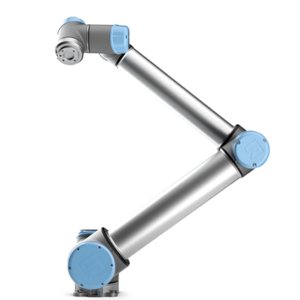
Robot visuomotor control

*Assessment 3 - PDE3433 Advanced Robotics*

Program for catching a flying/static object thrown into the robot workspace with a UR10 arm, without direct axis movements, inside the ROS ecosystem.



***Code Explanation***

The robot only depends on the error obtained by checking its position in relation to the destination where the object is going to fall. By relying only on an error that doesn't specify which direction it is, the code is forced to check all, in the case of this code, just X and Y.

Chart

Description automatically generatedThe object will fall from a height of 2 m, among other specifications. Therefore, you can imagine the crash area in 2D. This being the case, it is not too necessary to review all the axes. The code checks the X and Y axes, through individual while loops for each one, which are inside another while loop, thus making each axis position search stop only if the minimum possible axis has been reached. Or if the robot has reached a Cartesian error less than 0.05.

The axes are changed through moves of a size of 0.02 or less, for better occurrence. As security measures, if at the end of the loop it indicates that the Cartesian error is less than 0.05, another position check will be made to confirm that the robot has not moved by mistake, or the user has not interacted with the simulator buttons. If, upon reviewing the error again, it is confirmed that it is less than 0.05, the code would continue with the orientation verification; however, if the error is greater, the loop would restart.

Figure 1: Object Launching

Text

Description automatically generatedOnce the rotation is reached, the X and Y axes are no longer referenced. What begins to be reviewed would be the specific axes of the rotary tool, in this case catalogued as RX, RY and RZ. These are receiving values in order, which allows them to minimize the eastern error. Again, the financing is done through loops, but this case is a for loop. This allows us to check the RX, RY and RZ in order and minimize it below 5 degrees. When the Cartesian and Oriental errors are below the required minimum, the Robot can capture the object and generating a new point at which the next one will arrive.

Figure 1: Object Launching

**Observations and Clarifications:**

* The movements of the axes are all linear, except for the “home waypoint” which is established through the angle of each joint and the rotation tool.
* The complete rotation of each joint is PI = 3.1416… so we can calculate where we want to set the degree we want. This can be done by dividing PI/360 degrees.
* The movements are 0.02 or less so that the final error is more precise and therefore the code slows down. On the other hand, if we change the facings and obtain a larger movement, there is a high probability of the slightest error of each axis incorrectly.
* If the minimum error of X is miscalculated, for example, this would make the error of Y larger.