University of Technology Sydney
Faculty of Engineering and IT
Robotics

Lab Assignment 1
"SafeCo" Assembly Task

Student: Thiago Lages Rocha | 13357191

Professor: Gavin Paul

Sydney 2019.1

Summary

List of Figures

1 Introduction

This assignment proposes an assembly task utilizing two robot arms capable of grasping different objects. The arms should grasp the objects on a pre-determined order, assembly them and put the complete part away, once completed. Safety measures needed to be provided in order for this task to actually be implemented in real life, so students had to take that in consideration when developing their solution.

2 Objective

The main objective of this work is to provide students with a real world application of robotics in a way they can use the knowledge acquired so far on the subject, utilizing the tools and the references provided, in order to perform the task in a safe manner.

3 Design of the workspace and environment

One of the main concerns in robotics when it comes to having machines working on the same environment as humans, is safety. But not only that, both arms should also never hit each other.

That is why I proposed an approach to solve that problem, when defining the workspace for each of the robots. As we can see on figure 2, there is a clear separation between the two workspaces. To do that, I followed a few steps:

1. Obtain the two base transforms for each arm;
2. Calculate the median point of the line that goes from one base to another (p_0);
3. Get the unit vector that points in the direction from arm1 to arm2 (v). This will be the normal of the plane;
4. Given a point and a vector, we have a plane. With that, for each point that a given arm can reach, I calculate if it is above or below the plane, and thus, if it's a safe position or not.

To do the last part, I needed to properly calculate whether the point of interest p was above, below, or on the plane. Given the notation above, I just calculated the dot product between v and $(p - p_0)$, as show in figure 1 and by the equation below:

$$t = v \cdot (p - p_0). \tag{1}$$

This gives us the projection of $(p - p_0)$ over the normal of the plane, which means if we get a positive value for t, point p is above the plane; if it's a negative value, it's below the plane, and if it equals to zero, then it is on the plane. For the sake of this exercise, I considered all points on the plane to be part of arm1's workspace.

With that information in hands, it has been turned into some code to determine if that point was safe or not for a given robot. From that, I could simply iterate though the robot's joints and move them around, by fix increments, in order to plot the entire workspace, as seen on image 2.

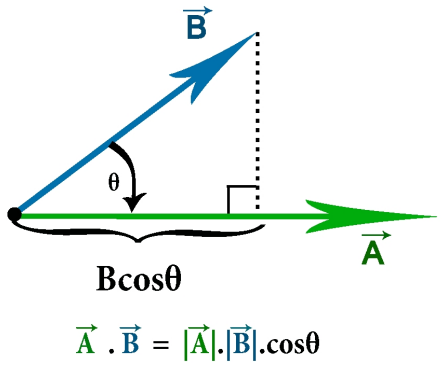

Figure 1: Spacial representation of the dot product between two vectors.

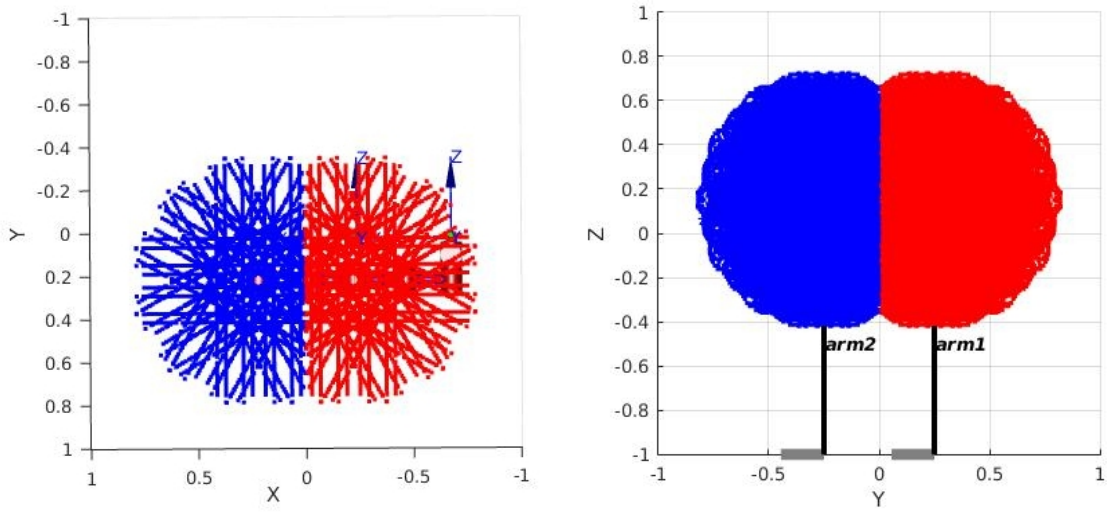

Figure 2: Workspace of two robot arms. Points in red refer to arm1 and points in blue refer to arm2. Left image is top view, right image is side view

Image 3 shows the entire environment around the robot arms.

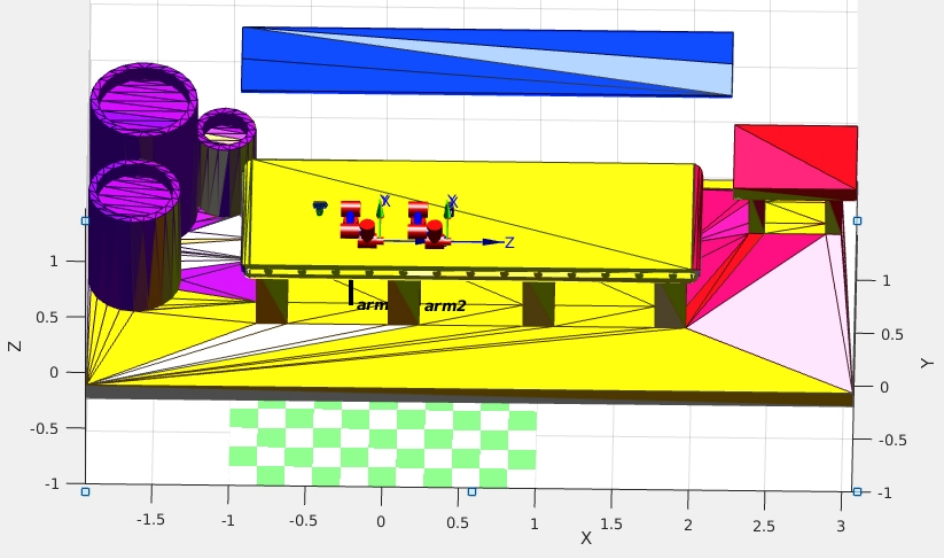

Figure 3: Environment around the robot

4 Robot of choice

We chose the UR3 robot, from Universal Robots, seen on figure 4. It is a really nice robot arm to use on an academic environment since it's easy to use and program, it has safety measures like stop/panic buttons, and it is capable of being physically controlled by a human being, moving its parts around with almost no risk.

Besides that, there are certainly a few limitations, like torque limits, payload, reach, and others. We don't expect it to be a real industrial robot capable of doing heavy lifting, or doing tasks extremely fast. For the sake of the exercise, it is a great choice since the task does not demand such characteristics.

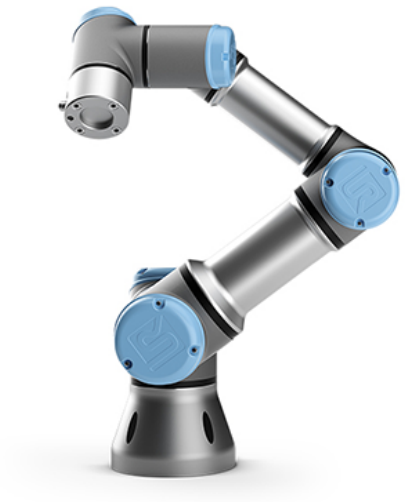

Figure 4: Universal Robot's UR3.

5 Development Tools

5.1 MATLAB

MATLAB is an easy-to-use IDE that allows developers to quickly debug their code, to see what's inside each variable, download and use many different toolboxes available, build Live Scripts, which are a kind of notebook that allows users to write text and code in order to edit their code in a more presentable way, among many other things. The use of MATLAB facilitate a lot of the things students needed to do, like simulating the environment around the robot and also its' movements.

5.2 Peter Corke's Robotics Toolbox

That brings us to the main component of this assignment, in terms of code. Peter Corke's Robotics Toolbox provided the students with many different solutions to most, if not all, of the problems students faced in robotics so far. It provided us with:

1. Defining a robot thought DH parameters;
2. Plot the robots' pose given joint angles;
3. Get the robots' joint states;
4. Do Forward Kinematics in order to find end-effector pose;
5. Do Inverse Kinematics in order to find joint angles, given an end-effector pose;
6. Create trajectories given two desired points.

That allowed us to solve the problem in a much easier way than implementing everything from scratch.

6 Task related challenges

Grasping objects with complex shapes is not an easy task, so students were allowed to consider that the object was grasped once the robot reached its' position. That made things way more simple and so we wouldn't have to worry about the actual grasping procedure. Otherwise, we would have to consider many things, shown further on this section.

6.1 Precision

Dealing with objects with a few centimeters in length, and assembling them on a millimeter scale is a really hard challenge. For that, all the motors would have to be really well calibrated and precise in order to achieve that precision. A minor drift on a motor's position would probably propagate, since we have motors on each joint, and the actual position of the end-effector would probably change a lot from desired.

6.2 Sensing

When it comes to sensing, we would need not only really precise methods for getting each motor's position (like high-precision encoders), but also sensing how close the robot arms are to the objects they are grasping, and how close they are to achieve the perfect assembly of the parts. That would take either computer vision methods (that could be hard to achieve due to the position of the parts, or the positioning of the cameras on site), or really high-precision sensors, which could cost thousands of dollars.

6.3 Grasping

The actual grasping is hard if we consider how complex theses parts are, as shown on figure 5. To achieve the actual grasping on a good position, it would take a good amount of time to first determine which position in that, given the available gripper, and second, to determine in which orientation the object is (if they were loose, and not on a pre-determined position), and in which position the gripper needed to be in order to grasp the object.

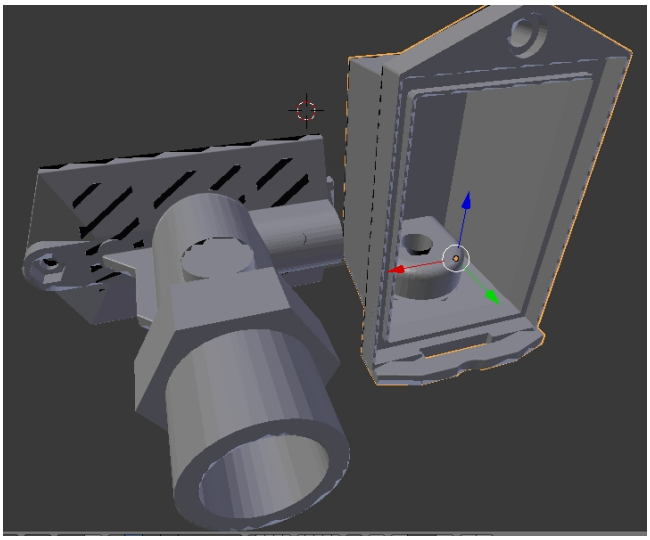

Figure 5: Parts that needed to be assembled (Circuit Board not shown).

7 Conclusion

This was a great assignment to gather all the information students acquired on the past weeks, that made them think about the problem and a good way to solve it with the tools provided on the subject. A few obstacles were found on the way, but with some effort they have been solved. The fact that we had to consider safety measures on the assignment also made the work harder, but necessary. Finally, a few assumptions like ignoring the actual grasping procedure made the work simpler and easier.

8 References

References

[1] Peter Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*, Springer Publishing Company, Incorporated, 1st edition, 2013.

[2] Gavin Paul, "Robotics lecturers and lab exercises," 2019.

[3] Universal Robotics, "Ur3 collaborative table-top robot arm that automates almost anything," https://www.universal-robots.com/products/ur3-robot/, April 2019, Accessed on 2019-04-01.