README FILE

PDE-3433 Advanced Robotics – Robot Visuomotor Control

Elena Zoretich, M00725319, 3rd year Robotics student (2022/2023).

Task Description

For this third assessment we have been asked to create an algorithm which will allow a UR10 manipulator to catch a flying object, specified as a ball, within a simulated environment. The simulation will be run in a Virtual Machine (VM) running on a distribution of Linux and it will be launched thanks to the use of ROS and Python 3 scripts. The simulation environment will show the UR10 manipulator, placed in a centred, fixed position, although neither the flying object nor its trajectory will be visible. In order for the manipulator to catch the ball, the algorithm must work out a method to reduce the distance between its end effector and a series of given errors, described as:

- Cartesian Error: describing the distance, in meters, between the end effector and the falling object trajectory. This error does not specify the distance along the x, y and z coordinates individually, but rather return a generic error message.

- Angular Deviation Error: describing the rotational distance, in degrees, between the direction of the falling object and the end effector orientation. The error message will print a singular value, not specifying the individual angular disparity of Roll, Pitch and Yaw.

- Ball_Caught Error: this error message will return a Boolean value, indicating if the ball has been caught (True) or not (False).

For the ball to be considered caught, both cartesian and angular deviation errors must reach a value equal or below a specified threshold, set as respectively 0.05m and 5°.

Installing & Libraries

Since this assignment requires a distinct ROS environment specifically created and set in our VM, it is implied that the user has already installed and properly configured the VM settings, therefore, no guidance on the software installation will be mentioned. Furthermore, not additional libraries have been used to carry on the task.

Repository Content

The “ElenaZoretich_A3_Robot_Visuomotor_Control” repository includes a total of 6 files:

README File

A PDF file describing the project’s main specifications, the files found in the submission folder and ‘How To’ guidelines to launch and execute the simulations and given files.

UR10_Report_EZ_M00725319

A report thoroughly describing the task environment, the problem analysis and the functionality of the designed algorithm.

A3_Demo_ElenaZoretich_M00725319_Robot_Visuomotor_Control.mp4

A recorded video demonstration of the UR10 successfully executing the algorithm, with additional voiceover narrating how to properly setup the simulation environment and code's logic.

P.S.: If any issue is encountered while trying to play the .mp4 video, the demonstration is also available as YouTube listed video. Please find its link below:

https://youtu.be/gKXV0nrc2Ts

Control_Lib.py

This script includes the UR_Controller() Python class. The UR_Controller objects contains a series of functions used in the main scripts, to either receive or send messages between the ROS UR10 and the Python scripts.

Single_Throw_UR10_CatchBall_A3_ElenaZoretich

This file contains the designed algorithm. Once run, it will allow the UR10 manipulator to perform a series of movements until the flying object has been caught. This script will execute the algorithm only once, therefore when the ball will be caught, its execution will terminate.

Multiple_Throw_UR10_CatchBall_A3_ElenaZoretich

This file also contains the designed algorithm. Unlike the above described file, this script will continuously run the algorithm, re-starting its sequence every time the flying object has been caught.

Locating Python Scripts

Once the submission folder has been downloaded, all files have been extracted and the user is now within the Virtual Machine environment, it is recommended to place the 'Control_Lib.py', 'Single_Throw_UR10_CatchBall_A3_ElenaZoretich' and 'Multiple_Throw_UR10_CatchBall_A3_ElenaZoretich' files in the 'scripts' folder located inside the 'robot_control_simulation' package.

[catkin_ws → src → robot_control_simulation → scripts]

By doing so, it will be possible to run the scripts with the same method that has been used for the existing examples files during in class demonstration.

How to Launch the Simulation

Run the Virtual Machine. Once inside the Linux environment, open the ROS terminal and follow the instructions below:

Step 1: Setup URSIM

Navigate to the directory where the URSIM has been installed.

1. cd to ~/ursim-3.15.4.106291

Run the simulator with Admin privileges.

2. sudo ./start-ursim.sh UR10

Wait for the simulator to open and click through all the error messages. Click on the 'About' button and copy the IP Address displayed, then close the simulator window.

Reopen the simulator without admin privileges.

3. ./start-ursim.sh UR10

Once open, to visualize the UR10, click on the 'Move' tab above.

Step 2: UR-Ros drivers

Open a new terminal window.

Option 1: Use provided launch file.

Prior the first launch of the simulation, the launch file needs to be edited.

4. gedit ~/catkin_ws/src/robot_control_simulation/launch/ur_ros_drivers.launch

Once the file opened, change the "robot_ip" parameter to match your URSIM's IP

When done, close the file, and back in the same terminal launch the robot drivers.

5. roslaunch robot_control_simulation ur_robot_drivers.launch

Option 2: use the packages directly.

4a. roslaunch ur_robot_driver ur10_bringup.launch robot_ip:="insert_URSIM's_ip_here"

Step 3: Start ball trajectory spawner.

Open a new terminal window.

Run the trajectory service to spawn the ball

6. rosrun robot_control_simulation ball_traj

You will be asked to input either option 1 or 2. Since to catch the ball both cartesian and angular deviation error must reach below a specified threshold, it is extremely encouraged to select input number 2.

7. Input either 1 (discouraged) or 2 (encouraged)

Additionally to the instructions provided in this ReadMe file, a 'How to use the package' guide should have been provided with the robot_control_simulation package.

Running the Script's Algorithm

To run the script containing the algorithm, open a new terminal, type 'rosrun', the name of the package containing the algorithm script, and the name of the script itself.

If the user wishes to run the algorithm only once, the command to run in the terminal should be as the following:

8a. rosrun robot_control_simulation Single_Throw_UR10_CatchBall_A3_ElenaZoretich

If the user wishes to run the algorithm continuously, the command should instead be the following:

8b. rosrun robot_control_simulation Multiple_Throw_UR10_CatchBall_A3_ElenaZoretich

Although the scripts are extremely similar, with only a main, external While Loop as difference, it is encouraged to try and run both scripts.

Authors

Elena Zoretich, Middlesex University, Student ID M00725319.

Email = ez111@live.mdx.ac.uk

Github IS & URL: EZoretich, https://github.com/EZoretich

Acknowledgements

The problem-solving approach used to complete the task personally consisted in a blend of critical and logical thinking, and heuristic methods. Brief research on greedy algorithms has been done, but the used resources remain limited to:

- Library and examples Python scripts provided by the lecturer.

- Introduction to A* Algorithm:
  https://www.redblobgames.com/pathfinding/a-star/introduction.html

- Classification of Greedy Algorithms:
  S. A. Curtis, 'The classification of greedy algorithms', *Science of Computer Programming*, vol. 49, no. 1–3, pp. 125–157, 2003.