ROBOT MODELING AND CONTROL

ECE470S

LAB4 : Motion Planning and Obstacle Avoidance with the KUKA Robotic Arm

1 Purpose

The purpose of this lab is to adapt the results of Lab 3 for implementation in a real robot. The objective is to use the artificial potential algorithm to make the KUKA robotic arm pick up an object and drop it in a cup while avoiding obstacles that are placed in the workspace.

2 Introduction

In lab 3 you experimented with the artificial potential motion planning algorithm. This algorithm generates attractive and repulsive potentials, which are used in conjunction with a gradient descent algorithm to generate a trajectory through obstacles from one point to another.

In this lab, you will implement this algorithm with the KUKA robotic arm. During the laboratory, you will experiment with various parameters of the algorithm and investigate its strengths and weaknesses. Further, you will see how a simulated experiment translates to real mechanical hardware. You will also have the chance to improve the algorithm as much as you can, utilizing the Manipulator API.

3 Preparation

Please submit a **complete** preparation at the beginning of the lab session. Prior to the lab, you should make sure that your repulsive and attractive force functions return the expected values. In this lab, instead of having a pencil attached to the gripper, we will be using the gripper to grasp objects. The offset a_6 now corresponds to the length of the gripper which is approximately $a_6 = 156$mm.

1. Modify your motion planning algorithm from Lab 3 (developed for the PUMA 560 manipulator) to work for the KUKA manipulator. The main difference is the DH table. For better results, set $a_6 = 0$ when deriving the attractive and repulsive force functions. You can create a second DH table called `DH_forces` for this purpose. This shifts the point on the gripper where the attractive and repulsive forces act. However you must still use $a_6 = 156$ in your inverse kinematics computations. Otherwise the gripper will collide with the ground.
2. Write neatly on paper the expression of repulsive forces for the following two objects: 1) repulsion upward from the Workspace plane, which we assume to be parallel to the $x_0 - y_0$ plane, and have a z_0 value of 32 mm; 2) Repulsion from a cylinder of finite length. The bottom of the cylinder lies on the $x_0 - y_0$ plane and the height of the cylinder is a parameter h.
3. Modify your repulsive function to account for the two new objects described in point 2 above: the horizontal plane $z = 32$ mm and the finite length cylinder. Set $\eta = 1$ in the repulsive function.

4. Create a Matlab script `setupobstacle_lab4prep.m` containing the statements:

   ```
   prepobs{1}.R = 100;
   prepobs{1}.c = [250; 0];
   prepobs{1}.rho0 = 500;
   prepobs{1}.h = 300;
   prepobs{1}.type = 'cyl';
   ```

   Test your repulsive function; you should get:

   ```
   >>setupobstacle_lab4rep;
   >>tau = rep(myrobot,[pi/10,pi/12,pi/6,pi/2,pi/2,-pi/6],obs);

   tau = [0.1795    0.9540    0.2353    -0.0344     -0.0344    0.0000]
   ```

   In particular, the repulsive forces should be

   ```
   Frep =

      1.0e-06 *

      -0.1269    0.0059    0.0073    0.0168    0.0168    0.0189
       0.0043    0.0096    0.0092    0.0096    0.0096    0.0018
       0.1005    0.1037    0.0615         0         0         0
   ```

5. Test that the motion planning algorithm works. Set the attractive and repulsive force coefficients as $\alpha_{\text{att}} = 0.013$ and $\alpha_{\text{rep}} = 0.01$ respectively. The following trajectory must be demonstrated for the preparation (i.e., the arm should plan a trajectory over the cylinder successfully). Observe the effect of reducing `rho0`.

   ```
   >> kuka = mykuka(DH);
   >> kuka_forces = mykuka(DH_forces);
   >> p1 = [620 375 50];
   >> p2 = [620 -375 50];
   >> R=[0 0 1;0 -1 0;1 0 0];
   >> H1=[R P1';zeros(1,3) 1];
   >> H2=[R P2';zeros(1,3) 1];
   >> q1 = inverse_kuka(kuka, H1);
   >> q2 = inverse_kuka(kuka, H2);
   >> motionplan(q1, q2, kuka_forces, obs);
   ```

6. Read Section 4.3 of this document, and discuss with your partner ideas for creative motion planning.

Robot Manipulator API

For your reference, below are the Matlab commands you will use to control the Kuka robot arm. You have practiced these commands in Lab 0.

- `startConnection` establishes connection between Matlab and KUKA.

- `stopConnection` terminates connection between Matlab and KUKA.
- `getAngles()` returns the vector of current joint angles $q = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$.
- `stop()` stops KUKA motion.
- `moveAxis(axis,vel)` moves a single KUKA axis. `axis` takes a value between 1 and 6 that corresponds to the joint angle to be commanded. `vel` is a signed value that determines the commanded angular speed of the joint. In this lab `vel` should be set to no greater than 0.01.
- `setAngles(q,vel)` sets KUKA arm angles to those defined in `q` to within a small tolerance. `vel` corresponds to the speed of motion. For this lab, set `vel` to 0.04. To cancel the command during execution press `ctrl c` and run `stop()` in the Matlab command window.
- `setHome(vel)` sets KUKA arm to the HOME configuration in Figure **??**. `vel` corresponds to the speed of motion. For this lab, set `vel` to 0.04. To cancel the command during execution press `ctrl c` and run `stop()` in the Matlab command window.
- `setGripper(state)` sets the gripper state. `setGripper(0)` closes the gripper; `setgripper(1)` opens the gripper.

Warning: Never use the `clear all` command in Matlab. Instead use `clearvars -except udpObj` to avoid errors from occurring.

The steps to **connect KUKA to the external PC**:

- Run `startConnection` in Matlab.
- On the KUKA SmartPad run `RSI_Ethernet.src` until the line `RSI_MOVECORR()`.
- To run the program, hold down half-way one of the enable buttons on the back of the KUKA SmartPAD. The enable buttons are labelled 3 and 5 in Figure 1. If you press too hard, an emergency stop will be activated. While holding down the enable button, press and hold the run button labelled 10 in Figure 2. Note that when running the line `RSI_ON` a warning will appear saying 'Caution - sensor correction is activated". Simply confirm this warning to continue.

The steps to **command the KUKA arm from Matlab** are as follows:

- Run the desired command in the Matlab command window. The robot will not yet move.
- Hold down half-way one of the enable buttons on the back of the KUKA SmartPAD.
- While holding down the enable button, press and hold the run button. Now the robot will carry out the desired command as long as both the enable and run buttons are being held down.
- To make the robot stop moving, simply release the enable button. To cancel the current command, press `ctrl c` and run `stop()`.
- If the robot ever hits an object while moving (such as the ground), immediately release the enable button and the robot will stop. Then press `ctrl c` in the Matlab control window to cancel the command and run `stop()`. Immediately call a TA to resolve the collision.
- If the robot does not move when commanded, it means an error has occurred. Call a TA to resolve the issue.
- For safety, students must never open the safety gate enclosing the robot unless instructed to do so. The enable and start button must never be pressed while the safety gate is open.

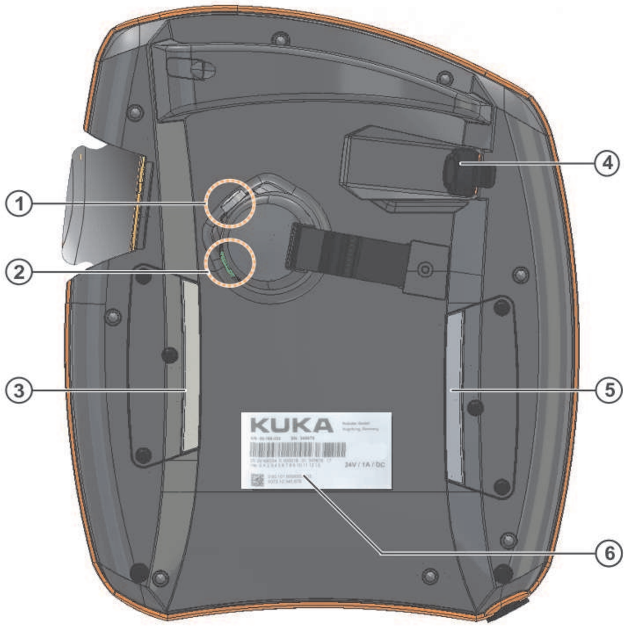

Figure 1: 3 and 5 are Enable buttons on the KUKA SmartPAD.

Warning: Make sure to FIRST connect KUKA to Matlab using `startConnection.m`. THEN run the RSI from the Kuka SmartPad

The steps to **disconnect KUKA from the external PC**:

- When finished running the lab, return to the program `RSI_Ethernet.src` on the SmartPAD, click on line 32 and then click `Block selection` to set the program cursor to the command `ret=RSI_OFF()`. Then continue running the program to the end. The program status indicator will turn black to indicate the program is complete.
- Run `stopConnection` in Matlab.

The steps to **deal with an RSI connection problem:**

- Type `CTRL+C` within the Matlab environment.
- There are two options available on the top portion of the SmartPAD screen: **cancel** or **reset**. Use **reset**, not cancel, to reinitialize the RSI program from the Kuka SmartPAD.
- After resetting the RSI program, push the play button multiple times until the program reaches the **RSI MOVECORR()** line.
- In Matlab, run `getAngles()` to verify that the RSI connection has been correctly re-established.

4 Experiment

The following steps should guide you in investigating the motion planning algorithm with the KUKA manipulator. You will be varying parameters to see how the actual implementation works and commanding

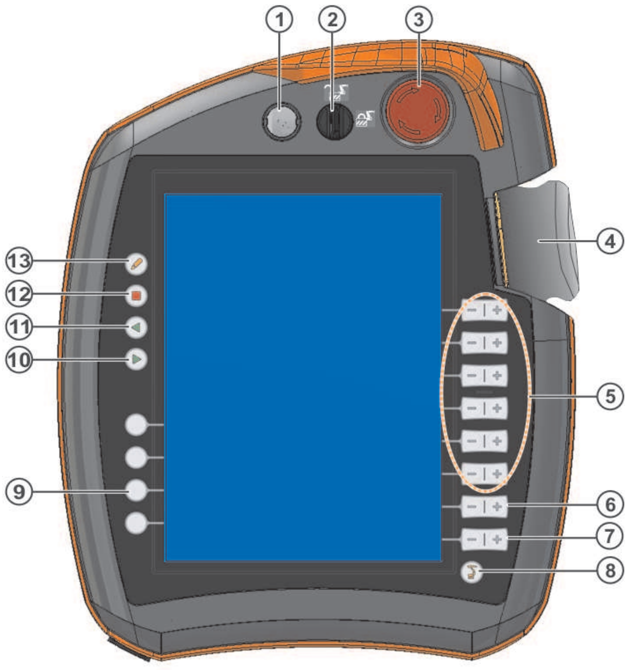

Figure 2: 10 is the Run button on the KUKA SmartPAD.

the robot to move while avoiding real obstacles in the workspace. You will also witness some challenges in working with hardware in contrast to simulations.

In this lab, you will *not* use the Matlab function `FrameTransformation.m` because all specifications will be directly expressed in the base frame of the robot.

4.1 Initial Motion Planning in Simulation

Now that you have calibrated the robot, you will perform motion planning with your algorithm and the manipulator.

1. Define the points:

   ```
   p0 = [370 -440 150];
   p1 = [370 -440 z_grid];
   p2 = [750 -220 225];
   p3 = [620 350 225];
   ```

   where `z_grid` $= 45$ mm. You might tweak this value later to make sure that the end effector correctly grabs the object.

2. Create a new function `setupobstacle.m` that has three obstacles. The ground plane $z = 32$ mm, with $\rho_0 = 150$, and two cylinders of radius 100 mm and height 572 mm with $\rho_0 = 150$. One cylinder is centred at $[620; 0]$ and the other cylinder is centred at $[620; -440]$

3. Set the coefficient of the repulsive force $\eta = 1$, and both repulsive and attractive force coefficients $\alpha = 0.01$.

4. Perform motion planning from the HOME configuration to p_0, from p_0 to p_1, from p_1 to p_2, and from p_2 to p_3 ensuring that the gripper is pointing towards the ground at each end point. That is,

$$R_6^0 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

   As in Lab 3, the outputs of `att` and `rep` do not affect `q(6)`. Simply use the command `linspace` to make `q(6)` transition from its initial value to the desired final value for each segment connecting p_i to p_{i+1}. Test your algorithm in simulation. You may have to tune the parameters to make this succeed.

Show these results to your lab TA.

4.2 Initial Motion Planning with Kuka

Now you will use the motion planning algorithm tested in simulation to command the actual robot arm.

1. Send the robot to the HOME configuration.
2. Place the provided target object at p_1. Place the provided target basket at p_3. Align the target object so that the gripper will be able to grasp it. Finally, place the obstacles at the appropriate positions.
3. Send the angles generated by the motion planning to the robot arm. Once again, you may have to vary parameters so that the trajectory succeeds without hitting the obstacles. Your software should start with the gripper open and move to point p_0 which is just above the target object. Once at the point p_0, use `setAngles` to move directly to point p_1 without obstacle avoidance. Then close the gripper to grasp the target object at p_1, move through the trajectory, and open the gripper at p_3 to drop the object into the target basket.
4. Without changing the force parameters, change the cylinder position within a 5-10 cm radius of the original position. To do this, update the appropriate cylinder parameters in `setupobstacle.m` and move the physical obstacle to the corresponding position. Then perform motion planning and send the angles to the robot arm.
5. What effect does varying the algorithm parameters have? Discuss repulsive and attractive α, repulsive η, and the ρ_0 associated with each obstacle.

Show these results to your lab TA.

4.3 Creative Motion Planning with Kuka

1. This section is to challenge you to come up with your own creative obstacle avoidance algorithm. You should utilize the gripper function to pick up a provided object and drop it at some target place. Some ideas are below:
   - Picking up multiple objects for multiple trajectories.
   - Stacking some items one on top of each other.
   - Add more cylinders, or bring new obstacles and develop functions for them.
   - The potential force method is only one algorithm for obstacle avoidance. Develop a different motion planning algorithm that does not make use of potential forces. For instance, you could exploit the specific geometry of the KUKA robot to come up with an intuitive algorithm that plans a trajectory that avoids the obstacles.

Show your results to the lab TA. Shoot a movie of the trajectory.

2. We will run a competition. The top movies will be posted on the web.

5 Submission

No later than one week after your lab session, submit on Quercus a zipped folder containing:

1. ALL of your Matlab functions. Thoroughly comment your code. Your folder may contain intermediate Matlab functions, but calls to the main functions must work without any modification required.
2. Any movies captured throughout the lab.