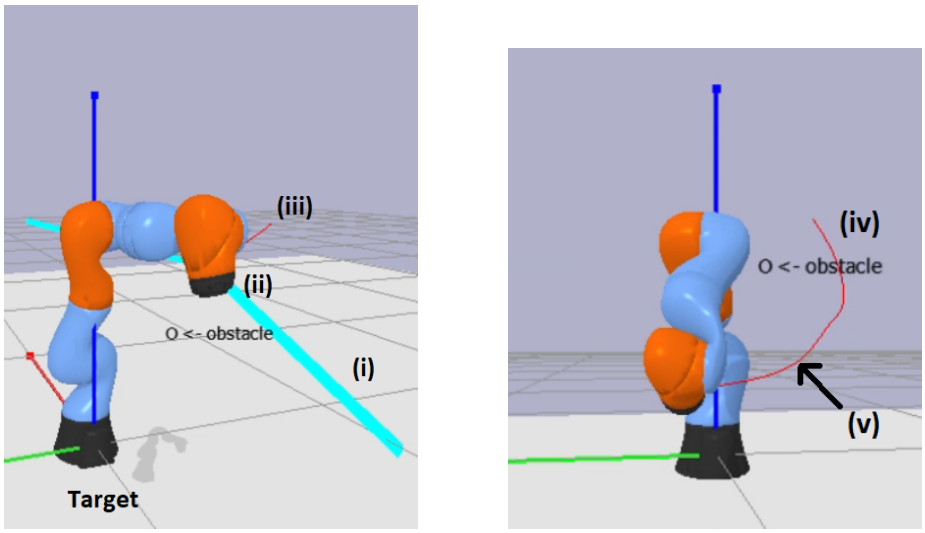
In the scenario of the KUKA Robot for this assignment, following features are addressed or modified from the original algorithm to adapt to this scenario:

1. Obstacle is a point. Thus, Normal direction from the surface of Obstacle is considered as the direction from Obstacle to Robot.

2. A value $d_{min} = 0.2$ is considered as the minimum acceptable distance between the robot and the obstacle. Thus, the boundary of the obstacle forms a sphere of radius 0.2.

3. One of the main challenges in 3D boundary following, especially with spheres, is that 90 degree rotation is not limited to clockwise or anti-clockwise. In fact, for any Normal direction to a sphere, the Tangent directions to the sphere at that point form a plane. In the case of this assignment, a Rotation matrix R is randomly chosen and fixed to decide the direction to follow along the boundary.

4. One Rotation matrix does not sufficiently define the movement along the boundary efficiently. This is because the Rotation matrix can create a spiral trajectory on the surface of the boundary sphere. To mitigate this in the assignment, we use the previously calculated direction defined by R and the direction from robot to obstacle to calculate a third direction



(a) When Obstacle is encountered

(b) After Obstacle

Figure 2: Figures describe in 3D, the Boundary Following algorithm. In 2a, the robot plans a different strategy to avoid obstacle. In 2b, the planner (1) has progressed towards the target, and (2) has a clear shot of the target.

(using cross product). This ensures that the end-effector follows a "longitudinal" trajectory along the sphere and does not spiral around the sphere during the boundary following.

By adding the features described above to the Boundary following algorithm, this algorithm can be successfully applied to the robot and the experimental setup described in Section 2.

When applied to the robot, the simulations look like those shown in Figure 2. In these Figures, x-axis is shown in red and as described before, the value is fixed to -0.4. Thus, all the information on the figures is expected to represent only at the plane of $x = -0.4$, except when the robot is following the boundary.

In Figure 2a, three points highlighted describe:

- (i) Blue tangents: These show the direction of motion of the end-effector when performing Boundary Following. In the figure, it is visible that the tangents are away from the obstacle in order to avoid it.
- (ii) End-effector: The black cylinderical shaped component on the KUKA IIWA robot is the end-effector.
- (iii) Red line: A faint red line indicates the path that the robot has already travelled. As we can see in the figure, before encountering the minimum distance to the obstacle, the end-effector's trajectory was directly pointed towards the target (and hence the obstacle).

In Figure 2b, two highlighted points describe:

- (iv) Red line: A faint red line indicates the end-effector's past trajectory when following the boundary. As visible, it is a "longitudinal" trajectory along the invisible minimum distance sphere having obstacle as its center. This improves robot performance.
- (v) Point of switch-back: This point approximately indicates the situation when the robot satisfied both the sufficient conditions to switch back from Boundary Following and aim towards the target (Go to Goal strategy). This switching is important to achieve the final objective.

The main advantages of this method are:

- This algorithm does not cancel out its effect with a Go to Goal algorithm. In fact, the algorithm stops the Go to Goal strategy, avoids the obstacle and then resumes the Go to Goal strategy. Thus, Deadlock will not be reached.
- This algorithm is applicable to obstacles of any shape and size and is not limited to point obstacles [2].
- This algorithm optimizes the 3-dimensional path along the obstacle by following the boundary. Thus, it is efficient in using physical space as well as time.

4 Limitations and Improvements

Although this algorithm works well for the scenario described here, there can be few improvements which further improve the algorithm in this environment and setup. There are few other improvements which might be useful in environments different from the one in this assignment. The limitations and possible improvements are detailed below:

1. A Rotation Matrix R is fixed, as described. This Rotation matrix can be arbitrary only because the assignment instructions clearly define that the robot end-effector position, the obstacle and the target lie exactly in one straight line. If this was not the case, Rotation matrix can be optimized to favor the direction closer to the target position.
2. In obstacles with different shapes, the Rotation matrix might fail to lead to a viable boundary following direction.
3. Because of the cross-product calculation, this algorithm cannot be applied back in 2 dimensional examples such as wheeled robots. Alternative method to the cross product to fit both dimensions is required.

4. Although the algorithm may be efficient in terms of using physical space around the obstacle and time to avoid the obstacle, the algorithm may not execute the path with the least effort. Maintaining path around the boundary takes high value actions in various directions, which might lead to a high-effort plan.

5. Since this method uses Inverse Kinematic control, it does not take into consideration the various inertial effects and dynamics on the robot system. This could be improved using Inverse Dynamics control.

6. In real robots, sensors are subject to noise. Because of the switching strategies, it is possible, that noise can make the algorithm jitter between the Attractive Potential and Boundary Following algorithm. This can be avoided by introducing an ϵ-gap between the two states, known as fat guard [3].

5 Results

The experiments were conducted on over 50 trajectories with random initial and final targets within the bounds specified in Section 2. Over 10 trajectories outside those bounds were explored, keeping $x = -0.4$. All 60+ trajectories were successfully executed. Many such trajectories were also utilized for validation to tune the hyperparameters such as Attractive Potential coefficient, etc.

The Inverse Kinematics based control applied on the KUKA IIWA Robot was adapted from the original code here [4]

One such result is demonstrated in the video of the assignment, along with the debug lines. It is visible that the robot starts by aiming for a straight line trajectory towards the goal. Near the obstacle boundary, the robot switches to boundary following. It continues following the boundary until the sufficient conditions for boundary following have been satisfied. Then, it switches back to aiming for the target and finally reaches the goal.

References

[1] V. J. Lumelsky and A. A. Stepanov, "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," *Algorithmica*, vol. 2, no. 1, pp. 403–430, 1987. [Online]. Available: https://doi.org/10.1007/BF01840369

[2] Z. Shiller, "Online suboptimal obstacle avoidance," *The International Journal of Robotics Research*, vol. 19, no. 5, pp. 480–497, 2000. [Online]. Available: https://doi.org/10.1177/02783640022066987

[3] M. Egerstedt, GeorgiaTech, and Coursera, "Control of mobile robots," 2013. [Online]. Available: https://www.coursera.org/course/conrob

[4] E. Coumins, "Inverse kinematics example," 2020. [Online]. Available: https://github.com/erwincoumans/bullet3/blob/master/examples/pybullet/examples/inverse_kinematics.py

6 Remarks

As a part of the assignment, the following are some remarks:

- Better diagrams, especially in 3 dimensions, could have better explained the mathematically complex yet visually simple algorithm. Animations of the sphere to show spiral movements, longitudinal movements and Tangent plane could better show the method.
- The Repulsive Potential Algorithm is implemented in order to test the deadlock. However, it is observed that the algorithm always avoids the deadlock (after a small pause) using its feedback controller. This is clearly visible from every experiment run with Repulsive Potential Algorithm, as an underdamped oscillation in the position when avoiding the obstacle. This is a standard feedback controller used by PyBullet, and could be directly used to avoid the deadlock. However, the theoretical basis for the approach used by the feedback controller has to be explored in detail, as it is not very clear why the controller chooses a single direction to correct its velocity or position during every experiment.