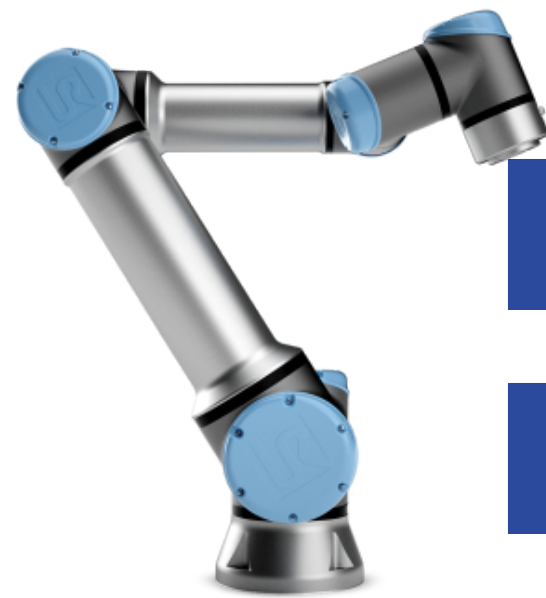





PROJECT EXAM



**FONDAMENTI
DI ROBOTICA**

Large, stylized blue geometric shapes, including a large arrow pointing right and a smaller triangle, located on the right side of the slide.

ASSIGNMENTS

There are four different assignments with increasing difficulty. In each of them, the 3D camera has to recognize the blocks on the table, retrieve the position and the orientation of them, and finally the blocks have to be taken by the robot arm and moved in a certain position (drop area). The assignments are the following:

- **Assignment 1:** There is only one object in the initial stand, positioned with its base “naturally” in contact with the ground.
- **Assignment 2:** Multiple objects on the initial stand, one for each class. The base is “naturally” in contact with the ground.
- **Assignment 3:** Multiple objects on the initial stand that can be rotated and lying on one of their lateral sides. They can't be one on each other.
- **Assignment 4:** The objects on the initial stand are those needed to create a composite object like a castle.

ENVIRONMENT

The software used to allow the communication between the different parties of the project is the middleware **ROS** (Robot Operating System).



Topics and services are used to share messages between the nodes, such as the robot, the vision node and the motion plan node.

The project is divided into two big areas:

- the **computer vision part**, to recognize the objects and retrieve their position, developed entirely in Python.
- the **motion planning part**, taking care of the kinematics and the movement of the robot, mainly developed in C++.

COMPUTER VISION

Detection

In the detection phase, it's possible to choose between two types of detection: *live detection* and *background detection*.

- *Live detection*: it shows the processed image previously captured by the zed camera.
- *Background detection*: it doesn't show the processed image previously captured by the zed camera.

Image processing

In this phase, the yolo model is firstly loaded and then used to detect any objects inside the captured image. Additionally, a threshold is used in order to discard uncertain values.

Once done, a dictionary of the detected objects is ready to be processed in the next phase.

COMPUTER VISION

Object processing

Firstly, the box image detected by yolo is filled of black except for the pixels representing the object. Then, all the 2D points are converted to their corresponding 3D points. The characterization of the object consists of these parts:

- *Gripper point*: To calculate a valid point to pick the object, we first retrieve the maximum and minimum x and y 3D coordinates. The gripper point will be determined by calculating the center of the rectangle formed using the x and y coordinates. From this rectangle, the gripper point will be selected as the point with the maximum z-coordinate along the z-axis.
- *Bottom side*: It's identified by selecting the largest row among the last three rows of pixels, ensuring a more accurate representation of the object's resting surface on the table.
- *Left/Right side*: This points are determined by comparing their x-coordinates to the leftmost/rightmost point of the bottom side. Points with similar z-coordinates to the bottom reference point are selected, and the farthest point from the bottom side is identified. If it aligns with the leftmost/rightmost point, it is aligned with the reference point.

COMPUTER VISION

- *Orientation*: The method used to calculate the orientation of the object involves different techniques:
 - Z-axis orientation, the angle is calculated using the left side and leftmost bottom points. The arc-tangent of their y-coordinate and x-coordinate difference determines the angle, with an adjustment of 90 degrees if the left side is shorter than the right side.

Main

The script subscribes to the ROS topic where the zed camera publishes the captured image and once done, it invokes a custom callback function.

ROBOT MOTION

Kinematics

The kinematics describes the motion of points and bodies (objects).

- The *direct kinematics* takes as input the set of the 6 values of the robot joints and returns the transformation matrix (composed by position and rotation matrix) of the end effector.
- The *inverse kinematics* takes as input a 3-dimension vector representing the final position of the end effector, and calculates eight possible combinations of values for the six joints.

Function "p2pMotionPlan"

The p2pMotionPlan function computes a path of points between a starting point and a final point.

Function "threep2p"

The threep2p is the function used to build an entire movement from a starting point to a final point. It joins three "p2pMotionPlan" functions, in order to reach two different intermediate points along his path to the final position.

ROBOT MOTION

Check functions

There are three main function that performs some checks:

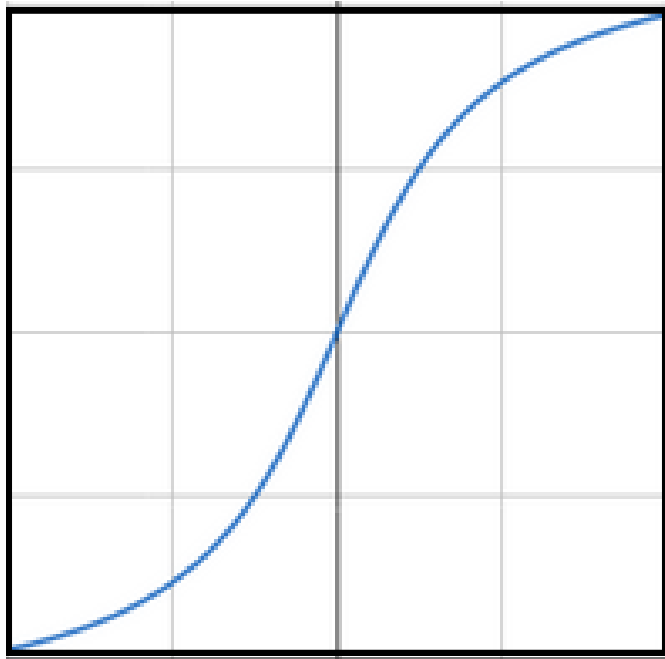
- *check_point*, checks if the inserted position is reachable by the robot, by comparing the result of the inverse kinematics and the result of the direct kinematics applied to the joints returned by the inverse.
- *check_angles*, checks if the computed joint by the inverse kinematics and the p2pMotionPlan are reachable by the ur5 robot. The joint values are checked to be within a range of allowed values.
- *check_collision*, checks if the final pose of every single link of the robot is over some space limits, such as the table height and the backward wall.

Main

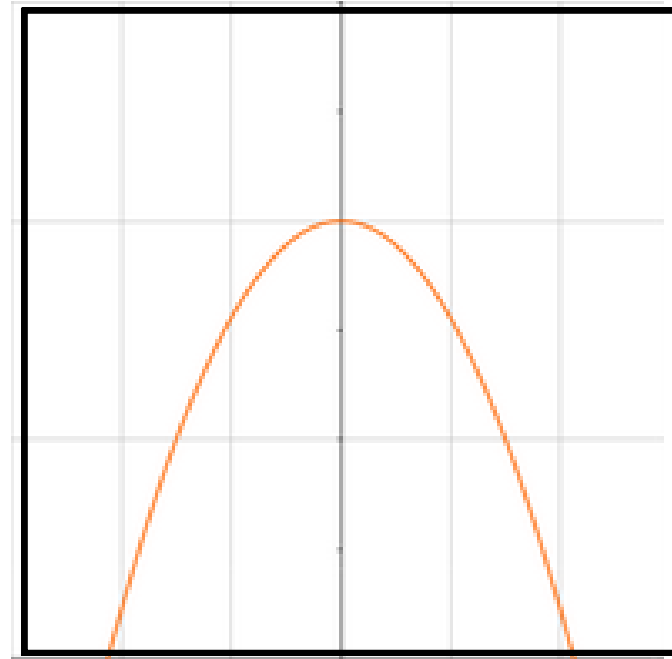
- A starting procedure is performed, to take the robot in a well known position. Then the node enters a while loop in which, firstly waits for the position to reach, sent by the vision node. Then, checks if the inserted position is reachable by the robot and if a rotation is needed. Once done, compute the movement by managing the opening and closing of the gripper. Finally, publishes the computed joint configurations in a topic.

ROBOT MOTION

Motion Plan characteristics



position



velocity



acceleration

CONCLUSIONS



<https://github.com/christiansassi/robotics-project>



Christian Sassi - 217640
christian.sassi@studenti.unitn.it

Luca Pedercini - 218551
luca.pedercini@studenti.unitn.it

Pietro Bologna - 218186
pietro.bologna@studenti.unitn.it

**THANKS
FOR YOUR
ATTENTION**