UNIVERSITÀ
DI TRENTO

Department of Information Engineering and Computer Science

Report of

Robotics Project

Authors

Badole Elena

Bonaccorsi Riccardo

Caporale Federico

Lorengo Sofia

Professors

Falqueto Placido

Focchi Michele

Palopoli Luigi

Sebe Niculae

Academic year 2022/2023

Indice

1 Abstract

The objective of this project is to locate some objects, detect their positions and move them from a initial position to a final position through a robotic arm.

In particular we have a manipulator, a 3D sensor and a set of mega-blocks. The first one is an anthropomorphic arm with a spherical wrist and a three-fingered gripper as and-effector. The second one is a calibrated 3D sensor that is used to locate the different objects and to detect their initial position. As far as objects are concerned, they are represented by mega-blocks whose geometry is known. They belong to different classes each of which denotes a final position to put the block.

The environment of assignment 1 is characterized by the presence of a single block that can belongs to any classes and that is positioned with its base in contact with the ground.

2 Perception

Perception is an essential aspect of robotics that can be explained as the ability of a robot to perceive and understand its surrounding environment. It can carry out these activities thanks to the presence of various type of sensors.

In this initial part of the project we need to recognize and classify the blocks that are present within the workspace of the robotic manipulator. To do this we use a ZED camera that is placed on the side of the table where the blocks are placed. This camera uses stereo vision, i.e. it captures images from two slightly offset cameras in order to perceive depth information.

2.1 Roboflow

For this project we decided to use the external service made available by Roboflow to train the robot's perception system. Roboflow is a platform that offers tools for image processing and analysis, using machine learning techniques. We made use of its performance to train our model through a series of labelled images containing blocks.

2.2 Program functioning

The program *cameraListener.cpp* manages the images coming from the ZED camera. These images are sent to the external object detection service mentioned above (Roboflow), which returns the location and class of the objects. After various data processing, the program publishes on a ROS topic the necessary information about the objects,that are the coordinates and the class. This information is relevant because it will then be useful for deciding the movement of the robotic arm.

2.3 Analysis of what happens to an immage

The image is captured from the camera and sent to the ROS node on the topic. The subscriber of the program subscribes to the topic and attends the arrival of the image. Once the image is received, the program calls the *imageCallback* function for its management. Within the function, the image is converted from ROS to OpenCV. Several operations are performed on the image, such as changing the contrast and cropping the area of interest. The image is then saved to a JPEG file that is sent to the Roboflow object recognition service. It processes the image and returns a string containing predictions about detected projects. Predictions are processed and transformed using geometric transformations. Finally, predictions are published on a ROS topic using a publisher. This allows other nodes to access this information.

2.4 Code explanation

2.4.1 main

The main takes care of initializing the ROS node, creating the subscriver and starting the execution loop that allows the node to listen and process images.

2.4.2 toString

This function receives a 3D matrix and converts it into a string.

2.4.3 ritagliaImmagine

This function takes in input an image and, given a rectangle of interest, it cuts out the image. To avoid image recognition errors, the image is cropped with predefined criteria so that you only have the area of interest.

2.4.4 extractIntegerWords

This function extracts predictions from a string.

2.4.5 sostituisciCarattereNewline

This function receives an input string and replaces the newline character with a space. It is used when concatenating strings that contain the information of reconnected objects.

2.4.6 exec

This function runs a system command. In this case it sends the image to Roboflow, that is the object recognition system, and returns the string containing the information sent by Roboflow.

2.4.7 imageCallback

This function handles the reception of an image. In detail the function converts the ROS image to the OpenCV format, manipulates it and sends it to the image reconfiguration system. Based on the predictions received, the function calculate the coordinates for each object and concatenates them with the class of the object. Finally it sends this information to the ROS node with the publisher declared within the feature.

3 Motion

After that the class and initial position of the blocks has been correctly identified through perception, the robotic arm must move correctly.

3.1 Algorithm

Motion is a fundamental but complex aspect of robotic arm. There are in fact multiple possible algorithms that can be used to move the robot from one position to another. The best idea is to make the arm travel as little road as possible, therefore the arm should follow a linear trajectory. Knowing the start point and the end point we can calculate the distance and modifying the angles so that the robot follows this line.

3.1.1 Forward kinematics

Forward kinematics makes possible to calculate the position and orientation of the end effector of a robot. It is possible thanks to the knowledge of the known joint parameters.

Direct kinematics is often represented through homogeneous matrix transformation, which allows to rappresent in a compact way the position and orientation of a robot system using a 4x4 matrix. This matrix takes into account the translations and rotations between the different parts of the robot, allowing to calculate the final position of the system.

3.1.2 Inverse kinematics

Inverse kinematics is the opposite of forward kinematics. In practice it determines the position that the robot must assume in order to reach a given position or a desired orientation of the robot's end effector.

To solve inverse kinematics are often used numerical algorithms such as the Newton-Raphson method or the inverse Jacobian method. These algorithms exploit the geometric and algebraic equations describing the kinematic structure of the robot in order to determine the internal configurations of the robot that satisfy the desired position.

3.2 Code

The *move* method was implemented to make the robotic arm move. It requires as input the final position of the robot, i.e. the position where the block resides, and the speed of movement. Once the execution has started, the initial position of the robot is calculated and the time it will take to reach the desired position. Once these data have been obtained, it is possible to start the iteration of the for loop to periodically calculate the temporary position of the robot through the inverse kinematic. The next position that the robot has to reach along the trajectory is then calculated and the angles of the robotic arm are modified accordingly.

4 Planning

This piece of code is used to plan how a block should be moved. In practice it manages the entire movement part of the robot.

The *plan* function takes as input the data provided by the vision and transforms them into data to be given to the *move* method, in order to move the various blocks. The method also separates the maxi string given in input and saves the position of the block in a structure when the camera performs the recognition. Subsequently the robotic arm moves up above the block, descends just above it and moves the gripper in order to grab it. Once the arm is caught, it moves above the final position, lowers itself again and releases the block. Finally the robot gets up so as not to be in the way.

The final position of the block is determined on the basis of the class to which the block itself belongs