# Introduction to Vision and Robotics 2020-21

Assignment

The assignment is due by 4 pm on Friday 27th of November

November 2020

This assignment will be extending a lot of what was previously done in the labs to a non-planar (3D) robot with 4 degrees of freedom. You will be working in groups of two. There will be, however, no extra credit for completing the assignment alone. The assignment is going to be broken down into 3 sections that cover robot vision and control.

## 1 Overview of the Simulator

### 1.1 Installation of Lab Library

1. Use UBUNTU as instructed in labs.
2. Create a folder in your home directory, here we named the folder **catkin_ws**, enter the folder and create another folder named **src**. You can do it via the terminal:
   **mkdir catkin_ws**
   **cd catkin_ws**
   **mkdir src**
3. The package for the lab is in a repository (repo) on Github which you can find through the following link: `https://github.com/mohsenkhadem1/ivr_assignment.git`. You can download it, unzip, and put in a folder named **ivr_assignment** inside the **src** folder you just created. **Important:** Make sure the folder that includes the downloaded files is named **ivr_assignment** and is inside the **src** folder.
4. Get back to your workspace folder **cd ∼/catkin_ws**. Use the following command to install the package you just downloaded:
   **catkin_make**
   **source devel/setup.bash**
   **Important:** The setup.bash file will have to be sourced as above in every new terminal. You can also add this to the end of your .bashrc file to automatically do it whenever a new terminal is opened. To do so you can open  **/.bashrc** using **gedit  /.bashrc**. Add the following two lines at the end of the opened file and save:
   **source /opt/ros/melodic/setup.bash**
   **source ∼/catkin_ws/devel/setup.bash**

5. Once the installation is completed you can navigate to the ∼**/catkin_ws/src/ivr_assignment/src** folder and run the following command to make the python files executable:

   **chmod +x image1.py image2.py target_move.py**

   Note: If you create any python files in this folder you should make them executable using **chmod** command.

Once the installation is completed you can navigate to the ∼**/catkin_ws** folder and run the robot simulator using:

**roslaunch ivr_assignment spawn.launch**

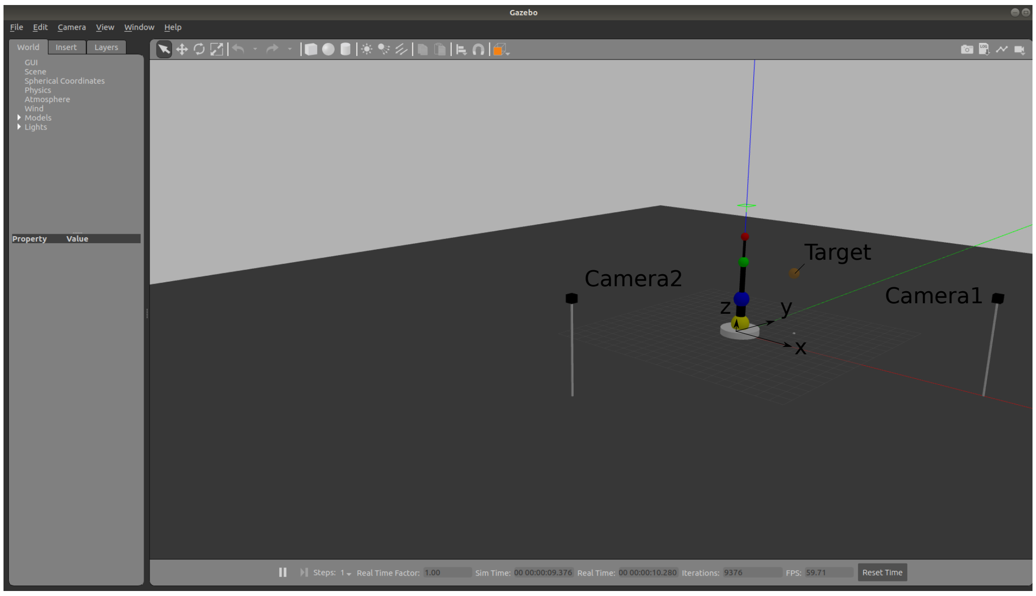

Figure 1: The robot in Gazebo simulation environment, and two camera and a moving target.

This opens the simulated robot in ROS simulation environment. The robot has 4 degrees of freedom and moves in a 3D world. The joints of the robot are shown by yellow (joint 1), blue (joint 2 & 3), and green (joint4) spheres. The red sphere is the robot end effector. Joint 1 is a revolute joint rotating around its z axis. Joint 2 and 3 are located at the blue circle and are revolute joints moving around the drawn x and y axes, respectively. Joint 4 is a revolute joint rotating around the x axis. The table below gives a description for each link:

| Link | Length | Rotation axis | Joint location |
|---|---|---|---|
| 1 | 2.5 [m] | z | Yellow |
| 2 | 0 [m] | x | Blue |
| 3 | 3.5 [m] | y | Blue |
| 4 | 3 [m] | x | Green |

The robot is in position control mode. You can move the robot's joints using the following command in a new terminal:

**rostopic pub -1 /robot/joint1_position_controller/command std_msgs/Float64 "data: 1.0"**

This line of code will move the joint 1 by 1 Radian. You can start to get familiar with the simulator by changing the joint angles. If you want to move another joint change the joint number in the above code. Joint 1 can be moved between $-\pi$ and $\pi$ and the rest of the joint can be moved between $-\pi/2$ and $\pi/2$.

The simulator is also designed to return two RGB arrays from two cameras placed around the robot. The images from the cameras are what you will be performing computer vision algorithms on.

The main codes which you shall use in the labs is inside **~/catkin_ws/src/ivr_assignment/src** and named **image1.py** and **image2.py**.

This codes receive the images from the cameras, process it and publish the results. To run the codes, open up a new terminal and use these commands:

**rosrun ivr_assignment image1.py**

It should show the image received from camera1.

**rosrun ivr_assignment image2.py** should show the image received from camera2.

Try to move the robot and check if the robot moves in the camera views. You should modify these codes as you did in the labs.

## 2 Robot Vision (20 marks)

### 2.1 Joint state estimation (15 marks)

As part of the new simulation you are given two orthogonal views on the zy plane and the xz plane. Similar to the labs you will need to calculate the joint angles for the robot to use. For this part of the assignment assume joint 1 is fixed. Next, update the python codes (image1.py or image2.py) to move the other three joints using the following sinusoidal signals:

joint2= $\frac{\pi}{2}\sin(\frac{\pi}{15} * t)$

joint3= $\frac{\pi}{2}\sin(\frac{\pi}{18} * t)$

joint3= $\frac{\pi}{2}\sin(\frac{\pi}{20} * t)$

where $t$ denotes the time.

Estimate the value of the 3 joints using computer vision. You may use whichever technique you want. Note that some of the spheres might not be clearly visible in one camera when the robot moves in 3D or when the robot tip is touching the target. Your algorithm should be able to handle these scenarios.

In your report, describe your algorithm (3 marks) and provide 3 plots (each 4 marks). In each plot, you should compare the estimated value of one joint with the sinusoidal signal that you used to move the joint. The plot should show the estimated joint values for at least 10 seconds.

### 2.2 Target detection (5 marks)

The second part of the vision component will be to detect the target in the image. Open up a new terminal and use the command:

**rosrun ivr_assignment target_move.py**

This will move the two orange objects around the robot. There are two moving orange objects, a box and a sphere. You should write an image processing code to estimate the position of the sphere in meters with respect to the robot base frame. As mentioned earlier both targets have the same RGB values, so colour alone will not disambiguate them. Feel free to crop any part of the images to gather samples if you need.

In your report, describe your algorithm and comment on the sources of error in your measurements (2 mark). Provide a plot (3 marks) showing the estimated x, y, and z position of the sphere. The plot should show the estimated joint values for at least 10 seconds.

### 2.3 Expected outcome

By the end your system should be able to accurately acquire the joint states of the robot (maximum of 0.15 radiant error), and be able to identify the correct target for the robot to reach to, i.e. the orange sphere.

To plot the results you can publish your estimated joints/measurments on a topic and use **rqt_plot** command. We have used several codes in the labs to create a publisher and publish data. Check out the lab solutions if you don't know how to do it. To plot two topics vs each other you can use

**rqt_plt [TOPIC1_NAME] [TOPIC2_NAME]/data[num]**

use TOPIC1_NAME if the topic is only one number and [TOPIC2_NAME]/data[num] if the topic is an array . The actual position of the target are published under the following topics:

**/target/x_position_controller/command**

**/target/y_position_controller/command**

**/target/z_position_controller/command**

To change the limits of rqt plot click on the arrow shaped button. Select a reasonable length to show the results, e.g. 50 seconds. You can save your plot by clicking on the save button. Additionally, you can record your data using **rosbag record** command and plot them in Matlab. See the following link for more information `https://uk.mathworks.com/help/ros/ug/work-with-rosbag-logfiles.html`.

## 3 Robot Control (20 marks)

### 3.1 Forward Kinematics (10 marks)

In this section we will be using the joint angles you have obtained from the previous part. Calculate the equations for robot Forward Kinematics (FK). We are interested in controlling the x,y, and z position of the robot end-effector. The FK should look like this:

$$\begin{bmatrix} x_e \\ y_e \\ z_e \end{bmatrix} = K(q) \text{ ,where } q = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix} \tag{1}$$

$K$ is a $3 \times 1$ vector, $x_e$, $y_e$ and $z_e$ are the robot end-effector positions, and $q$ is a vector of joints' angles.

Present the results of your Forward kinematics calculation (5 marks). To verify your FK, move the robot to 10 different points across it's workspace using **rostopic pub** command. Do not keep any of the joints zero. Compare the estimated end-effector position via the images to the estimated end-effector position via FK in a table and comment on accuracy (5 marks).

### 3.2 Closed-loop Control (10 marks)

From the previous tasks you can acquire an estimation of the joint states and the location of the correct target. Calculate the Jacobian of the robot. Develop a Controller similar to lab3 to follow the target sphere.

By the end your robot should be moving to the correct target. Not only this you should be able to report quantitatively the accuracy of the arm movement. Present the results of your Velocity kinematics calculation (4 marks). Present three plots (2 marks for each) comparing the x,y, and z position of the robot end-effector with the x,y, and z position of the target (i.e, the sphere). The plot should show the positions for at least 10 seconds.

## 4 Perform one of the following tasks (10 marks)

### 4.1 Joint state estimation (10 marks)

Repeat the process described in section 2.1. This time do not fix joint 1. Move the joints using the following sinusoidal signals:
joint1= $\pi\sin(\frac{\pi}{15} * t)$
joint2= $\frac{\pi}{2}\sin(\frac{\pi}{15} * t)$
joint3= $\frac{\pi}{2}\sin(\frac{\pi}{18} * t)$
joint3= $\frac{\pi}{2}\sin(\frac{\pi}{20} * t)$
where $t$ denotes the time.

Estimate the value of the 4 joints using computer vision. You may use whichever technique you want. In your report, describe your algorithm (2 marks) and provide 4 plots (each 2 marks). In each plot, you should compare the estimated value of one joint with the sinusoidal signal that you used to move the joint. The plot should show the estimated joint values for at least 10 seconds.

### 4.2 Null-space Control (10 marks)

Develop a null space controller that uses the robot's redundancy to make sure the robot doesn't hit the orange box while the robot's end-effector follows the spherical target.

Discuss your algorithm and how it is different from the previous controller. (4 marks). Present three plots (2 marks for each) comparing the the position of the robot end-effector with the position of the sphere and the position of the box. 1 plot for x positions, one for y, and one for z.

### 4.3 Joint state estimation (10 marks)

Copy the robot.urdf file inside the ivr_assignment folder into the urdf folder and replace the existing robot.urdf file. Launch the gazebo simulator. You will see that the robot's joints are all black. Repeat the process described in section 2.1 using the newly updated robot. Move the joints using the following sinusoidal signals:
joint2= $\frac{\pi}{2}\sin(\frac{\pi}{15} * t)$
joint3= $\frac{\pi}{2}\sin(\frac{\pi}{18} * t)$
joint3= $\frac{\pi}{2}\sin(\frac{\pi}{20} * t)$
where $t$ denotes the time.

Estimate the value of the 3 joints using computer vision. You may use whichever technique you want. In your report, describe your algorithm (4 marks) and provide 3 plots (each 2 marks). In each plot, you should compare the estimated value of one joint with the sinusoidal signal that you used to move the joint. The plot should show the estimated joint values for at least 10 seconds.

## 5 Submission

You will need to submit a report in the format of studentNo1_studentNo2.pdf. Each group should submit one file. At the beginning of the report explain who has worked on what part of the project. In your report give us a link to your GitHub account for downloading your final ROS library. Your repository should be public. Only one link is needed for each group. We should be able to download your library and run your executable files and get the results of part 3 of the project (in section 4) without errors. Your GitHub account should clearly show the history of development of your code. Do not copy and paste the final library in GitHub. Gradually update it (commit and push) during the course of four weeks as you are writing your code.

The final report should include plots and short answers to the questions asked in the assignment. Your figures should be clear, readable, with legend, and all axis labeled properly. The report must be maximum of 5 pages, minimum of 1 cm margin on all sides, single spacing, and font of 11. Reports that don't follow these guidelines will not be marked.