Fundamentals of robotics

Adami Federico - Olivieri Francesco - Veronese Eddie

Kinematics - Vision

1 Project description

The project aims to employ a robotic manipulator, specifically an anthropomorphic arm with a spherical wrist and a two-fingered gripper as its end-effector, to pick up objects from an initial stand where they are stored without any specific order. The goal is to place these objects in a specified order on different stands, referred to as the final stand. The objects belong to different classes, but their geometries are known and encoded in STL files. To accomplish this task, a calibrated 3D sensor is employed to accurately locate the objects and determine their relative positions on the initial stand.
The project is divided into 4 tasks:

1. Detect and move a single object from the initial stand to its assigned position on the final stand
2. Move multiple objects from the initial stand to their respective positions on the final stand
3. Move multiple objects from the initial stand to their prescribed positions, stacking objects of the same class to form towers
4. Sequentially pick up objects from the initial stand and assemble them to create a composite object on the final stand

2 Vision

The project involves using a combination of artificial vision and computer vision techniques to detect and determine the position and orientation of the blocks. Initially, the YOLO (You Only Look Once) model is used to detect the blocks, subsequently the ZED Camera obtains depth information and creates a three-dimensional point cloud around the detected block area. This point cloud is then compared to a basic SDF model to determine the orientation of the blocks.

2.1 Object recognition

YOLO, is an object detection algorithm, it's goal is to detect and classify objects in an image or video in real-time. YOLO tackles the task as a single regression problem: algorithm divides the image into a grid of cells and predicts a set of bounding boxes and the probabilities associated with different object classes for each cell.
The process involving YOLO can be summarized in the following steps:

2.1.1 Training

The training phase begins with data collection, which involves taking photos of the objects in various positions and orientations on the workspace. Subsequently, these images were annotated in a YOLO-compatible format, assigning each object its class and a bounding box. Finally this dataset was passed to YOLO, which initiated the training process using pretrained weights as a starting point. The algorithm iteratively conducted forward passes on the training images, computed the loss by comparing the model's predictions with the ground truth annotations, and updated the model's weights using gradient descent optimization

```
2 0.349487 0.608597 0.051990 0.092760
0 0.692234 0.578620 0.031451 0.078054
```

Figure 1: YOLO format annotation

2.1.2 Detection

In the detection phase of YOLO, the trained model applies a single-shot object detection technique. This involves dividing the input image into a grid and making predictions for each grid cell, eliminating the need for sliding windows or region proposals. By efficiently processing the entire image in a single pass, YOLO achieves

real-time object detection. It utilizes convolutional neural networks (CNNs) to learn and extract features from images, enabling accurate and efficient object detection. The predictions include bounding boxes and class probabilities for detected objects, and non-maximum suppression is applied to eliminate duplicate detections. The final output consists of bounding box coordinates, class labels, and confidence scores

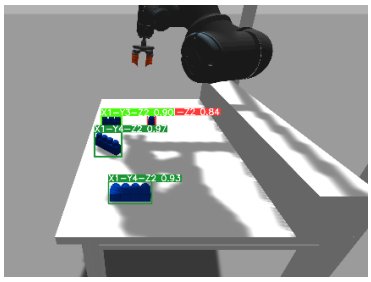

Figure 2: Blocks recognition

2.2 Objection localization

Object localization involves using a ZED Camera to capture depth information and generate a point cloud representation of the scene. Open3D's global registration algorithm is then used to align and determine the position and orientation of objects by extracting features, matching correspondences, and optimizing the alignment.

2.2.1 Object position

To determine the position of objects, a ZED Camera is used, this camera captures depth information and creates a three-dimensional representation of object points in the scene as a "point cloud". The ZED Camera utilizes stereo vision technology to calculate the depth of points in the image, it takes advantage of the disparity between the images captured by two lenses positioned at a known distance to estimate the distance of points from the camera. The z-coordinate represents the distance of each point from the camera along the optical axis. By utilizing knowledge of the camera's geometry and calibration parameters, the physical distance of objects from the camera can be obtained and can be converted into different reference frames.

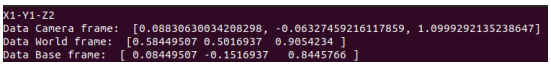

Figure 3: Estimated coordinates

2.2.2 Object orientation

Open3D's global registration algorithm is used to determine the relative position and orientation between two point clouds. The algorithm first extracts distinctive features from both point clouds. These features represent key points or regions in the point clouds that can be matched. Then, the algorithm compares the extracted features to find correspondences between the two point clouds. To improve the accuracy of the registration, point cloud filtering techniques are applied, so as to work only around the area of interest. The estimated transformation is refined using optimization algorithms. These algorithms iteratively adjust the transformation parameters to minimize the discrepancies between the matching features. The optimization process continues until convergence, resulting in an optimized transformation. By utilizing feature extraction, feature matching, point cloud filtering, and optimization, Open3D's global registration algorithm it aims to achieve a precise alignment between two point clouds

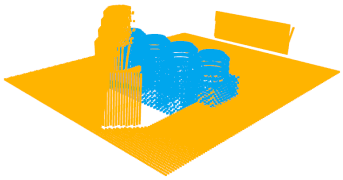

Figure 4: Point clouds before comparison

Figure 5: Point clouds after comparison

3 Kinematics

3.1 Context

A manipulator can be schematically represented from a mechanical viewpoint as a kinematic chain of rigid bodies (links) connected by means of revolute or prismatic joints, one end of the chain is constrained to a base, while an end-effector (EF) is mounted to the other end, in our case the UR5e robot has 6DoF (revolute joints) and a spherical wrist. The resulting motion of the structure is obtained by composition of the elementary motions of each link with respect to the previous one. Therefore, in order to manipulate an object in space, it is necessary to define the direct and inverse kinematic equations which describe the end-effector position and orientation as a function of the joint variables and viceversa. This allows the end-effector position and orientation (pose) to be expressed as a function of the joint variables of the mechanical structure with respect to a reference frame and with the inverse kinematics the joint variables can be described as a function of the EF pose. In our project the basic task we need to accomplish through kinematics is the movement of the EF from a starting pose to the desired one with a linear movement expressed in the working space coordinates. Then these movements can be composed into bigger actions that move and manipulate blocks from one place to another. To reach these objectives we used the inverse differential kinematics.

3.2 Inverse Differential Kinematics

In order to control the robot movement in the working space it is needed to implement the Inverse Differential Kinematics (IDK) which gives the relationship between the end-effector linear and angular velocity and the corresponding joint velocities. In order to obtain the IDK equation we need to invert the direct differential kinematics (DDK) equation, defined as:

$$v_e = \begin{bmatrix} \dot{p_e} \\ \omega_e \end{bmatrix} = J(q)\dot{q}$$

with

$$J = \begin{bmatrix} J_P \\ J_O \end{bmatrix}$$

where v_e indicates the EF translational and rotational velocities, J represents the Jacobian matrix and $\dot{q}$ represents the joint velocities. In our case the UR5e has 6DoF and the jacobian matrix is square and full rank and can be simply inverted to obtain the IDK equation:

$$\dot{q} = J^{-1}(q)v_e$$

If the initial manipulator posture $q(0)$ is known, joint positions can be computed by integrating velocities over time, in our case we use a technique based on Euler integration method to integrate in discrete time. Given the joint positions known at time t_k the joint positions at time $t_{k+1} = t_k + \Delta t$ can be computed as

$$q(t_{k+1}) = q(t_k) + \dot{q}(t_k)\Delta t$$

To calculate the Jacobian inverse solution for a non reduntant manipulator the following equation was used:

$$\dot{q} = J^{-1}(q) \begin{bmatrix} K_P e_P \\ K_O e_O \end{bmatrix}$$

$$e_P = p_d - p_e(q)$$
$$e_O = \phi_d - \phi_e(q)$$

where K_P and K_O are positive defninite matrices, e_P and e_O represent respectively the poistion error and the orientation error expressed in Euler angles. An important issue that needs to be addressed is the occurance of singularities when using Euler angles representation, to deal with such problem we explored two solutions. The first one was to calculate the orientation error with quaternions but we had some issues with convergence. The second solution with the angle and axis method turned out to be the most effective one and consists in calculating the orientation error from the rotation matrix that represents the relative rotation from the actual end-effector orientation R_e to the desired orientation R_d as:

$$R_d^e = R_e^T R_d$$

from this we would then calculate the corresponding euclidean vector $e_o^e = \delta\hat{r}$ that would then need to be mapped into the world frame with R_d rotation.

3.3 Vector Field

During the operations of the robot it may happen a case called shoulder singularity where the wrist origin alignes on the world z-axis with the shoulder of the robot. This situation is quite troublesome even in its neighbourhood because small velocities in the operational space can result in large velocities in the link space, moreover at this singularity the IDK could have infinite solutions. Due to these issues it is necessary to avoid this area during the motion planning phase. One of the first ideas we had, was to define a fixed circle around the center of the base of the robot and calculate tangential trajectories to this circle, this solution worked fine when the movement was outside the circle but on the inside it was difficult to handle. A simpler approach we implemented includes a radial vector field (VF) centered in the origin of the base frame of the robot which repels away the end-effector from the singularity.

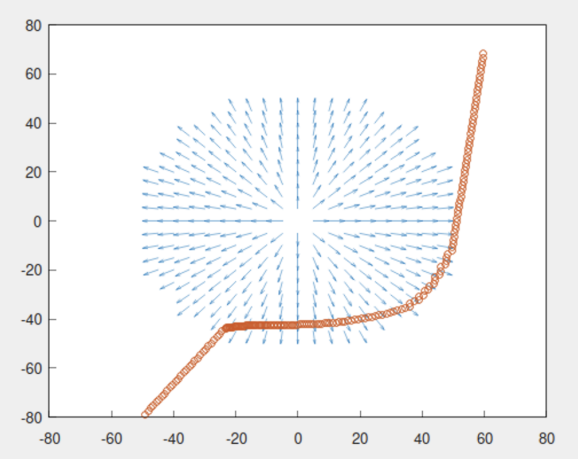

Figure 6: EF trajectory with radial vector field, for simplicity only the repulsive VF is visualized

The VF is implemented in the path planning and it computes the position error e_P of the EF. In the following formulas we intend as "x" the position in the space and not just along one axis.

$$\nabla U(x) = \nabla U_{att}(x) + \nabla U_{rep}(x)$$

In order to obtain the position error we calculate the gradient of the vector field.

$$e_P(x_0) = -\left(\frac{\delta U(x)}{\delta x}|_{x_0}\right)^T$$

e_P is determined by the composition of two VFs , U_{att} which attracts the EF of the robot to it's final destination x_{end} and a second VF U_{rep} which pushes the end-effector away from the shoulder singularity (center of the repulsive VF).

$$\nabla U_{rep} = \begin{cases} -k_{rep}\frac{d(x)}{\|d(x)\|}, & \text{if } \|d(x)\| < R \\ 0, & else \end{cases}$$

figurex$_0$*isthecurrentposition*

In ∇U_{rep} , K_{rep} is a factor which affects the intensity of the repulsive field, while R represents the maximum distance where the repulsive fields has effect.

$$\nabla U_{att} = \begin{cases} k_{att}\frac{x-x_{end}}{\|x-x_{end}\|}, & \|x - x_{end}\| > \rho \\ \mathrm{k_{att}}\left(\mathrm{x} - \mathrm{x_{end}}\right), & else \end{cases}$$

In the gradient of the attractive vector field k_{att} is used to improve convergence of the algorithm and affect the force that attracts the end effector to x_{end}. The parameter ρ represents the distance at which the attractive force of the vector field U_{att} is changed to help the convergence of the algorithm. The end effector stops its movements when its distance from the final position x_{end} is less then a predetermined threshold.

4 Task Planning

4.1 Object rotation

In order to set up an object correctly, we check the current block orientation (roll, pitch, yaw) and then perform appropriate actions to achieve the desired position. If both roll and pitch angles are close to zero or multiples of π, the block is considered already upright. Otherwise, we handle the case where the Lego is lying on the longer side ($|pitch| \approx \frac{\pi}{2}$) and when it is upside down ($|roll| \approx \pi \oplus |pitch| \approx \pi$). To optimize the code, we noticed that all the cases have a common step: the vertical position. From there, we perform the final rotations that bring the block to an upright position (roll=0, pitch=0, yaw=0). See the code for further details.

4.2 Task Making

Starting a task must be performed by running the main file and passing by argument the number of the task you are willing to play. For every task, initially we retrieve from the vision all the models found and their relative information (position and orientation), based on those data we approach the block and we bring it to his target position. All the translational movements from one place to another are made at a working height: 23 cm above the workbench.

5 KPI

- KPI 1.1 (time to detect the position of the object): **7**s
- KPI 1.2 (time to move the object between its initial and its final positions): **14**s
- KPI 2.1 (total time to move all the objects from their initial to their final positions): **34**s (3 objects)
- KPI 3.1 (total time to move all the objects from their initial to their final positions): **80**s (4 objects)
- KPI 4.1 (total time to move all the objects from their initial to their final positions): **86**s (4 objects)

6 Sources

- https://github.com/ultralytics/yolov5
- *http* : *//www.open3d.org/docs/release/tutorial/pipelines/global*$_r$*egistration.html*
- "Engineering Educator Academy" youtube channel
- Siciliano - 2009 - Robotics modelling, planning and control