Vision-Based Block Detection, Localization, and Assembly Using UR5 Robot Arm - Project Progress Report

Anh Tu Duong

July 29, 2023

1 Introduction

This progress report provides an overview of the work completed so far in the ”Vision-Based Block Detection, Localization, and Assembly Using UR5 Robot Arm” project. The project aims to achieve autonomous pick-and-place assembly of Lego blocks using a UR5 robot arm, with three main components: Vision for object detection and localization, Motion for robot arm manipulation, and the Planning component, which is yet to be developed.

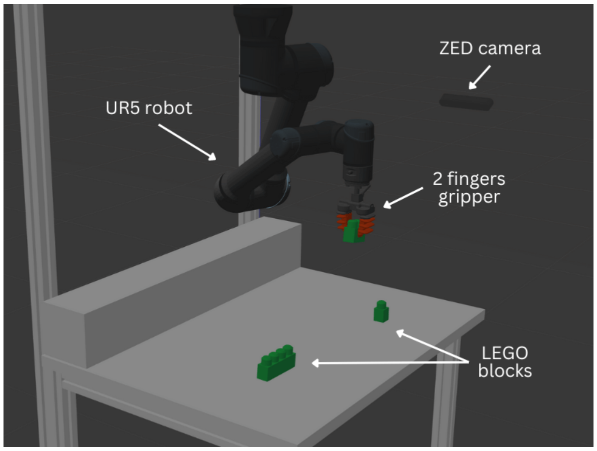


Figure 1: UR5 robot with ZED camera and Lego blocks on the table

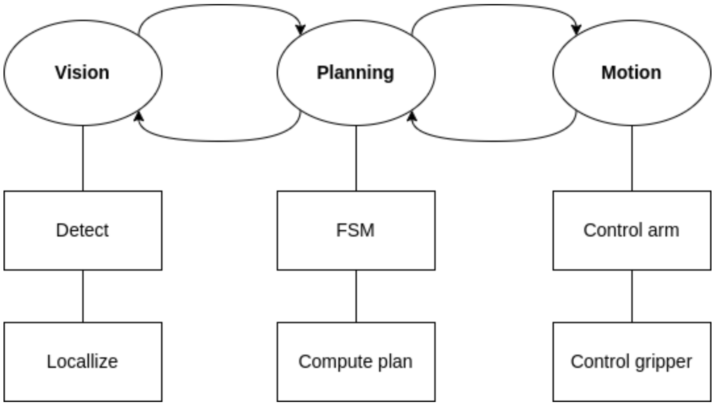


Figure 2: Main workflow

2 Vision

The Vision component serves as the perception module of the robot and has been successfully implemented.

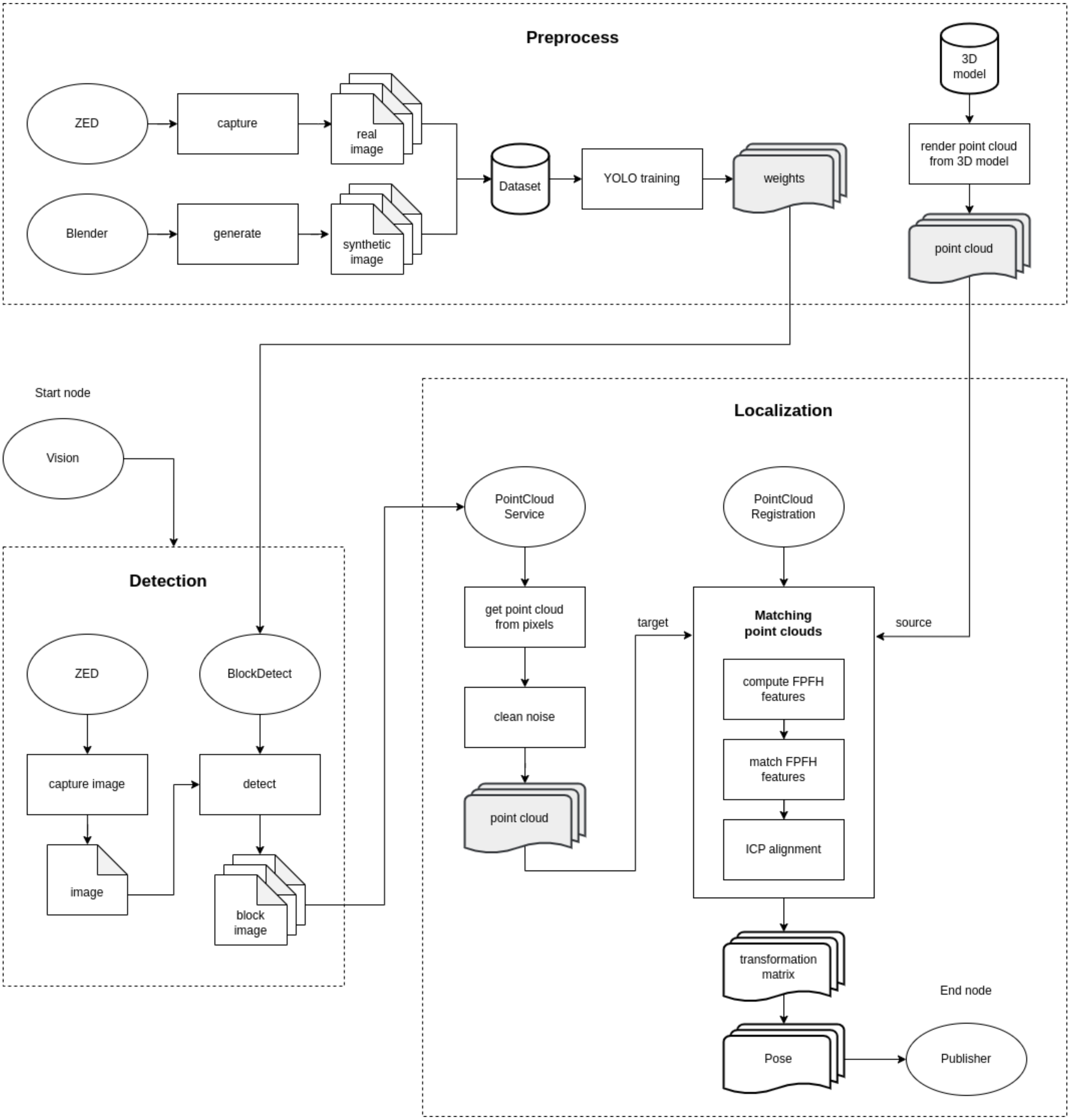


Figure 3: Vision workflow

The vision work starts with preprocessing the data: Dataset collection, YOLO training and Point cloud generation from 3D Lego model. The vision node runtime starts with the Detection and Localization of Lego blocks using point cloud registration technique. The result we want to achieve is Lego's translation and orientation, then send them to the Planning node.

2.1 Dataset collection

A dataset (about 2000 images) containing real images (10%) from the ZED camera and synthetic images (90%) generated from Blender with 3D models has been collected. I also put synthetic 3D models inside real background images to enrich dataset.

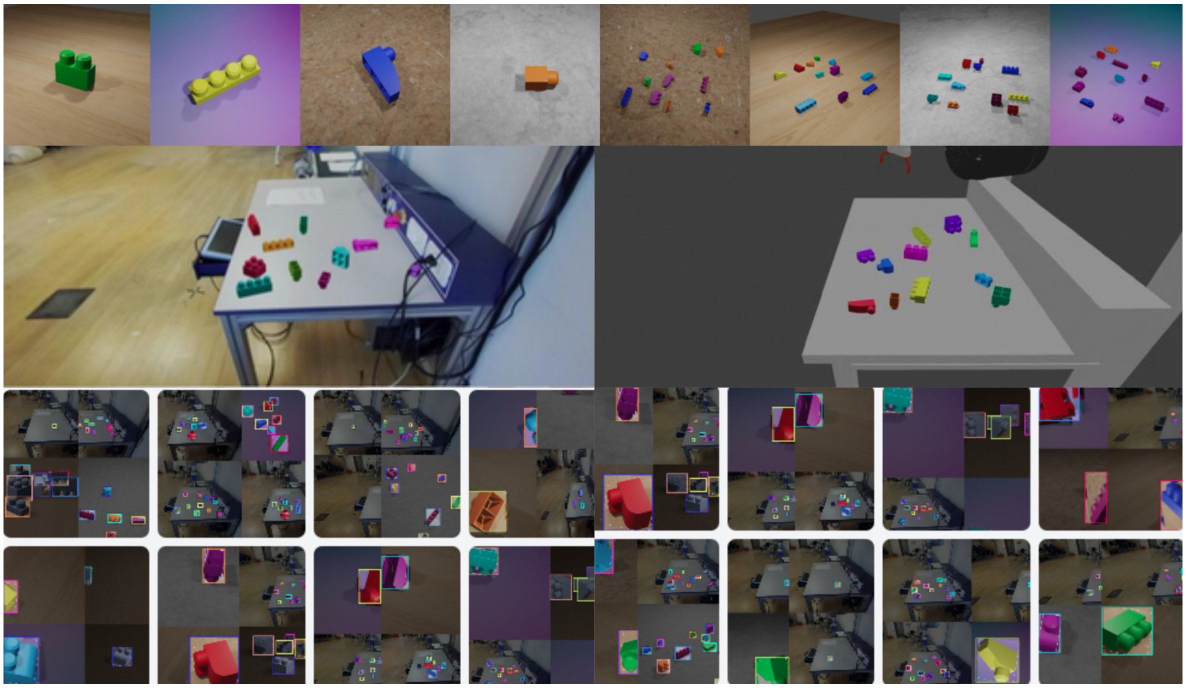


Figure 4: Dataset

The dataset is divided into:

- Training set (70%): 1.5k images
- Validation set (20%): 411 images
- Testing set (10%): 205 images

2.2 YOLO training

YOLOv5 was employed to train an object detection model. The model gets 95% accuracy. However some poses of the Lego blocks are very similar. For example, the only different between block X1-Y2-Z1 and X1-Y2-Z2 is their height. I'm still looking for a way to solve this issue.

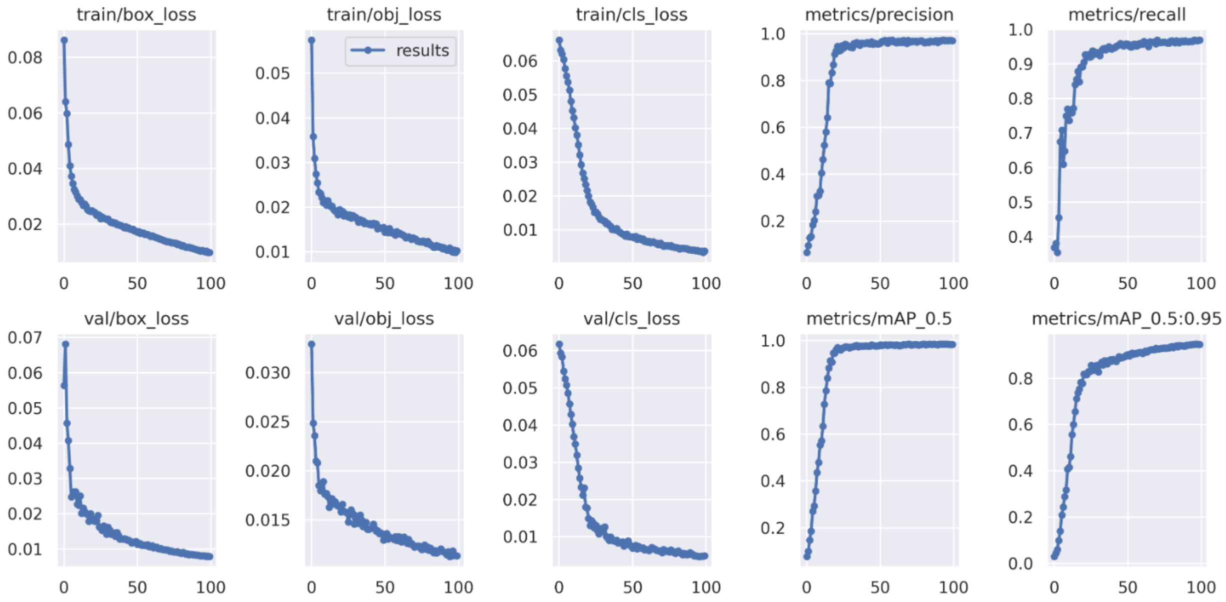
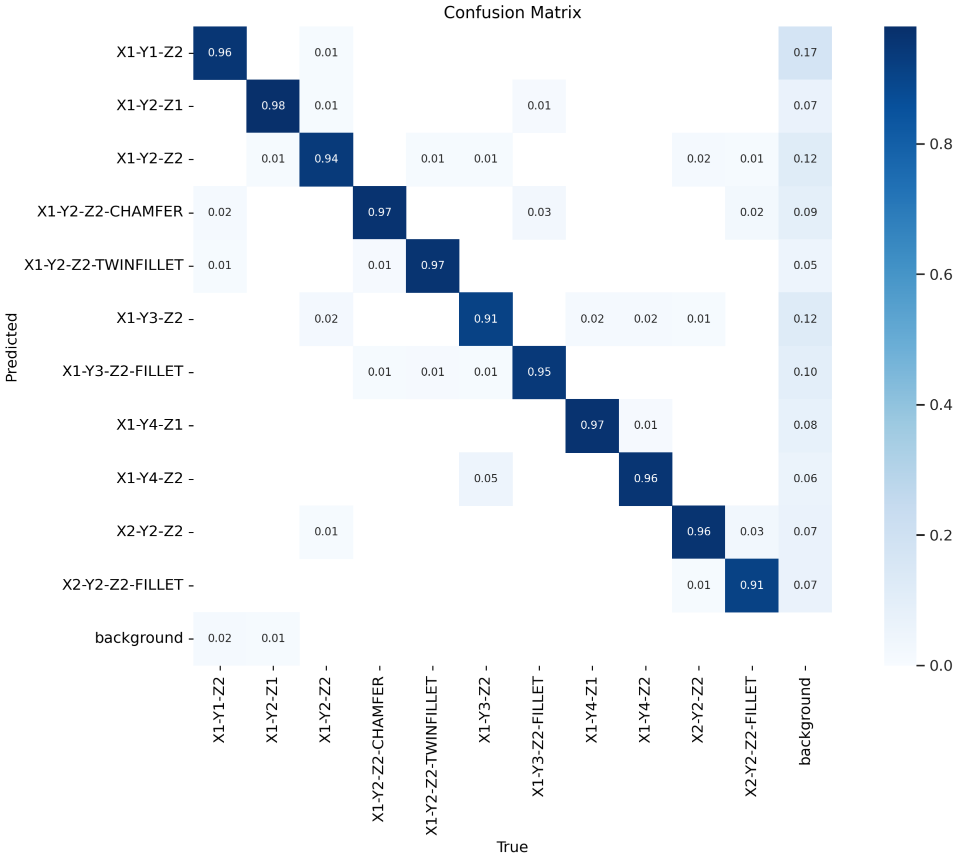


Figure 5: YOLOv5 train results



	X1-Y1-Z2	X1-Y2-Z1	X1-Y2-Z2	X1-Y2-Z2-CHAMFER	X1-Y2-Z2-TWINFILLET	X1-Y3-Z2	X1-Y3-Z2-FILLET	X1-Y4-Z1	X1-Y4-Z2	X2-Y2-Z2	X2-Y2-Z2-FILLET	background
X1-Y1-Z2	0.96		0.01									0.17
X1-Y2-Z1		0.98	0.01				0.01					0.07
X1-Y2-Z2		0.01	0.94		0.01	0.01				0.02	0.01	0.12
X1-Y2-Z2-CHAMFER	0.02			0.97			0.03				0.02	0.09
X1-Y2-Z2-TWINFILLET	0.01			0.01	0.97							0.05
X1-Y3-Z2			0.02			0.91		0.02	0.02	0.01		0.12
X1-Y3-Z2-FILLET				0.01	0.01	0.01	0.95					0.10
X1-Y4-Z1								0.97	0.01			0.08
X1-Y4-Z2						0.05			0.96			0.06
X2-Y2-Z2			0.01							0.96	0.03	0.07
X2-Y2-Z2-FILLET										0.01	0.91	0.07
background	0.02	0.01										

Figure 6: Confusion matrix

2.3 Point cloud generation from 3D Lego model

Point clouds were generated from the 3D models of the Lego blocks to represent their shapes. These point clouds will be matching with the point clouds of the Lego in the scene to localize it, which will be presented in Localization phase.

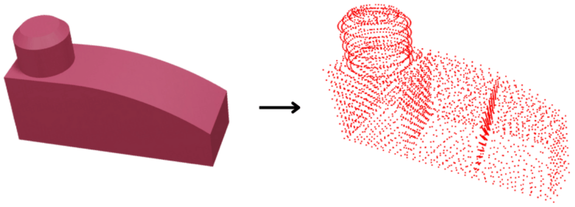


Figure 7: From 3D Lego model to point cloud

2.4 Detection

The Vision node starts with capturing images from the ZED camera and detecting Lego blocks using the trained YOLOv5 model. The bounding box information for detected objects is passed to the Localization process.

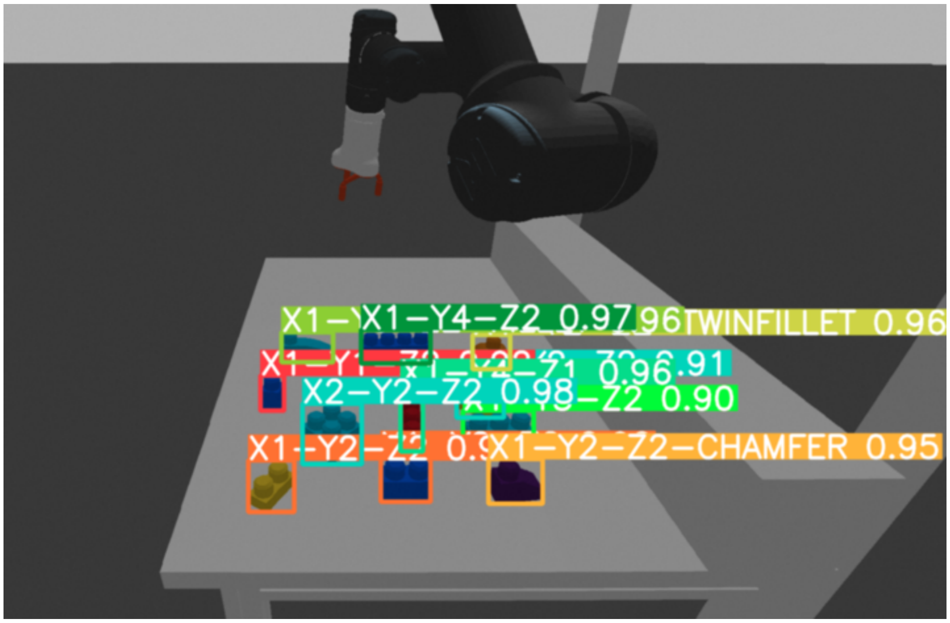
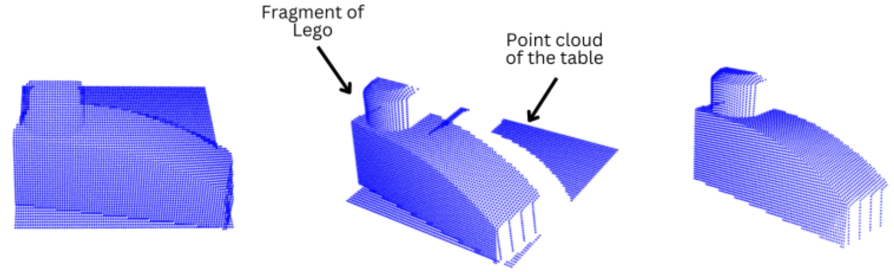


Figure 8: Object detection using YOLOv5

2.5 Localization

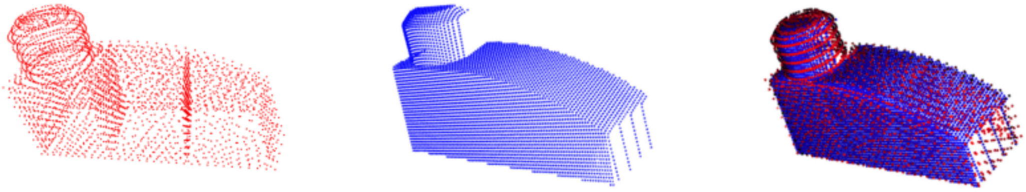
In the Localization process, the bounding boxes from the Vision component are used to extract point clouds of every pixels from the ZED camera data. After cleaning the point cloud belonging to the table, the rest represents a fragment of Lego blocks.



Point cloud of every pixels of the bounding box

Point cloud after cleaning the table

Point Cloud Registration is a fundamental problem in 3D computer vision and photogrammetry. Given several sets of points in different coordinate systems, the aim of registration is to find the transformation that best aligns all of them into a common coordinate system. Point Cloud Registration plays a significant role in this localization phase. Having point cloud rendered from the 3D model mesh (source point cloud) and the point cloud of an object detected from ZED camera (target point cloud), we need to perform some calculation to align the source point cloud to the target point cloud. The transformation matrix calculated after alignment is the pose of the Lego object in the scene.



Point cloud generated from 3D model (source)

Point cloud of detected object (target)

Align source with target point cloud

Figure 9: Point cloud registration

FPFH and ICP are 2 algorithms used to perform point cloud registration:

2.5.1 FPFH

FPFH (Fast Point Feature Histograms) is a feature descriptor commonly used in point cloud analysis and 3D computer vision tasks. It is an extension of the Point Feature Histograms (PFH) descriptor, designed to capture the geometric properties of 3D points. The FPFH descriptor calculates the local surface characteristics of a point in a point cloud by considering its neighbors. Here's a high-level overview of how FPFH features are computed:

- Select a point of interest (the query point) in the point cloud.
- Identify the k-nearest neighbors of the query point.
- For each neighbor, calculate the relative position with respect to the query point, forming a pairwise point-pair feature.
- Compute the difference in normal directions between the query point and each neighbor.
- Calculate a weighted histogram of the pairwise point-pair features, where the weights are based on the differences in normal directions.

- Concatenate the histograms from all the neighbors to obtain the final FPFH feature for the query point.

The FPFH descriptor takes into account not only the geometric positions of the neighboring points but also the differences in their surface normals. This combination allows for a more robust representation of local surface properties and provides a measure of distinctiveness between different points.

2.5.2 ICP

ICP stands for Iterative Closest Point. The ICP algorithm iteratively finds the best transformation that minimizes the distance between corresponding points in the two point clouds. The steps of ICP can be summarized as follows:

- Initialization: Define an initial transformation matrix that represents the rough alignment between the two point clouds. This initial transformation can be computed using some other method, or simply set to the identity matrix if no prior information is available.
- Correspondence: Find the corresponding points between the source point cloud and the target point cloud. One common method is to find the nearest neighbors in the target point cloud for each point in the source point cloud.
- Compute transformation: Compute the transformation that best aligns the corresponding points. One common method is to use Singular Value Decomposition (SVD) to compute the rotation and translation that minimize the sum of squared distances between corresponding points.
- Update transformation: Update the transformation matrix with the computed rotation and translation.
- Check convergence: Check if the convergence criterion is met. This can be done by comparing the change in the transformation matrix between iterations with a predefined threshold. If the convergence criterion is not met, go back to step 2 and repeat until convergence is reached.
- Output: The final transformation matrix gives the registration result. Apply this transformation to the source point cloud to align it with the target point cloud.

3 Motion

The Motion component is responsible for trajectory planning and robot arm manipulation.

3.1 Command Extraction

A high level API has been implemented to specify different commands, mainly used to control robot arm and gripper. These commands are taken from the Planning node.

3.2 Trajectory Planning

Has been developed with the help of the MoveIt package. MoveIt is a powerful motion planning framework in ROS that simplifies trajectory planning and robot arm manipulation. Using the target poses and orientations provided by the Planner's commands, the MoveIt component calculates feasible trajectories for the robot arm. These trajectories ensure safe and efficient pick-and-place movements, taking into account the robot's kinematics and workspace limitations. (To be studied and written in detail in the next weeks)

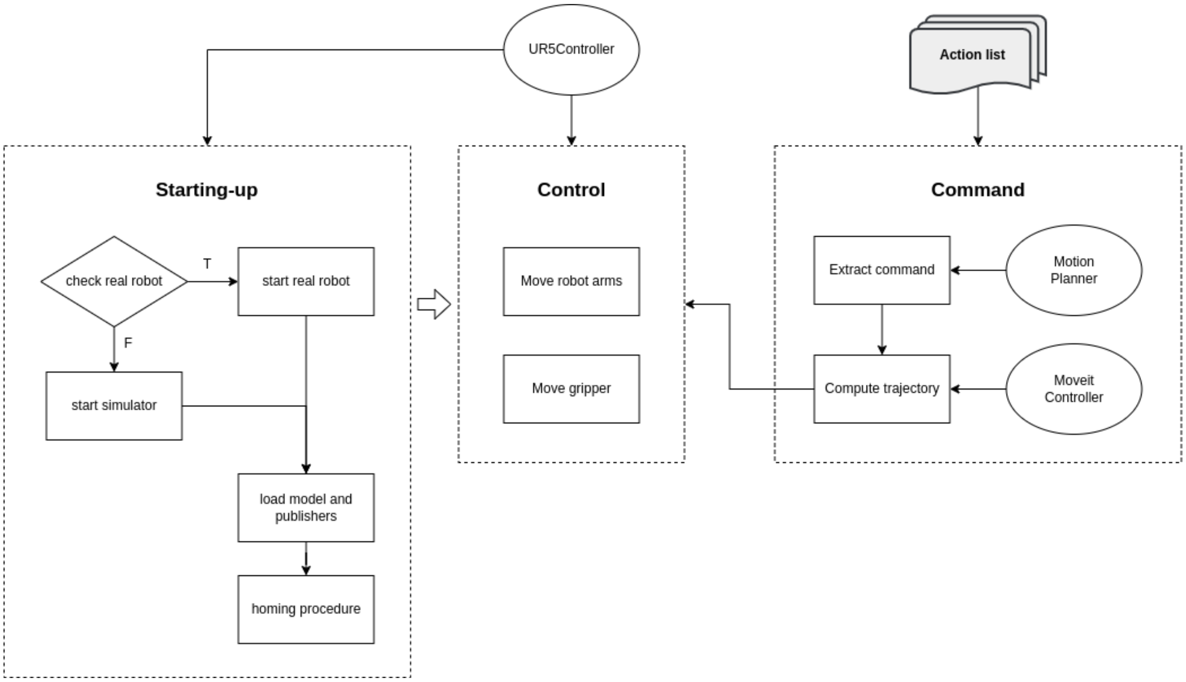


Figure 10: Motion node workflow

3.3 Robot Arm Manipulation

With the trajectories determined, the Motion node sends control commands to the UR5 robot arm, enabling it to navigate to specified locations, grasp Lego blocks and accurately place them at the desired positions to construct the predefined structure. This is currently developing in Gazebo simulation, which has a lot of bugs (for example, when placing a Lego block on top of the other, they will ”dance”, pop out of the table, or sometimes blend together). I have been discussing this with prof Focchi and we are planning to move to another simulation framework.

4 Planning: (Work in Progress)

The Planning component is yet to be developed, and this presents the primary challenge in the project.

4.1 Difficulty and Challenges

- The Planning component involves high-level coordination and task planning, integrating the information received from the Vision and Motion nodes. Designing an efficient planning algorithm that considers the availability of Lego blocks, sequence of pick-and-place operations, and error handling is a complex task.
- Error handling and recovery mechanisms need to be incorporated in the planning process to handle unexpected events, such as incomplete object detection or localization failures.

5 Next Steps

To complete the project, the following steps need to be taken:

- **Develop the Planning Component:** The Planning component needs to be designed and implemented to coordinate the communication between the Vision and Motion nodes, generate a high-level plan for the robot arm, and handle error scenarios.

- **Testing and Refinement:** Thorough testing of the entire system should be performed to identify any issues or inconsistencies. Refinements may be required in the YOLO training model, localization and motion phase for improved performance and reliability.
- **Documentation and Final Report:** Proper documentation of the code, algorithms, and the overall system should be completed. A comprehensive final report detailing the project's methodology, results, and conclusions should be prepared.

6 Conclusion

The "Vision-Based Block Detection, Localization, and Assembly Using UR5 Robot Arm" project has made significant progress in implementing the Vision and Motion components. However, the Planning component remains to be developed, which poses the primary challenge. Addressing the difficulties in planning and implementing robust error handling will be crucial to achieving a fully autonomous Lego block assembly system. With further development and refinement, the project has the potential to demonstrate the capabilities of vision-based robotics in real-world applications.

Note: This report provides an overview of the progress made so far. Detailed technical information and results will be provided in the final report upon completion of the Planning component.