École Centrale de Lyon

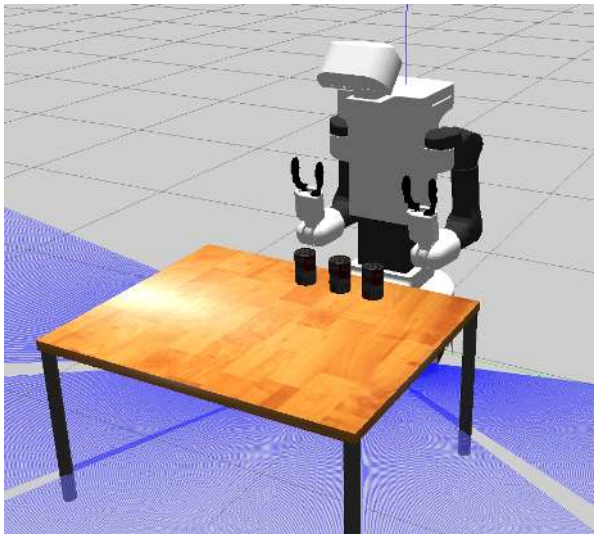
TFE 2024

Intelligent Autonomous Systems
Technische Universitaet Darmstadt
Darmstadt

Rapport Final de Travail de Fin d'études
Master thesis report

REFINING VISION-BASED 6D POSE ESTIMATION USING TACTILE SENSING

COSSERAT Esteban



Option: Informatique - *IT*
Filière: Vision et intelligence artificielle - *Vision and artificial intelligence*
Métier: Intrapreneur Entrepreneur

Tutors:		
Pedagogical:		CHEN Liming
head of the host organisation :		PETERS Jan
Enterprise:	IAS - TUD	HANSEL Kay
		SCHNEIDER Tim
	LIRIS - ECL	DURET Guillaume
		CHAPIN Alexandre

ÉCOLE CENTRALE LYON

membre de : UNIVERSITÉ DE LYON

FINAL YEAR INTERNSHIP

Academic year 2023-2024

This form should be inserted after the cover page of the FYI Report. It certifies that the Report has been validated by the firm and can be submitted as it stands to the Registrar's Office of Ecole Centrale de Lyon.

COMPANY VALIDATION OF FYI REPORT

Final Year Internship references

Student's name : COSSERAT Esteban

Report title : REFINING VISION-BASED 6D POSE ESTIMATION USING TACTILE SENSING

Firm : Intelligent Autonomous Systems of Technische Universitaet Darmstadt

Name of Company tutor : HANSEL Kay, SCHNEIDER Tim, PETERS Jan and DURET Guillaume

Name of School tutor : CHEN Liming

The company acknowledges having read the report mentioned above and authorizes its transmission to the Ecole Centrale de Lyon.
The company authorizes the distribution of the report on the catalog of the library of the Ecole Centrale de Lyon:

- Information on the TFE (title – author – keywords – summary...) will be public
- Access to the pdf report, upon authentication, will be limited to students and staff of the Ecole Centrale de Lyon.

Firm's representative :

Name :
Position :

Date:
Signature and stamp :

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisors Kay HANSEL, Tim SCHNEIDER, Guillaume DURET and Alexandre CHAPIN for orienting my research and providing weekly guidance throughout my work. Special thanks to Guillaume DURET for helping me find this internship by connecting me with the host laboratory.

I am also grateful to Professor Jan PETERS for leading the Intelligent Autonomous Systems laboratory and accepting me into their team during this work. Additionally, I would like to thank Professor Georgia CHALVATZAKI, who leads the Interactive Robot Perception & Learning (PEARL) laboratory, for opening their experimental room to me.

I extend my thanks to the administrative staff, Nanette SCHREIBER and Sabine SCHNITT, for helping me set up in my Darmstadt office and assisting with my administrative tasks.

I would also like to thank Maha ZAAFRANE of the international relations services at Centrale Lyon for guiding me through my scholarship application process.

Special thanks to Professor Liming CHEN for being my Centrale Lyon tutor during this internship.

Finally, I would like to thank all the jury members, Alexandre DANESCU, Liming CHEN, Jan PETERS and Kay HANSEL, for taking the time to consider and evaluate my work.

Résumé du rapport:

Ce rapport présente le travail de fin d'étude de cycle ingénieur au sein du laboratoire IAS pour l'amélioration de processus de saisie sur le robot Tiago de constructeur PAL robotics. Il s'agit de recherche axée sur l'amélioration de l'estimation de la pose d'objet par intégration de données tactiles lors de tâches de manipulation robotique. Les objectifs principaux étaient doubles : (1) développer un processus de manipulations robustes pour la préhension robotique, et (2) améliorer l'estimation de la pose d'objets en fusionnant les données tactiles et visuelles. Initialement, les efforts se sont concentrés sur la maîtrise des outils robotiques et la création d'une interface pour contrôler les actions robotiques, aboutissant à un programme documenté pour une gestion efficace du robot. Ensuite, une solution a été élaborée pour combiner les données d'entrée sur la pose de l'objet cible, conduisant à des estimations de pose plus précises. Les simulations utilisant l'algorithme Iterative Closest Point (ICP) ont démontré des améliorations significatives de l'estimation de la pose, validées par des expériences réussies en laboratoire sur un robot.

Mots-clés libres: Manipulation robotique ; Estimation de la pose des objets ; Données tactiles ; Tâches de préhension ; Simulation ; Validation en conditions réelles ; ROS (Robot Operating System) ; Docker ; Moveit ; Open3d ; Algorithme Iterative Closest Point (ICP)

Abstract:

This report presents the master thesis work carried out in the IAS laboratory on the grasp process improvement of the Tiago robot from PAL robotics. It is about research focused on enhancing vision-based pose estimation through tactile data integration during robotic manipulation tasks. The primary objectives were twofold: (1) to develop a robust manipulation pipeline for robotic grasping, and (2) to improve object pose estimation by combining tactile and visual data. Initially, efforts concentrated on mastering robot tools and creating a user interface for controlling robotic actions, resulting in a well-documented package for efficient robot management. Subsequently, a solution was designed to combine input data on the target object's pose, leading to more accurate pose estimations. Simulations employing the Iterative Closest Point (ICP) algorithm demonstrated significant improvements in pose estimation, which were further validated through successful real-world experiments on a laboratory robot.

Keywords: Robotic manipulation ; Object pose estimation ; Tactile data ; Grasping tasks ; Simulation ; Real-world validation ; ROS (Robot Operating System) ; Docker ; Moveit ; Open3d ; Iterative Closest Point (ICP) algorithm

Contents

List of Figures

Introduction

Estimating the position of an object in 6-degree-of-freedom space (3 translations and 3 rotations) is a crucial task for robots. A robot will have physical interactions with its environment. To define this environment, the robot will receive sensor data and interpret them to understand its surrounding scene. Pose estimation is the task that consists of using this incoming data to define the robot's environment. This part has to be accurate to allow the robot to interact with the defined environment.

Current advancements in computer vision enable good pose estimation from color images using deep learning models. This point has been raised by the BOP challenge 2022 report [24], which evaluates new solutions to the pose estimation problem.

However, these solutions heavily rely on high-quality image input. When part of the object is occluded, the efficiency of these methods significantly decreases. During the grasping process, occlusion of the object is often unavoidable, leading to the loss of accurate pose estimation.

The following work aims to address this issue by integrating tactile sensors on the gripper, thereby enhancing the pose estimation of the object during the picking process and enabling object tracking throughout the manipulation.

This master thesis report is structured into several chapters.

The first chapter provides the context of the research, including a presentation of the laboratory and an overview of the global project to understand the final goals of solution to develop.

The second chapter presents a state-of-the-art review of the project before its initiation, including the tools used, the capabilities of the robot, and the starting grasping pipeline (or process). Following this, the report details the work conducted along this project through the third and fourth chapters.

The third chapter focuses on the development of a grasping pipeline to manipulate tomato soup cans. This pipeline will be instrumental in developing the plugin tools that will improve pose estimation, as it will facilitate contact between the tactile sensor and the target object. This section will also discuss the MoveIt package, which enables various robot manipulations.

The fourth chapter addresses the second development phase dedicated to the improvement of pose estimation. It includes the problem statement, potential solutions, the selection of one solution, its development, an analysis of the results, and finally, experimentation in both simulation and real-world environments.

The report concludes with a summary and suggestions for future work.

1 Context

This chapter provides a view of the work's context. It starts globally by positioning the student's curriculum and then zooms in on the laboratory and the project to finally understand the mission's goal.

1.1 Master Thesis

This report is part of the generalist engineering courses at the French engineering school Ecole Centrale de Lyon, where a six-month internship is required during the final year. The internship took place at the Intelligent Autonomous Systems (IAS) laboratory [25] of the Technical University of Darmstadt (TUD) in the Computer Science department [10].

The internship in the research field is considered as a master thesis in Germany, following the Centrale Lyon courses with a specialization in informatics, specifically in computer vision and artificial intelligence. Furthermore, this specialization was chosen to be applied in the robotic field, making this internship in a robotic laboratory a continuation of previous courses.

1.2 The Laboratory

The IAS group at TUD is specialized in the implementation of AI for the robotic field. Led by Professor Jan Peters, it works on several projects aimed at improving robot control using machine learning to assist people in difficult tasks.

During the research, experiments took place at the PEARL lab [12], which specializes in the perception and intelligence of interactive robots.

1.3 The Project

The present work takes place in the Chiron project which involves collaboration among three laboratories, each specializing in distinct fields:

- The "Intelligent Robotics and Biomechatronics Laboratory" [11] at Nagoya University in Japan focuses on the mechatronics aspect of the robot.
- The "Intelligent Autonomous Systems" [25] group at the Technical University of Darmstadt in Germany specializes in robot control and teleoperation.
- The "Laboratoire d'InfoRmatique en Image et Systèmes d'information" (LIRIS) [5] at Ecole Centrale Lyon in France is dedicated to computer vision research.

Figure 1.1: This figure shows the mobile dual-arm manipulation robot Tiago from PAL Robotics[26]. The CHIRON project uses this robot to enable people with reduced mobility to perform difficult activities by manipulating it remotely.

Figure 1.2: This figure shows the target of the first step mission: grasping and manipulating objects is important to manage the introduction of a tactile sensor to the pipeline.

The Chiron project [2] aims to provide a robotic solution for individuals with mobility limitations. This solution is based on the Tiago dual-arm robot (Figure 1.1) from the robotic maker PAL Robotics [20]. The robot will be teleoperated (controlled remotely with a remote and virtual reality view) by individuals with mobility limitations to perform difficult tasks from home. One example is remote grocery shopping.

1.4 The Mission

In this context, the mission was to improve the current object pose estimation pipeline by adding tactile sensor modality.

This mission serves as a link between the work of LIRIS and IAS. It considers the computer vision work of LIRIS, which already estimates the object pose from a color image, and the manipulation work of IAS, which updates the grasping pipeline to consider the tactile sensor.

To quantify and discretize this mission, a first step was defined. This step, shown in Figure 1.2, involved building a can tower from several cans placed on a table in front of the robot. This first step aimed to understand how the robot works and how to make it move to grasp an object securely. The object grasping step is essential for the subsequent implementation of the tactile sensor, as it is necessary to put the sensor in contact with the target object.

2 Initial State

2.1 Pose Estimation Pipeline

The current grasping pose estimation pipeline for Tiago, including 6D pose estimation, is defined in Figure 2.2. The pipeline consists of three main model functions all trained on the YCB (Yale-CMU-Berkeley) dataset [28] visible figure 2.1.

A model function is a function defined by a machine learning model, so it has been trained on a dataset to reproduce a task. In our case all functions use the same dataset. The YCB dataset is very famous on the 6D pose estimation research field because it contains a clean stock of data for some daily items.

- **YOLOX** (You Only Look Once X)[7]: This model takes one color image **(1)** as input and returns all rectangles in the picture that contain the found object (also named the bounding boxes)**(2)**. As it has been trained on the YCB dataset, it is able to find all items of this dataset in the input image.
- **GDR-Net** (Geometry-Guided Direct Regression Network)[27]: This model takes a color image centered on a known object obtained by the previous step YOLOX **(3)** as input and returns its 6D pose, so the 6 coordinates needed to represent perfectly its position and orientation in the space.

  As the input has to be centered on the object, this model will apply to each object found by the first model and for each application it will crop the image with the YOLOX function output bounding box.

  The GDR-Net output can be represented by a coordinates frame **(4)**.

  We can notice that this model is, today, one of the most advanced for this task and well recognized as it emerged victorious in the BOP Challenge 2022 [24].
- **PointNetGPD** (Grasp Pose Detection)[17]: This model takes as input the object mesh **(5)** and its position given by the GDR-Net to provide several grasping configurations **(6)**. A grasping configuration is a pose that the robot gripper has to reach to grasp the target object efficiently.

2.2 Robot Structure

The Tiago dual-arm robot is managed using the ROS [22] (Robot Operating System) framework. PAL Robotics (the Tiago robot manufacturer) shared the ROS documentation and a tutorial to simulate and manage the robot.

Figure 2.1: This image represents all items content on the YCB (Yale-CMU-Berkeley) dataset[28] which the original pipeline of the project is based on.

2.2.1 ROS

ROS (Robot Operating System) is a widely used communication framework in the robotics field. It allows for the parallelization of multiple processes through normalized and secured communication between them. Furthermore, it is an open-source framework with a large community that develops numerous packages that can be easily integrated into any ROS system.

For a better understanding of how ROS works, it is based on nodes and topics. Nodes are programs that receive information, process it, and then publish the results. Topics are normalized messages published and received by nodes to allow them to communicate their respective inputs and outputs. ROS is particularly interesting in robotics because topics can be read by another system connected to the server.

For example, you can have a node that controls motors running on the robot embedded computer, and an image processing node running on a powerful computer connected to the server. These nodes can communicate through a topic, with the image processing node telling the motor control node where the robot should go.

The ROS packages are a group of nodes used to execute one task. For example, in this project, the (`Image_process_ws`) mentioned in Figure 2.2 is a package that will be executed in a Docker container. It will contain the node for processing the image through the YOLOX and GDR-Net functions, which will publish the pose estimation topic.

Finally, ROS offers great flexibility, as a node can easily be replaced by another one that respects the same topic inputs and outputs. This is especially true for simulating a robot. If you replace your motor controller node with a simulation node, as long as it publishes similar topics, the rest of the system will work seamlessly. This makes it easy to develop a process node by simulating the robot.

2.2.2 Gazebo and Rviz

Gazebo [6] is the most used simulation environment for ROS. It contains physical world laws to approximate real-life conditions as closely as possible. In the Tiago robot ROS tutorial shared by their designer, PAL Robotics, all packages can be executed on the Gazebo robot simulation.

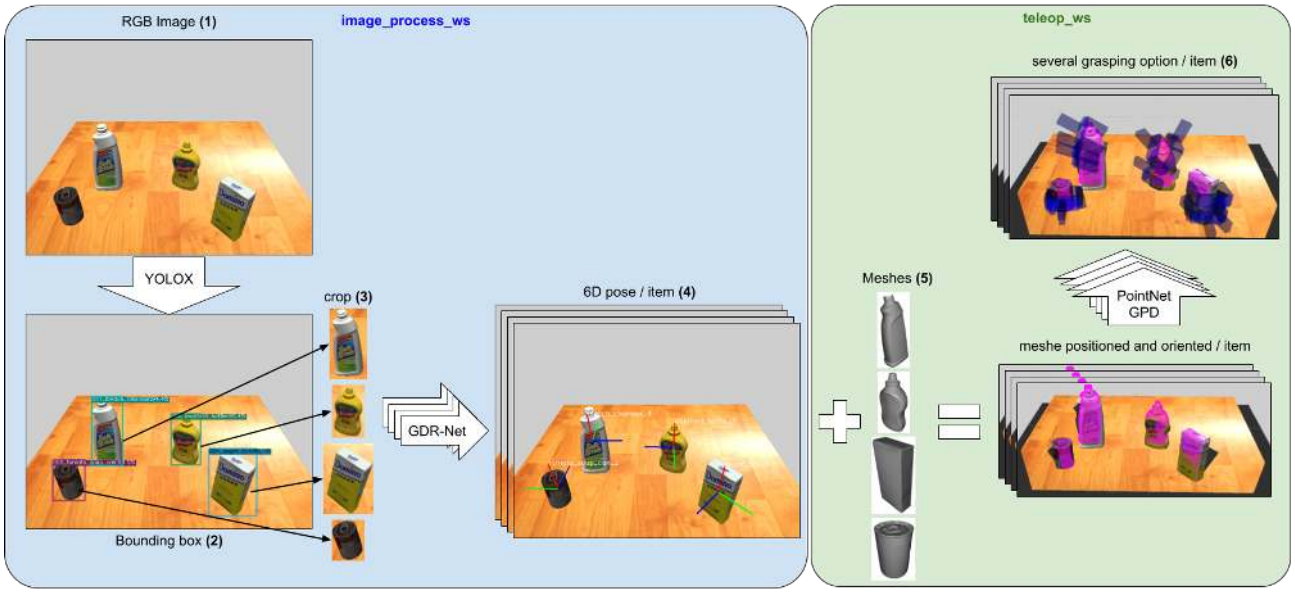


Figure 2.2: This graph presents the initial pipeline. There appear the 3 machine learning models used, their inputs and outputs. The two groups (`Image_process_ws` and `Teleop_ws`) represent the docker structure.

Therefore, anyone would be able to work on this robot without needing to have a real one.

Another very useful ROS package is Rviz [23], which is used to visualize topics. The topics can be published from a real system or a simulated one. For example, the simulated Tiago robot, shared by PAL Robotics, will reproduce the real robot topics such as motor feedback, camera images, and any other topic that the real robot publishes.

For some packages, Rviz can also be used to publish some topics and interact with the robot.

Figure 2.3 shows the Gazebo and Rviz interfaces. Gazebo is an environmental simulator, so it considers the properties of each element to apply physical laws to them. Rviz is only a topic viewer, so it generates a visualization without physical considerations.

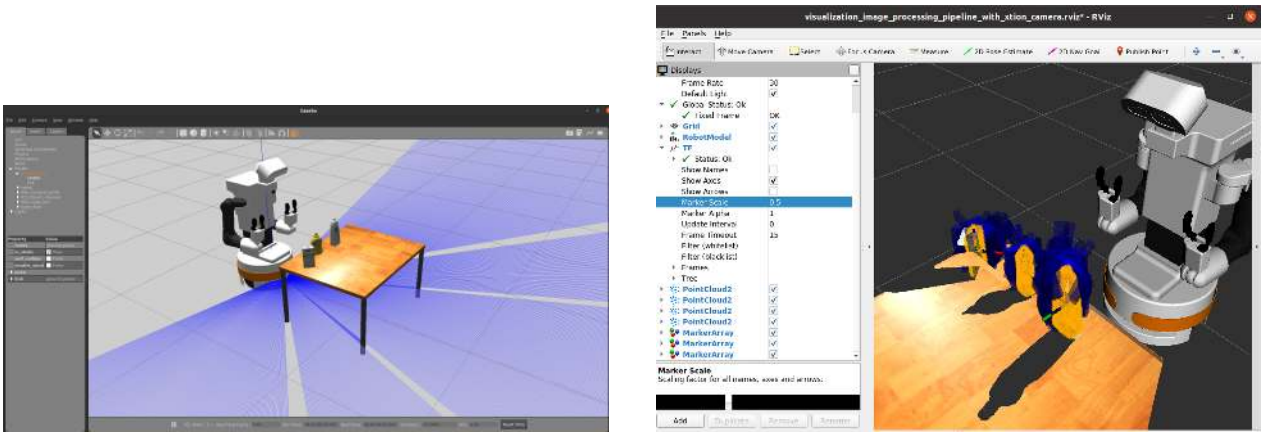


Figure 2.3: This figure shows the windows of Gazebo (left) and Rviz (right) of the Tiago dual-arm robot in front of three items from the YCB dataset placed on a table. The Rviz window (in blue) visualizes the grasping pose estimation for each item.

3 Grasping Process Setting

This chapter presents the first phase of the work, which focuses on designing a grasping pipeline to enable tactile sensor contact with the target item. This section details the development method and structure employed to achieve this goal, utilizing various tools presented throughout the chapter. The chapter concludes with a presentation of the grasping pipeline and the experimental results obtained.

3.1 Simulation

The first step in setting up the grasping process is to learn how to move the robot. To ensure the robot's safety during this learning phase, it will be simulated. The previous IAS works and the ROS Tiago tutorial were useful for setting up the robot simulation and provided some ideas and tools for the grasping process.

The simulation can be designed to include interactive objects. As the first goal is to stack tomato soup cans from the YCB dataset, three cans and a table have been added to the simulation environment.

3.2 Docker

An important point to note is that all processes will run under Docker containers. Docker is an OS-level virtualizer, meaning it virtualizes and sets up an environment on a machine. It permits easy transitioning from one machine to another, as the support environment is the same simulated one.

Moreover, it allows for multiple environments on the same machine. This is especially interesting with ROS, as different packages are developed by the community over time. Some of these packages use fixed environment versions that may not be compatible with others.

For example, in our case, the Tiago simulation is based on the ROS Noetic version, and the teleoperation process package (which generates a grasping configuration list from an object pose) is based on the ROS Melodic version. Since we cannot have two different versions of ROS in one environment, we must set one and simulate the other.

Nonetheless, ROS is designed to allow communication between environments and machines, so running nodes in different Docker virtual machines is not difficult.

Regarding Docker vocabulary, a Docker image is a template of a virtual machine. From this image, we can generate a container. An image is independent of any support machine, whereas

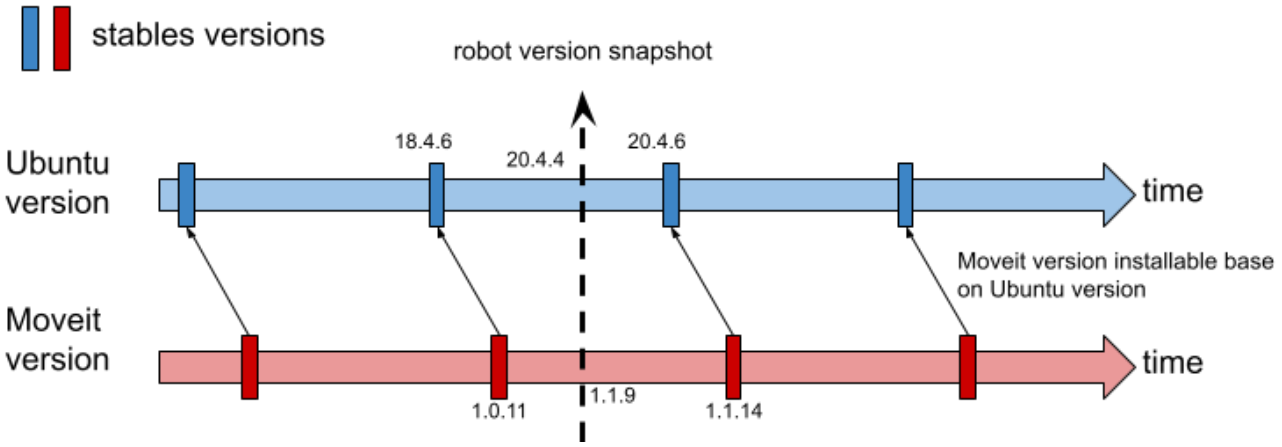


Figure 3.1: This schematic of Ubuntu and MoveIt versioning over time shows the versioning issue encountered between the real robot version and the stable one usually used to develop a MoveIt node.

the container is attached to some physical resources.

This point is important in robotics, as a robot will use actuators and sensors to communicate with the real world, and access to devices has to be given to our container if we want to use them through it.

Docker was helpful in addressing a significant issue that occurred during experiments on the real robot. The robot's embedded computer could not be updated and was running an unstable version of Ubuntu (as represented in Figure 3.1). To easily use the packages developed with the simulation on the robot, the best method was to use Docker images that matched the unstable robot configuration.

Some packages check the ROS topic signature to ensure compatibility between the publisher and the subscriber. The MoveIt package (which will be further developed) generated this issue on the real robot. To fix it, the solution was to find the MoveIt version of the real robot and install the same version in the Docker environment. This was challenging because the Moveit version was linked to the Ubuntu one, so the Docker images had to be based on a specific Ubuntu snapshot. Finally, the package version had to be used in the simulation to align with the real robot environment and facilitate a switch from to the other.

Initially, the project consisted of three Docker containers:

- **image_process_ws**: This container is based on Ubuntu 20.04 and uses ROS Noetic. Its specificity is that it requires the computational accelerator CUDA to process (as it uses machine learning models trained with CUDA).

- **teleop_ws**: This container is based on Ubuntu 18.04 and uses ROS Melodic. It receives

the pose estimation from the image processing container and publishes the list of grasping poses for each object found.

- **tiago_dual_sim_ws**: This container is based on the tutorial Docker image with a downgraded version of MoveIt to align with the real robot and uses ROS Noetic. It is used for simulation and topic visualization with Rviz.

Additionally, two Docker images we developed:

- **manipulation_ws**: This container was developed to use the MoveIt package. It is based on a specific Ubuntu version (20.04.4) and a specific ROS Noetic and MoveIt version to be compatible with the real robot (as presented in Figure 3.1).

  Another interesting point with Docker is the Docker Hub platform, where anyone can share their Docker images. The one designed to fix this issue has been published and documented [4] to be reused on other IAS projects.

- **gelsight_ws**: This container will be further developed in the next chapter. It is based on Ubuntu 20.04 with ROS Noetic. Its specificity is that it considers the tactile sensor device plugged in via USB to interact with it within the container.

To automate the launch of all these Docker containers, it is possible to use a Docker Compose file that describes the container parameters of each image for a group launch. With this setup, the entire pipeline can be launched with a single command.

3.3 Update the Pipeline

The initial code provided to find the grasping pose list from a color image considered only one instance of each object. Therefore, the first update was to permit multiple instances of each object.

The second update was to add a kind of object tracker in the image processing step. This easy tracker will register the item pose and attribute them a name. When a new set of items poses is received from the GDR-Net function, the tracker looks for possible poses association, based on the distance between registered and new items. The new items with an association will be used to update the registered item pose and the others will generate new items in the register.

After naming the items, the messages published in the topic will be sorted by name. This step is needed because of the grasp generator structure, which has several nodes running in parallel, and each of them will treat an item based on its place in the received message topic.

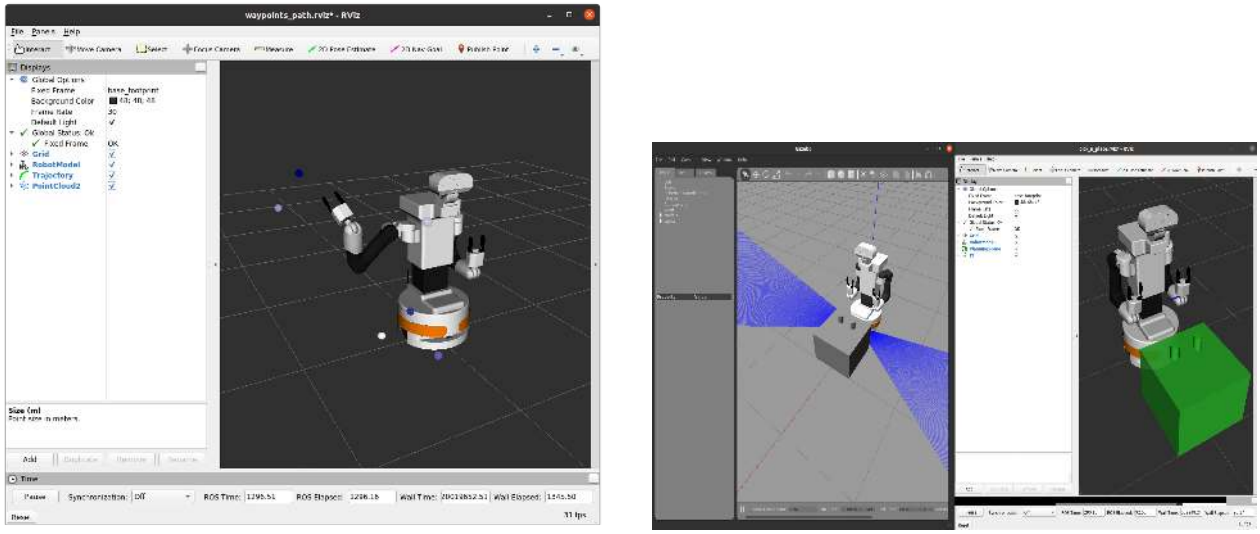


Figure 3.2: To document the MoveIt experiments, some example code was cleaned up to show what can be done with the MoveIt package. On the left, the code generates n points to reach (in blue intensity order) by the right gripper. On the right, three interactive geometries have been generated: one table to avoid items and two cylinders. The code will pick and place the left cylinder in front of the robot over the other one.

The tracking will ensure object pose continuity. It is not very robust but is enough to avoid rapid switching between items and the grasp generator.

3.4 ROS MoveIt Package

The ROS tutorial suggested using the MoveIt [18] package to control the robot. It is easy to implement in Python and offers many interesting functions. Despite the initial versioning issues, it was still beneficial to use this package.

Numerous tests and functionalities were experimented with this package to reach a pose from a specific point on the robot, prevent collisions, and grasp and place objects. As the MoveIt package was not widely used in the IAS laboratory, some example code and experiments were documented and shared. Notably, there is a code that generates random poses to reach and a code that generates interactive geometries for picking and placing, as shown in Figure 3.2.

The most interesting functionality of the MoveIt package is the ability to generate a collision environment with interactive objects and the path planning method that finds a path to reach a target frame while avoiding collisions.

3.5 Grasping Pipeline

With this package, a full pipeline was developed to grasp an object. To do this, the Python code interacts with the user to choose the next action to be performed by the robot. Figure 3.3

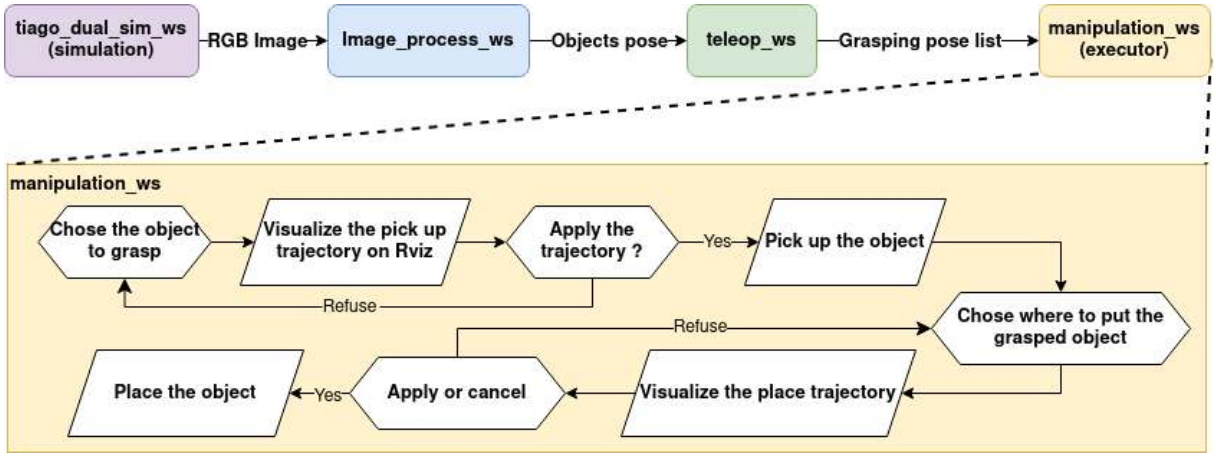


Figure 3.3: This graphic shows the grasping pipeline with the interaction to manipulate the robot. This interaction is possible through the command line user interface. The recognized items have a number used by the user to select which object to grasp. To place the object, two choices are possible: place the object back where it was picked up, or stack it on another can by selecting its referential number.

shows the new pipeline including the pick and place interaction.

Finally, with this grasping pipeline, the robot was able to grasp a can and stack it over another can or put it back to its original position. With this new pipeline, it was easier to build a soup can tower by stacking them rather than building a pyramid. As the final goal of this work is to improve pose estimation using tactile sensors and not robot manipulation, the build structure goal changed.

Moreover, the process of stacking cans is even more interesting than pyramid building for 6D pose estimation considering tactile information. During the grasping operation, the can may move slightly from the image pose estimation. As the gripper hides part of the can, it is particularly interesting to use tactile information to gain further details about the can's pose. Finally, the ability to stack cans will be a result of :

- the image based pose estimation of the supporting can
- the robot capacity in reaching a target point accurately
- and the pose estimation of the picked can, which is improved thanks to the tactile sensor.

3.6 Experiment

Following the successful simulation work, the next step was to test the setup on the real robot. This part was not so difficult as expected, since it mainly consisted in replacing the Gazebo simulation node with the real robot node. Consequently, there were minimal switching issues (except for the Docker issues addressed earlier).

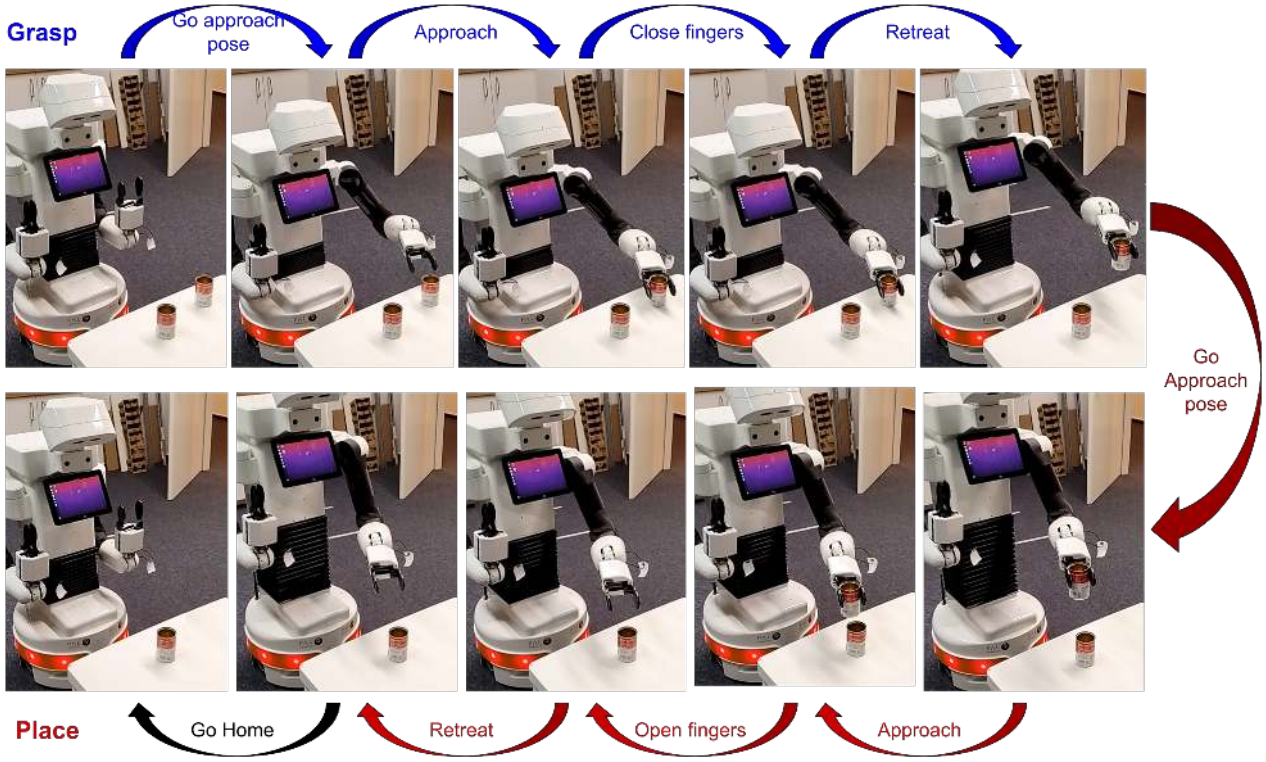


Figure 3.4: This figure show the experiments robot steps for stacking cans without the tactile sensor improvement.

Especially, the results were even better than in the simulation, as the laws of physics were difficult to model realistically. For example, the cans tended to slip in the simulation but not in the real experiment.

To ensure the safety of the robot, a small vertical offset, of a few centimeters, was set to avoid hitting the table during the picking action, and another offset was set for the placing action. This second offset can be seen in Figure 3.4 after the place approach action.

However, we see that the can is not perfectly aligned with the support can, so when the gripper opens its fingers, the can falls to the ground. This misalignment represents the pose error that we aim to reduce in the next chapter by adding a tactile sensor to the robot gripper.

4 Estimation Pose Improvement

This chapter presents the second part of the work, which is focused on the visuotactile improvement of pose estimation. It begins by presenting the material and problematic, followed by a simulation of the problem to be resolved. Subsequently, the research work is explained to reach a final solution, which is then analyzed and evaluated.

4.1 Tactile Sensor

Now that the gripper and the target object are in contact, the tactile sensor can be added to the robot and the pipeline.

The tactile sensor used is the Gelsight mini [8], as shown in Figure 4.1. This sensor uses a gel material, three lights, and a camera to estimate the depth image on the gel surface. The surface of the Gelsight is opaque, and when an object deforms it, some light is hidden in the surface due to the relief. Therefore, the camera sees a color image where the color depends on the surface relief. With a trained image processing model, the sensor converts the color image into a depth image.

Some related works exist using this sensor for pose estimation during manipulation tasks [13][16].
This sensor can be mounted on the robot with a specific finger design, as shown in Figure 4.2.

4.2 Problematic Options

To improve the pose estimation with a tactile sensor, two main options are possible:

1. Use a machine learning model that takes as input the image and tactile sensor point cloud output and gives the pose estimation of the object [3][21].

   This solution means not using the current visual pipeline for the grasping operation. The main reason for doing this is that during the grasping operation, when the gripper hides part of the object, the visual pose estimation accuracy will decrease. Therefore, a new model allows keeping a maximum of information from the image input during the grasping process.

2. Consider the two inputs separately and manage their weight in the pose estimation [1]. This solution allows keeping the current pipeline and just adding a process block to im-

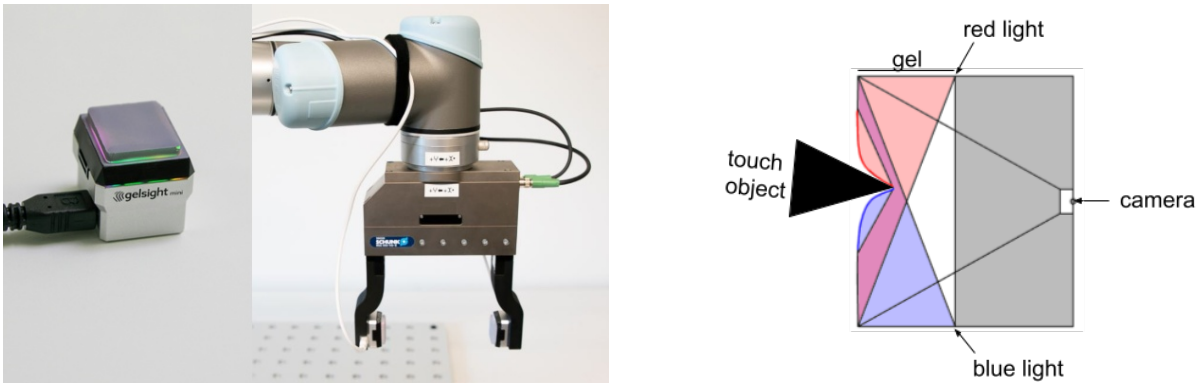


Figure 4.1: This figure shows, on the left, the GelSight-Mini tactile sensor alone and two of them mounted on a gripper. On the right, a schematic illustrates how this sensor works.

prove the pose estimation.

The second solution was chosen because the first one would be very time-consuming to set up and would require a large dataset.

4.3 Simulation With Open3D

This part presents the problematic representation in a 3D environment (treated by Open3D), the solution research to solve this problematic, and the solution analysis.

4.3.1 Setting Up

Currently, we have three input data to consider:

1. The visual pose estimation, which is a 6D pose of our object given by the current visual pipeline. It is the center of the can defined, on the robot references, by the 6 coordinates.
2. The proprioception of the tactile sensor, which is the pose estimation of the tactile sensor based on the kinematic chain of the robot joints. The Tiago robot is very accurate, so this estimation can be considered very close to the true pose.
3. The point cloud of the tactile sensor.

Two point clouds can be generated from these inputs: the tactile sensor one and the object one (based on the mesh of the object). In our work, we will only consider solid known objects, specifically the tomato soup cans from the YCB dataset.

To visualize these point clouds, the Python package Open3D [19] is useful. Besides visualization, this package contains some very interesting registration functions to align point clouds.

Figure 4.2: This figure shows, on the left, the robot finger design to add the Gelsight sensor to the Tiago robot. On the right, the 3D printing finger and the Gelsight sensor mounted.

The first step is to simulate our problem. For this, we will consider being in an environment with 16 variables, as shown in Figure 4.3.

In our case, every position is defined as a 6-coordinate transformation from another pose. We consider the base of the robot as the origin. The 6 coordinates are 3 translations $\mathbf{X} = (x, y, z)$ in millimeters and 3 rotations $\mathbf{\Theta} = (l, m, n)$ in radians. With these coordinates, we can generate a transformation matrix, which is a 4-dimensional square matrix. The ground truth transformation matrix of the can pose from the robot base will be an example of the transformation construction from the 6 coordinates.

$$
\begin{cases} \mathbf{X_{gt}} = (x_{gt}, y_{gt}, z_{gt}) \in \mathbb{R}^3 \\ \mathbf{\Theta_{gt}} = (l_{gt}, m_{gt}, n_{gt}) \in [0, 2\pi]^3 \end{cases}
$$

$$
\begin{aligned}
R_x(l_{gt}) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(l_{gt}) & -\sin(l_{gt}) \\ 0 & \sin(l_{gt}) & \cos(l_{gt}) \end{bmatrix} \\
R_y(m_{gt}) &= \begin{bmatrix} \cos(m_{gt}) & 0 & \sin(m_{gt}) \\ 0 & 1 & 0 \\ -\sin(m_{gt}) & 0 & \cos(m_{gt}) \end{bmatrix} \\
R_z(n_{gt}) &= \begin{bmatrix} \cos(n_{gt}) & -\sin(n_{gt}) & 0 \\ \sin(n_{gt}) & \cos(n_{gt}) & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{4.1}
$$

$$
R(\Theta_{gt}) = R_x(l_{gt}) \cdot R_y(m_{gt}) \cdot R_z(n_{gt})
$$

$$
T_{base \to ground\_truth} = T(\mathbf{X_{gt}}, \mathbf{\Theta_{gt}}) = \begin{bmatrix} & \mathbf{R}(\mathbf{\Theta_{gt}}) & & \mathbf{X_{gt}}^T \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

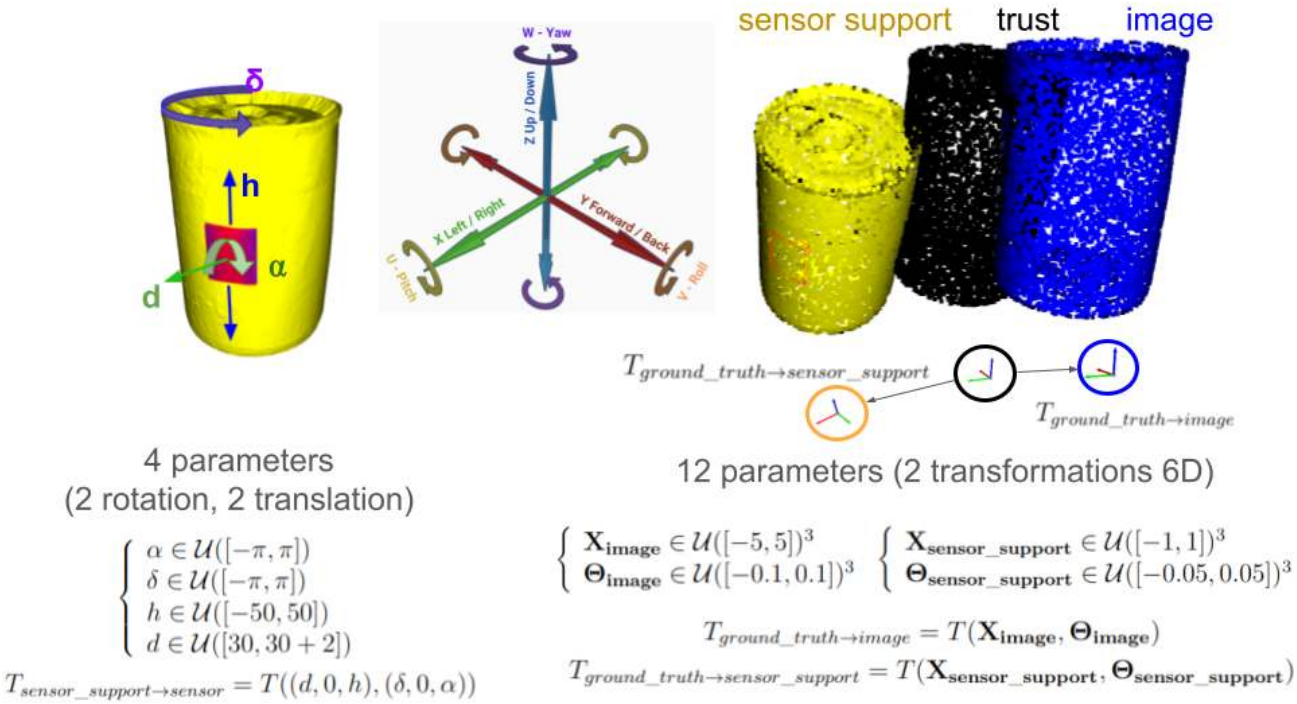


Figure 4.3: This figure shows a summary of our problem. We can see three cans representing the true pose of the can (black), the image pose estimation based on the current pipeline (blue), and the proprioception can estimation (yellow). The Gelsight point cloud will be on the yellow can surface. The sensor can be set anywhere on the lateral surface of this can, so we add four parameters linked to the Gelsight pose on this surface. The dimensions have been extended for better comprehension. In fact, the yellow and black cans are almost identical, and the blue one also has less error.

For the image pose estimation, the transformation is a chain of the proprioception between the robot base and the robot camera and the image pose process that estimates the object pose from the camera.

$$\begin{aligned} T_{base \to image} &\in \mathbb{R}^4 \times \mathbb{R}^4 \\ T_{base \to image} &= T_{base \to camera} \cdot T_{camera \to image} \end{aligned} \tag{4.2}$$

The sensor pose is a proprioception from the robot base to the robot finger.

$$T_{base \to sensor} \in \mathbb{R}^4 \times \mathbb{R}^4 \tag{4.3}$$

With these parameters, we can define our error problem. The image-based estimation pose error is a sum of proprioception and image process. As the kinematic chain between the base and the camera is not very long and the robot is supposed to be accurate, we can neglect the proprioception error. Furthermore, this error can be estimated with the Gazebo simulation as we can know the ground truth pose and the image pose estimation. For easy consideration, we will consider this error as respecting a uniform distribution ($\mathcal{U}$) error of a range of 5 centimeters

and 0.1 radians around the ground truth pose.

$$\begin{cases} \mathbf{X_{image}} \in \mathcal{U}([-5,5])^3 \\ \mathbf{\Theta_{image}} \in \mathcal{U}([-0.1,0.1])^3 \end{cases} \tag{4.4}$$
$$T_{ground\_truth \to image} = T(\mathbf{X_{image}}, \mathbf{\Theta_{image}})$$

If our kinematic chain was perfect, the sensor output should be on the true pose surface. As the sensor is at the end of a long kinematic chain, we will consider an error but smaller than the image estimation one. We do not have an error estimation by the robot manufacturer and finding it would require full experimental work. So we will just take the arbitrary values of 1 millimeter and 0.05 radians range error.

$$\begin{cases} \mathbf{X_{sensor\_support}} \in \mathcal{U}([-1,1])^3 \\ \mathbf{\Theta_{sensor\_support}} \in \mathcal{U}([-0.05,0.05])^3 \end{cases} \tag{4.5}$$
$$T_{ground\_truth \to sensor\_support} = T(\mathbf{X_{sensor\_support}}, \mathbf{\Theta_{sensor\_support}})$$

Now, to define our problem, we will start from the true pose, where we will add the image and proprioception error to find our inputs. In that way, we find the blue and yellow poses on the figure 4.3.

However, the tactile sensor does not give us the full yellow point cloud but only a section of this one. To simplify our problem, we will consider the can as a cylinder (radius 30 mm, height 100 mm) and the tactile sensor always normal and in contact with this cylinder. We define the sensor pose with 4 parameters on this yellow can: α which is the orientation of the sensor around the can normal, δ which is the orientation of the normal around the can revolution axis, h which is the height of the sensor along the can surface (can of a 100 mm height) and d which is the depth of the sensor through the normal of the can, defining the surface contact (can of 30 mm radius).

This gives us the sensor pose from the can sensor support:

$$\begin{cases} \alpha \in \mathcal{U}([-\pi,\pi]) \\ \delta \in \mathcal{U}([-\pi,\pi]) \\ h \in \mathcal{U}([-50,50]) \\ d \in \mathcal{U}([30,30+2]) \end{cases} \tag{4.6}$$
$$T_{sensor\_support \to sensor} = T((d,0,h),(\delta,0,\alpha))$$

Now, we have to find the true pose with the image pose estimation (blue point cloud) and the tactile output (a cropped part of the yellow point cloud).

There is still a step to have a good approximation of our problem, which is the tactile sensor output point cloud. To have this good simulation, we use the Open3D raytracing function. The tactile sensor output is supposed to be a depth image, so with uniform pixels on the XY plane and a depth along the Z axis, but from the mesh point cloud charged, we do not have enough points in the tactile sensor part (red box) to have a good simulation of the sensor.

To resolve this issue, we use the meshes surface of the can and a group of rays to generate a point cloud similar to the Gelsight output. The result is visible in Figure 4.3.

4.3.2 ICP

Now that we have a simulation of our inputs, we search for a point cloud alignment solution to estimate the true pose from the two inputs.

As mentioned above, Open3D has some algorithms to do this, and one of them is the ICP (Iterative Closest Point) algorithm [9], which gives the transformation matrix between two input point clouds.

The points of a point cloud are fixed together as a solid object, so it is possible to apply a transformation to a point cloud to move all its points. The ICP algorithm takes an origin point cloud and a target point cloud and looks to optimize the transformation of the origin to fit with the target.

The ICP algorithm works as follows:

1. **Initialization**: Start with an initial estimate of the transformation matrix. This can be the identity matrix if no prior knowledge is available.
2. **Correspondence Search**: For each point in the origin point cloud, find the closest point in the target point cloud. This step establishes correspondences between the two point clouds.
3. **Transformation Estimation**: Compute the transformation that minimizes the distance between the corresponding points. This can be done using methods like the Singular Value Decomposition (SVD) or the Kabsch algorithm.
4. **Transformation Application**: Apply the computed transformation to the origin point cloud to align it more closely with the target point cloud.
5. **Iteration**: Repeat steps 2-4 until the transformation converges or a maximum number of iterations is reached. The algorithm iteratively refines the transformation to minimize the distance between the corresponding points.

There were several possible solutions for how to apply this function to improve our grasping process:

The first idea was to apply this registration between two tomato soup can point clouds and apply a rate of this transformation. That means estimating a can point cloud from only the tactile sensor point cloud. This does not work efficiently as we cannot find the inverse transformation of $T_{sensor\_support \to sensor}$ with only the sensor output.

The second idea was to try to grasp the object on an interesting zone. An interesting zone is a group of points that contain position information of the point cloud. For example, on a

cylinder point cloud, the top and bottom circle edges give more information about the position than the lateral surface. This interesting zone can be found with some algorithms, and as we work only on known objects, we can easily find them before touching the can. However, when we define the grasp target, we sum the errors $T_{trust \to sensor\_support}$ and $T_{trust \to image}$, so we are not sure to finally reach this zone. Furthermore, these zones are usually not able to be caught as they are supposed to contain important curves.

Finally, the last and chosen option was to apply the ICP method between the two input point clouds. That means, from the tactile sensor at the pose $T_{base \to sensor}$ as the origin and the can point cloud at the pose $T_{base \to image}$ as the target. This will look for the shortest transformation to apply to the sensor output to be stuck to the image estimation point cloud. As the sensor pose is more accurate, we will apply the inverse transformation to the image point cloud. In that way, the unknown parameters such as the height of the can or its revolution axis rotation are defined by the image point cloud, and the other coordinates are defined by the tactile sensor.

Finally, to have the visuotactile pose estimation, we are combining the image pose estimation transformation with the ICP one.

$$\begin{aligned} T_{ICP,sensor \to image} &\in \mathbb{R}^4 \times \mathbb{R}^4 \\ T_{visuotactile,base \to sensor} &= T_{base \to image} \cdot T^{-1}_{ICP,sensor \to image} \end{aligned} \tag{4.7}$$

4.3.3 First Result

This solution was evaluated with the Open3D simulation. It performed N tests where the 16 parameters are randomly generated, and the results are given as a violin graph for each position coordinate in Figure 4.4.

We can see in this result a mean improvement and for some directions also reduce the variance. But we see that we still have some outliers resulting in a pose-estimation error. Some experiments led to the finding that when the sensor is too near an extremity of the can, it can consider the top or bottom surface of the can as the tactile point cloud, so it applies a 90° rotation of the image estimation.

Furthermore, we can observe an accuracy decrease along the z-axis on rotation and translation. As the tactile sensor output does not give any information on these two dimensions (due to the revolution shape of the can), the best possible new estimation pose is the image one. And even if the algorithm is supposed to stick the image pose on the sensor, so keeping the image error on these two dimensions, it still can add a little up shift or yaw rotation that decreases the result.

The yaw orientation is not very important in our task of stacking cans, but the up shift is caused by this error can imply can collision or over offset during the place manipulation.

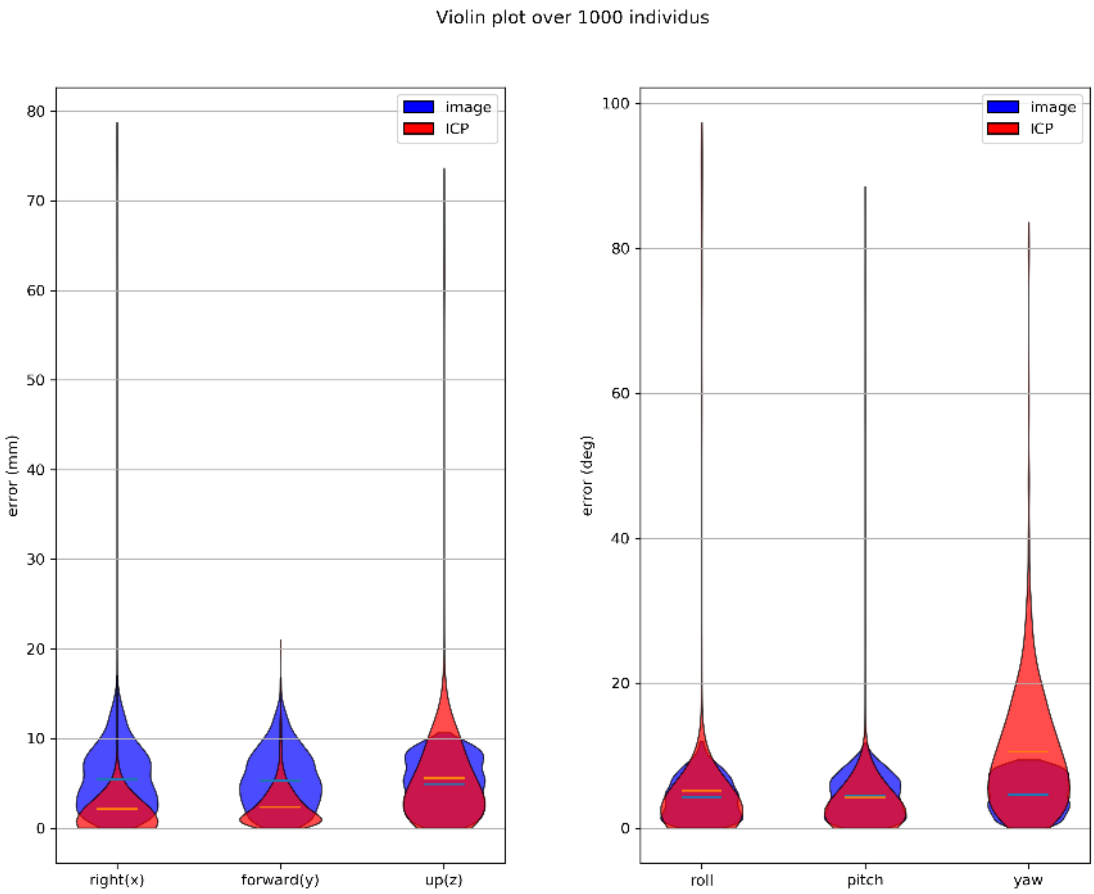


Figure 4.4: This violin graph shows the error distribution for the image (in blue) and the ICP (in red) estimation. It is the result of 1000 tests with random values of the 16 parameters.

4.3.4 Inertial Image Estimation

The first idea to solve this issue was to add a kind of inertia to the image pose estimation. In this way, the final pose estimation will not be only the image pose attached to the Gelsight sensor but will keep an offset in the image pose estimation direction.

To achieve this, the ICP output pose estimation will be used as a target for a new ICP operation with the image pose estimation as the origin. Finally, we apply only a part of the output transformation to the visuotactile pose to obtain our final lambda pose.

$$T_{visuotactile_\lambda, base \to sensor_\lambda} = T_{visuotactile, base \to sensor} \cdot \lambda T_{ICP_\lambda, sensor \to image} \tag{4.8}$$

To find a good lambda for this problem, statistical experiments were conducted. These experiments consisted of generating 50 vectors of random values for our 16 problem parameters. With these vectors, we varied the λ value from 0 to 1 and calculated the mean error over the 50 tests.

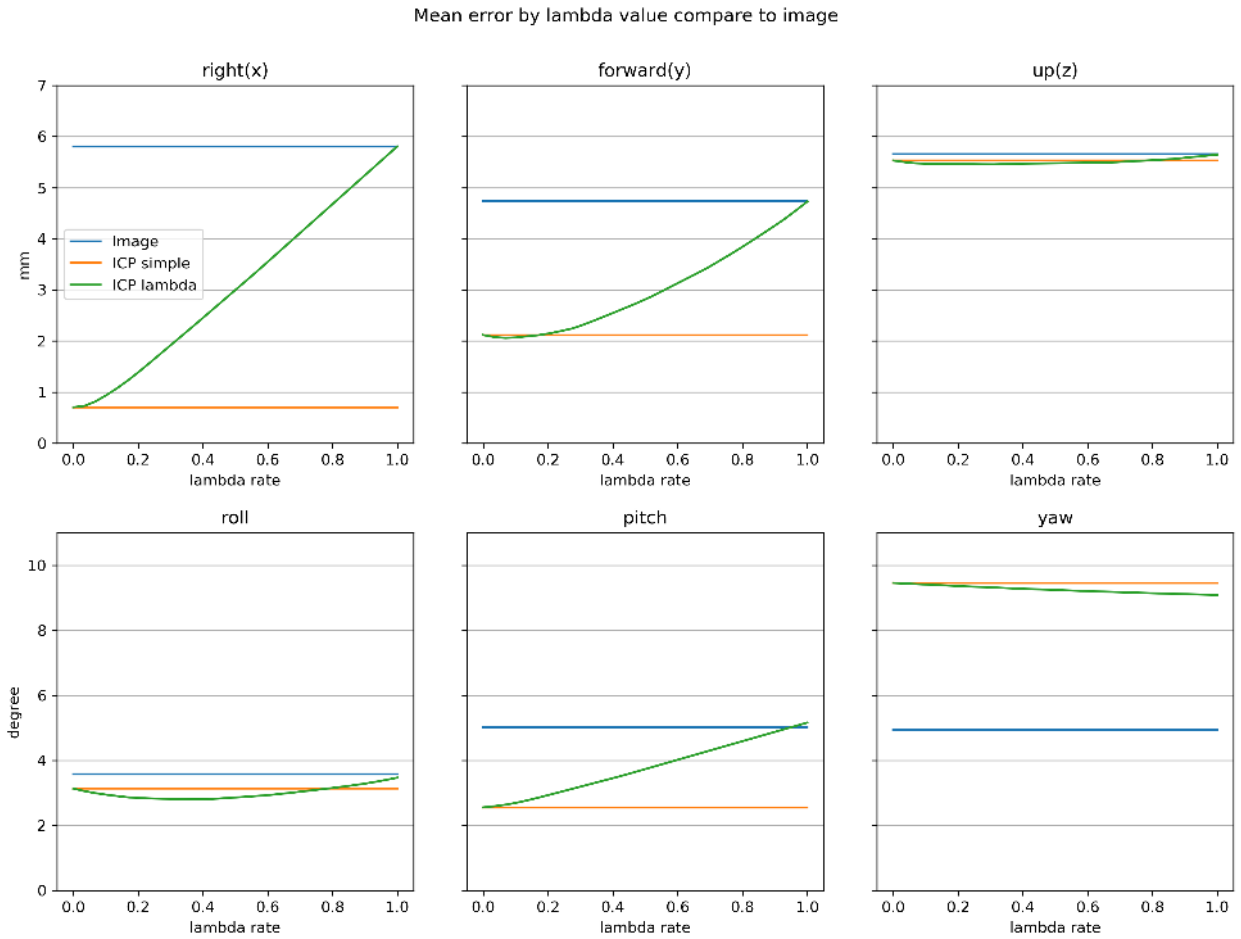


Figure 4.5: This graph plots the error of the 6 pose parameters through the value of λ. This value is the consideration rate of the ICP transformation obtained by applying the ICP function from the visuotactile position and the image pose estimation. For each value, it is based on a mean of 50 experiments.

The results are shown in Figure 4.5. We can see that as lambda increases, the error generally increases as well. Even though some minima appear for other values than $\lambda = 0$, the global error across the 6 coordinates does not look significantly better.

We also highlight the yaw plot where, even at lambda equal to one, the error is not the same as the image one. This is because the ICP transformation looks for the matching point solution, and the revolution shape of the can allows a good correspondence without a common yaw.

4.3.5 Grasp Configuration Tune

The second idea to fix the bad values around the can extremities was to add conditions to the grasping configuration pose. As we can choose the four parameters of the tactile position on the can, an analysis of the accuracy over these parameters was interesting. Figures 4.6 and 4.7 show the error variation over the height pose of the sensor and the orientation of the sensor.

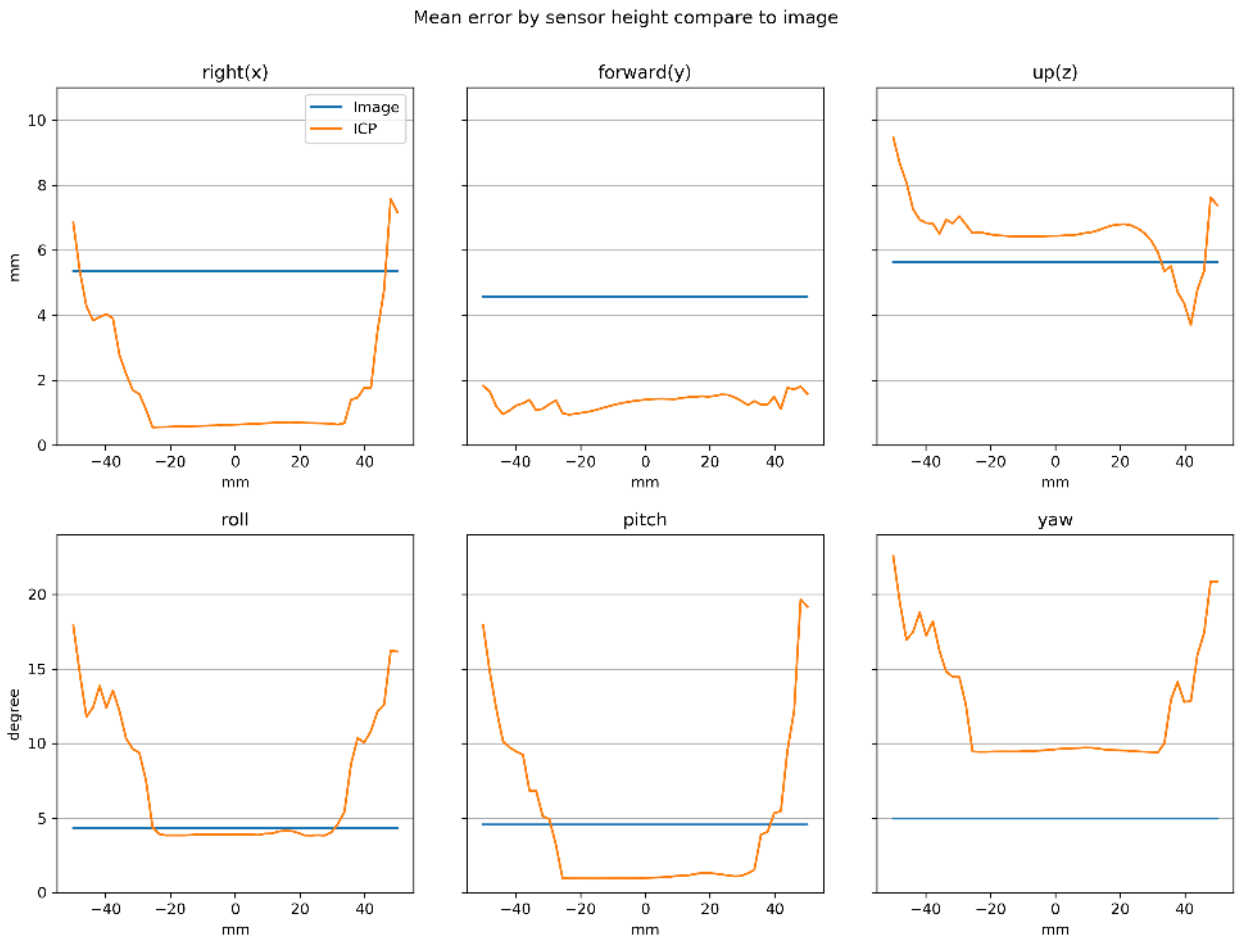


Figure 4.6: This graph plots the error of the 6 pose parameters through the height pose of the sensor over the can. For each position, it is based on a mean of 50 experiments. The 50 experiments are generated with 15 random parameters at the beginning, and their only variable parameter is the sensor height.

These graphs show a significant fall in accuracy near the extremities but not a very important effect of the tactile angle.

Following this result, a new analysis of the accuracy improvement with a filter of the tactile pose in a range of four centimeters around the middle of the can was conducted. The result graph in Figure 4.8 shows the final improvement possible through the ICP method.

We see that the error pose estimation decreases significantly compared to the image pose estimation (the original one), except for the z parameters. As mentioned earlier, these up shift and yaw rotation errors are, in the best case, equal to the image-based estimation error.

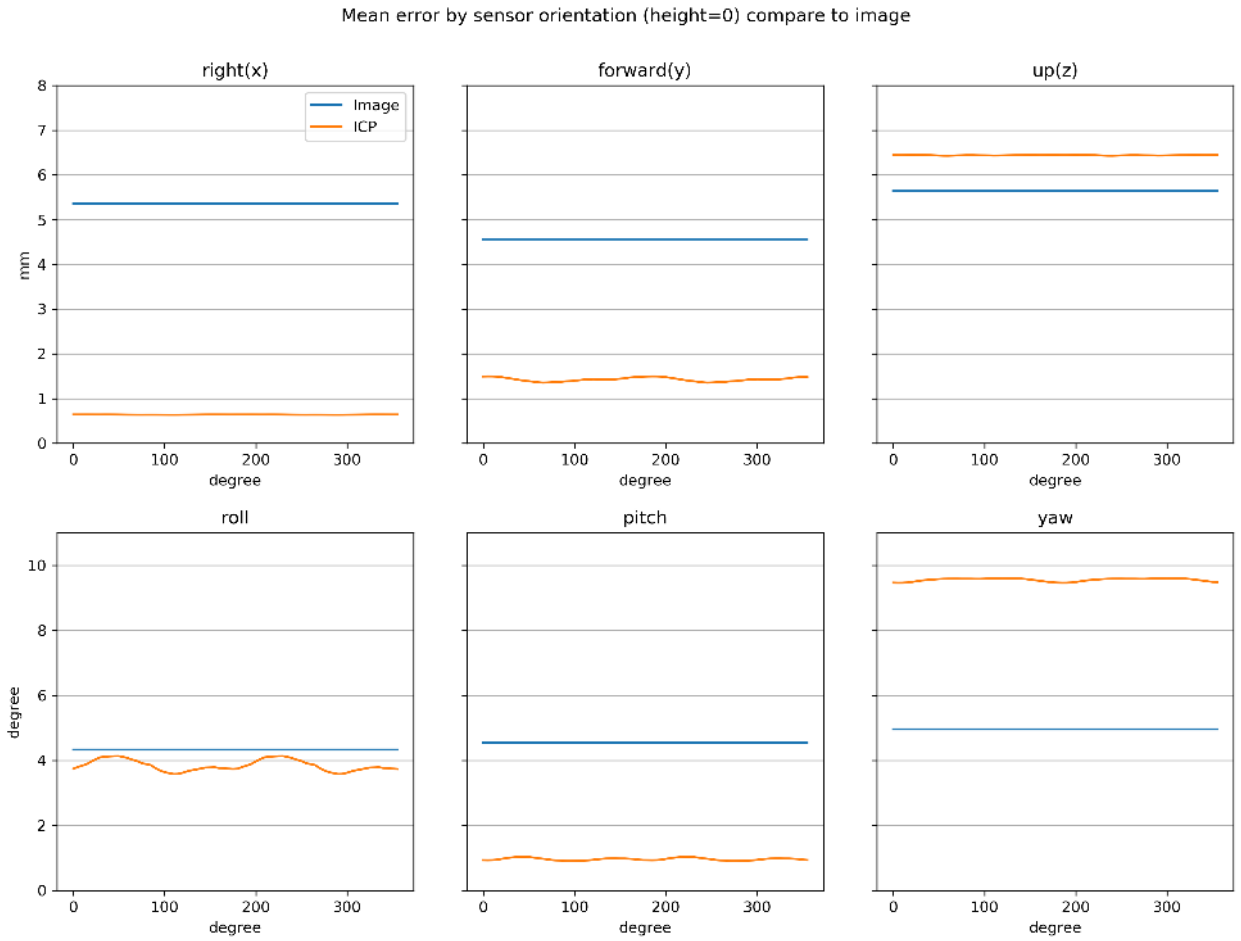


Figure 4.7: This graph plots the error of the 6 pose parameters through the orientation pose of the sensor. For each position, it is based on a mean of 50 experiments. The 50 experiments are generated with 14 random parameters (the sensor is always on the middle circle of the can) at the beginning, and their only variable parameter is the sensor orientation.

4.4 Pipeline Update

The ICP task now has to be added to the pipeline. The main challenge is to know which image pose estimation to use for applying the ICP with the tactile sensor point cloud, as when the tactile sensor touches the can, the visual pose estimation will be perturbed by the gripper.

As in the grasping pipeline, we already use some MoveIt virtual collision objects to manipulate the real object. We also use this virtual object for the ICP. These collision objects are updated by the user interface, so once the user likes the virtual object pose, they can proceed to the grasping operation, and the visual pose estimation error will not be read. So when the gripper has been closed on the can (we can estimate that from this point the can will not move anymore, and even if it moves, we could be able to track it through the tactile point cloud), we apply the ICP process and update the pose estimation. The figure 4.9 shows where the ICP is applied in the pipeline.

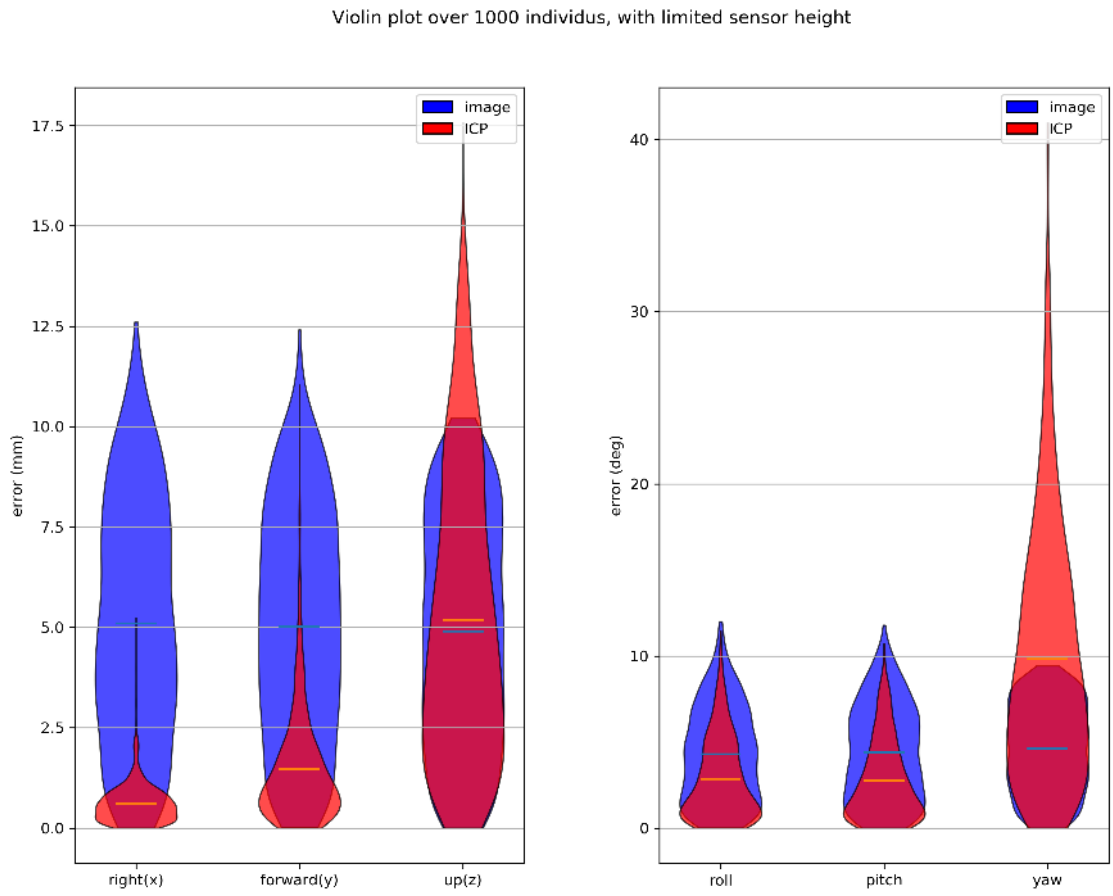


Figure 4.8: This violin graph shows the error distribution of the error pose over 1000 experiments with 16 random parameters where the sensor height is limited to a range of 4 cm around the center circle of the can.

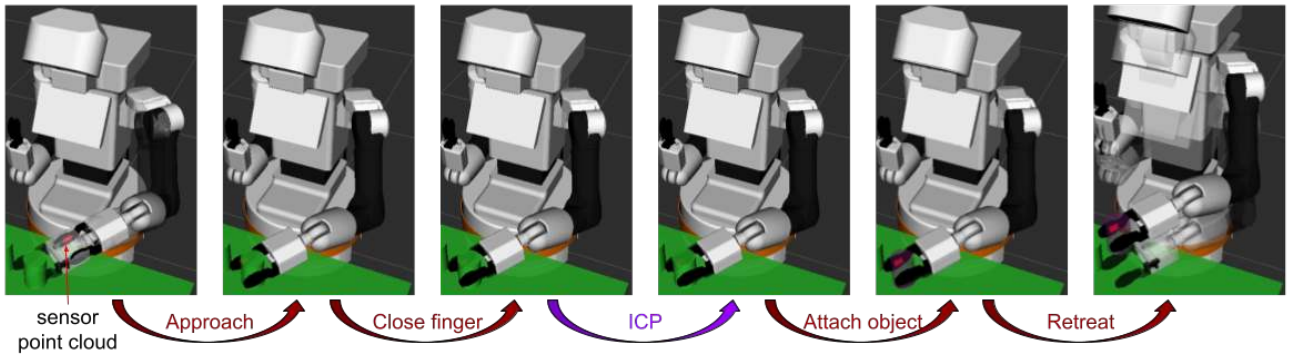


Figure 4.9: This figure shows the grasping pipeline update with the ICP operation.

5 Future Works

The current state of this research represents a good step and an easily reproducible method. However, some improvements can be made following this work.

The first follow-up work would be to extend the code to the full YCB dataset. This does not require a lot of work, as the starting pipeline already works with this full dataset. However, some important updates are still needed. For example, currently, the MoveIt collision objects are generated as cylinders, so to add new items, they would need to be changed to the corresponding mesh items.

Another good implementation is the consideration of the environment. During the development of the ROS manipulation workspace with MoveIt, some experiments have been conducted regarding the MoveIt collision octomap. This is a voxel map of collision objects, defining points to avoid in the space. This map could be very useful for defining the table, for example (which is currently defined by a box with a hardcoded pose). As the MoveIt package is already developed for the Tiago robot by PAL Robotics, this map just needs to be activated to work. A final interesting point about this octomap is that the MoveIt collision objects added to the environment clear the octomap around them. Therefore, the octomap collision points due to the target object surface are removed, and the object can be manipulated without collision consideration by the MoveIt path planner.

Regarding object tracking, currently, the grasped object is considered to be stuck to the gripper once it is grasped. If it moves, the MoveIt collision object (which describes the object pose estimation in the final pipeline) will not be updated and will have an offset with the true can. It would be very useful to add tracking on the Gelsight sensor to ensure the object does not slip.

The current pipelines use a snapshot of the visual pose estimation and the tactile point cloud output to estimate the final pose estimation. Adding a temporal meaning to these objects could be very effective in decreasing input errors. Furthermore, it would be interesting to keep a visual process during the manipulation to fix the issue of the z-axis errors. Some methods to consider several input data exist, such as the Kalman filter [15], which is easily used in SLAM (Simultaneous Localization and Mapping) algorithms [14].

An interesting experiment could also be to add a second tactile sensor on the other finger. In this way, the ICP function would be much more accurate.

A final possible improvement could be to conduct an optimization study to, for example, define a confidence rate for each degree of freedom parameter based on the input. For our case, with the tomato soup can, we would give more confidence to the image pose estimation about the z-axis (translation and rotation).

Conclusion

To summarize our work, we have, in the first phase, developed a process to grasp a tomato soup can using the Tiago robot arm and to place this can over another one. In the second phase, we simulated the can pose errors of this process to develop a visuotactile solution based on the ICP algorithm. Finally, this visuotactile solution has been implemented into the process to improve the pose estimation.

The first phase enabled us to become familiar with the robot tools and capabilities. It contained some difficulties, such as the MoveIt robot version issue, which was fixed through a Docker environment. However, it allowed for fluent and easy robot manipulation through a user interface, enabling us to experiment with the task of picking and stacking cans using the robot.

The second phase, developed using the Python package Open3D, included a problem-defining part and a solution research part. The problem-defining part permitted the simulation of the robot inputs and the position errors, which can be used later to improve the problem solution. The solution research concluded with an ICP-based solution that allowed for an improved pose estimation.

The problem had some interesting specific challenges, such as the cylindrical geometry of the manipulated item. This revolving geometry implied an inability to describe the sensor pose over the can surface, which generated an error improvement along and around the revolution axis of the item.
Furthermore, this solution is static and uses the image-based pose estimation and the tactile sensor output sequentially.
Some possible improvements to the developed pipelines are described in the future works chapter.

Finally, we achieved our goal as our final pipeline improved the image-based pose estimation by adding a tactile sensor to the process. It can be launched in simulation or on the robot and allows for the evaluation of the error by the result shift of the stacking operation.

This task will be helpful in the Chiron project for manipulating objects by teleoperation. It was also personally very interesting, as it allowed me to discover many tools and methods used in the robotics field, which will be useful in the future.

Bibliography Papers

[1] Tomoki Anzai and Kuniyuki Takahashi. "Deep Gated Multi-modal Learning: In-hand Object Pose Changes Estimation using Tactile and Image Data". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA: IEEE, Oct. 2020, pp. 9361–9368. ISBN: 978-1-72816-212-6. DOI: 10.1109/IROS45743.2020.9341799. URL: https://ieeexplore.ieee.org/document/9341799/ (visited on 05/23/2024).

[3] Snehal Dikhale et al. *VisuoTactile 6D Pose Estimation of an In-Hand Object Using Vision and Tactile Sensor Data*. Vol. PP. Journal Abbreviation: IEEE Robotics and Automation Letters Pages: 1 Publication Title: IEEE Robotics and Automation Letters. Jan. 2022. DOI: 10.1109/LRA.2022.3143289.

[7] Zheng Ge et al. "YOLOX: Exceeding YOLO Series in 2021". In: arXiv:2107.08430 [cs]. arXiv, Aug. 2021. DOI: 10.48550/arXiv.2107.08430. URL: http://arxiv.org/abs/2107.08430 (visited on 09/07/2024).

[13] Gregory Izatt et al. *Tracking objects with point clouds from vision and touch*. Pages: 4007. May 1, 2017. 4000 pp. DOI: 10.1109/ICRA.2017.7989460.

[14] Alif Khairuddin, Shukor Talib, and Habibollah Haron. *Review on simultaneous localization and mapping (SLAM)*. Pages: 90. Nov. 1, 2015. 85 pp. DOI: 10.1109/ICCSCE.2015.7482163.

[15] Qiang Li et al. *Kalman Filter and Its Application*. Pages: 77. Nov. 1, 2015. 74 pp. DOI: 10.1109/ICINIS.2015.35.

[16] Rui Li et al. "Localization and Manipulation of Small Parts Using GelSight Tactile Sensing". In: *IEEE International Conference on Intelligent Robots and Systems* (June 30, 2014). DOI: 10.1109/IROS.2014.6943123.

[17] Hongzhuo Liang et al. "PointNetGPD: Detecting Grasp Configurations from Point Sets". In: *2019 International Conference on Robotics and Automation (ICRA)*. arXiv:1809.06267 [cs]. May 2019, pp. 3629–3635. DOI: 10.1109/ICRA.2019.8794435. URL: http://arxiv.org/abs/1809.06267 (visited on 09/07/2024).

[21] Alireza Rezazadeh et al. "Hierarchical Graph Neural Networks for Proprioceptive 6D Pose Estimation of In-hand Objects". In: arXiv:2306.15858 [cs]. arXiv, June 2023. DOI: 10.48550/arXiv.2306.15858. URL: http://arxiv.org/abs/2306.15858 (visited on 05/23/2024).

[24] Martin Sundermeyer et al. "BOP Challenge 2022 on Detection, Segmentation and Pose Estimation of Specific Rigid Objects". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. ISSN: 2160-7516. June 2023, pp. 2785–2794. DOI: 10.1109/CVPRW59228.2023.00279. URL: https://ieeexplore.ieee.org/document/10208680 (visited on 08/21/2024).

[27] Gu Wang et al. "GDR-Net: Geometry-Guided Direct Regression Network for Monocular 6D Object Pose Estimation". In: arXiv:2102.12145 [cs]. arXiv, Mar. 2021. DOI: 10.48550/arXiv.2102.12145. URL: http://arxiv.org/abs/2102.12145 (visited on 09/07/2024).

Bibliography Websites

[2] *Chiron Project*. URL: https://chiron.website/ (visited on 09/07/2024).

[4] *estebancosserat/ros-noetic-moveit general | Docker Hub*. URL: https://hub.docker.com/repository/docker/estebancosserat/ros-noetic-moveit/general (visited on 09/07/2024).

[5] *Fiche équipe imagine*. fr. URL: https://liris.cnrs.fr/equipe/imagine (visited on 09/07/2024).

[6] *Gazebo*. URL: https://gazebosim.org/home (visited on 09/07/2024).

[8] *GelSight | Tactile Robotics | Tactile Sensing | Digital Touch*. URL: https://www.gelsight.com/gelsightmini/ (visited on 09/07/2024).

[9] *ICP registration - Open3D 0.18.0 documentation*. URL: https://www.open3d.org/docs/release/tutorial/pipelines/icp_registration.html (visited on 09/12/2024).

[10] Informatik. *Profil*. de. URL: https://www.informatik.tu-darmstadt.de/fb20/profil/index.de.jsp (visited on 09/07/2024).

[11] *Intelligent Robotics and Biomechatronics Laboratory, Nagoya University*. URL: https://www.mein.nagoya-u.ac.jp/en/index.html (visited on 09/07/2024).

[12] *Interactive Robot Perception & Learning*. URL: https://pearl-lab.com/ (visited on 09/07/2024).

[18] *MoveIt Motion Planning Framework*. URL: https://moveit.ai/ (visited on 09/07/2024).

[19] *Open3D – A Modern Library for 3D Data Processing*. URL: https://www.open3d.org/ (visited on 09/07/2024).

[20] *PAL Robotics | Home*. en-US. URL: https://pal-robotics.com/ (visited on 09/07/2024).

[22] *ROS: Home*. URL: https://www.ros.org/ (visited on 09/07/2024).

[23] *rviz - ROS Wiki*. URL: https://wiki.ros.org/rviz (visited on 09/07/2024).

[25] Jan Peters and the IAS Team. *Intelligent Autonomous Systems | Main / Home Page*. en. URL: https://www.ias.informatik.tu-darmstadt.de/ (visited on 09/07/2024).

[26] *TIAGo - Mobile Manipulator Robot*. en-US. URL: https://pal-robotics.com/robot/tiago/ (visited on 08/21/2024).

[28] *YCB Benchmarks – Object and Model Set | Benchmarking for robotic manipulation*. en-US. URL: https://www.ycbbenchmarks.com/ (visited on 09/07/2024).