VISION-BASED ROBOTIC PICK-AND-HANDOVER MANEUVERS USING VARIABLE IMPEDANCE CONTROL

by

Qiguang Chen

Submitted in partial fulfillment of the requirements
for the degree of Master of Science

at

Dalhousie University
Halifax, Nova Scotia
June 2023

Table of Contents

List of Tables

List of Figures

Nomenclature

Abbreviations

2D	Two-dimensional
3D	Three-dimensional
DMPs	Dynamic Movement Primitives
DOF	Degree of Freedom
FE	Franka Emika
H2R	Human-to-robot
IoT	Internet of Things
IRL	Inverse Reinforcement Learning
LfD	Learn from Demonstration
mAP	Mean Average Precision
PCA	Principal Component Analysis
R-CNN	Region-Convolutional Neural Network
R2H	Robot-to-human
RCM	Reliability Centered Maintenance
RL	Reinforcement Learning
ROS	Robot Operating System

SME	Small and Medium-sized Enterprises
YOLOv5	You Only Look Once version five

Symbols

δ	The difference between two values
$\in$	Belongs to
$\mathbb{R}$	Set of Real Number
$\mathbb{R}^{k*k}$	Set of $k * k$ Real Matrix
$\mathbb{R}^{k}$	Set of $k * 1$ Real Vector
Ψ	Constructed as a radial basis function (RBF)
$\sum$	Summation
ζ	Control-value Parameter

Abstract

The adoption of advanced intelligent manipulator systems to carry out pick-and-handover tasks has seen a rise in both the manufacturing and healthcare sectors, thanks to their impressive precision, adaptability, and operational efficiency. Execution of these tasks demands the synergistic functioning of various modules, encompassing the sensor system used for gathering environmental data, the control algorithm used for manipulator, and the trajectory planning algorithm. The primary objective of this thesis focuses on building a framework, which integrates these modules. A significant merit of this framework is its inherent capacity for easier upgrades. Due to its modular structure, it allows for the modification or replacement of individual components without causing any disruptions to the overall system.

Initially, a vision-based impedance control method is employed with a 7-degree-of-freedom (7-DOF) Franka Emika (FE) Panda robotic manipulator to accomplish pick-and-handover tasks, featuring human-like fruit grasping capabilities, which ensures a basic framework with different modules is built. Subsequently, two low cost vision modules are established for both two-dimensional (2D) and three-dimensional (3D) object recognition, localization, and anthropomorphic manipulation, utilizing the You Only Look Once version five (YOLOv5) system. The salient attribute of this segment revolves around attaining a commendable level of accuracy using low-cost cameras. The innovative modular-based framework also allows for a smooth transition between different types of camera modules, such as shifting from a standard camera module to a depth camera module or a laser module.

To facilitate the end-effector in picking up objects with varying characteristics, a novel Difference-based Dynamic Movement Primitives (DMPs) algorithm is utilized for trajectory planning module to generate human-like trajectories. Finally, a variable impedance controller is designed for control module to strike an optimal balance between precision, safety, and efficiency during the object pick-and-handover task.

Acknowledgements

Firstly, I would like to express my deepest gratitude to my supervisor, Dr. Ya-Jun Pan. Her consistent support, insightful advice, and exemplary conduct have been instrumental in this journey. She has provided guidance not just in the academic realm, but also taught me invaluable input on becoming a responsible individual and a professional researcher. I would like to thank Dr. Darrel Doman and Dr. Serguei Iakovlev for their input and assistance as my supervisory committee members. In addition, I extend my gratitude to Dr. Doman for his unwavering support throughout my undergraduate and graduate studies. I am also grateful to Dr. Serguei Iakovlev for his understanding and tolerance for my spelling errors.

Thank my colleagues in the Advanced Control and Mechatronics Lab for their help and company throughout this program. In particular, I would like to thanks Lucas Wan and Ryan Adderson. Thanks for Lucas for showing me how to solve problems effectively and calmly, for guiding me in starting my research career, and for his insightful analyses completed with pen and paper. I would like to thank Ryan for his advice, kindness, and for being the talkative one in the lab. Thanks for their big hearts which build the great researching atmosphere for our lab. It is my pleasure to have you as my seniors and elder brothers. To my mentors and seniors, I will make your input worthwhile.

Finally, I would like to express my gratitude towards my family and friends for their support in my study career and my life.

Chapter 1

Introduction

This chapter introduces the significance of intelligent control system for robotic manipulators in diverse industries, providing an overview of the motivation, contributions, and structure of this thesis.

1.1 Application of Intelligent Robotic Manipulators

Intelligent control systems for robotic manipulators have been widely researched and developed over the years. They are applied in various industries such as manufacturing, healthcare, logistics, and others. Two applications to be discussed in this chapter are manufacturing and healthcare.

1.1.1 Manufacturing

Fig. 1.1 depicts an instance of a manipulator utilized in the manufacturing industry. This particular system is designed to efficiently carry out assembly tasks.

Intelligent manipulator systems have become increasingly popular in manufacturing applications due to their ability to provide high accuracy, flexibility, and efficiency. [12] discusses the development and implementation of an online compliance error compensation system for industrial manipulators, aimed at improving accuracy and reducing deformation in high-force processes by using an elasto-geometric robot model and force sensor measurements. [13] develops a deep learning-based object detection solution using 3D point clouds for a collaborative mobile robotic manipulator, aimed at automating Small and Medium-sized Enterprises (SME) production processes, with detailed principles, procedures, and experimental validation in automatic name tags production and plug-in charging tasks. A framework for task allocation in human-robot collaborative assembly planning, using a multiagent human-robot team approach with two abstraction layers, where nominal coordinated

skill sequences are generated and executed through complex hierarchical and concurrent hybrid state machines to handle unpredictable events in dynamic environments is proposed in [14].

Figure 1.1: Manipulator used in the industry [1]

Fig. 1.2 shows an example of employing computer vision system in robotic manipulator object detection and path planning system by constructing a 3D gridded workspace with two cameras and planning the path for robotic manipulator with trained neural network [2].

1.1.2 Healthcare

Fig. 1.3 presents the concept of a robotic manipulator used in hospital settings to assist doctors during surgical procedures. Manipulators are important for healthcare due to their ability to perform precise, consistent, and repetitive tasks, assisting in various medical procedures, reducing human error, and improving patient safety. They can be used in surgeries, rehabilitation, diagnostics, and drug dispensing, enabling minimally invasive procedures, faster recovery times, and enhanced accuracy in medical interventions.

Fig. 1.4 shows an example of surgical robot with ability to perform automated lung cancer treatment utilizing pre-operative images and real-time shape sensing through

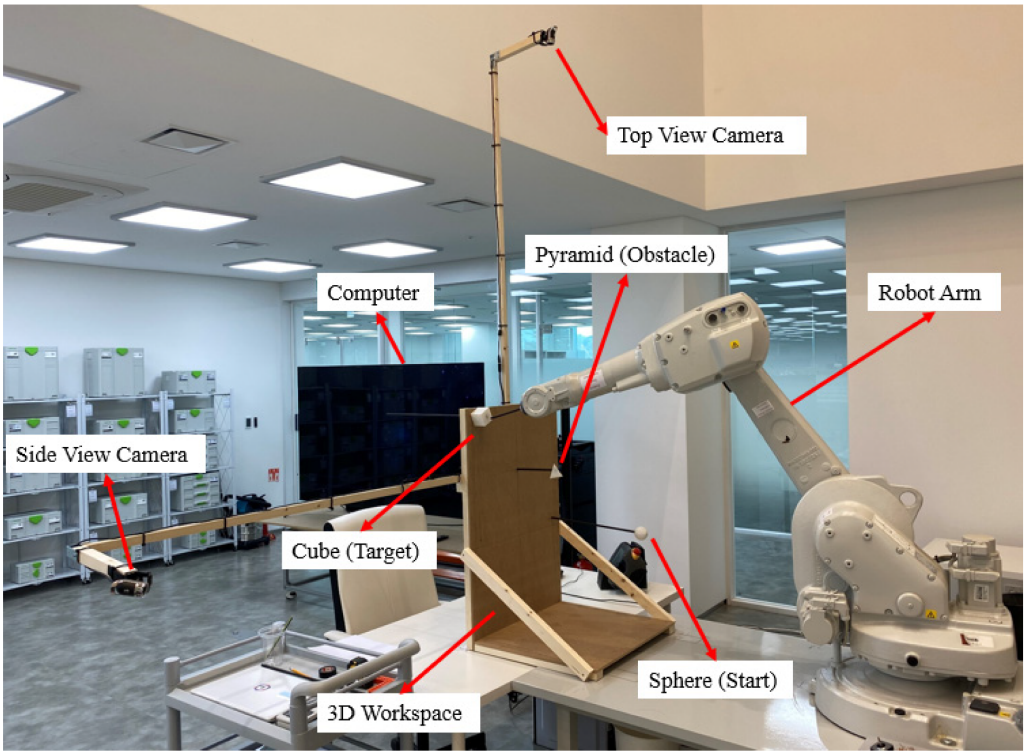

Figure 1.2: Robotic manipulator path planning system with computer vision system [2]

Figure 1.3: Manipulator used in the hospital [3]

optical fiber sensing techniques [4].

[15] presents an IoT-based human-robot collaborative control scheme for robot-assisted minimally invasive surgery, using a hierarchical operational space formulation and IoT technology to facilitate task execution, collision detection, and smooth swivel motion, with improved performance in RCM constraint and surgical tip accuracy demonstrated through experiments on a patient phantom. A motion planning system for assistive or rehabilitation robotics using a learning by demonstration approach based on dynamic movement primitives, offering high generalization, accurate user motion reproduction, and efficient adaptation to object position changes, with reduced memory allocation and computational time is introduced in [16].

Figure 1.4: Surgical robot with ability to perform automated lung cancer treatment [4]

1.2 Research Motivation

To improve robotic manipulators' service quality, better intelligent control systems need to be developed for pick-and-handover task. Such system should have the following functions: 1) the robot is aware of the surrounding environment (such as the positions of objects, targets, and obstacles in the environment) with visual sensors; 2) based on the start point, obstacles, and target position from the gathered

environment information, perform advanced trajectory/path planning; 3) when the manipulator encounters its environment or interacts with humans or environment, it receives force/torque information through various sensors, and performs real-time adjustments to planned trajectory/path to improve force and position convergence performance based on the advanced control method.

In response to the complexities inherent in the intelligent control of robotic manipulators, this thesis endeavors to establish an integrated manipulator system that incorporates all crucial challenges as distinct signal modules. The primary objective is to empower this system with the ability to accurately grasp an object and transfer it to a moving human hand.

1.3 Thesis Contributions and Outline

This thesis first presents the development and integration of an overhead web camera, a qb robotic hand, and a Panda robotic manipulator using the Robot Operating System (ROS). The system demonstrates human-like identification and grasping of delicate fruit in an experimental setting, with the robot maintaining awareness of its 2D surrounding environment. A custom YOLOv5 dataset is trained to facilitate human hand detection and enable successful pick-and-handover maneuvers. The study also develops a high-accuracy object recognition and localization vision system, with a depth error of less than 1.5%, using two low-cost webcams and YOLOv5. Furthermore, a novel DMP algorithm is introduced to learn and generate human-like motions specific to tasks associated with different objects. In the end, a variable impedance control algorithm is introduced to generate various interaction forces between the end-effector and the environment. The benefit of this system lies in its modular approach, affordability, and adaptability. It allows for easy and cost-effective upgrades or modifications, enhancing the system's flexibility and operation life. It imitates human-like motions, enabling delicate handling of various objects, making it adaptable for diverse applications. Moreover, its high-accuracy vision system is particularly noteworthy for its precision despite the use of low-cost equipment.

The intelligent manipulator system created in this study has been published in the 2022 International Congress of Canadian Mechanical Engineering (CSME). Furthermore, the cost-effective yet high-precision camera vision modules, along with the

unique difference-based DMPs algorithm crafted for this work, have been accepted by the 2023 International Federation of Automatic Control (IFAC). The work in completing a pick-and-handover task, utilizing this specialized vision module and variable impedance control, was submitted to the Transactions of the Canadian Society for Mechanical Engineering (TCSME) in December 2022 and revised in April 2023. Please read the Appendix A for my publication list.

The contents of this thesis are organized as follows. Chapter 1 highlights the significance of intelligent control for robotic manipulators in the fields of manufacturing and healthcare. Chapter 2 reviews the foundational literature related to pick-and-handover maneuvers, object detection algorithms, sensors used to get the information of surrounding environment, Learn from Demonstration (LfD) trajectory planning algorithms, and Variable impedance control. Chapter 3 introduces the general system integration for an intelligent manipulator system used for pick-and-handover task, which encompasses a wide range of functions and algorithms, from low-level motor control algorithms to high-level impedance control algorithms. Chapter 4 delves into the development of more accurate vision modules for a manipulator's pick-and-handover task. The DMP algorithm, which enables the manipulator to effectively handle objects of varying shapes is introduced in chapter 5. Chapter 6 discusses the implementation of a variable impedance control method, guiding the manipulator to interact with the environment using different force levels. Finally, Chapter 7 concludes the work presented in this thesis and proposes several prospective areas for future research.

Chapter 2

Literature Review

This chapter presents an overview of recent literature in the fields of pick-and-handover maneuvers, sensors used to get the information of surrounding environment, object detection algorithms, LfD trajectory planning algorithms, and variable impedance control. The primary objective of this chapter is to outline the current state of these topics.

2.1 Pick-and-Handover Maneuvers

Figure 2.1: The concept of human-robot pick-and-handover task [5]

Fig. 2.1 shows the concept of human-robot pick-and-handover task. In the literature review of human-robot object handovers, [5] suggests dividing the manipulator pick-and-handover task into two distinct parts: human-to-robot (H2R) [11] [17] and robot-to-human (R2H) [18] [19]. This subsection helps to clarify different challenges

and considerations involved in each stage of the handover process.

Moreover, [5] recommends distinguishing between sensors used for pre-handover and physical handover phases. The primary purpose of pre-handover sensors is to relay object information to the manipulator, encompassing Kinect cameras (RGBD), optical sensors, motion capture systems, and additional cameras. These will be elaborated upon in the following section. Conversely, the main aim of physical handover sensors is to notify the manipulator when to open its end-effector to release the object into the human hand, or close it to retrieve the object. Force/torque sensors are widely employed for this task. [20] completed a manipulator task utilizing an OptiTrack system as the pre-handover sensor, paired with an ATI Gamma force/torque sensor mounted at the wrist for the physical handover phase. This setup measures the interaction force exerted on the object as it transitions from the giver to the receiver. [21] explores how task-oriented robotic grasping strategies influence the efficiency and human perception of collaborative tasks, indicating that such strategies can significantly enhance human-robot interaction by reducing task completion time and improving the user experience. The main sensor used to collect the data in this paper is the force/torque sensor mounted on the robot's wrist, and the IMU.

2.2 Sensors Used for Surrounding Environment Information

To successfully interact with moving objects or a human's hand, the robot requires the use of different sensors to continuously track their coordinates in real-time. Popular choices for such sensors include the Kinect camera, optical sensors, motion capture systems and cameras [5]. Fig. 2.2 shows the concept of sharing environment between manipulator and different sensors, cameras [6], RGBD cameras [7], motion capture systems [8] and optical sensors [9].

RGBD cameras provide both color and depth information, allowing for accurate tracking of objects and humans in 3D, making them useful for object recognition, tracking, manipulation, and human-robot interaction [22]. [23] presents a system for multimodal human-robot interaction, allowing the user to use natural language to ask the robot to grasp objects detected using a low-cost RGBD camera. [24] describes the use of an RGBD camera in conducting experiments to integrate human-robot and human-human object handover interactions for the purpose of developing and

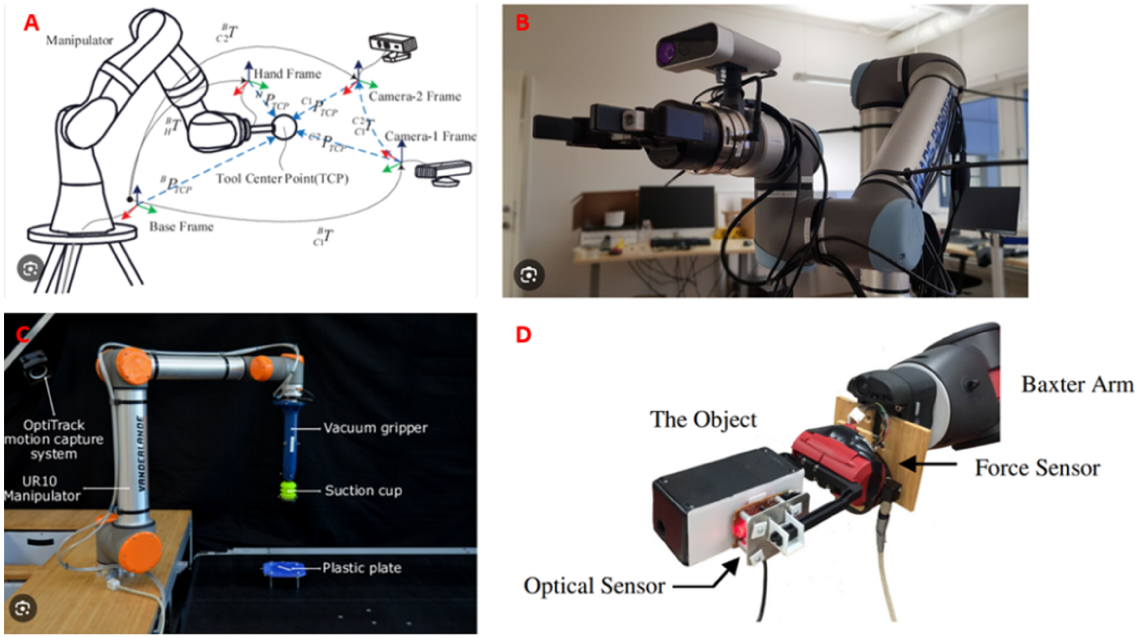

Figure 2.2: The concept of sharing environment between manipulator and different sensors [6] [7] [8] [9]

evaluating a robot prototype system.

Optical sensors offer a high precision and accuracy in tracking object and hand movements. [25] describes the implementation and experimental validation of a control system for human-robot physical interaction during object handover. The design and implementation of a teleoperated robotic arm with joint impedance control and optical torque sensors, enabling safe interaction with unstructured environments and humans is described in [26].

Motion capture systems are specialized systems that use multiple cameras and markers to track the movements of objects or people with high precision and accuracy. [27] presents an exploratory user study of human-to-robot handovers which utilized a set of 12 OptiTrack Flex 13 motion capture cameras to accurately track the baton, participant's dominant hand, and end-effector. [28] proposes a method for enabling robots to determine appropriate grasp configurations for handovers by building a knowledge base of objects and their proper handover grasp configurations, organizing them based on movements and inter-object interaction features, and recognizing similarity in affordances, allowing for successful generalization to new objects by using motion capture system.

Cameras can capture high-resolution images with color information, making it easier to identify and distinguish between different objects based on their visual characteristics, such as shape, color, texture, and size. These advantages also make cameras useful for gathering information of the surrounding environment, such as detecting and tracking objects or people. [29] describes the design of a human-robot co-working scheme using natural language and computer vision, demonstrating the integration of Google voice recognition and YOLO software with a KINOVA robotic arm for collaborative robot interaction. [30] presents a vision-based dynamic object recognition system by using high quality industrial camera for pick-and-place tasks, which detects landmark features and provides grasping points for randomly located objects, with evaluations demonstrating accurate detection of location, posture, distance, and object type, and successful pick-and-place task execution by a robotic manipulator.

In this thesis, cameras were chosen as the sensor to gather information of the surrounding environment, as they offer many advantages compared to other sensor setups. One of the most important advantages is that images captured by cameras can be supplied to most object detection algorithms directly, which is useful for determining the coordinates of the object and hand during interactions.

2.3 Object Detection Algorithms

Fig. 2.3 shows the concept of object detection algorithm for this thesis. You only look Once (YOLO) [31], Single-Shot Detector (SSD) [32], Faster Region-Convolutional Neural Networks (R-CNN) [33], and Mask R-CNN [34] are popular object detection algorithms used in computer vision and machine learning applications. YOLO and SSD are fast and simple to implement but have lower accuracy, while Faster R-CNN and Mask R-CNN offer higher accuracy but are slower and require more computational resources. YOLO and SSD are better for detecting small objects and have fewer false positives, while Faster R-CNN and Mask R-CNN can handle more complex object shapes and sizes, and can detect object masks. Ultimately, the choice of which algorithm to use depends on the specific application and the trade-off between speed, accuracy, and computational resources available. For this thesis, YOLO was chosen as the object detection algorithm due to its fast, real-time performance, ability to detect small objects well, and ability to reduce false positives. These characteristics

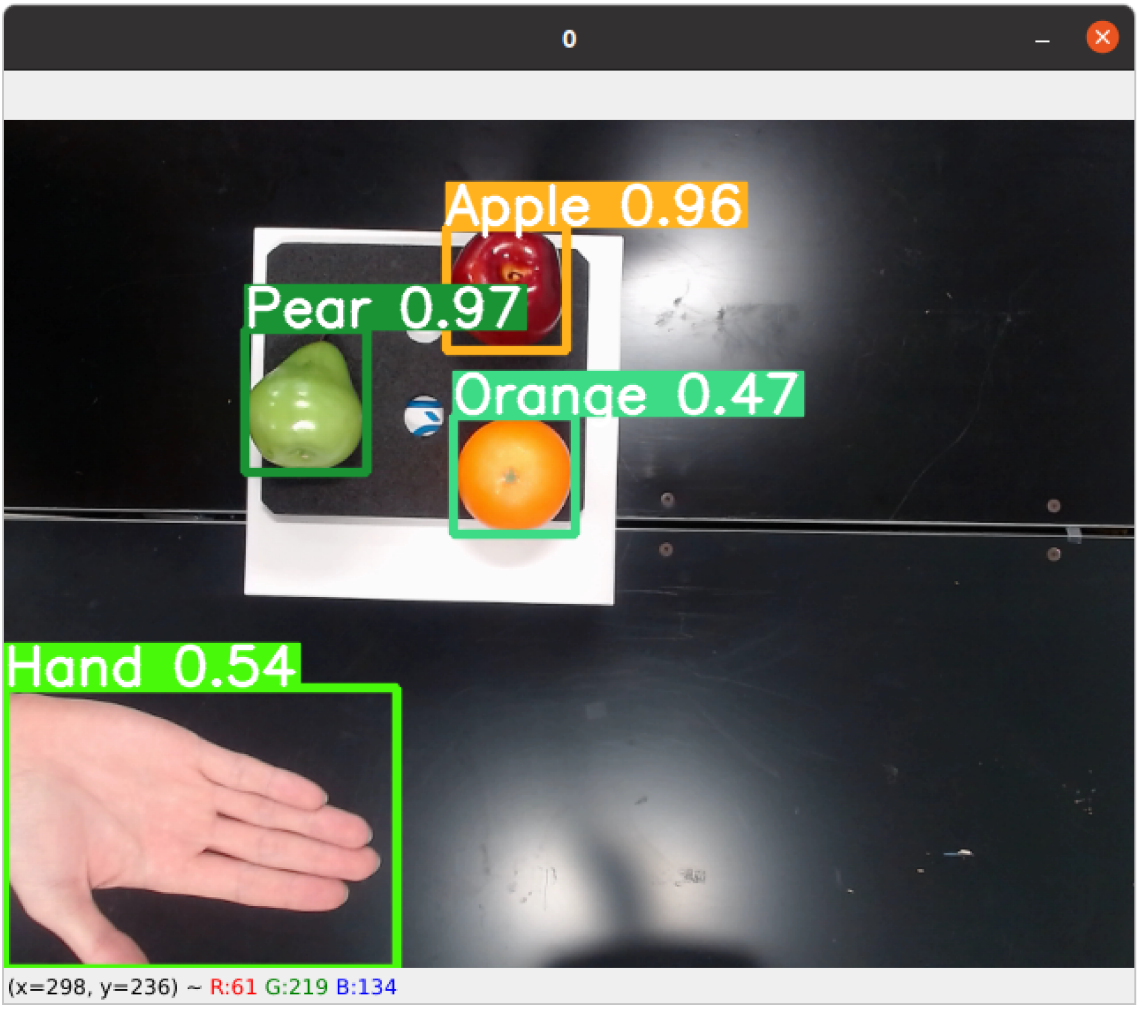

Figure 2.3: An example of object detection algorithm

make it well-suited for the real-time, dynamic human-robot interaction tasks that the thesis focuses on.

2.4 Learn from Demonstration Trajectory Planning Algorithms

Learn-from-Demonstration (LfD) trajectory planning algorithms involve recording human demonstrations and extracting features such as joint positions and velocities, and then using this data to train a model to regenerate trajectory for the robot. Fig. 2.4 shows an example of the general process for LfD, which includes multi-modal interfaces module, communication module, teleportation control module, and

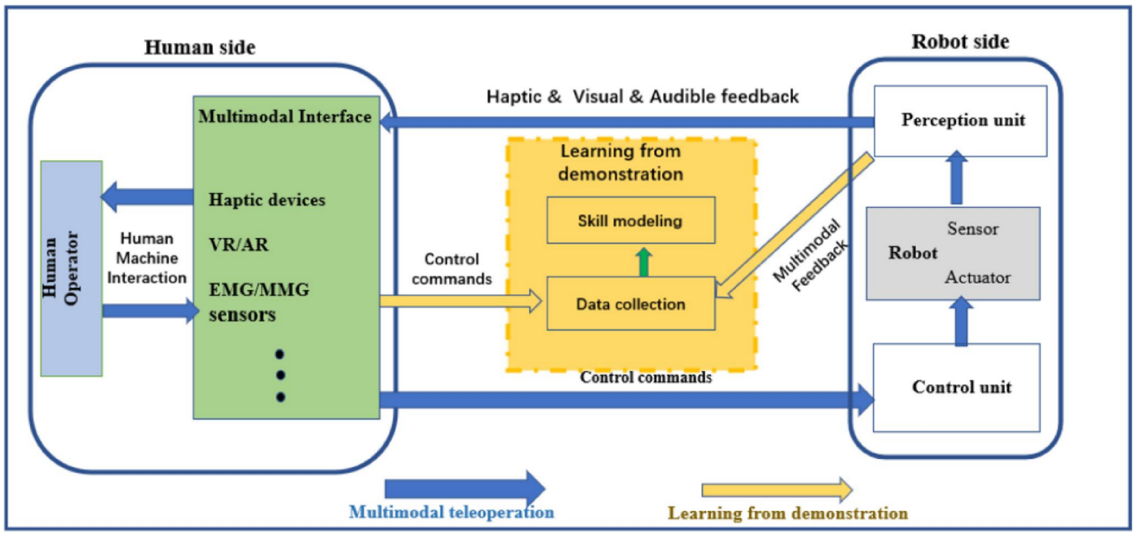

Figure 2.4: An example of the general process for LfD [10]

skill learning and generalisation module. Some popular LfD algorithms used for manipulator are Inverse Reinforcement Learning (IRL) [35] and Dynamic Movement Primitives (DMPs) [36].

IRL learns a reward function from human demonstrations to generate new trajectories that optimize the reward. [37] proposes a method for personalizing the behavior of assistive robotic arm manipulators for individuals with upper-limb motor disabilities, using Gaussian process-based inverse reinforcement learning to infer user's preferences and adapt the robot's obstacle avoidance strategy, validated through experiments with thirteen able-bodied subjects. [38] explores the use of two versions of inverse reinforcement learning and Principal Component Analysis (PCA) weighting for transferring task knowledge from a human expert to a robot in a dynamic environment.

Dynamic Movement Primitives models the trajectory of the demonstrated task as a nonlinear differential equation and generates new trajectories by modifying the parameters of the equation. The work in [39] involves a robot learns from human demonstrations for human–robot handovers and the accuracy of the task is improved by using an adaptive learning and control framework-based DMPs. A method was proposed in [13] to improve the accuracy of regenerated trajectories used for assisting humans in their daily activities. They achieved this by selectively increasing the kernel density in specific areas of the trajectory. [40] developed a method to optimize

the parameters of motion primitives to improve efficiency and robustness in high dimensional problems. In our approach, we implemented a difference-based DMPs algorithm to enhance the practicality and safety of the regenerated trajectory.

DMPs have several advantages over IRL, making them suitable for real-world applications. One of the most significant advantages is their adaptability to changes in the task and environment. Another advantage is that DMPs can adjust the shape and timing of trajectories based on feedback from the environment, allowing the robot to adapt to new situations. A difference-based DMPs was employed as the trajectory planning algorithm in this thesis.

2.5 Impedance Control

Impedance control is a popular control method for human-robot interaction and has been applied in numerous studies [41] [42] [43].

It is used in this framework because it is easy to implement, robust, and can achieve safe, compliant physical interaction with human users. The controller feedback is designed in Cartesian space because the position of the objects are given in Cartesian space. The general control torque is computed as

$$\boldsymbol{\tau} = J^T(-K\tilde{\boldsymbol{x}} - B(J\dot{\boldsymbol{q}})) + C(\dot{\boldsymbol{q}}, \boldsymbol{q})\dot{\boldsymbol{q}}, \tag{2.1}$$

where

$$K = \begin{bmatrix} K_t & 0 \\ 0 & K_r \end{bmatrix}, \tag{2.2}$$

$$B = \begin{bmatrix} B_t & 0 \\ 0 & B_r \end{bmatrix}, \tag{2.3}$$

$\tilde{\boldsymbol{x}} \in \mathbb{R}^6$ contains the Cartesian position and orientation errors, $J \in \mathbb{R}^{6\times k}$ is the Jacobian matrix, $K_t, K_r, B_t, B_r \in \mathbb{R}^{3\times 3}$ are diagonal matrices that contain the translational and rotational impedance stiffness and damping parameters, respectively.

[44] provides an overview and comparison of key concepts and principles, implementation strategies, important techniques, and real-world applications related to the impedance control of robotic manipulation. Several advanced impedance control methods were summarized in this article, such as force-tracking impedance control,

hybrid impedance control, robust impedance control, and adaptive impedance control. Force-tracking impedance control allows a robot to maintain a predetermined force on an object while also adapting to changes in the object's position or movement, thereby achieving a balance between stiffness and flexibility in handling tasks [45]. Hybrid impedance control combines force and position control, allowing a robot to maintain a specified force along a certain direction (usually normal to the contact surface) while accurately controlling position in the other directions, thus enabling effective interaction with both its environment and humans [46]. Robust impedance control ensures stability and desired dynamic performance when interacting with uncertain environments, even in the presence of unknown or variable parameters such as changes in mass, stiffness, or damping [47]. Variable impedance control automatically adjusts the control parameters in real-time to match the changing dynamics of the environment or task, thereby optimizing performance and interaction with unpredictable surroundings [48]. In human-robot interaction, the advantage of adaptive impedance control lies in its ability to safely and efficiently respond to unpredictable human behavior in real-time, adjusting the robot's force and motion to maintain smooth, natural interaction, which enhances user experience, safety, and the overall effectiveness of the collaborative task. For this reason, variable impedance control was chosen for this thesis.

Chapter 3

The Integration of Intelligent Manipulator System

This chapter introduces the general system integration for an intelligent manipulator system, which encompasses a wide range of functions and algorithms, from low-level motor control algorithms to high-level control algorithms for human-robot interaction and environment-robot interaction. To verify the feasibility of the system, a vision-based impedance control method is applied to a 7-degree-of-freedom (7-DOF) Franka Emika robotic manipulator to complete pick-and-place tasks with human-like grasping of fruits.

3.1 Background

As robotic manipulators becomes more prevalent in various fields such as manufacturing, healthcare, and education, it is crucial to develop a general flowchart that outlines the different stages of research in the system of intelligent manipulator control. This flowchart would help researchers to identify the position of their research in the broader context of the control system, facilitating the integration of their work into the existing framework. A flowchart for the system of intelligent manipulator control of this thesis could be structured as shown in Fig. 3.1, which is the summary of Fig. 3.2, Fig. 3.4, and Fig. 3.5. The first step is to determine the end-effector's Cartesian coordinates based on the kinematic and dynamic models of the manipulator. This involves using mathematical models to calculate the position, velocity, and acceleration of the end-effector based on the joint angles of the manipulator. The second step is to generate the desired trajectory of the end-effector based on sensory information, such as the coordinates of the start point, goal point, and obstacles in between. This involves using algorithms to plan a safe and efficient trajectory for the manipulator to follow. The third step involves considering the force interaction between the manipulator and its environment, including any human interaction. This includes developing control algorithms that can allow the end-effector to apply

a desired force to the environment.

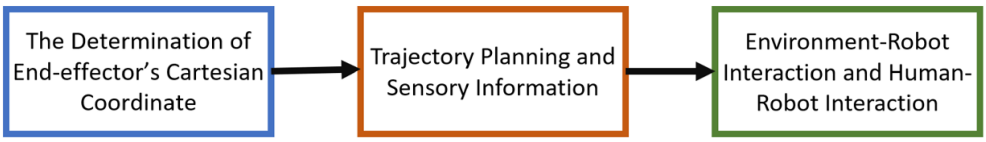

Figure 3.1: A flowchart for an intelligent manipulator system

3.1.1 Cartesian Coordinates and Kinematics

As shown in Fig. 3.2, the actual motor positions are measured, and the joint angles are calculated by using the kinematic model of motors. The dynamic model of the manipulator is then used to relate the joint angles, joint velocities, joint accelerations, and external forces/torques acting on the manipulator together. The kinematic model of the manipulator is used to calculate the position and orientation of the end-effector based on the updated joint angles, which are then converted to the world coordinate system and corrected for any necessary transformations or errors. Finally, the corrected end-effector coordinates are output as the result of the process.

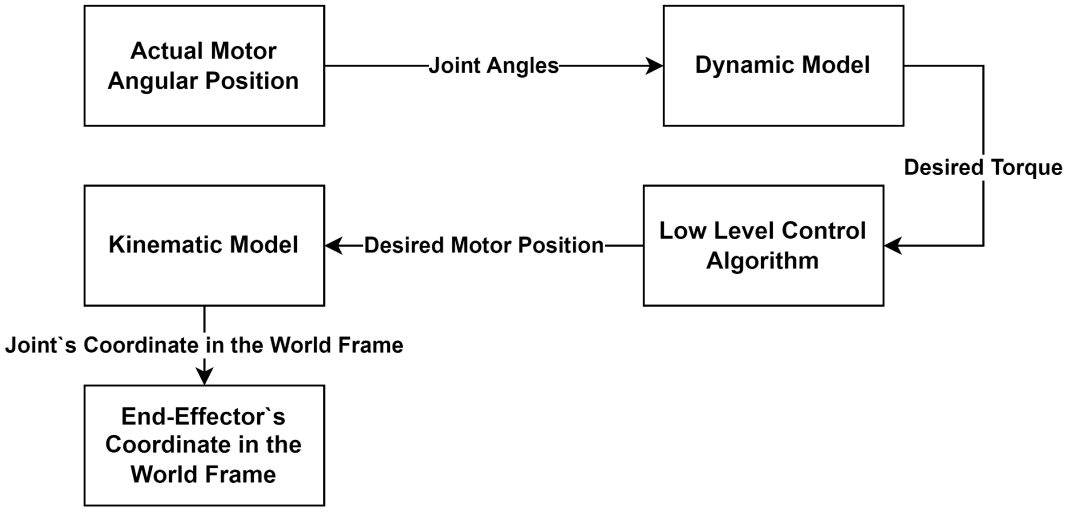

Figure 3.2: A flowchart for the determination of the end-effector's Cartesian coordinates

The rest of this section introduces the kinematic model of a manipulator, using a two-degree-of-freedom (DoF) manipulator as an illustrative example. The forward kinematic model details how the position and orientation of the manipulator's end-effector is a function of each joint's angular or linear position, whether the joint is

revolute or prismatic. For a robot with n links, the relationship between the position of the end-effector and the first joint can be expressed as follows:

$$T_n^0 = T_1^0 T_2^1 T_3^2 \cdots T_n^{n-1} \tag{3.1}$$

Where T_n^0 is the transformation matrix between the frame of the first joint and the end-effector, and T_n^{n-1} is the transform matrix between adjacent joints. A transformation matrix is made up of a rotation matrix, R_i^{i-1}, and a prismatic matrix, P_i^{i-1} as follow:

$$T_{i-1}^i = \begin{bmatrix} \boldsymbol{R_i^{i-1}} & \boldsymbol{P_i^{i-1}} \\ 0 & 1 \end{bmatrix} \tag{3.2}$$

$\boldsymbol{P_i^{i-1}}$ is composed of three prismatic vectors, in which a joint's frame travelled in X, Y, and Z direction as shown as follow:

$$\boldsymbol{P_{i-1}^i} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} \tag{3.3}$$

$\boldsymbol{R_i^{i-1}}$ is used to express the orientation angle of a frame as shown as follow

$$\boldsymbol{R_{i-1}^i} = \begin{bmatrix} \hat{x}_i^{i-1} & y_i^{i-1} & z_i^{i-1} \end{bmatrix} \tag{3.4}$$

Where $\hat{x}_i^{i-1}$, $\hat{y}_i^{i-1}$, and $\hat{z}_i^{i-1}$ are the projection matrix for the three axes of the original frame between the rotated frame and the original frame about the X-, Y-, and Z-axis of the original frame, respectively. To describe the transformation matrix as easily as possible, a popular convention, Denavit-Hartenberg diagram, will be introduced.

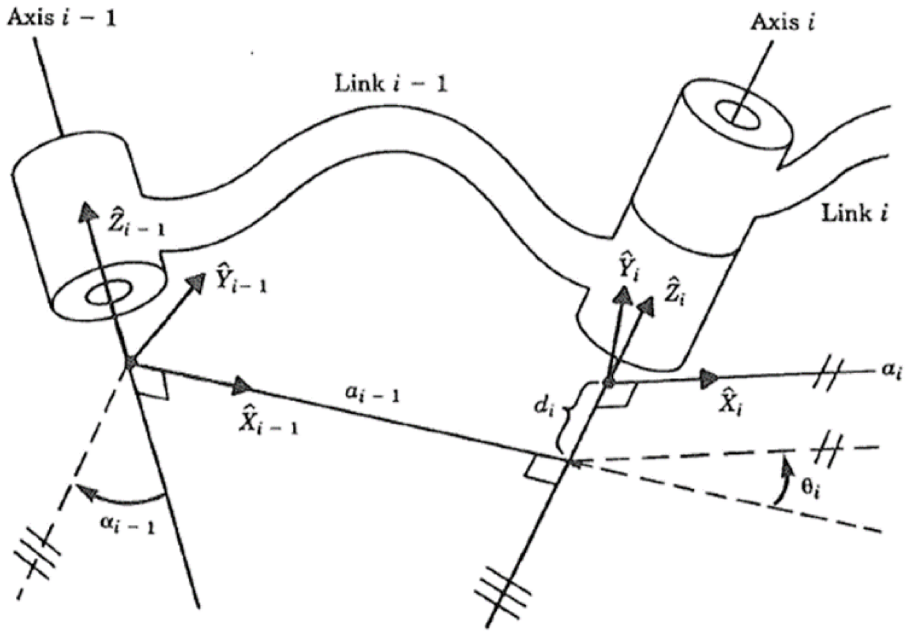

Figure 3.3: Illustration of modified Denavit-Hartenberg convention[11]

Symbols and assumptions, labeled in Fig 3.3, are to be established and introduced:

- The order of the links and joint axes is determined from base to tip.
- The Z_{i-1} axis is aligned with the $i-1$ joint axis.
- The X_{i-1} axis is aligned with the common normal between the $i-1$ and i joints.
- The Y_{i-1} axis is established using the right-hand rule.
- α_{i-1} (link twist) is defined as the angle from Z_{i-1} to Z_i, measured along X_{i-1}.
- a_{i-1} (link length) is defined as the distance from Z_{i-1} to Z_i, measured along X_{i-1}.
- d_i (link offset) is defined as the distance from X_{i-1} to X_i, measured along Z_i.
- θ_{i-1} (joint angle) is defined as the angle from X_{i-1} to X_i, measured about Z_i.

And then the transformation matrix, T_i^{i-1}, can be expressed by:

$$
{}^{i-1}T_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_{i-1}) & \sin(\theta_i)\sin(\alpha_{i-1}) & a_{i-1}\cos(\Theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_{i-1}) & -\cos(\theta_i)\sin(\alpha_{i-1}) & a_{i-1}\sin(\theta_i) \\ 0 & \sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}
$$

3.1.2 Trajectory Planning for the End-Effector

Several steps are required to generate a desired trajectory for the end-effector, as shown in Fig. 3.4.

To begin the process, the path planner must be provided with the Cartesian coordinate of the end-effector or a particular joint, along with the coordinates of obstacles marked as No-go areas and the target location. The coordinates of the joints and end-effector are commonly obtained through the manipulator's encoder, while sensors on the manipulator or in the surroundings are usually used to detect obstacles and target coordinates.

After obtaining this information, the path planner takes charge and generates a path for the end-effector to follow. This path is based on the sensory data and considers the position of any obstacle in the workspace. Once the desired path is generated, it is sent to the trajectory planner for further processing.

The trajectory planner takes the desired path generated by the path planner and generates a trajectory for the end-effector to follow as an output for this section. This trajectory is created based on the manipulator's specific capabilities, such as the joints' range of motion, and considers any physical constraints or limitations present. The trajectory planner ensures that the manipulator moves in a smooth and efficient manner, while also meets any performance requirements or safety standards.

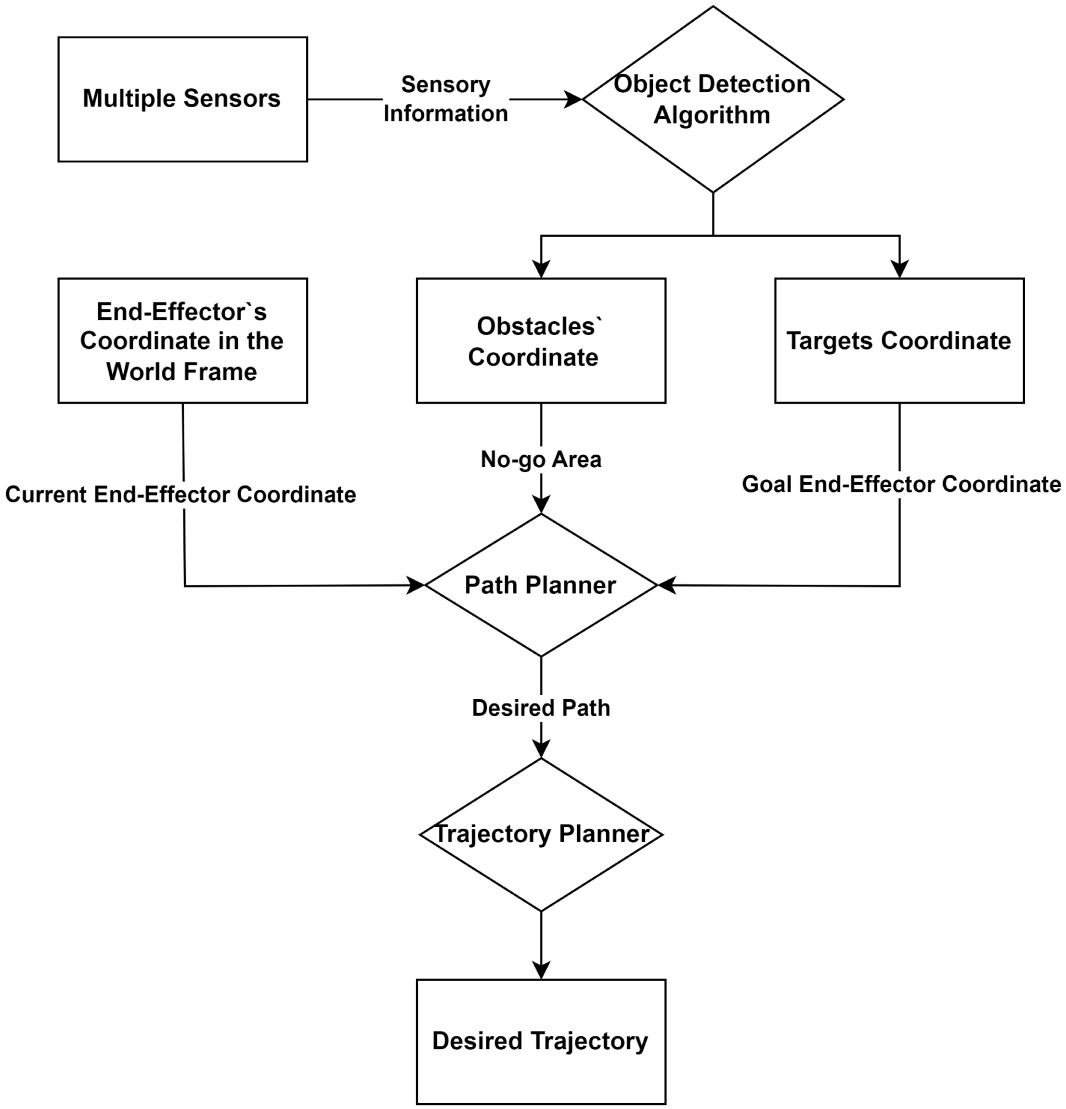

Figure 3.4: A flowchart for the trajectory planning for the end-effector

3.1.3 Human-Robot and Environment-Robot Interaction

This section introduces the process of environment-robot interaction as shown in Fig. 3.5, which includes human-robot interaction as a type of interaction. The process involves providing the environment-robot interaction controller, which is typically an impedance controller, with the force/torque requirement for the interaction, target object location, and desired trajectory as inputs. The controller processes the input data and generates an updated trajectory that incorporates the specific impedance properties required for the interaction. The updated trajectory is then sent as output, and the manipulator adjusts its movements accordingly to move the end-effector to the desired target location while satisfying the force/torque requirements.

In conclusion, the process introduced in this section enables the manipulator to perform complex tasks while interacting with humans or other objects in the environment with high precision and safety.

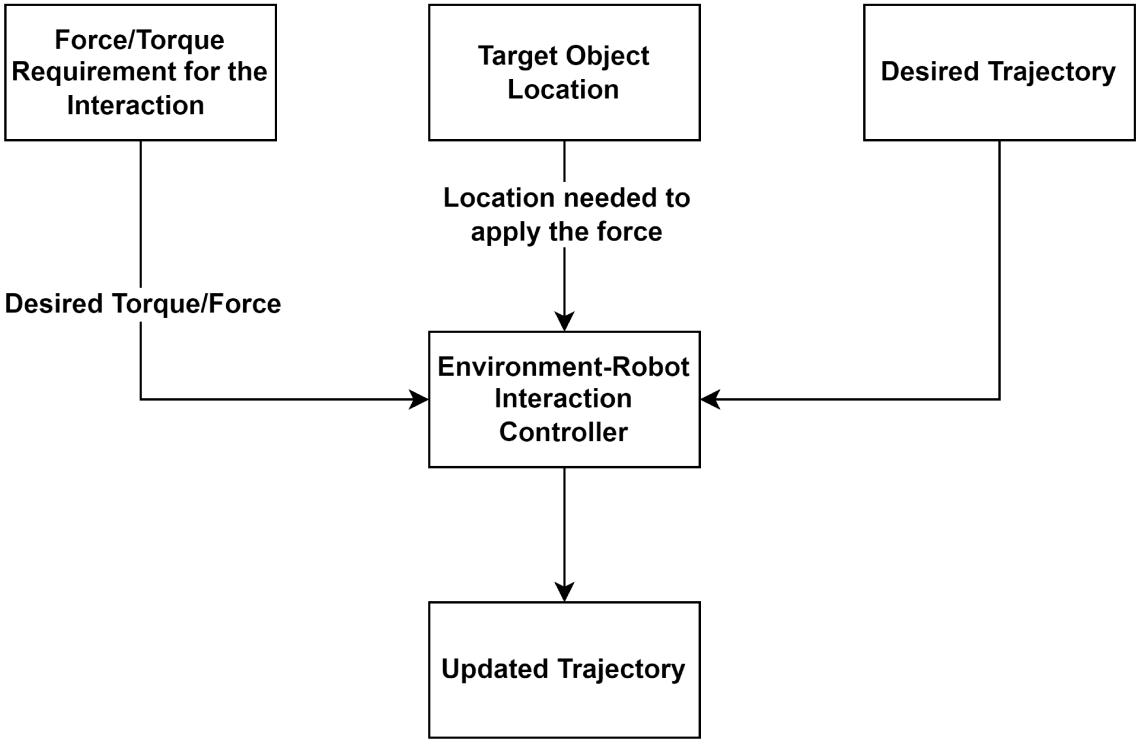

Figure 3.5: A flowchart for the advanced control algorithm for the end-effector

3.2 Problem Formulation

To verify the feasibility of the intelligent manipulator system, a vision-based impedance control method is applied to a 7-DOF Franka Emika robotic manipulator for completing pick-and-place tasks with human-like grasping maneuvers of fruits.

The general form for a k-DOF serial robotic manipulator can be written as:

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\dot{\boldsymbol{q}}, \boldsymbol{q})\dot{\boldsymbol{q}} + G(\boldsymbol{q}) = \boldsymbol{\tau} + \boldsymbol{\tau_f}, \tag{3.6}$$

where $M(q) \in \mathbb{R}^{k\times k}$ is the symmetric and positive definite inertia matrix of the robot, $\boldsymbol{C}(\dot{\boldsymbol{q}}, \boldsymbol{q}) \in \mathbb{R}^{k\times k}$ is the Coriolis and Centrifugal torque matrix, $G(\boldsymbol{q}) \in \mathbb{R}^k$ is the gravity torque matrix, $\boldsymbol{\tau} \in \mathbb{R}^k$ is the control input torque, $\boldsymbol{\tau_f} \in \mathbb{R}^k$ is the joint friction torque, and $\boldsymbol{q} \in \mathbb{R}^k$, $\dot{\boldsymbol{q}} \in \mathbb{R}^k$, and $\ddot{\boldsymbol{q}} \in \mathbb{R}^k$ are the joint positions, joint velocities, and joint accelerations, respectively. The dynamic matrices, M, C, and G are calculated internally by the FE Panda robot model library and the friction, $\boldsymbol{\tau_f}$, is internally compensated for. The simulations in this thesis use the estimated dynamic parameter values as determined by [49].

The integration of the system components is facilitated through ROS, allowing for efficient communication and coordination between different hardware and the controller. YOLOv5 object detection algorithm is trained to detect fruits and used to capture their locations in the workspace. For the grasping tasks, a qb-SoftHand robotic hand is employed as the end-effector. The desired experimental results will demonstrate the successful achievement of autonomous human-like pick-and-place tasks by the Franka Emika robot. While the primary aim of this thesis is to accomplish a pick-and-handover task, successfully completing this task will lay a solid foundation for future work in this area.

3.3 Experiments and Results

The experiments and results used to verify the intelligent manipulator system will be introduced in this section.

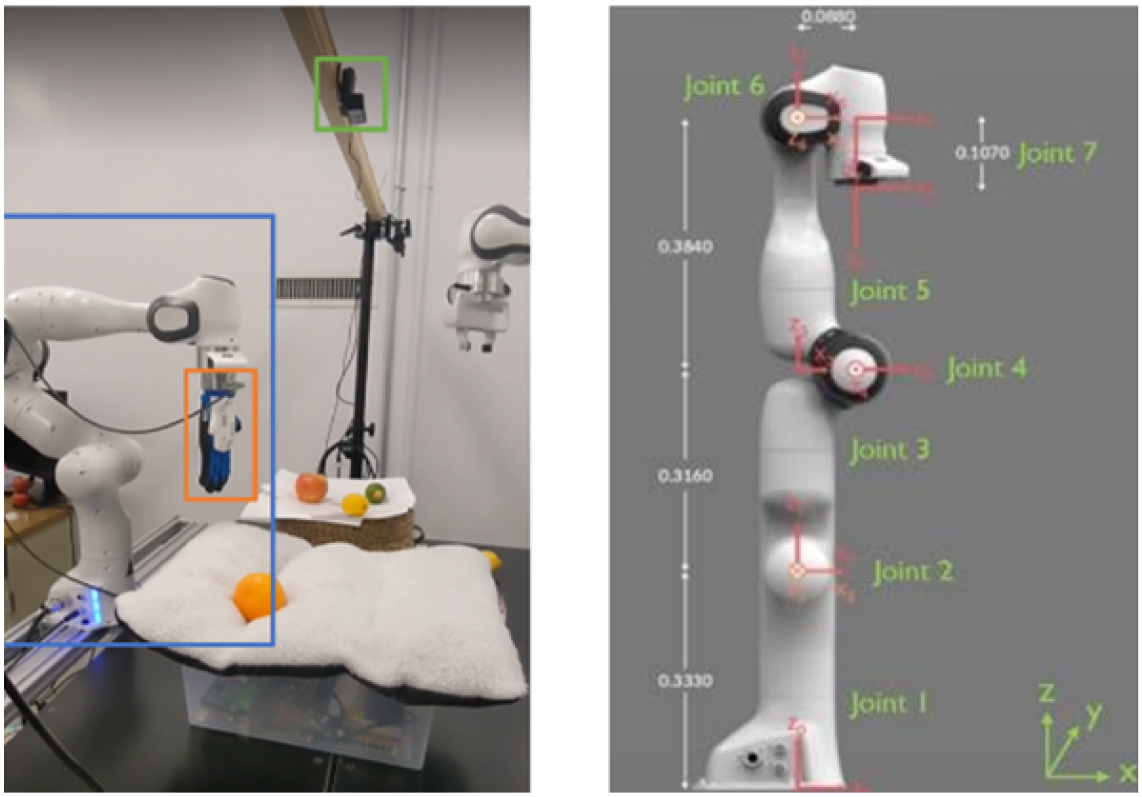

Figure 3.6: Equipment setup in the Advanced Control and Mechatronics (ACM) Lab at Dalhousie University.

3.3.1 Franka Emika Robotic Manipulator

Fig. 3.6 shows the equipment setup in the Advanced Control and Mechatronics (ACM) Lab at Dalhousie University. As shown in Fig. 3.6, the 7-DOF robotic manipulator, labeled blue, is the Franka Emika Panda research robot; the end-effector, labeled orange, is the qb Softhand research setup; and the camera, labeled green, is a Logitech C922 HD webcam.

3.3.2 qb-Softhand Research Setup

A qb-SoftHand anthropomorphic robotic hand is employed as the end-effector on the Franka Emika Panda robot for pick-and-place tasks. The qb-SoftHand closely resembles a human hand and has several advantages. First, the qb-SoftHand is compliant and safe to operate near humans. Second, the qb-Softhand's five mechanically independent fingers allow this end-effector to have both a large gripping surface and excellent adaptability to grasp objects of different sizes and ductility. Third, the

qb-SoftHand has 19 anthropomorphic DOFs and only one motor, allowing for easy control implementation. An example of performing human-robot interaction by using qb-Softhand is shown in Fig. 3.7.

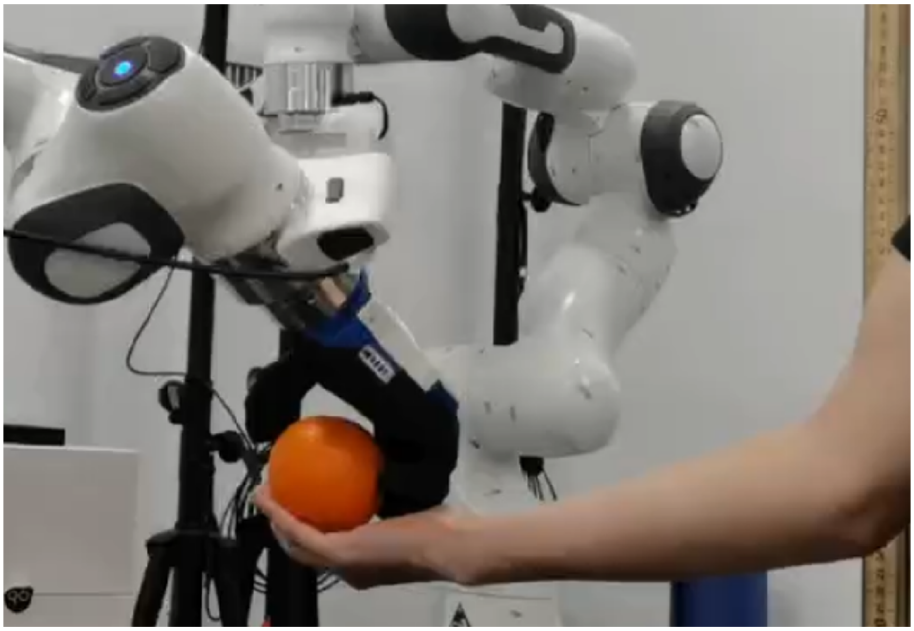

Figure 3.7: An example of performing human-robot interaction by using the qb-Softhand

3.3.3 Robot Operating System

Robot Operating System (ROS) [50] is a system architecture designed for interfacing with robot hardware and integrating different robotic hardware together. The function of ROS is like a universal plugin, which has the ability to connect different software and coding languages. ROS has built-in developing tools, such as Gazebo and Rviz, which provide a powerful simulation environment that allows developers to test their results before applying them to the real environment.

The ROS communication system is composed of several elements: the ROS network, ROS master, ROS nodes, ROS publisher, and ROS subscriber. The ROS network consists of different components of a mobile robot like the controller, camera, path planner, and motors, all of which communicate via the ROS network. Notably, the ROS network can be distributed across multiple machines. The ROS master serves to manage and coordinate the different parts of the ROS network.

ROS nodes are differentiated based on their functions, including publishers, subscribers, and services. These nodes exchange data using ROS messages. Publishers send messages to a specific topic, and subscribers then opt to receive the desired messages.

ROS commands are utilized to invoke desired functions. To commence using ROS, the initial step involves running 'roscore', or alternatively, launching a launch file which will auto-execute 'roscore' using the command "roslaunch package_name package_launchfile_name.launch". Once 'roscore' is operational, several commands become available to retrieve the desired information.

The command "rosnode list" is employed to display all active ROS nodes. Similarly, "rostopic list" showcases all active ROS topics. To echo the information within a 'rostopic', the command "rostopic echo topic name" is used. This command essentially mirrors what a ROS subscriber would do within a node, i.e., print the data on the terminal each time the subscriber callback is triggered.

The command "rostopic hz" is useful for calculating the publishing rate (or frequency) of a given topic. Lastly, the command "rostopic info" presents the fundamental information of a 'rostopic'.

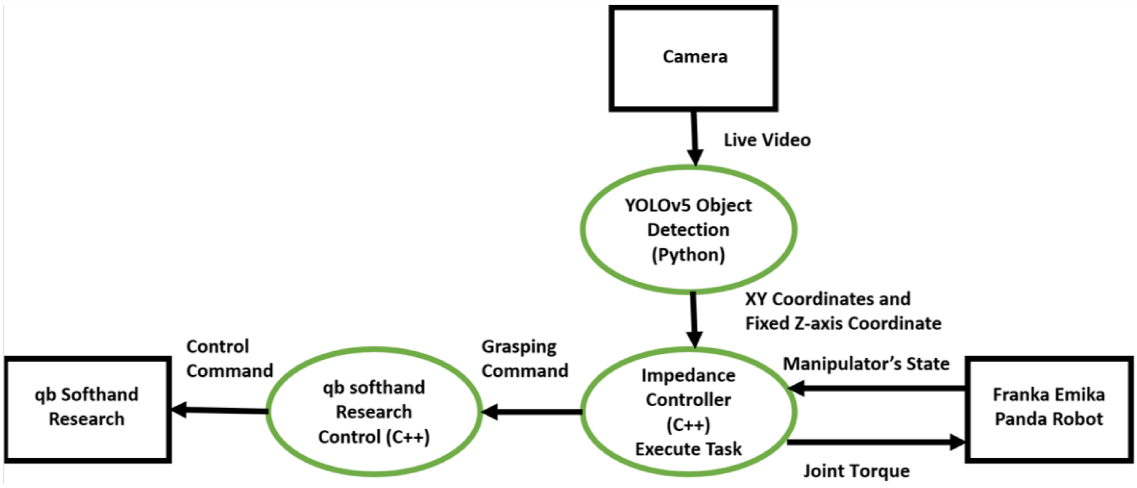

Figure 3.8: ROS block diagram of the experiment

The ROS block diagram of the experimental hardware framework is shown in Fig. 3.8. Black square boxes are used to identify hardware, and green ellipses are used to identify ROS nodes. First, live video is taken by the web camera and input to the YOLOv5 object detection algorithm. Next, the XY-coordinates of the desired target

(apple and orange for the experiment in this paper) are calculated. The computer vision-based XY-coordinates and a fixed Z-coordinate are input to the impedance controller as the desired coordinates of the robotic manipulator's end-effector. A set of actions are prescribed in the impedance example controller to perform the pick-and-handover task, which is shown in the following section. The desired joint torque is commanded to the robot, and the manipulator's state is measured by encoders and torque sensors and sent back to the controller to finish the closed-loop control. On the other side, open and close commands are commanded to the qb-Softhand to let it execute desired actions.

3.3.4 YOLOv5 Overview

For pick-and-handover task, the YOLOv5 model was trained on custom images of fruits and hands. Fig. 3.9 shows the series of tasks that were completed to train and optimize the YOLOv5 model to detect custom objects. The process started with image acquisition where the fruits (apples, oranges, lemons, and pears) and hands were placed under an RGB camera. A total of 1800 frames/images with fruits and hands placed at different orientations and positions in the frame were obtained, with fruits and hands of different classes placed in equal proportions to create a balanced dataset.

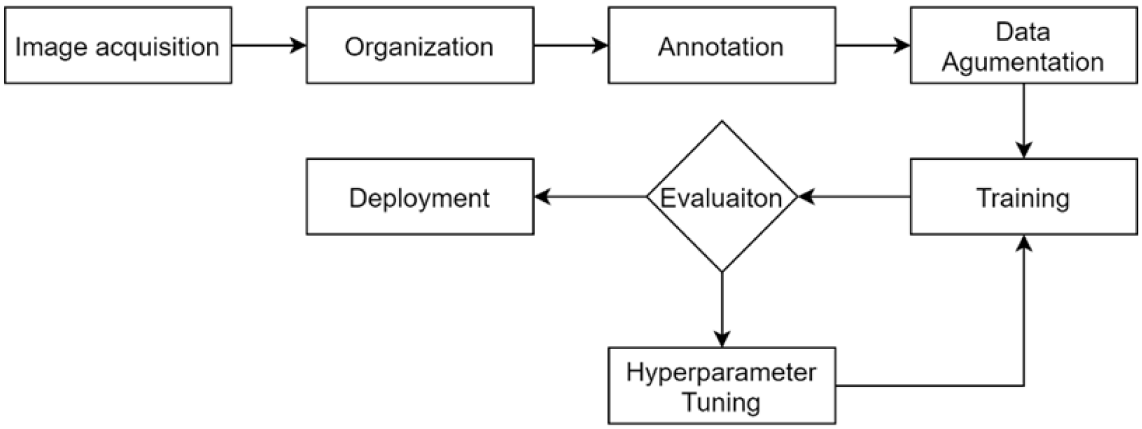

Figure 3.9: Block diagram for training YOLOv5 on custom objects for the pick-and-handover task.

The training images were uploaded to Roboflow, an online platform used for organizing, annotating, and augmenting images. The images were annotated by drawing a bounding box around the object and identifying its class. The annotated images were randomly split into training, validation, and test datasets. The data was further augmented by randomly flipping, rotating, and adjusting the brightness and hue of the images. In total, 1300 images were used for training, 400 images for validation, and 100 images for an independent test set.

The YOLOv5 model was trained in a Google Colab environment with GPU resources to accelerate the training process. A code snippet from Roboflow was copied into the Colab environment to download the prepared dataset. Taking the implementation computer configuration into consideration, the smallest and fastest pre-trained YOLOv5 model was chosen for fine-tuning.

The YOLOv5 model was repeatedly trained by optimizing key hyperparameters, such as image size, batch size, and learning rate, and evaluated based on metrics such as mean Average Precision (mAP@0.5:0.95), precision, and recall. The model was initially trained with an image size of 416 and gradually increased to 624, 832, and 1024 with a batch size of 16 for 300 epochs. An increase in (mAP@0.5:0.95) was observed from 0.652 to 0.835 when the image size was increased from 416 to 1024. An inference made on a image (A) and a live video (B) for the pick-and-handover task is shown in Fig. 3.10. The trained weights were downloaded for the deployment onto the application.

3.3.5 Task Execution

The task execution process is developed in C++ in sequence with the impedance controller provided by franka. The flowchart for the pick-and-place task is provided in Fig. 3.11. The controller is executed at 30 Hz. For each subtask, while the end-effector is moving, the error between the current and desired positions is calculated until it reaches a specified threshold. Once the error has been calculated according to the appropriate subtask, the impedance controller is computed, and the control torque is applied to the robot.

Additionally, counters are implemented in each subtask to check that the error is within the error threshold for 500 loops. This is to avoid moving on to the next

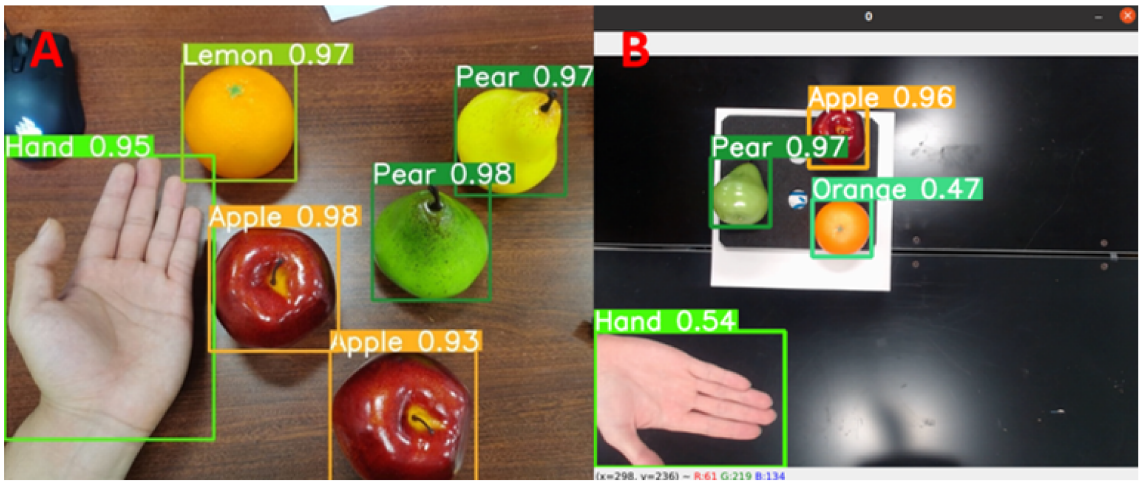

Figure 3.10: An inference made on a image (A) and a live video (B) for the pick-and-handover task.

subtask without ensuring that the robot is locked on to the desired position in the case of overshooting or external interference. To avoid sudden motions while maintaining position accuracy, the error is calculated incrementally as

$$\Delta \boldsymbol{X} = \frac{C}{C_{max}}(\boldsymbol{X} - \boldsymbol{X_d}), \tag{3.7}$$

where $\boldsymbol{X}$ is the robot state in Cartesian coordinates, $\boldsymbol{X_d}$ is the desired state, C is a counter, and C_{max} is a maximum counter threshold. In this application, the path that the robot takes to the goal points is not prioritized, and therefore the robot takes the shortest path as generated by the impedance controller.

3.3.6 Coordinate Transformations

The method used to transform the target's pixel coordinates in the camera's frame to its Cartesian coordinates in the frame of the base link of the robotic manipulator is introduced in this section.

1) Pixels to Cartesian Coordination Transformation

This section will introduce how to get the target's Cartesian coordinates from the video's pixel coordination. A screenshot of YOLOv5 object detection video is shown in Fig. 3.12. First, the center pixel coordinate of the target will be calculated based on (3.8).

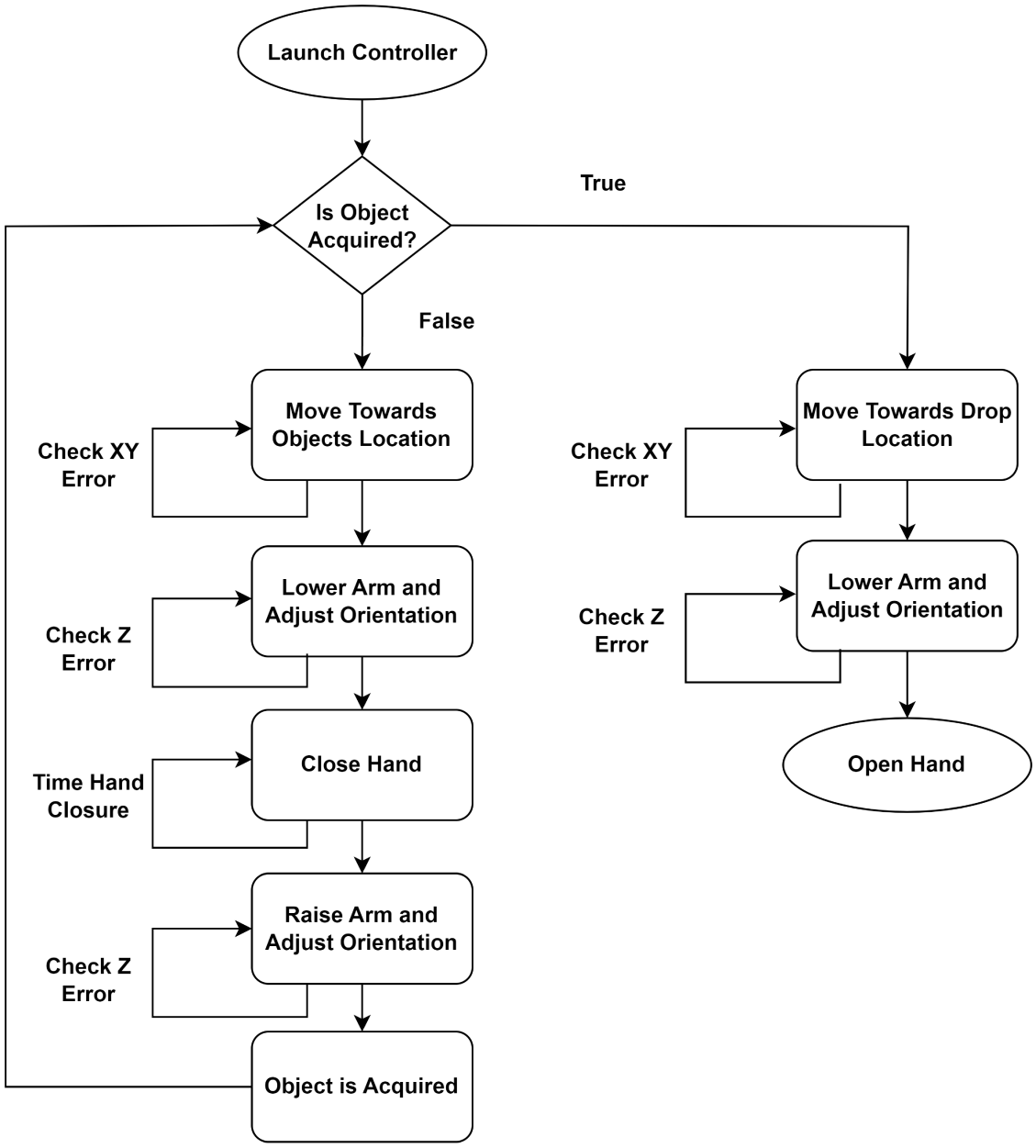

Figure 3.11: The flowchart for the pick-and-place task

$$\begin{bmatrix} P_{cx} \\ P_{cy} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} P_{leftx} + P_{rightx} \\ P_{topy} + P_{bottomy} \end{bmatrix}, \tag{3.8}$$

where P_{cx} and P_{cy} are the X-coordinate and Y-coordinate of the center of the object detection box, labeled red in Fig. 3.12. P_{leftx} and P_{rightx} are X-coordinates for the left and right edge of the object detection box, respectively. P_{topy} and $P_{bottomy}$ are Y-coordinates for the top and bottom edges of the object detection box, respectively.

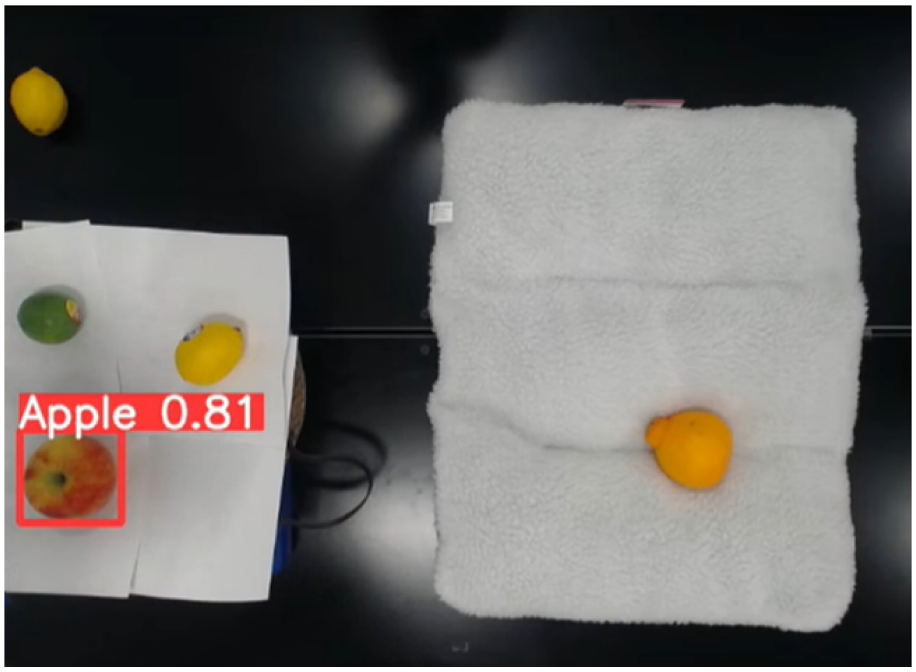

Figure 3.12: A screen shot of the YOLO object detect video

The custom pixel resolution is set as 640×480 for Logitech web camera, and the table's dimension under the camera's view is measured as $1.07 \times 0.8025m$. With those parameters, (3.9) is used to calculate the target's Cartesian coordination under the camera's frame.

$$\boldsymbol{P_{cartesian}} = \frac{(\boldsymbol{P_{c(pixel)}} - \frac{1}{2}P_{max(pixel)})P_{max(Cartesian)}}{P_{max(pixel)}}, \tag{3.9}$$

where $\boldsymbol{P_{cartesian}}$ is the Cartesian coordination in the camera's frame. $\boldsymbol{P_{c(pixel)}}$ is the pixel's coordinate of the target's center. $P_{max(pixel)}$ is the maximum pixel range for the web camera, which is $\{640, 480\}$. $P_{max(Cartesian)}$ is the maximum Cartesian range, which is $\{1.07, 0.8025\}m$.

2) Cartesian Coordination Transformation

To let the camera's Cartesian coordination system match the robotic manipulator's coordination frame, the following equation is used

$$\boldsymbol{P_m} = \boldsymbol{P_c} \cdot \boldsymbol{T_A} + \boldsymbol{T_B}, \tag{3.10}$$

where $\boldsymbol{P_m}$ and $\boldsymbol{P_c}$ are the coordinates of robotic manipulator and camera's Cartesian coordinates frame, respectively. $\boldsymbol{T_A}$ and $\boldsymbol{T_B}$ are the transformation vectors. Since the

XY-plane of the camera is mirrored compared with the manipulator's XY-plane, $\boldsymbol{T_A}$ is set as $\{-1, -1, 0\}$. $\boldsymbol{T_B}$ is used to adjust the camera's position relative to the centroid's position of the base link, link 1, of the manipulator and is set as $\{0.55, -0.14, 0.5\}$.

3.3.7 Orientation Transformation

As shown in Fig. 3.13, it is hard to grasp fruit with the default end-effector orientation, and note that the soft hand has no joint at the wrist. To simulate the natural grasping behavior of humans, the end-effector needs to be rotated by certain angles about the XYZ-axes. Based on several experiments, the suitable rotating vector around the X-axis, Y-axis, and Z-axis, respectively, in degrees is set as $\{170, 80, 30\}$.

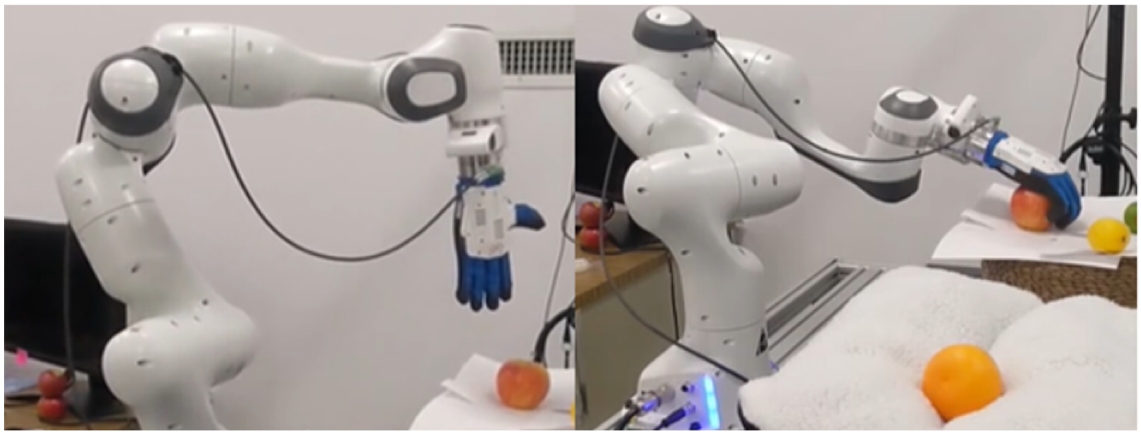

Figure 3.13: Default and pick-and-place orientation for the manipulator

There are two methods to express orientation coordinate, Euler angles and quaternion. Euler angles, which represent three-dimensional orientations via three specific rotations (such as roll, pitch, and yaw), can be converted to a quaternion format to efficiently calculate 3D rotations which is easier to be understand by human. A quaternion, defined as $q = w + xi + yj + zk$ where w, x, y, z are real numbers and i, j, k are the fundamental quaternion units, is a mathematical structure used extensively for calculations involving three-dimensional rotations. To express this desired orientation in Franka-Ros, the Euler angle needs to be converted to a quaternion.

The method introduced in [9] is used to convert the comfortable pick-and-place Euler angle to the quaternion value $\{0.23, 0.723, 0.252, 0.601\}$.

3.3.8 Gazebo Simulation

The controller for pick-and-place tasks is tested and the controller parameters are tuned in the Gazebo simulation environment before applying it to the hardware setup. A urdf/xacro-based robotic manipulator model provided by franka-ros is used for the Gazebo simulation. Fig. 3.14 shows the Franka Emika robot in the Gazebo environment.

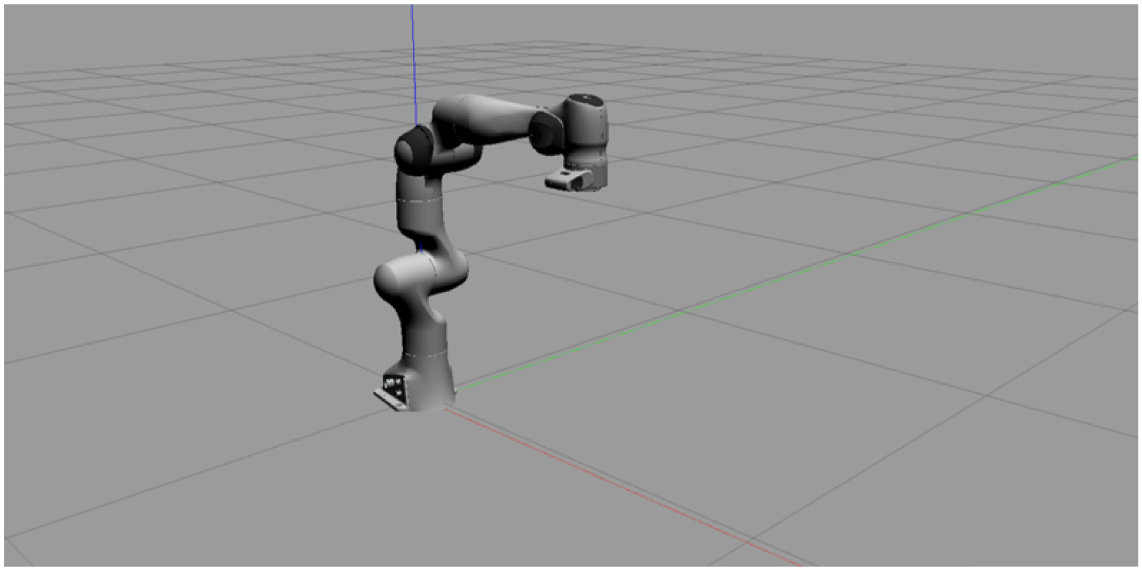

Figure 3.14: The Franka Emika robot in Gazebo

3.3.9 Experiment Execution

Pick-and-place tasks were executed for two different fruits, apple and orange, in different locations to test the performance of the developed framework. The unit of coordinates introduced in this paragraph is meter, and they are considered in the Cartesian coordination frame of the base link of the robotic manipulator. The XY-coordinates are $\{-0.44, -0.32\}m$ for the location of the apple, labeled as the blue point, and $\{-0.6, -0.34\}m$ for the orange, labeled as the orange point. The XY-coordinates of the drop location, labeled as the green point, are set as $\{0.6, -0.2\}m$. Fig. 3.15 shows the points of interest in the overhead camera's view.

The initial position and orientation are set as $\{0.555, 0, 0.521\}m$ and $\{0, 1, 0, 0\}m$, respectively for the end-effector. The impedance parameters are set as $K_t = 525N/m$, $K_r = 52.5N/m$, $B_t = 35N \cdot s/m$, and $B_r = 11N \cdot s/m$. During the execution of the

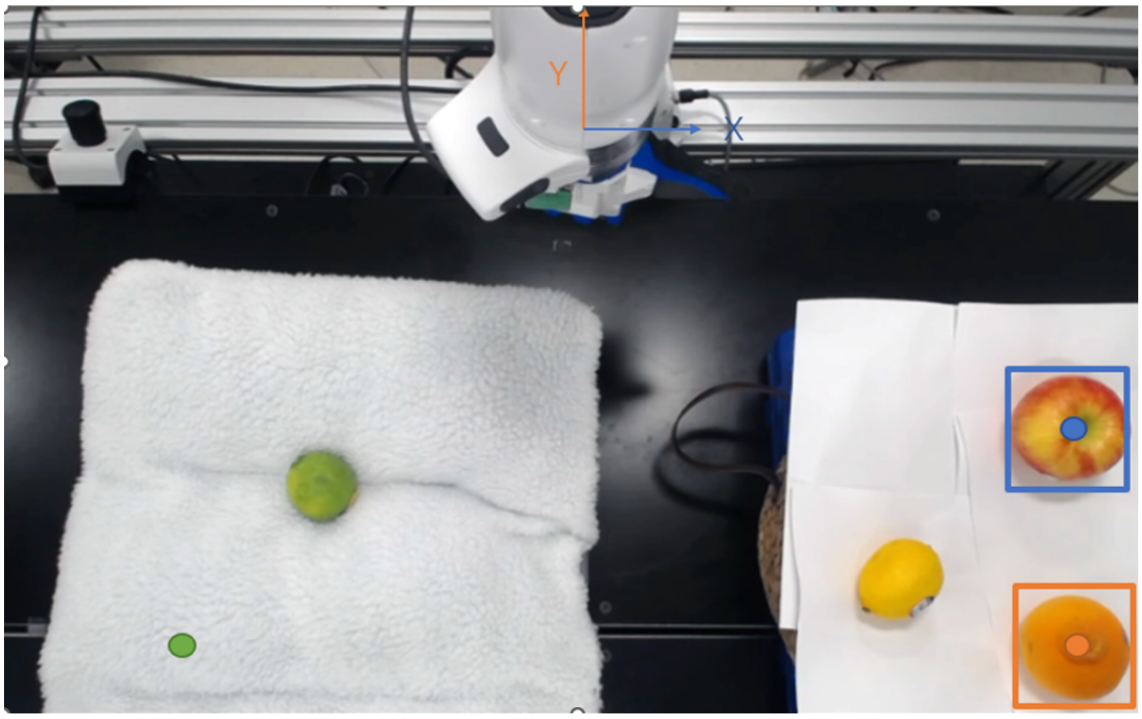

Figure 3.15: Important locations under camera view

code, the qb-SoftHand will perform the process which is introduced in the section task execution and move the targeted fruit to the drop location.

Fig. 3.16A shows the end-effector in the initial position. Fig. 3.16B shows the end-effector move towards the object location. Fig. 3.16C shows that the manipulator lowers and changes the orientation of the end-effector. Fig. 3.16D shows the action "grasping".

Fig. 3.17A shows the arm changes its end-effector's orientation to the default orientation to ensure that the object stays in the qb-Softhand. Fig. 3.17B shows that the end-effector moves to the drop location. Fig. 3.17C shows that the manipulator lowers and changes the orientation of the end-effector again. Fig. 3.17D shows the action "place". The same process is done for the orange pick-and-place task to validate the performance of the experimental setup. The full video of the entire pick-and-place task is available at: `https://youtu.be/snimlSePF5Q`.

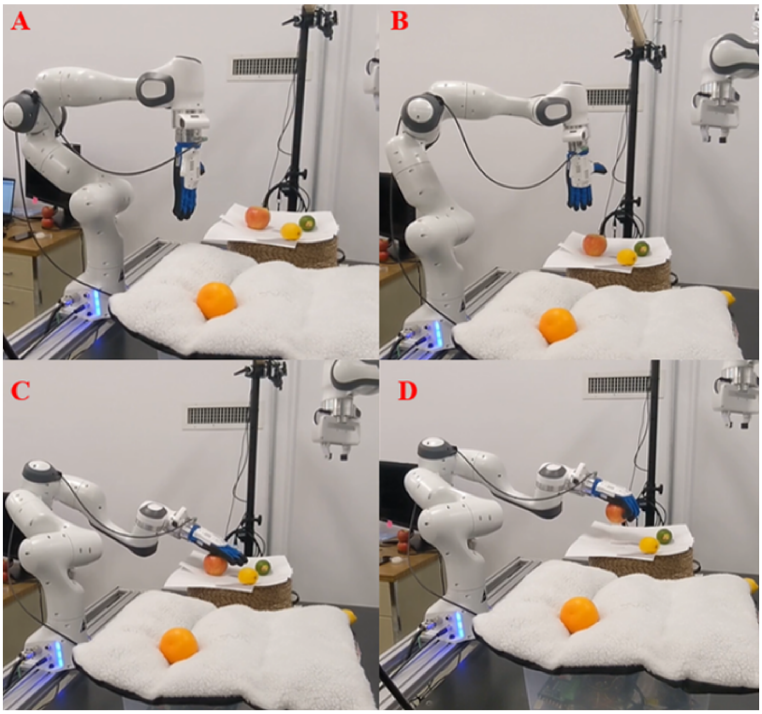

Figure 3.16: Action "Pick" for the apple pick-and-place task

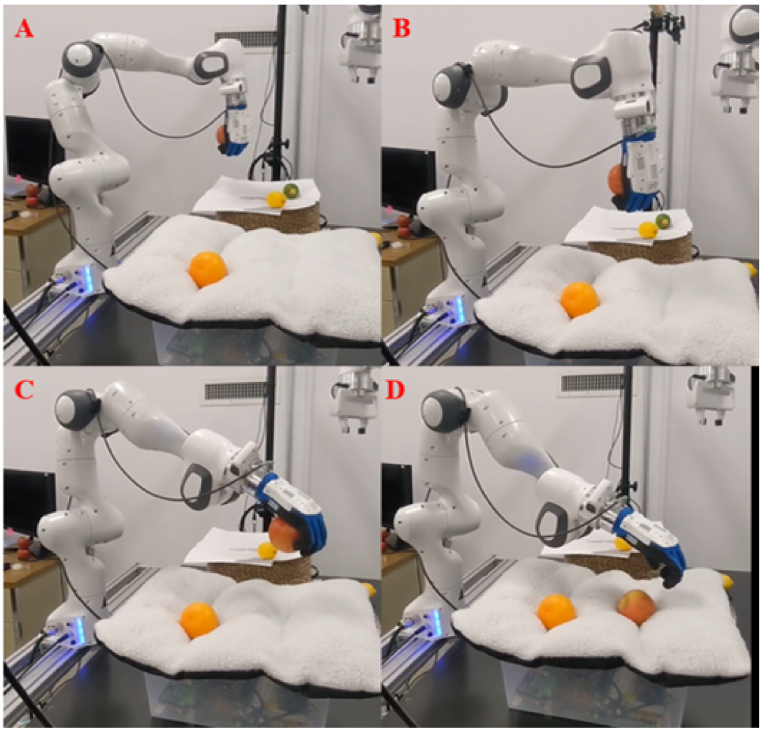

Figure 3.17: Action "Place" for the orange pick-and-place task

Fig. 3.18 shows the recorded end-effector's trajectories for the apple and orange experimental pick-and-place tasks. The trajectory plots show that both trials follow similar paths. The sharp motion in the initial approach to the object in the apple pick-and-place is due to a misdetection of the manipulator for an apple. This can be improved in the future by providing more training images to the YOLOv5 model and improving the object detection process.

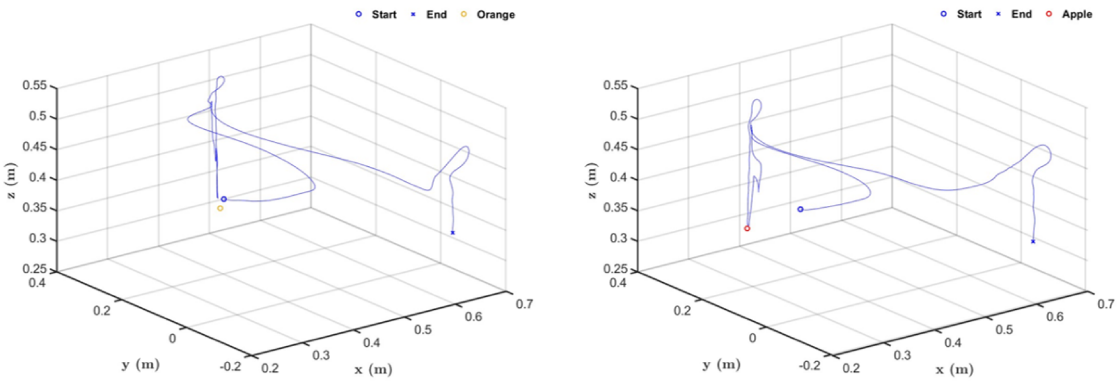

Figure 3.18: The end-effector's trajectories for the pick-and-place task.

3.4 Summary

This section presents a flowchart for integrating an intelligent manipulator system for pick-and-place tasks using a vision-based impedance control method. An experimental setup using a robotic manipulator, a robotic hand, and a camera for object detection is performed. However, there are limitations to the methodology and setup. The limitations include low accuracy of the vision module, limited applicability of the object detection module to 2D environments, and difficulties in task execution planning due to complex error-checking and encoder systems. To address these limitations, the next chapter introduces a more accurate vision system and a simpler grasping plan.

Chapter 4

Object Detection and Localization Module

As mentioned earlier, the selected calculation method for the vision modules was found to be inaccurate, which in turn affected the determination of the target and obstacle coordinates. This chapter discusses the development of more accurate vision modules for a manipulator pick-and-handover task. The first part focuses on a 2D vision module, while the second part introduces a 3D vision module for more complex environments.

4.1 Background Theory

The dimensional estimations are converted from pixel to millimeter using the formula:

$$l = nd\tan(\frac{\theta}{N}), \tag{4.1}$$

where l is the real length of the line in mm, n is line's length in pixels in the picture, d is the distance between camera and object, θ is the view angle of the camera, and N is the diagonal resolution of camera.

Based on the Logitech Pixel-Perfect technology, each pixel in an image represents the same distance in the real world. As shown in Fig. 4.1, $\frac{\theta}{N}$ represents the angle of one pixel in the image, and $d\tan(\frac{\theta}{N})$ represents the real length of a pixel. Therefore, the real length of an object, l, can be calculated by multiplying the number of pixels, n, by the real length of a pixel.

4.2 Problem Formulation

To verify the accuracy of the vision module and apply it to real-time situations, the equation will be modified to include the YOLOv5 object detection box in this chapter. A manipulator pick-and-handover task is performed to demonstrate the feasibility of

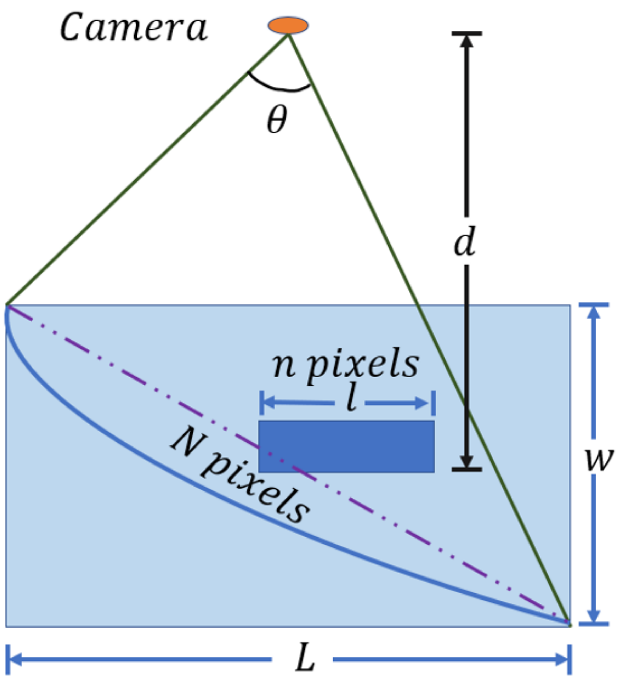

Figure 4.1: Webcam view field diagram

the 2D vision module. The next chapter presents the task used to verify the 3D vision module, which involves the trajectory planning algorithm.

4.3 Proposed Algorithm

4.3.1 One-Camera Based 2D Vision Module

Using the YOLOv5 algorithm with a single camera and a fixed Z-coordinate, the 3D coordinates of an object can be determined. The camera is placed at the top of the workspace, and the algorithm identifies different objects, encompassing their positions in a bounding box and returning them in pixel values. With the known view angle, diagonal resolution of the camera, and distance between the camera and the object, the relative X- and Y-coordinates between the center of the bounding box and the camera can be calculated. The relative distance between the camera and the original point of the manipulator is known, so the coordinates of the detected object in the manipulator's frame can be calculated by using the transformation matrix.

The distances between the center line in X- and Y-axes of the camera and object, C_x and C_y are calculated as:

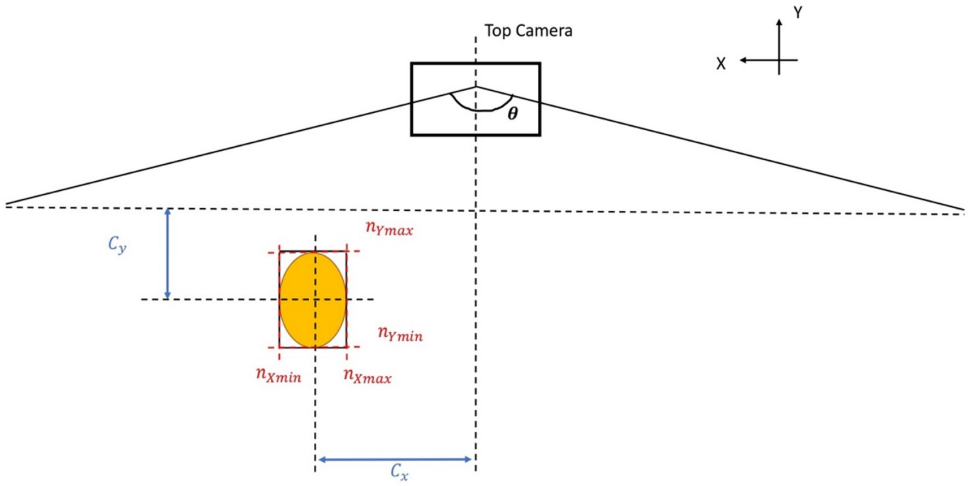

Figure 4.2: Schematic dimensions of the top view

$$C_x = d\left(\frac{N_1 - (n_{X_{min}} + n_{X_{max}})}{2}\tan\left(\frac{\theta}{N}\right)\right), \tag{4.2}$$

$$C_y = d\left(\frac{N_2 - (n_{Y_{min}} + n_{Y_{max}})}{2}\tan\left(\frac{\theta}{N}\right)\right), \tag{4.3}$$

where N_1 and N_2 are the horizontal and vertical dimensions of the camera's frame in pixels, $n_{\min}$ and $n_{\max}$ are the pixel values of the detection box edges in X- and Y-axes, θ is the view angle of the camera, and N is the diagonal dimension of the camera's frame in pixels.

The X, Y and Z distances between the detected object's center and the camera are represented as:

$$\boldsymbol{P_C^O} = \begin{bmatrix} C_x & C_y & C_z \end{bmatrix}, \tag{4.4}$$

where C_z is a known constant with a preset height of the location of the object.

To calculate the object's position in the manipulator's reference frame, the following equation is applied:

$$\boldsymbol{P_O^M} = \boldsymbol{P_C^O} + \boldsymbol{P_M^C}, \tag{4.5}$$

where $\boldsymbol{P_O^M}$ is the object's position in the manipulator's reference frame and $\boldsymbol{P_M^C}$ is the camera's position in the manipulator's reference frame.

4.3.2 Two-Camera Based 3D Vision Module

The localization of an object in the 3D space can be achieved with two cameras and the YOLO algorithm. As shown in Fig. 4.3, one camera is placed at the top of the workspace, and another camera is placed at the side of the workspace. The position of the side camera is adjusted so that it is at the center of the left boundary of the top camera's frame. A YOLO algorithm is deployed on each camera. The YOLO algorithm can identify the object in the frame, encompass the object with a bounding box, and return the dimensions and the position of the bounding box in pixels. With known view angle and position of the cameras, the distance between the center of the bounding box and the cameras can be determined, so that the position of the object can be calculated and sent to the robotic arm. Figs. 4.3 and 4.4 show the schematics of the side and top view of the experimental setup. The distances between the center

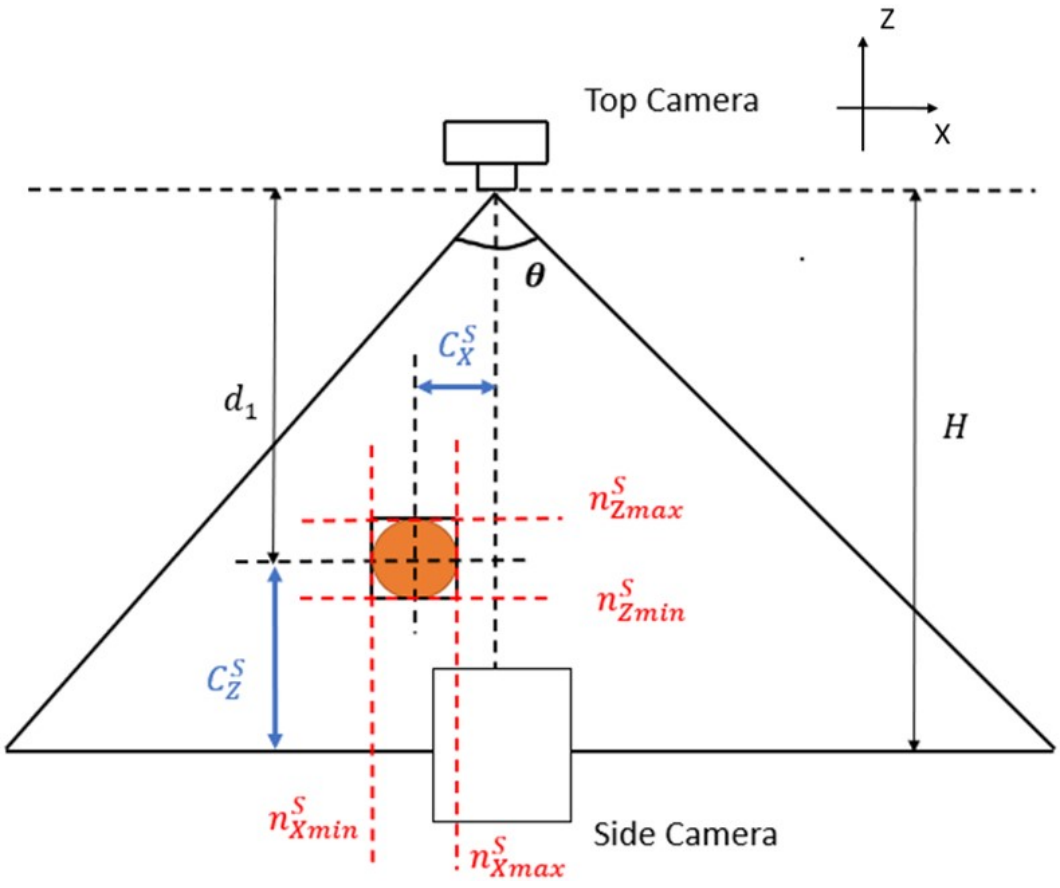

Figure 4.3: Schematic of the side view

line in X-, Y-, and Z-axes of the side camera and top camera and object, $C_\bullet^*$ where

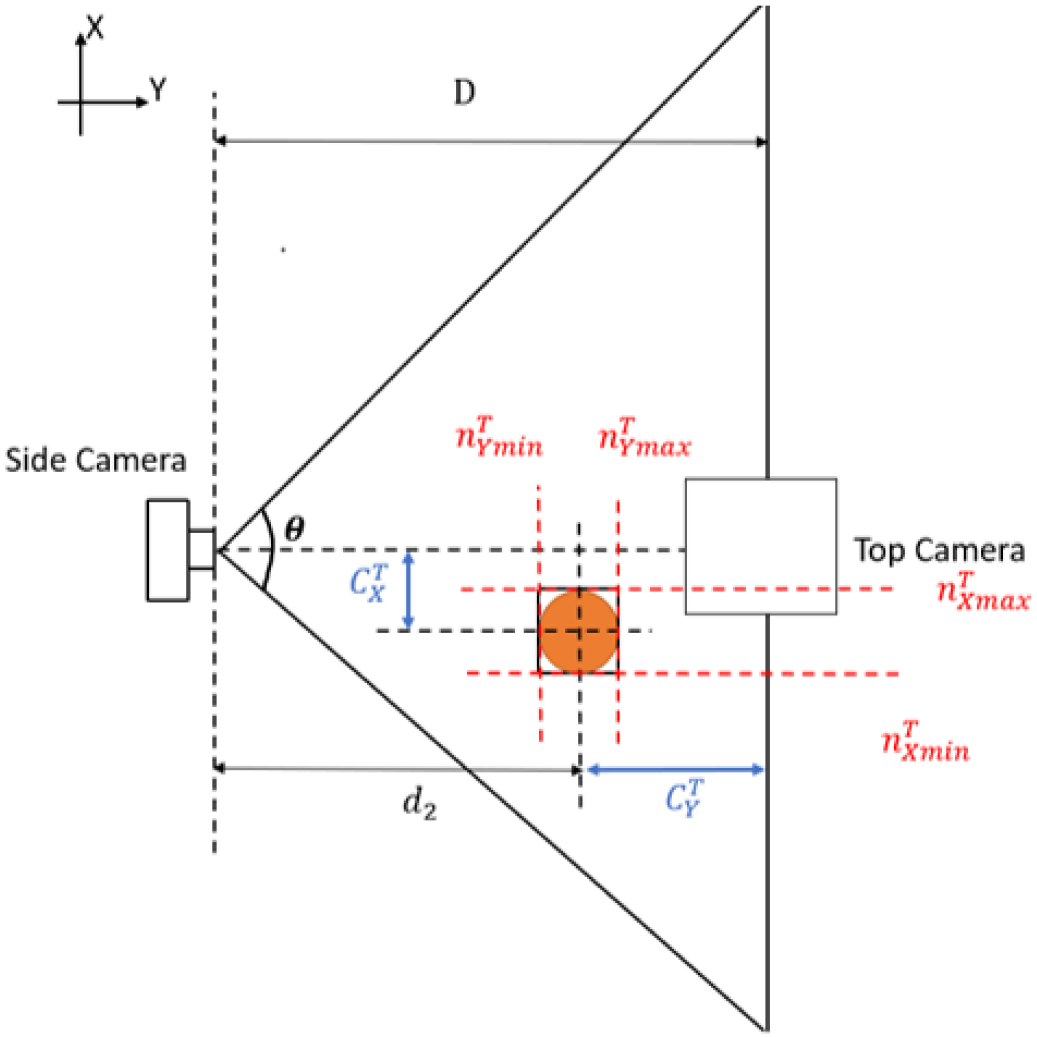

Figure 4.4: Schematic of the top view

$* = T, S$ for top and side and $\bullet = X, Y, Z$, can be calculated as

$$C_X^T = \frac{N_1 - (n_{X_{min}}^T + n_{X_{max}}^T)}{2} d_1 \tan(\frac{\theta}{N}), \tag{4.6}$$

$$C_Y^T = \frac{N_2 - (n_{Y_{min}}^T + n_{Y_{max}}^T)}{2} d_1 \tan(\frac{\theta}{N}), \tag{4.7}$$

$$C_X^S = \frac{N_2 - (n_{X_{min}}^S + n_{X_{max}}^S)}{2} d_2 \tan(\frac{\theta}{N}), \tag{4.8}$$

$$C_Z^S = \frac{N_1 - (n_{Z_{min}}^S + n_{Z_{max}}^S)}{2} d_2 \tan(\frac{\theta}{N}), \tag{4.9}$$

where N_1 and N_2 are the horizontal and vertical dimensions of the top and side cameras' frame in pixels, $n_{\bullet_{min}}^*$ and $n_{\bullet_{max}}^*$ are the pixel values of the detection box edges in X-, Y-, and Z-axes in side and top camera's frame, θ is the view angle of the cameras, and N is the diagonal dimension of the top and side cameras' frame in pixels. Note that the distance in X-axis can be determined by either camera in (4.6) and (4.8). The distances between the object and top camera, d_1, and side camera, d_2 can be calculated as:

$$d_1 = H + C_Z^S, \tag{4.10}$$

$$d_2 = D + C_Y^T. \tag{4.11}$$

Substituting 4.7 and 4.9 into 4.10 and 4.11, d_1 and d_2 can be calculated by:

$$d_1 = H + \frac{N_1 - (n_{Z_{min}}^S + n_{Z_{max}}^S)}{2} d_2 \tan(\frac{\theta}{N}), \tag{4.12}$$

$$d_2 = D + \frac{N_2 - (n_{Y_{min}}^T + n_{Y_{max}}^T)}{2} d_1 \tan(\frac{\theta}{N}), \tag{4.13}$$

where H and D are the distances between two cameras in X and Z directions, respectively. By combining (4.6)–(4.13), the X, Y, and Z distances between the object's center and the top camera, $\boldsymbol{P_O^C}$, can be calculated as:

$$d_x = \frac{N_1 - (n_{X_{min}}^T + n_{X_{max}}^T)}{2} d_1 \tan(\frac{\theta}{N}), \tag{4.14}$$

$$d_y = \frac{N_2 - (n_{Y_{min}}^T + n_{Y_{max}}^T)}{2} d_1 \tan(\frac{\theta}{N}), \tag{4.15}$$

$$d_z = d_1, \tag{4.16}$$

$$\boldsymbol{P_O^C} = \begin{bmatrix} d_x, & d_y, & d_z \end{bmatrix}. \tag{4.17}$$

To calculate the object's position in the manipulator's reference frame, the following equation is applied

$$\boldsymbol{P}_{\boldsymbol{O}}^{\boldsymbol{M}} = \boldsymbol{P}_{\boldsymbol{C}}^{\boldsymbol{M}} + \boldsymbol{P}_{\boldsymbol{O}}^{\boldsymbol{C}}, \tag{4.18}$$

where $\boldsymbol{P}_{\boldsymbol{O}}^{\boldsymbol{M}}$ is the object's position in the manipulator's reference frame and $\boldsymbol{P}_{\boldsymbol{C}}^{\boldsymbol{M}}$ is the top camera's position in the manipulator's reference frame.

4.4 Results

This section introduces the experiments used to verify the 2D and 3D vision modules.

4.4.1 One-Camera Based 2D Vision Module

Fig. 4.5 showcases the nine separate test points within the camera's field of view for a single object used in the experiment. These test points encompass corners, edges, and the center within the camera's view.

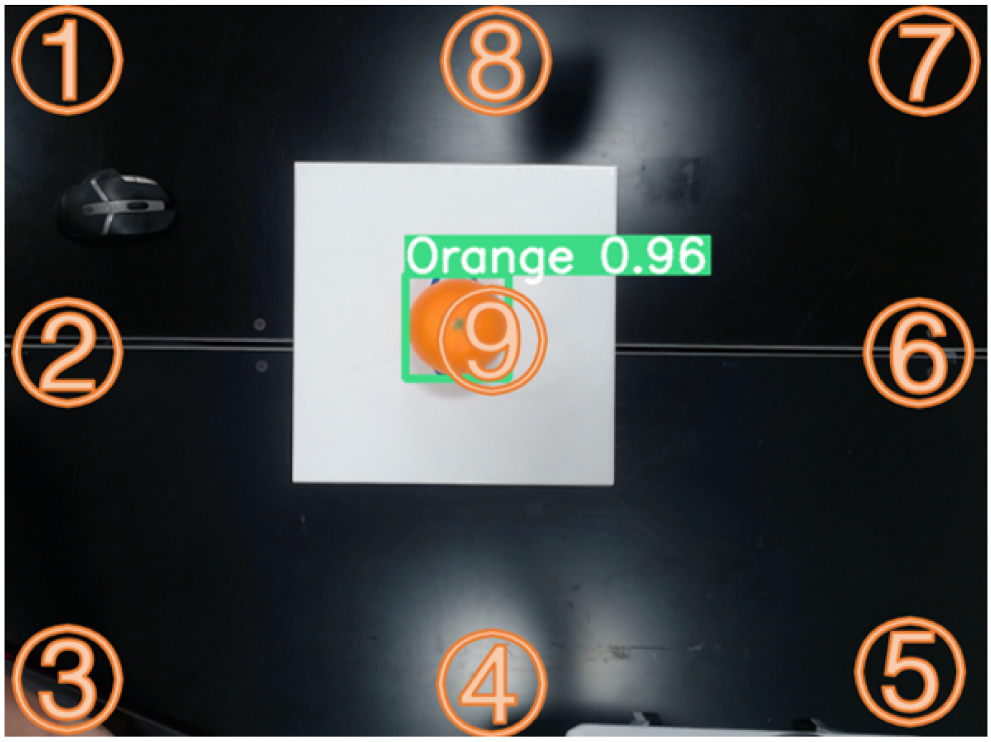

Figure 4.5: A top view for the different test points

Table 4.1 shows the experimental data for calculating the position of an object,

including the calculated distance between the camera and the object, and the pixel information of the bounding box.

Table 4.2 shows the measured distance from the object to the camera, the distance calculated by the vision module in the frame of the camera, the absolute error and the relative error. The absolute error between measured and calculated distances is calculated as:

$$e_{absolute} = |d_{measured} - d_{calculated}|, \tag{4.19}$$

The relative error between measured and calculated distances is calculated as:

$$e_{relative} = \frac{|d_{measured} - d_{calculated}|}{d_{measured}} \times 100\%, \tag{4.20}$$

where $e_{absolute}$ and $e_{relative}$ are absolute and relative errors between measured and calculated distances, respectively. $d_{measured}$ and $d_{calculated}$ are measured and calculated distances from the object to the camera, respectively.

The maximum error is $0.013m$, which is suitable for the manipulator pick-and-handover situation. The error may be caused by: 1) The inaccuracies of the bounding boxes, which can be improved by using a polygon bounding box during the training process. 2) The orientation of the camera is not completely parallel with the work bench.

Location	Pixel Information (X_min, X_max, Y_min, Y_max)	Calculated Distance (m) (X, Y)
1	(568, 640, 414, 480)	(0.348, 0.254)
2	(567, 640, 207, 273)	(0.347, 0)
3	(568, 640, 0, 67)	(0.348, -0.253)
4	(288, 351, 0, 68)	(0, -252)
5	(0, 76, 0, 65)	(-0.349, -0.254)
6	(0, 74, 207, 273)	(-0.347, 0)
7	(0, 75, 416, 480)	(-0.347, 0.255)
8	(287, 350, 412, 480)	(0.001, 0.252)
9	(288, 350, 207, 274)	(0.001, 0)

Table 4.1: Pixel information and calculated distance (m)

Point	Calculated Distance (m) (X, Y)	Measured Distance (m) (X, Y)	Absolute Error (m) (X, Y)	Relative Error (%) (X, Y)
1	(0.348, 0.254)	(0.355, 0.242)	(0.007, 0.012)	(1.97, 4.96)
2	(0.347, 0)	(0.355, 0.00)	(0.008, 0.00)	(2.25, 0.00)
3	(0.348, -0.253)	(0.355, -0.242)	(0.007, 0.011)	(1.97, 4.55)
4	(0, -252)	(0, -0.242)	(0.00, 0.01)	(N/A, 4.13)
5	(-0.349, -0.254)	(-0.355, -0.242)	(0.006, 0.012)	(1.69, 4.96)
6	(-0.347, 0)	(-0.355, 0)	(0.008, 0.00)	(2.25, 0.00)
7	(-0.347, 0.255)	(-0.355, 0.242)	(0.008, 0.013)	(2.25, 5.37)
8	(0.001, 0.252)	(0, 0.242)	(0.001, 0.01)	(N/A, 4.13)
9	(0.001, 0)	(0,0)	(0.001, 0)	(N/A, N/A)

Table 4.2: Absolute error (m) and relative error (%)

4.4.2 Two-Camera Based 3D Vision Module

This part of experiment is used to verify the 3D vision module. Fig. 4.6 shows the screenshots of YOLO systems running with top and side cameras. The object is identified and bounded by each YOLO algorithm. Each YOLO algorithm will return the pixel location of bounding box centers in the camera's frame to the computer. Using (4.6)–(4.18) listed in previous sections, the coordinates of the object are determined.

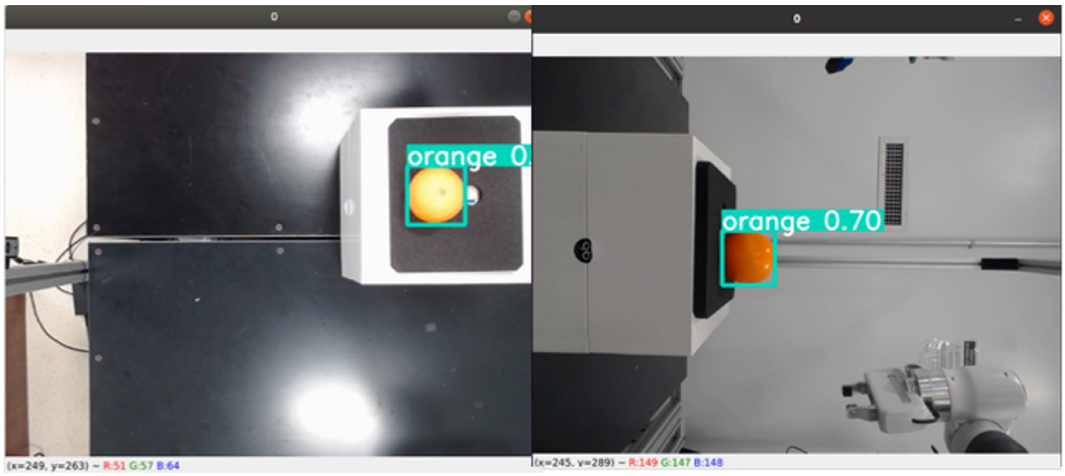

Figure 4.6: Screenshots of YOLO system from top and side cameras

Table 4.3 presents the validation data for calculating the position of an object, including the the measured distance between the top camera and the object, the distance between the top camera and the object calculated by our system, and the absolute error. As shown in Table 4.3, the maximum error is 1.9 cm and the maximum

Table 4.3: Object positions with reference to the top camera $[x,\ y,\ z]$

Measured (cm)	Calculated (cm)	Absolute Error (cm)
$[-9.5,\ 23.2,\ 93.3]$	$[-9.3,\ 21.2,\ 91.9]$	$[0.2,\ 0.1,\ 0.4]$
$[7.5,\ 32.1,\ 93.5]$	$[8.0,\ 34.0,\ 92.4]$	$[0.5,\ 1.9,\ 0.9]$
$[0.0,\ 24.1,\ 93.5]$	$[0.0,\ 24.7,\ 92.1]$	$[0.0,\ 0.6,\ 1.4]$
$[0.8,\ 29.8,\ 74.4]$	$[0.3,\ 28.9,\ 74.2]$	$[0.5,\ 0.9,\ 0.2]$
$[-9.0,\ 1.9,\ 74.4]$	$[-8.7,\ 1.1,\ 74.1]$	$[0.3,\ 0.8,\ 0.3]$
$[8.5,\ 1.2,\ 74.6]$	$[8.0,\ 1.0,\ 74.2]$	$[0.5,\ 0.2,\ 0.4]$

percentage error is under 1.5% in Z-axis, which is suitable for this application. Some errors may be due to the inaccuracies of the bounding boxes, which can be improved by additional training of the YOLO algorithm.

4.5 Experiment

Pick-and-handover tasks are executed for two different fruits (orange and apple) in different locations to test the performance of the developed framework. The unit of coordinates introduced in this paragraph is meter, and they are considered in the Cartesian coordination frame of the base link of the robotic manipulator. The parameters used for the vision system are $N_1 = 640$, $N_2 = 480$, $N = 800$, and $\theta = 78°$. The impedance parameters with proper units are set as $K_t = 525N/m$, $K_r = 52.5N/m$, $B_t = 35N \cdot s/m$, and $B_r = 11N \cdot s/m$. During the execution of the code, the robot can perform the following process: i) identify the location of the fruit; ii) pick up the fruit; iii) handover the fruit to the location of the user's hand.

Fig. 4.7 and 4.8A show the end-effector in the initial position. Fig. 4.7 and 4.8B show the end-effector grasping the object. Figs. 4.7 and 4.8C show that the manipulator moves to the suitable moving orientation. Figs. 4.7 and 4.8D show the action "handover". The same process is executed to handover an apple where the user's hand is in a different location after the orange handover. The same process is done for the apple pick-and-handover task to validate the performance of the experimental setup. The videos of the pick-and-handover tasks for orange and apple are available on YouTube: `https://youtu.be/N73M2FvOR84` and `https://youtu.be/JFarYWA5sEM`.

Fig. 4.9 and Fig. 4.10 show the recorded end-effector's trajectories for the orange and apple experimental pick-and-handover tasks, respectively. The trajectory plots

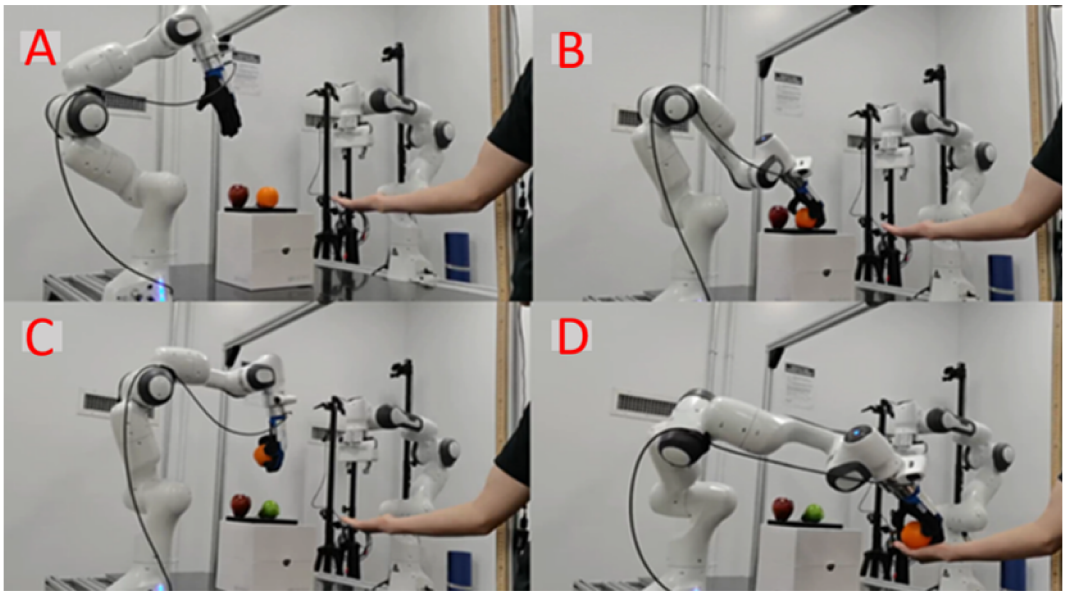

Figure 4.7: Orange pick-and-handover task

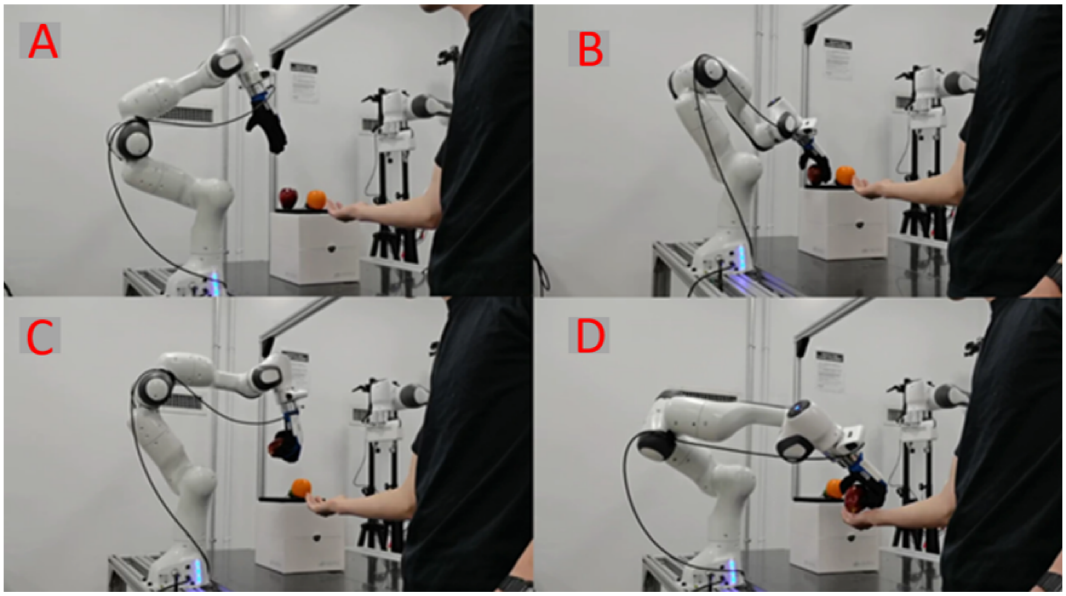

Figure 4.8: Apple pick-and-handover task

show that both trials follow similar paths. Comparing to the previous work, the trajectory of performing task is smoother. However, the turning trajectory can be curved to reduce the wear and tear of the manipulator and improve the stability of the pick-and-handover task performance in the future.

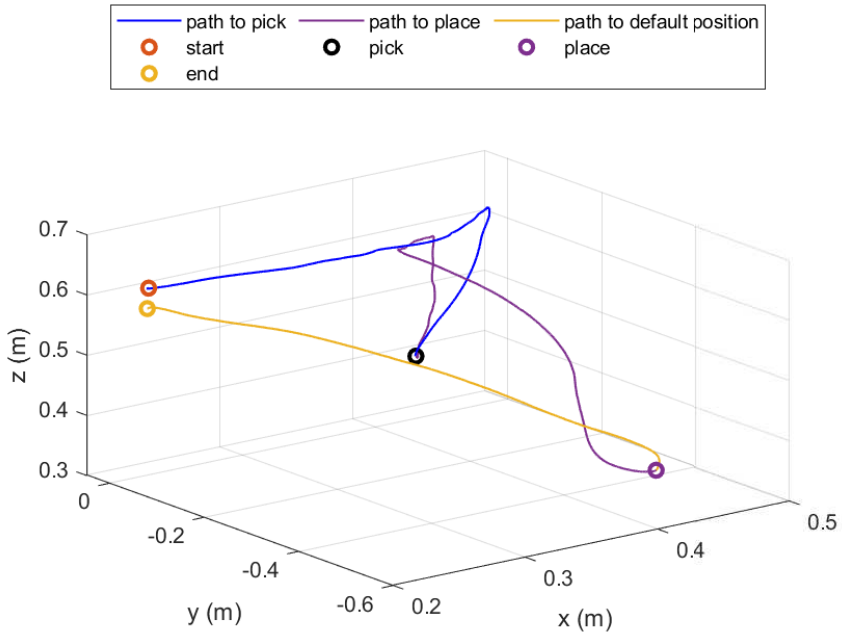

Figure 4.9: The plot of end-effector's position for orange pick-and-handover task

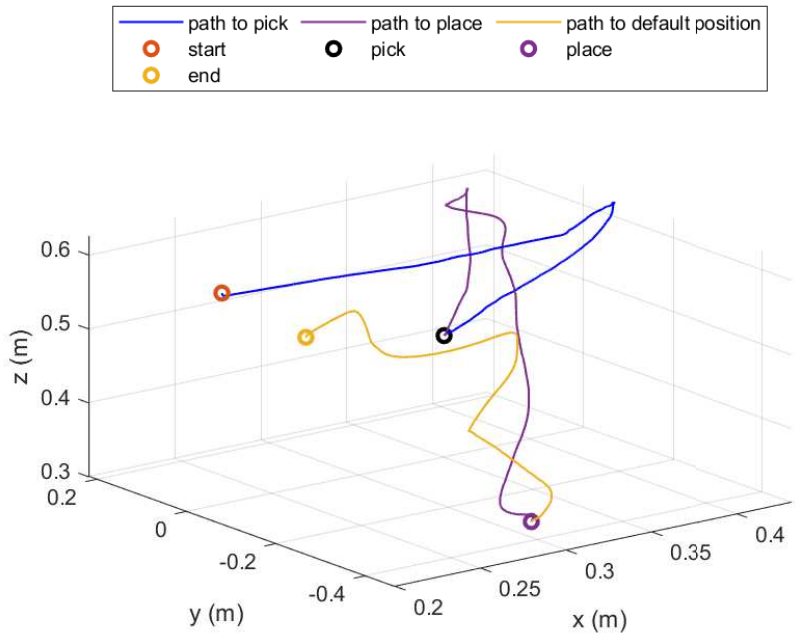

Figure 4.10: The plot of end-effector's position for apple pick-and-handover task

4.6 Summary

In this chapter, the development of accurate vision modules for manipulator pick-and-handover tasks is discussed. A one-camera based 2D vision module is presented, which uses the YOLOv5 object detection algorithm and a single camera to determine the 2D position of an object. The 2D vision module is tested in real-time situations and demonstrates a suitable accuracy for manipulator pick-and-handover tasks. The maximum error in the 2D vision module is 0.013 m, which is acceptable for most pick-and-handover tasks.

Additionally, a two-camera based 3D vision module is introduced for more complex environments. The 3D vision module uses two cameras and the YOLO object detection algorithm to determine the position of an object in 3D space. The experimental results show that the 3D vision module is capable of accurately determining the object's position, with small absolute errors in the X, Y, and Z dimensions.

Two fruit pick-and-handover experiments have been successfully carried out to confirm the efficacy of the 2D vision modules. However, the equal step end-effector trajectories utilized in this chapter are only appropriate for objects that can be

grasped from above, such as oranges and apples. They are not well-adapted for objects necessitating a side grasp, like bottles. To overcome this limitation, a more complex trajectory planning algorithm will be presented in the following chapter. An experiment designed to evaluate the performance of the 3D vision module in the pick-and-handover process will be conducted using this advanced trajectory planning algorithm in chapter 5.

Chapter 5

Proposed Advanced Trajectory Planning Algorithm

In the chapter 4, it was discussed that picking objects with various shapes, such as cups and bottles, using the equal step end-effector trajectory developed by a coder can be challenging. In this chapter, the Difference-based DMPs algorithm will be introduced, which can effectively handle objects with different shapes.

5.1 DMPs Theory

The aim of DMPs is to build an attractor model with a stable second-order system [36]. To change the target trajectory, the final state of the system can be changed by adjusting the attracting point. An n-dimensional DMPs trajectory can be represented by following second-order ordinary differential equations,

$$\zeta \dot{\boldsymbol{v}} = \alpha(\beta(\boldsymbol{g} - \boldsymbol{y}) - \boldsymbol{v}) + f(t), \tag{5.1}$$

$$\zeta \dot{\boldsymbol{y}} = \boldsymbol{v}, \tag{5.2}$$

where ζ is a value-control parameter, $\boldsymbol{g} \in \mathbb{R}^n$ is the goal state, $\boldsymbol{y} \in \mathbb{R}^n$ current state, $\boldsymbol{v} \in \mathbb{R}^n$ is the current velocity, and α, β are constants similar to P and D gains in the PD controller. The f term is a trajectory learning term which is expressed as

$$f(t) = \frac{\sum_{i=1}^{N} \Psi_i(t) w_i}{\sum_{n=i}^{N} \Psi_i(t) w_i}, \tag{5.3}$$

where Ψ_i are the kernel functions, w_i are weight parameters, and N is the number of kernel functions. Using (5.3), a complex trajectory can be described by changing the weight of different kernels. Since the trajectory learning term f is highly dependent on time, it is difficult to directly add other dynamic systems to it or synchronize multiple DOF trajectories. Therefore, a time-independent term $\boldsymbol{x}$ is used to replace the time for discrete type DMPs,

$$\zeta \dot{\boldsymbol{x}} = -\alpha_x \boldsymbol{x}, \tag{5.4}$$

where α_x is a constant. The system will converge to zero for any initial value $\boldsymbol{x_0}$. The parameter α_x will influence the convergence speed of the system. The kernel function Ψ_i is constructed as a radial basis function (RBF),

$$\Psi_i(x) = \exp(-h_i(x - c_i)^2) = \exp(-\frac{1}{\sigma_i^2}(x - c_i)^2), \tag{5.5}$$

where σ_i and c_i represent the width and the center location of the kernel function, Ψ, which are obtained by the demonstration trajectory. To ensure that the forcing term f converges to zero when the state of the system converges to $\boldsymbol{g}$, the expression of f is changed to

$$\boldsymbol{f}(t) = \frac{\sum_{i=1}^{N} \Psi_i(t) w_i}{\sum_{n=i}^{N} \Psi_i(t) w_i} \boldsymbol{x}(\boldsymbol{g} - \boldsymbol{y_0}), \tag{5.6}$$

where $\boldsymbol{y_0}$ is the initial state. In (5.6), $\boldsymbol{x}$ is used to ensure that $\boldsymbol{f}$ is converging to zero with decreasing $\boldsymbol{x}$.

5.2 Problem Formulation

The DMPs algorithm has a primary advantage of generating demonstration-based trajectories with similar shapes for various start and goal points. This feature makes it a useful tool for tasks that require the repetition of the same movement pattern, such as in robotic assembly and manufacturing. However, DMPs may not perform optimally when there is a considerable difference between the execution start and goal points and the demonstrated start and goal points for manipulator pick-and-handover tasks.

Nevertheless, three essential elements are critical in pick-and-handover tasks: 1) Precise positioning and alignment of the end-effector when interacting with the environment, 2) Avoiding collisions during the end-effector's movement, and 3) Guaranteeing the robustness and fluency of the generated trajectory. Considering these challenges, conventional DMPs might struggle to create suitable trajectories. As a result, this section utilizes difference-based DMPs as the trajectory planning algorithm to tackle these issues effectively.

5.3 Proposed Algorithm

As depicted in Fig. 5.1, the first trajectory serves as the demonstration trajectory. The second trajectory, generated by the standard DMPs algorithm, is a generically regenerated trajectory between different starting and ending points, preserving a similar shape. The third trajectory is produced using the difference-based DMPs method. The black straight line illustrates an equal-step trajectory, generated via hard coding to minimize the distance between the starting and ending points. With a reduced difference, the red trajectory, created by the standard DMPs, successfully follows the desired shape in the intended position.

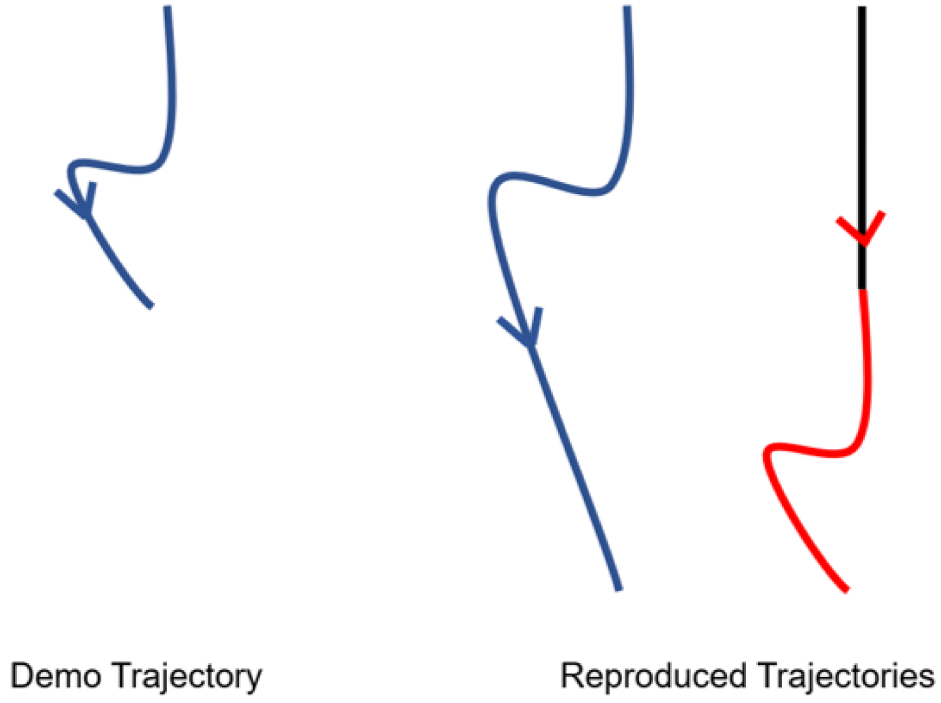

Figure 5.1: An example for demo and reproduced trajectories

In this case, the DMPs trajectory is regenerated based on the current state of the manipulator as follows

$$\begin{aligned} \bar{\boldsymbol{x}}_{\boldsymbol{start}} &= \boldsymbol{x}, \\ \bar{\boldsymbol{x}}_{\boldsymbol{end}} &= \boldsymbol{x} + \boldsymbol{\delta}_{\boldsymbol{DMP}}, \\ \boldsymbol{\delta}_{\boldsymbol{DMP}} &= \hat{\boldsymbol{x}}_{\boldsymbol{end}} - \hat{\boldsymbol{x}}_{\boldsymbol{start}}, \end{aligned} \tag{5.7}$$

where $\bar{\boldsymbol{x}}$, $\boldsymbol{x}$, $\boldsymbol{\delta_{DMP}}$, $\hat{\boldsymbol{x}} \in \mathbb{R}^6$. $\boldsymbol{\delta_{DMP}}$ is the difference between the start coordinate and goal coordinate of the demonstration trajectory. $\boldsymbol{x}$ is the current position of the manipulator, $\bar{\boldsymbol{x}}_{\boldsymbol{start}}$ and $\bar{\boldsymbol{x}}_{\boldsymbol{end}}$ are the start and end positions used to regenerate the DMPs trajectory, $\hat{\boldsymbol{x}}_{\boldsymbol{start}}$ and $\hat{\boldsymbol{x}}_{\boldsymbol{end}}$ are the start and end position of the demonstrated trajectory.

5.4 Experiment Result

5.4.1 Hardware Setup

Fig. 5.2 shows the equipment setup for this section in the Advanced Control and Mechatronics (ACM) Lab at Dalhousie University. As labeled in Fig. 5.2, the system includes a 7-DOF FE Panda robotic manipulator, one qb-SoftHand end-effector, a Logitech C920 HD webcam, and a Logitech C930E HD webcam.

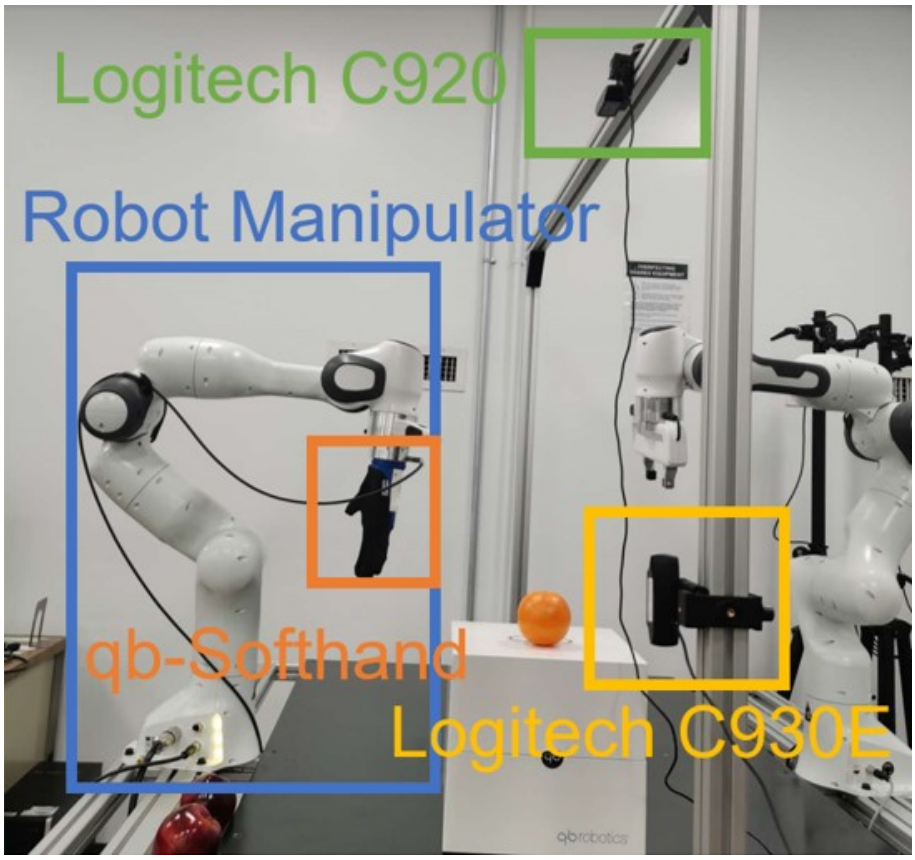

Figure 5.2: The equipment setup the Advanced Control and Mechatronics (ACM) Lab at Dalhousie University

5.4.2 ROS Implementation

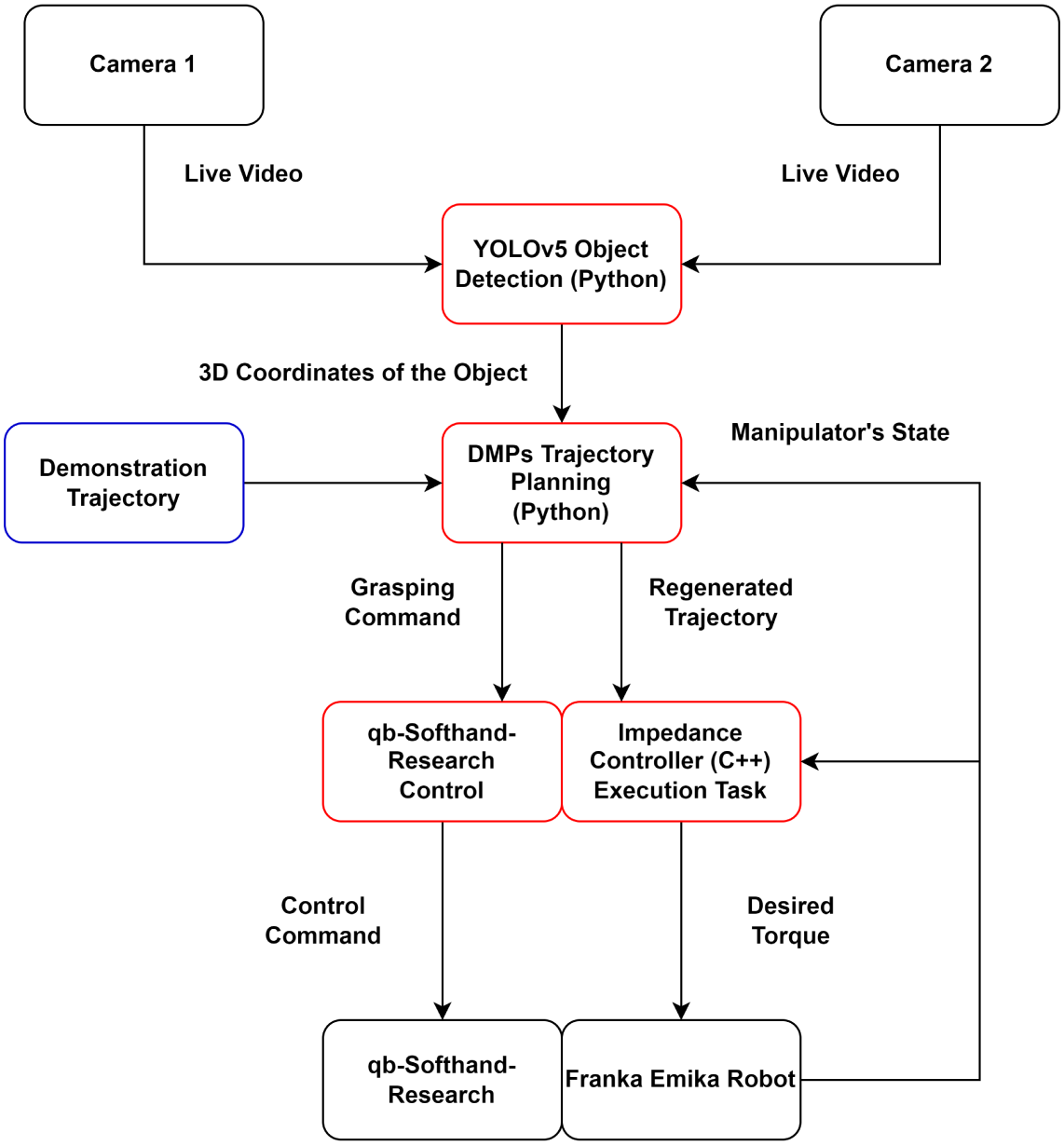

Figure 5.3: ROS block diagram of the experiment

ROS is used as the software to facilitate the communication between the multiple hardware components. Fig. 5.3 shows the ROS block diagram of the experimental framework, where black boxes represent the hardware, red boxes represent the ROS nodes, and blue boxes represent the data. First, two live videos are taken by two webcams and input to the YOLOv5 object detection algorithm. Next, the XYZ-

coordinates of the target object are calculated. The coordinates are input to the DMPs trajectory planning algorithm to generate a desired trajectory based on the current end-effector and object positions. A set of actions is prescribed to perform the pick-and-place task which is shown in the following section. The generated trajectory is then input to the impedance controller. The desired joint torques are commanded to the robot and the manipulator's state is measured by encoders and torque sensors and sent back to the controller to complete the closed-loop control. Simultaneously, open and close commands are sent to the qb-SoftHand to allow the hand to execute the desired actions at the designated times.

5.4.3 Task Execution

The process of the pick-and-place action of the robotic manipulator that is executed within the DMPs trajectory planning node is introduced in this section. First, the manipulator is directed to the X-coordinate of the object. It is intentionally not directed to the Y- and Z-coordinates of the object to avoid collision. Next, the DMPs trajectory is generated using (5.7) and executed to adjust the robotic arm's end-effector to the desired pick-up orientation. Then the manipulator is directed to the Y- and Z-coordinates of the object. When the object position is reached, the grasping action will be performed and the object can be picked up by the end-effector. The end-effector will then move to a suitable moving orientation and the robotic arm will move to the target place location. Once the target location is reached, the end-effector will be adjusted again to a suitable place orientation via the generated DMPs trajectory and drop the object at the desired location. Finally, the manipulator will return to the starting position and orientation.

5.4.4 Experiment Results

Two objects were considered separately in the experimental testing, an orange and a bottle. Considering that the orange must be approached from the top and the bottle must be approached from the side, two demonstration trajectories were recorded to regenerate different end-effector motions for each object. Because the coordinates of the object have already been determined at the beginning of the task, the movement of the robotic arm will not affect the experiment execution. The parameters used

for regenerating the trajectory in (5.6) are $\alpha = 60$, $\beta = 15$, $\alpha_x = 1$, $N = 1000$, $\tau = \frac{1}{N_{demo}}$, and N_{demo} is the number of data points for each demonstration. The impedance parameters are $K_t = 525N/m$, $K_r = 52.5N/m$, $B_t = 35N \cdot s/m$, and $B_r = 11N \cdot s/m$. The top camera's position relative to the manipulator's base is $\boldsymbol{P}_{\boldsymbol{C}}^{\boldsymbol{M}} = [0.64,\ 0.375,\ 1.065]\ m$. The parameter of cameras are $N_1 = 640$, $N_2 = 480$, and $N = 800$.

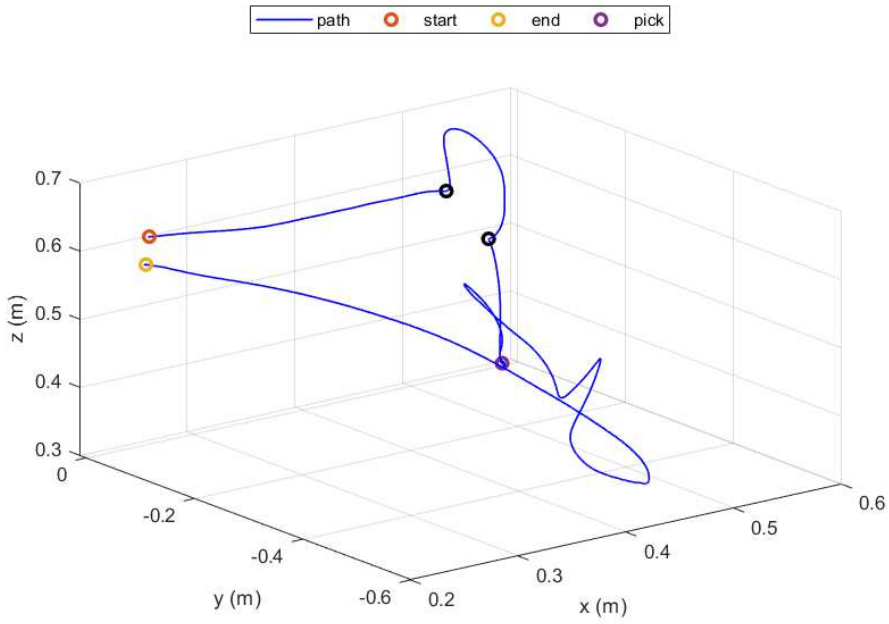

Figure 5.4: The profile of the end-effector position for orange pick-and-place task

Figs. 5.4 and 5.5 show the recorded end-effector's trajectories for the orange and bottle experimental pick-and-place tasks. The trajectories between black dots in Fig. 5.4 and Fig. 5.5 are generated by difference-based DMPs.

Fig. 5.6A shows the end-effector in the initial position. Fig. 5.6B shows the end-effector grasping the object. Fig. 5.6C shows the manipulator passes an orange to a human. Fig. 5.6D shows the manipulator finishes the task and returns to the default position. Fig. 5.7A-D show a similar task with a bottle with a different learned DMPs trajectory where the manipulator approaches the bottle from the side. The video of the pick-and-place task is available at `https://youtu.be/qlepM2Lrlno`.

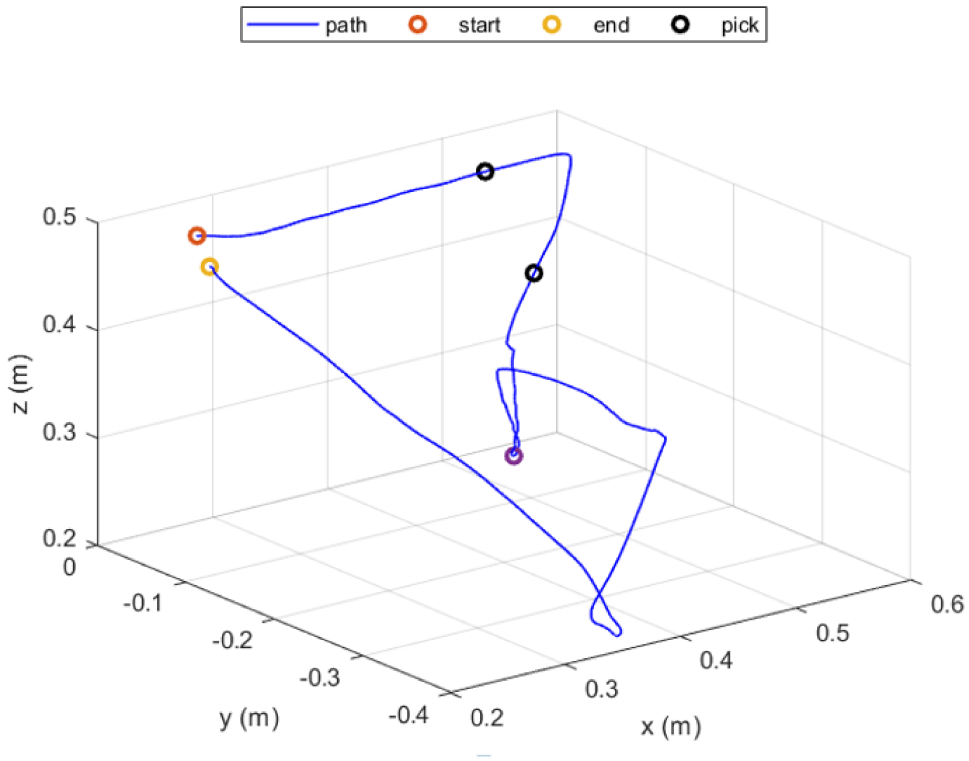

Figure 5.5: The profile of the end-effector position plot for bottle pick-and-place task

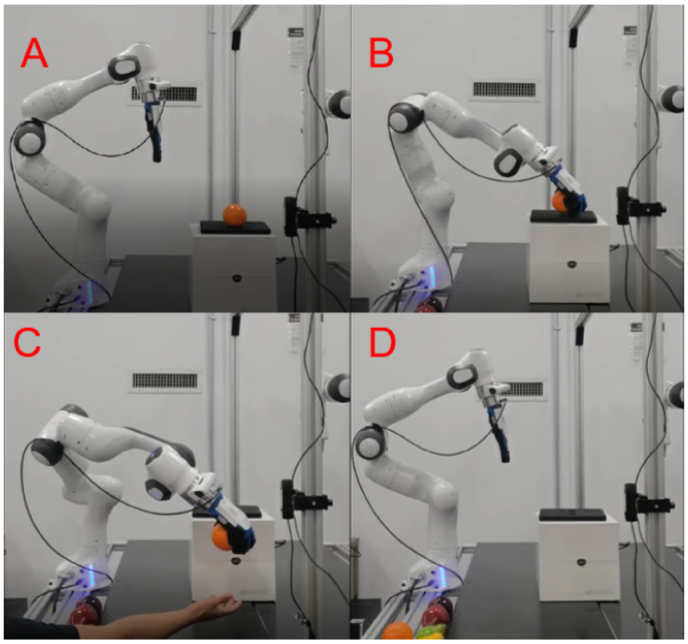

Figure 5.6: Orange pick-and-place task

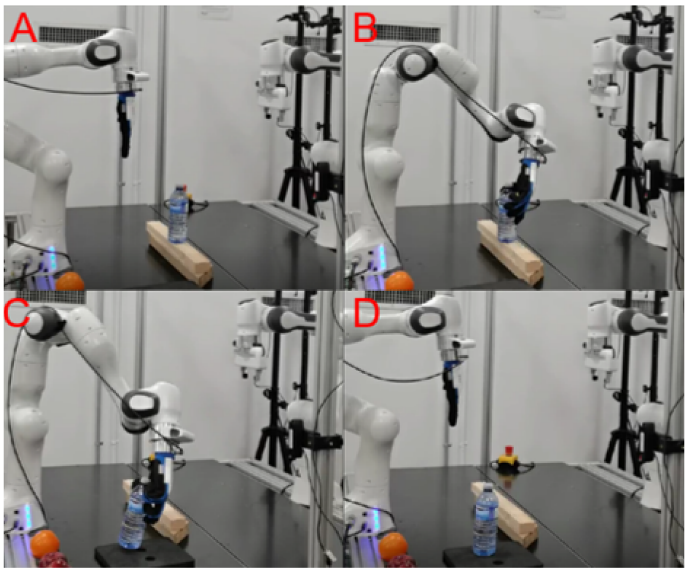

Figure 5.7: Bottle pick-and-place task

5.5 Summary

This chapter presents an intelligent manipulator system that utilizes the YOLOv5 algorithm with low-cost webcams to identify and locate objects in three dimensions with depth errors less than 1.5%. A difference-based DMPs trajectory planning algorithm was designed to autonomously complete a pick-and-place task in a human-like manner using impedance control and a 7-DOF FE robot.

However, the system has some limitations that need to be addressed. One limitation is that the position may not converge due to the impedance controller's property, which can make it challenging for the system to achieve precise positioning. Another limitation of the system is its inability to interact with objects of various weights. This can restrict the range of objects that the system can manipulate, impacting its overall usefulness in certain applications.

To overcome those limitations, a variable-impedance controller will be developed in the next chapter.

Chapter 6

Proposed Variable Impedance Control

6.1 Impedance Control Theory

The reference of this section is [51]. Impedance control is a method of controlling the interaction between a robot and its environment by regulating the robot's impedance or resistance to motion.

In this chapter, we assume that the environment is rigid. The stiffness of the environment is denoted as k_e. The mass, damping factor, and stiffness of the robot end-effector are represented by m, b, and k, respectively. Once a contact between the robot and the environment is established, the current contact force applied by the robot to the environment is denoted by f. The system model of robot and rigid environment is shown in Fig. 6.1. Fig. 6.1 (a) shows the situation without any contact between robot and environment, (b) shows critical point when contact occurs but $f = 0$, and (c) shows contact situation with $f \neq 0$.

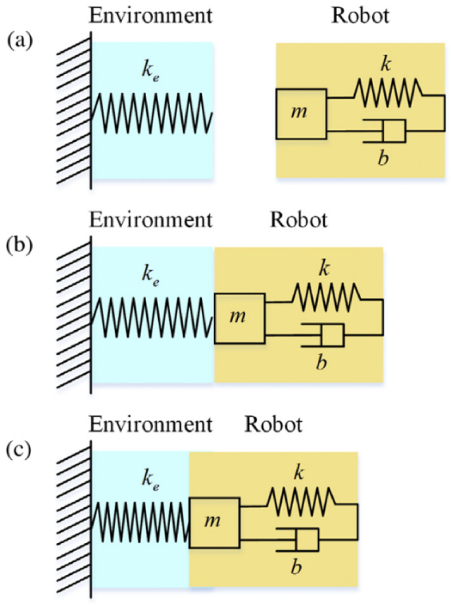

Figure 6.1: System model of the robot and rigid environment

6.2 Problem Formulation

Safety is paramount in the field of human-robot interaction. Traditionally, to protect humans from harm, robotic arms are designed to cease operation abruptly when a human approaches. However, this method is not suitable for scenarios where the robot must interact and collaborate with humans, as it interrupts the continuity of the task.

In these cases, a more complex approach that allows the robot to adapt its behavior based on the proximity and actions of humans is needed. This calls for the integration of variable impedance control into the robotic manipulator. This control scheme enables the robot to determine when it needs to complete tasks with more force or when a gentler, softer approach is necessary. The integration of such a control system will ensure a safe and efficient interaction between human and robot. This section introduces a pick-and-handover experiment to verify the feasibility of the variable impedance control in this scenario.

6.3 Proposed Algorithm Description

Consider the impedance controller introduced in the previous chapter. The controller feedback is designed in Cartesian space because the position of the objects are given in Cartesian space. The control torque is computed as

$$\boldsymbol{\tau} = J^T(-K\tilde{\boldsymbol{x}} - B(J\dot{\boldsymbol{q}})) + C(\dot{\boldsymbol{q}}, \boldsymbol{q})\dot{\boldsymbol{q}}, \tag{6.1}$$

where

$$K = \begin{bmatrix} K_t & 0 \\ 0 & K_r \end{bmatrix}, \tag{6.2}$$

$$B = \begin{bmatrix} B_t & 0 \\ 0 & B_r \end{bmatrix}, \tag{6.3}$$

$\tilde{\boldsymbol{x}} \in \mathbb{R}^6$ contains the Cartesian position and orientation errors, $J \in \mathbb{R}^{6\times k}$ is the Jacobian matrix, $\{K_t, K_r, B_t, B_r\} \in \mathbb{R}^{3\times 3}$ are diagonal matrices that contain the translational and rotational impedance stiffness and damping parameters, respectively.

Since $\tilde{\boldsymbol{x}}$ is assumed to be constant in this section, to generate an interaction force between the end-effector and the environment varying over time, the impedance stiffness matrix should be varying over time as a function of time:

$$K_t = \begin{bmatrix} K_{tx} \\ K_{ty} \\ K_{tz} \end{bmatrix} = \begin{bmatrix} f_x(t) \\ f_y(t) \\ f_z(t) \end{bmatrix}, \tag{6.4}$$

where K_{tx}, K_{ty}, and K_{tz} represent the stiffness gains in the X-, Y-, and Z-axes, while $f_x(t)$, $f_y(t)$, and $f_z(t)$ denote three time-dependent functions that are employed to adjust the stiffness values dynamically throughout the task execution.

6.4 Experiment Results

This section will showcase a experiment which uses the variable impedance to improve the efficiency and safety of the manipulator pick-and-handover task.

6.4.1 Task Execution

The process of the pick-and-handover action of the robotic manipulator that is executed within the trajectory planning code is introduced in this section. The flowchart for the pick-and-handover task is provided in Fig. 6.2. At first, the coordinates of the desired fruit and hand is detected by vision module. Afterwards, the end-effector's position is aligned to the object in X-axis to avoid collision. Next, the orientation of the end-effector is adjusted to a desired pick-up orientation. And then the end-effector is directed to the Y- and Z-coordinates of the object. The grasping command will be sent to end-effector when the object position is reached, and the object can be picked up by the qb-SoftHand. The qb-SoftHand then raises and adjusts the orientation to avoid collisions with the surface of the box. Once the manipulator is in a collision-free state, it will then move towards the position of the hand and adjust the orientation to a desired place once at the same time. The manipulator consistently tracks the operator's hand movements, allowing them to receive the objects in various locations. Once the manipulator reaches the position of the hand and is held above the hand for a set amount of time, the qb-SoftHand will receive a command to open and place

Table 6.1: The stiffness for each stage of the proposed pick-and-handover task

Step	Description	Stage	Stiffness
1	Hold at the default position	No interaction	High
2	Move towards object`s location and adjust orientation	No interaction	High
3	Pick the object, Raise arm and adjust orientation	Interact with object	Medium
4	Move towards hand`s initial location	Interact with object	Medium
5	Follow the movement of the hand	Interact with human	Low
6	Move back to default position	No interaction	High
7	Hold at the default position	No interaction	High

the fruit in the hand. Afterward, the manipulator will return to its original position and orientation for further use.

Generally, as shown in Table 6.1, the operation of a robotic manipulator is divided into three main stages: a) No interaction with either an object or a human; b) Interaction with an object; and c) Interaction with a human. These stages utilize varying degrees of stiffness for the end-effector, depending on the nature of the interaction.

1. When the end-effector is not interacting with any object or human, a high level of stiffness, denoted as $K_{t(high)}$, is employed. This is done to minimize the position tracking error and maintain precise control over the manipulator's movements.

2. When the end-effector interacts with an object, a medium level of stiffness, denoted as $K_{t(med)}$, is used. This setting ensures a balance between control and flexibility, preventing any potential damage to the object during interaction.

3. Lastly, when the end-effector interacts with a human operator, a low level of stiffness, denoted as $K_{t(low)}$, is applied. This softer approach prioritizes the safety of the human operator, allowing for a gentler interaction and minimizing any risk of harm.

By tailoring the stiffness according to the interaction, the system can provide an optimal balance between precision, safety, and efficiency.

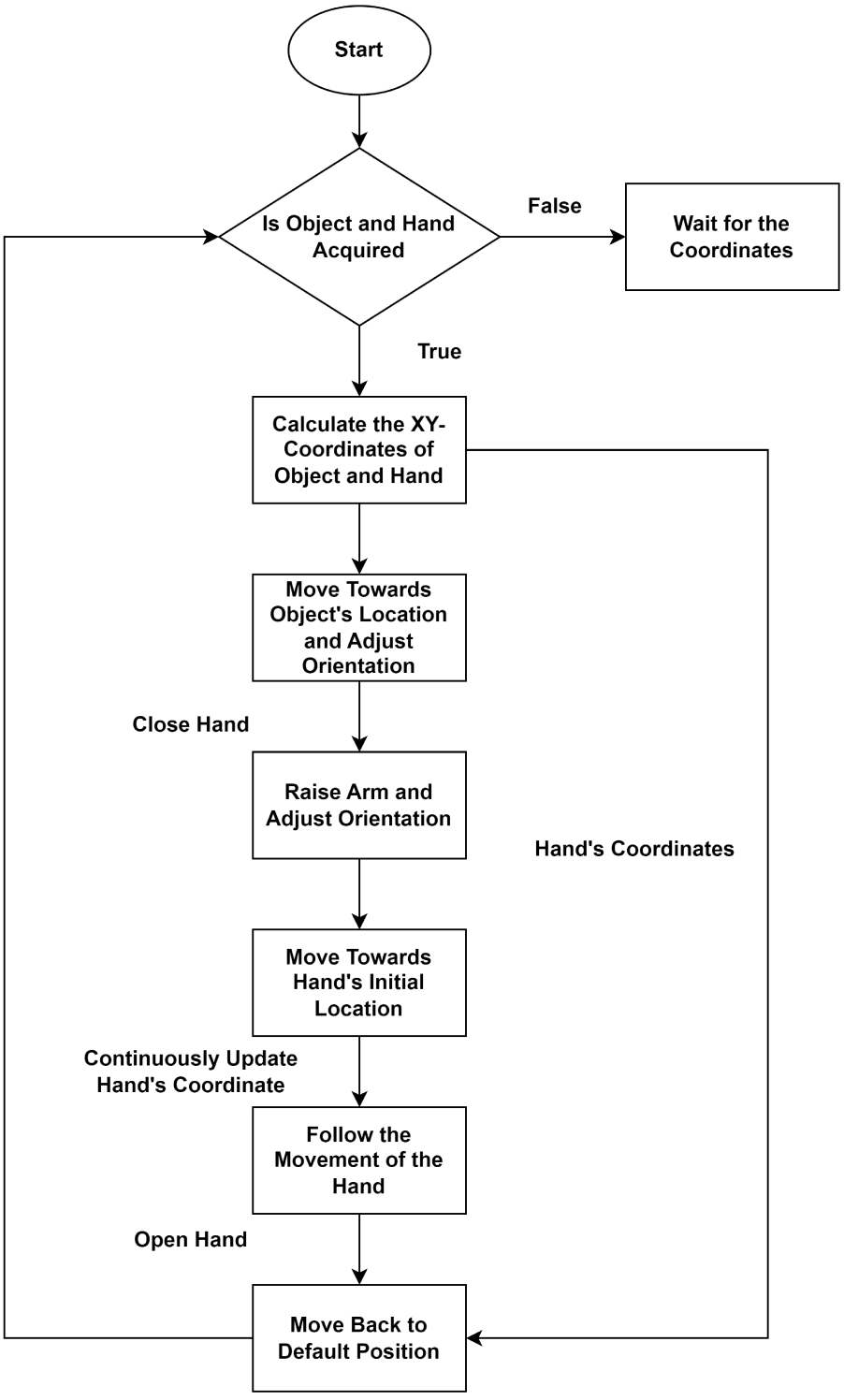

Figure 6.2: The flowchart for the proposed pick-and-handover task

6.4.2 Experiment Execution

The vision module used in this section is same as section 4.5. The impedance parameters with proper units are set as $K_{t(high)} = 300N/m$, $K_{t(med)} = 250N/m$, and $K_{t(low)} = 200N/m$. During the execution of the code, the robot can perform the following process: i) identify the location of the fruit; ii) pick up the fruit; iii) handover the fruit to the location of the user's hand. Fig. 6.3A shows the end-effector in the initial position. Fig. 6.3B shows the end-effector grasping the object. Fig. 6.3C shows that the manipulator moves to the initial hand location. Fig. 6.3D shows that manipulator follows the movement of the hand and finish the action "handover". The videos of the pick-and-handover tasks is available at `https://youtu.be/uXdda8pdA_U`.

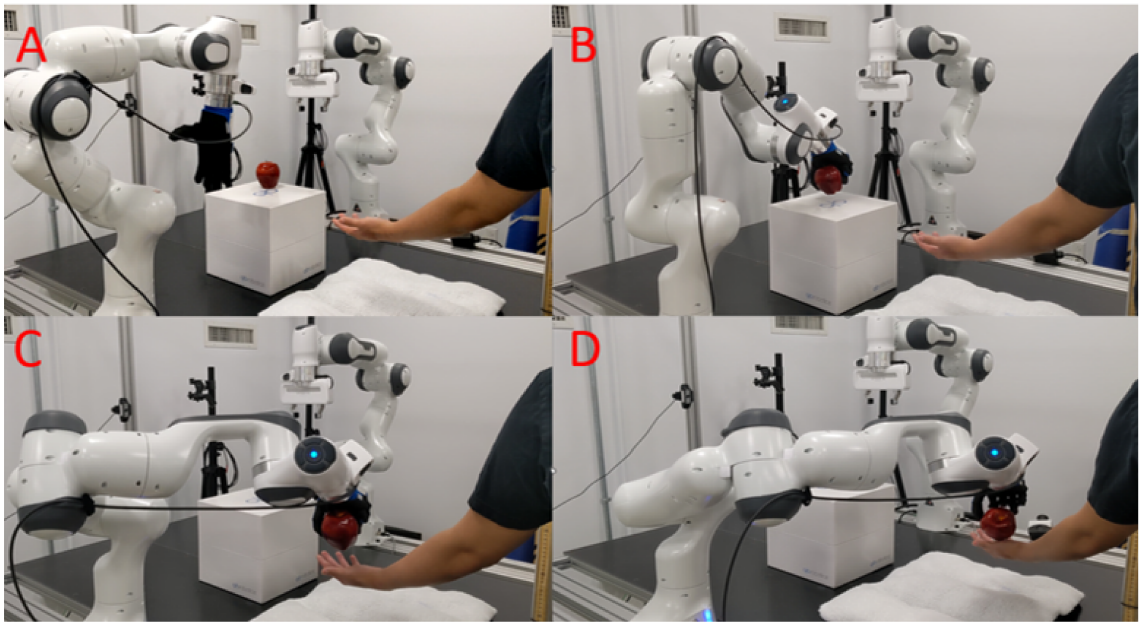

Figure 6.3: The process of pick-and-handover task

6.4.3 Experiment Result

Fig 6.4 illustrates the recorded end-effector trajectories for the experimental pick-and-handover tasks. To demonstrate the altering in the transitional stiffness during the task does not result in any instability or abrupt trajectory changes. Different colors are used to represent various transitional stiffness levels. The black plot depicts the trajectory where the end-effector does not interact with the object or the

operator, and thus a high stiffness, $K_{t(high)}$, is employed. The purple plot represents the trajectory where the end-effector interacts with the object, necessitating a medium stiffness, $K_{t(med)}$. Lastly, the green plot illustrates the trajectory where the end-effector interacts with the human operator, requiring a low stiffness, $K_{t(low)}$.

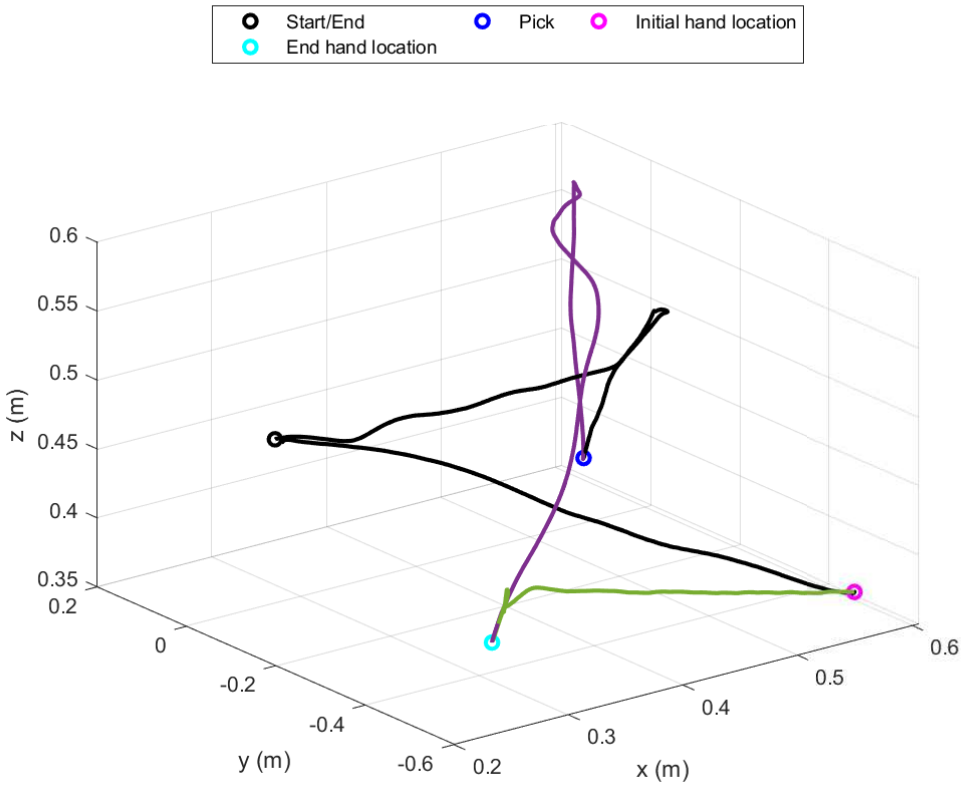

Figure 6.4: The end-effector's actual trajectories with varying stiffness

Fig. 6.5 displays the desired X-, Y-, and Z-coordinate plots alongside the actual X-, Y-, and Z- coordinate plots. The plots show a slight delay of approximately 0.7 seconds before 75 seconds, which may be caused by various factors such as network latency, processing delays, or hardware limitations. However, after 75 seconds, a larger delay of around 1.5 seconds is observed, which can be attributed to the continuous object detection of a moving hand. This process increases the computational load significantly and can lead to longer processing times, resulting in the observed delay. To improve the overall performance, optimizing the object detection algorithm could be considered. The plot indicates that the convergence in X and Y directions is better

than that in Z direction. This could be due to the weight of the grasping object, as it can significantly affect the convergence in Z-axis. The weight of the object creates a gravitational force that needs to be counteracted by the impedance controller, which can result in less precise control over Z-axis.

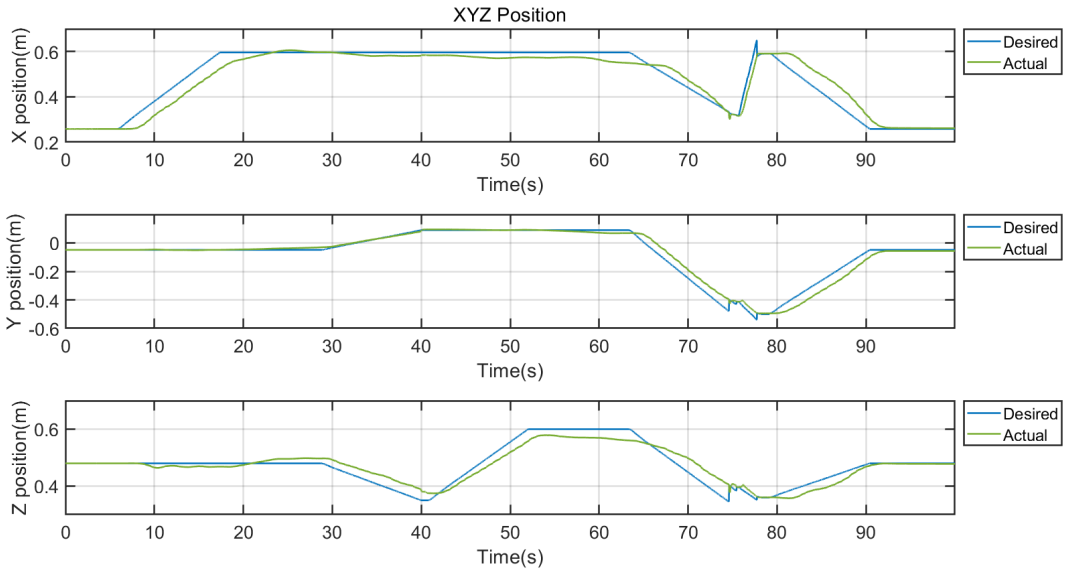

Figure 6.5: The end-effector's actual trajectories vs desired trajectory

6.5 Surface Cleaning Task Using Variable Impedance Control

This section introduces another application of variable impedance control - enabling the end-effector to apply force to its environment. This feature expands the range of tasks that the robot can perform. Considering the interaction between the end-effector and its environment as a process similar to compressing a spring allows for a better understanding of the forces involved. There are two methods to generate varying forces: 1) compress the same spring with a longer displacement, or 2) compress the same displacement using different springs. For manipulator applications, the second method is more convenient as the end-effector's trajectory would need to be replanned if the first method is used. As a result, this section will introduce how to generate the various force for the interaction between the end-effector and the environment by using variable impedance control.

6.5.1 Hardware Setup

As illustrated in Fig. 6.6, the manipulator is designed to perform a surface cleaning task by grasping a duster and traversing the top of the box, applying varying levels of force to effectively clean the surface.

Figure 6.6: Initial and goal state for the surface cleaning task

6.5.2 Experiment Execution

The surface cleaning process of the robotic manipulator follows an equal step trajectory planning. Initially, the manipulator applies low stiffness to the surface for a duration from 0 to 10 seconds. Afterward, the stiffness increases linearly from 10 to 48 seconds, and finally maintains a high level of stiffness between 48 and 60 seconds, as demonstrated in (6.5). It is important to note that the same value of K_t was utilized for K_{tx}, K_{ty}, and K_{tz} during this experiment (N/m).

$$K_t = \begin{cases} 150 & \text{for } 0s < t < 10s \\ 150 + 0.375t & \text{for } 10s <= t < 48s \\ 300 & \text{for } 48s <= t < 60s. \end{cases} \tag{6.5}$$

The force analysis diagram is shown in Fig. 6.7, where F_z represents the force applied by the gripper to the box, F_n denotes the reaction force exerted by the box

on the gripper, and F_f corresponds to the friction force between the duster and the box. The blue block symbolizes the gripper, while the orange block represents the box. Initially, the gripper makes contact with the box at a point between two edges of the box. It then moves in the positive Y direction until it reaches the edge of the box. Finally, the gripper proceeds to move in the negative Y direction until the completion of the experiment.

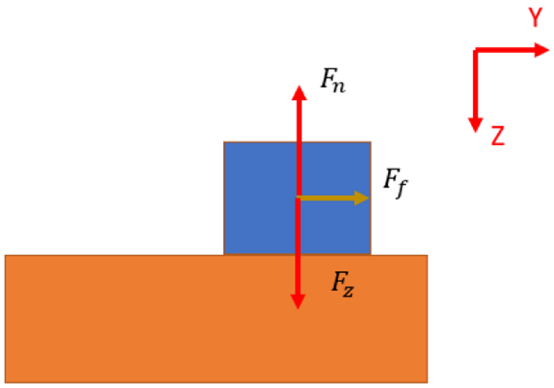

Figure 6.7: Force diagram

6.5.3 Experiment Result

Fig. 6.8 illustrates the interaction force throughout the experiment, with a 10-point moving average filter applied to the data for smoothing purposes. This results in a noticeable increase/decrease at the beginning of the plot. It is important to note that there is no force acting in the X direction during the experiment; thus, any fluctuations in the force plot along X-axis can be primarily attributed to noise. A peak in the force along Z-axis is observed when the gripper makes contact with the box between 0 - 10 seconds, followed by a slight decrease as the gripper begins to move. The contact force exhibits a linear increase between 10 - 48 seconds, corresponding to the rise in stiffness gain. After 48 seconds, the contact force remains approximately constant. Initially, a negative force appears in Y-axis as the gripper moves towards the positive Y direction. Following this, between 10 - 48 seconds, the force in Y-axis linearly increases, corresponding to the rise in force in Z-axis. After 48 seconds, the force in Y-axis also stabilizes and remains approximately constant.

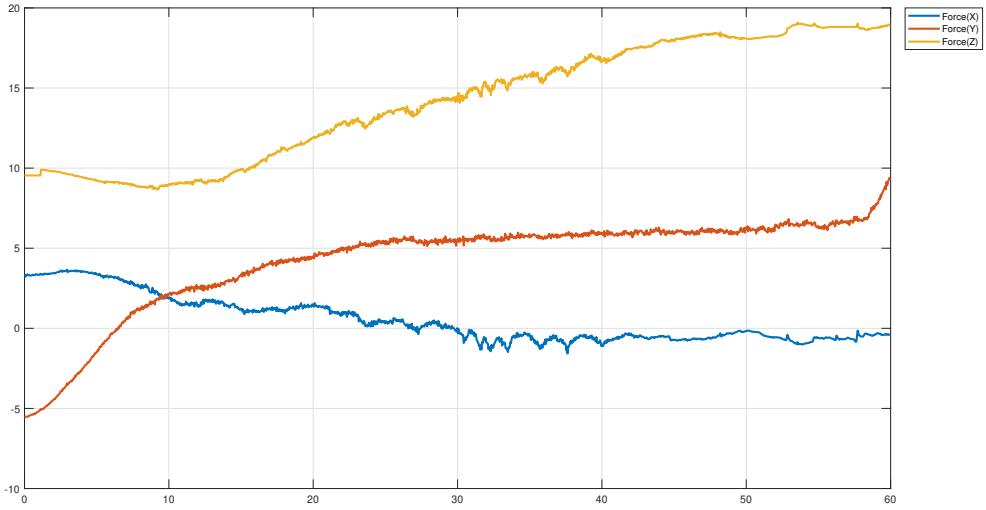

Figure 6.8: The interaction force (N) between the end-effector and environment

Fig. 6.9 illustrates the magnitude of the composition of force during the entire experiment. Initially, from 0 - 10 seconds, the composition of force decreases as the gripper moves in the negative Y direction. Between 10 seconds and 48 seconds, the composition of force exhibits a linear increase. This is because the impedance stiffness of the system also increases linearly during this time period. As the stiffness of the system increases, the force required to maintain the desired position also increases. After 48 seconds, the composition of force becomes more noisy. This change in behavior is a result of the gripper making contact with the edge of the box.

Fig. 6.10 presents a comparison between the desired and actual trajectories of the end-effector, as mapped in X-, Y-, and Z-coordinates. This visualization serves to confirm the efficacy of the variable impedance control algorithm.

The X-coordinate demonstrates a high degree of convergence, largely attributed to the absence of force in this specific direction. A slight, consistent discrepancy is observed in Y direction, indicate the manipulator has to counteract friction during task execution. Notably, a substantial constant difference between the desired and actual trajectories is apparent in the Z direction, which can be attributed to the pressure exerted by the end-effector on the box surface during interaction.

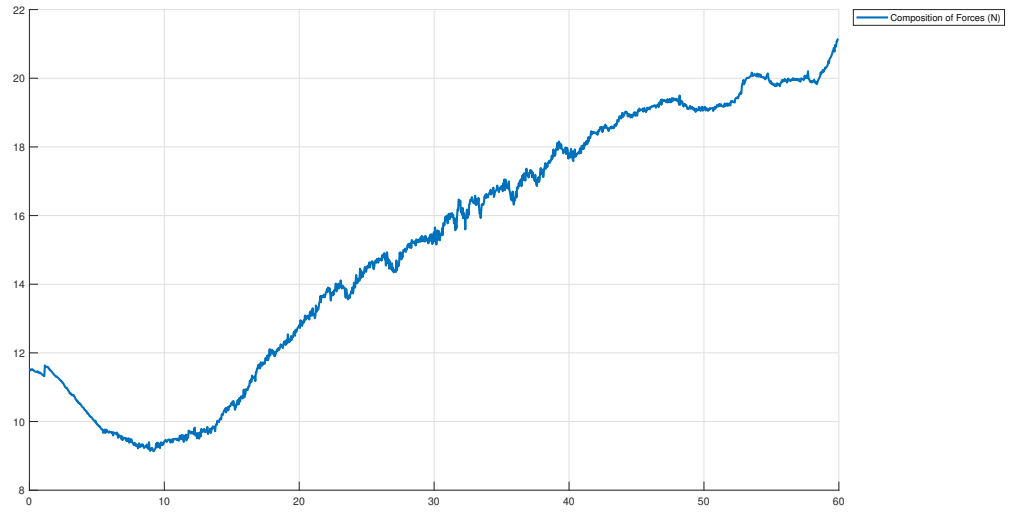

Figure 6.9: The forces (N) between the end-effector and environment

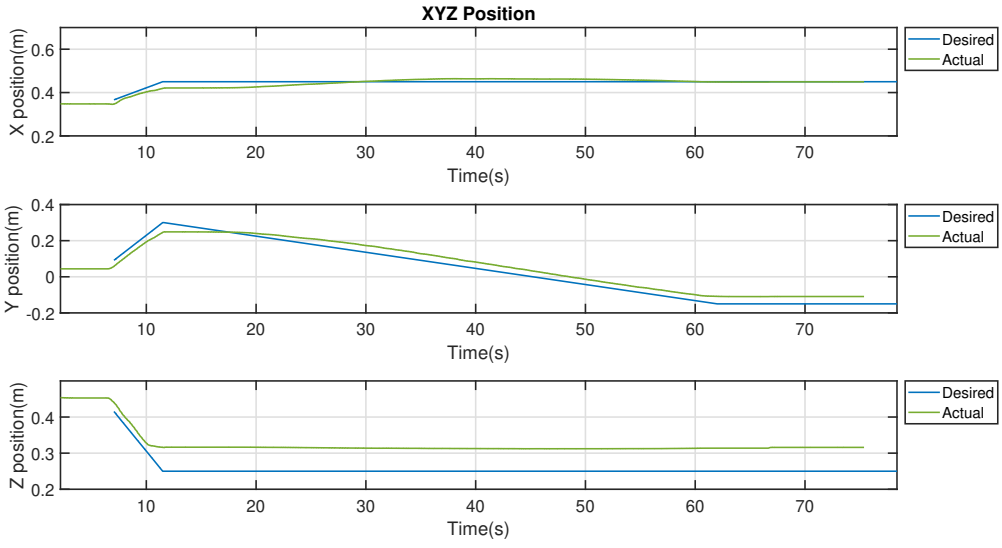

Figure 6.10: Desired coordinates VS. actual coordinates

6.6 Summary

This chapter introduces the utilization of the variable impedance control algorithm for two cases: 1) complete a object pick-and-handover task with different impedance stiffness, and 2) generate various interaction forces between the end-effector and the environment. In future research, machine learning algorithms will be integrated with the variable impedance control approach to develop more intelligent control methods to adapt stiffness during task execution.

Chapter 7

Conclusions and Future Work

This chapter provides a summary of the work presented in this thesis and proposes potential research areas that expand upon the concept of intelligent control for manipulators.

7.1 Conclusions

The first part of this thesis introduces the design, development, and incorporation of an advanced intelligent manipulator system for executing pick-and-place tasks employing a vision-based impedance control method. The intelligent manipulator system encompasses several modules, primarily focusing on object detection, trajectory planning, and the generation of varying interaction forces.

The second area of exploration involves the development of precise 2D and 3D vision modules. These modules serve as the "eyes" of the manipulator system, allowing it to identify, localize, and understand objects in its environment. This task is accomplished using the YOLOv5 object detection algorithm, known for its remarkable speed and accuracy. This high-performance deep learning algorithm enables the manipulator system to recognize various objects in real-time, regardless of their shape, size, or position.

The third part of the study concentrates on the establishment of an innovative trajectory planning strategy. The strategy is centered on the application of DMPs, which allows the manipulator to create and follow paths from a start point to an end point, replicating human-like motion. This enables the manipulator to not just move from one point to another, but also to adapt its movements based on the environment and the task at hand. The use of difference-based DMPs offers improved flexibility and adaptability in the system's performance.

Lastly, the research focuses on the application of a variable impedance control algorithm. This algorithm allows the manipulator to adjust the interaction forces

between its end-effector and the surrounding environment. This is a key feature in manipulator systems, as it enables the system to handle different objects with varying degrees of force and precision, thus performing tasks in a safe and efficient manner.

In order to verify the performance of the developed system, a series of experiments were carried out. The results show that the intelligent manipulator can perform pick-and-handover tasks with high accuracy and efficiency. Additionally, the system demonstrated an ability to autonomously execute tasks in a human-like manner, suggesting that it could smoothly interact with human operators and adapt to changes in the workspace.

In conclusion, this thesis provides a comprehensive exploration of a vision-based impedance control manipulator system. The combination of the YOLOv5 object detection algorithm, difference-based DMPs trajectory planning, and variable impedance control algorithm has resulted in a manipulator system that can perform pick-and-handover tasks accurately, efficiently, and in a human-like manner.

7.2 Future Work

In the future, various intelligent and cooperative control methods are required for multi-mobile-manipulator systems to complete different tasks. Some challenges exist in this field, such as swift shifting of control methods and controller parameters based on task requirements during execution, separating complex tasks into sub-tasks while keeping executing efficiency, human detection, and interaction while navigating and co-manipulating in a dynamic environment. The problems may be magnified when mobile collaborative robots are employed as service robots. To improve robotic manipulators' service quality when working with humans in a complex environment, novel control systems for robotic manipulators are needed. The objective of the future research is to propose a novel cooperative control approach and reinforcement learning (RL) method for collaborative robots to complete complex tasks with multiple robots in the system and with humans in a dynamic environment.

Bibliography

[1] Steven Gislam. 21st century industrial automation: From reactive to proactive, value add services, Sep 2019.

[2] Ali Abdi, Mohammad Hassan Ranjbar, and Ju Hong Park. Computer vision-based path planning for robot arms in three-dimensional workspaces using q-learning and neural networks. *Sensors*, 22(5):1697, 2022.

[3] Poojaallawadhi. Robotics innovation: Leading companies in surgical image manipulation applications for the healthcare industry, Feb 2023.

[4] Tamas Haidegger. Autonomy for surgical robots: Concepts and paradigms. *IEEE Transactions on Medical Robotics and Bionics*, 1(2):65–76, 2019.

[5] Valerio Ortenzi, Akansel Cosgun, Tommaso Pardi, Wesley P. Chan, Elizabeth Croft, and Dana Kulic. Object handovers: A review for robotics. *IEEE Transactions on Robotics*, 37(6):1855–1873, 2021.

[6] Yingli Li, Huibin Du, Yiwen Zhao, Zheng Wang, and Xingang Zhao. A fast calibration implementation for multiple depth cameras and manipulator based on invariance of the linear transformation. *2018 Chinese Automation Congress (CAC)*, pages 3666–3671, 2018.

[7] Sintef. Sintef digital robot manipulator lab.

[8] Menno Lubbers, Job Voorst, Maarten Jongeneel, and Alessandro Saccon. Learning suction cup dynamics from motion capture: Accurate prediction of an object's vertical motion during release. pages 1541–1547, 10 2022.

[9] Sina Parastegari, Ehsan Noohi, Bahareh Abbasi, and Miloš Žefran. A fail-safe object handover controller. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2003–2008, 2016.

[10] Weiyong Si, Ning Wang, and Chenguang Yang. A review on manipulation skill acquisition through teleoperation-based learning from demonstration. *Cognitive Computation and Systems*, 3(1):1–16, 2021.

[11] Wei Yang, Chris Paxton, Maya Cakmak, and Dieter Fox. Human grasp classification for reactive human-to-robot handovers. pages 11123–11130, 2020.

[12] Lars Johannsmeier and Sami Haddadin. A hierarchical human-robot interaction-planning framework for task allocation in collaborative industrial assembly processes. *IEEE Robotics and Automation Letters*, 2(1):41–48, 2017.

[13] Clemente Lauretti, Francesca Cordella, Anna Lisa Ciancio, Emilio Trigili, Jose Maria Catalan, Francisco Javier Badesa, Simona Crea, Silvio Marcello Pagliara, Silvia Sterzi, Nicola Vitiello, and et al. Learning by demonstration for motion planning of upper-limb exoskeletons. *Frontiers in Neurorobotics*, 12, 2018.

[14] Monica Katherine Gonzalez, Nikolas Alexander Theissen, Asier Barrios, and Andreas Archenti. Online compliance error compensation system for industrial manipulators in contact applications. *Robotics and Computer-Integrated Manufacturing*, 76:102305, 2022.

[15] Hang Su, Salih Ertug Ovur, Zhijun Li, Yingbai Hu, Jiehao Li, Alois Knoll, Giancarlo Ferrigno, and Elena De Momi. Internet of things (iot)-based collaborative control of a redundant manipulator for teleoperated minimally invasive surgeries. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[16] Zhengxue Zhou, Leihui Li, Alexander Fürsterling, Hjalte Joshua Durocher, Jesper Mouridsen, and Xuping Zhang. Learning-based object detection and localization for a mobile robot manipulator in sme production. *Robotics and Computer-Integrated Manufacturing*, 73:102229, 2022.

[17] Patrick Rosenberger, Akansel Cosgun, Rhys Newbury, Jun Kwan, Valerio Ortenzi, Peter Corke, and Manfred Grafinger. Object-independent human-to-robot handovers using real time robotic vision. *IEEE Robotics and Automation Letters*, 6(1):17–23, 2021.

[18] Antonis Sidiropoulos, Efi Psomopoulou, and Zoe Doulgeri. A human inspired handover policy using gaussian mixture models and haptic cues. *Autonomous Robots*, 43(6):1327–1342, 2018.

[19] Sina Parastegari, Ehsan Noohi, Bahareh Abbasi, and Miloš Žefran. A fail-safe object handover controller. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2003–2008, 2016.

[20] José R. Medina, Felix Duvallet, Murali Karnam, and Aude Billard. A human-inspired controller for fluid human-robot handovers. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 324–331, 2016.

[21] Valerio Ortenzi, Francesca Cini, Tommaso Pardi, Naresh Marturi, Rustam Stolkin, Peter Corke, and Marco Controzzi. The grasp strategy of a robot passer influences performance and quality of the robot-human object handover. *Frontiers in Robotics and AI*, 7, 2020.

[22] Kathrin Bothe, Alexander Winkler, and Leif Goldhahn. Effective use of lightweight robots in human-robot workstations with monitoring via rgbd-camera. In *2018 23rd International Conference on Methods Models in Automation Robotics (MMAR)*, pages 698–702, 2018.

[23] S. Rosa, A. Russo, A. Saglinbeni, and G. Toscana. Vocal interaction with a 7-dof robotic arm for object detection, learning and grasping. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 505–506, 2016.

[24] Ansgar Koene, Satoshi Endo, Anthony Remazeilles, Miguel Prada, and Alan M. Wing. Experimental testing of the coglaboration prototype system for fluent human-robot object handover interactions. *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 249–254, 2014.

[25] Miguel Prada, Anthony Remazeilles, Ansgar Koene, and Satoshi Endo. Implementation and experimental validation of dynamic movement primitives for object handover. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2146–2153, 2014.

[26] Dzmitry Tsetserukou, Riichiro Tadakuma, Hiroyuki Kajimoto, Naoki Kawakami, and Susumu Tachi. Towards safe human-robot interaction: Joint impedance control of a new teleoperated robot arm. In *RO-MAN 2007 - The 16th IEEE International Symposium on Robot and Human Interactive Communication*, pages 860–865, 2007.

[27] Matthew K. X. J. Pan, Elizabeth A. Croft, and Günter Niemeyer. Exploration of geometry and forces occurring within human-to-robot handovers. In *2018 IEEE Haptics Symposium (HAPTICS)*, pages 327–333, 2018.

[28] Wesley P. Chan, Yohei Kakiuchi, Kei Okada, and Masayuki Inaba. Determining proper grasp configurations for handovers through observation of object movement patterns and inter-object interactions during usage. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1355–1360, 2014.

[29] Shih-Hung Wu and Xie-Sheng Hong. Integrating computer vision and natural language instruction for collaborative robot human-robot interaction. In *2020 International Automatic Control Conference (CACS)*, pages 1–5, 2020.

[30] Kyekyung Kim, Jaemin Cho, Jihyeong Pyo, Sangseung Kang, and Jinho Kim. Dynamic object recognition using precise location detection and ann for robot manipulator. In *2017 International Conference on Control, Artificial Intelligence, Robotics Optimization (ICCAIRO)*, pages 237–241, 2017.

[31] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[32] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing.

[33] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[34] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[35] Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[36] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[37] Iason Batzianoulis, Fumiaki Iwane, Shupeng Wei, Carolina Gaspar Pinto Ramos Correia, Ricardo Chavarriaga, José del R Millán, and Aude Billard. Customizing skills for assistive robotic manipulators, an inverse reinforcement learning approach with error-related potentials. *Communications biology*, 4(1):1406, 2021.

[38] Emil Blixt Hansen, Rasmus Eckholdt Andersen, Steffen Madsen, and Simon Bøgh. Transferring human manipulation knowledge to robots with inverse reinforcement learning. In *2020 IEEE/SICE International Symposium on System Integration (SII)*, pages 933–937, 2020.

[39] Min Wu, Bertram Taetz, Yanhao He, Gabriele Bleser, and Steven Liu. An adaptive learning and control framework based on dynamic movement primitives with application to human–robot handovers. *Robotics and Autonomous Systems*, 148:103935, 2022.

[40] Freek Stulp, Evangelos A. Theodorou, and Stefan Schaal. Reinforcement learning with sequences of motion primitives for robust manipulation. *IEEE Transactions on Robotics*, 28(6):1360–1370, 2012.

[41] Lucas Wan and Ya-Jun Pan. Bilateral teleoperation of a multi-robot formation with time-varying delays using adaptive impedance control. In *2022 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1739–1746, 2022.

[42] Georgeta Bauer, Ya-Jun Pan, and Henghua Shen. Adaptive impedance control in bilateral telerehabilitation with robotic exoskeletons. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 719–725, 2020.

[43] Fanny Ficuciello, Luigi Villani, and Bruno Siciliano. Variable impedance control of redundant manipulators for intuitive human–robot physical interaction. *IEEE Transactions on Robotics*, 31(4):850–863, 2015.

[44] Peng Song, Yueqing Yu, and Xuping Zhang. A tutorial survey and comparison of impedance control on robotic manipulation. *Robotica*, 37(5):801–836, 2019.

[45] Homayoun Seraji and Richard Colbaugh. Force tracking in impedance control. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 499–506 vol.2, 1993.

[46] Robert J. Anderson. and Mark W. Spong. Hybrid impedance control of robotic manipulators. In *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, volume 4, pages 1073–1080, 1987.

[47] A. Ibeas and M. de la Sen. Robust impedance control of robotic manipulators. In *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, volume 2, pages 1258–1263 Vol.2, 2004.

[48] K. Lee and M. Buss. Force tracking impedance control with variable target stiffness. *IFAC Proceedings Volumes*, 41(2):6751–6756, 2008. 17th IFAC World Congress.

[49] Claudio Gaz, Marco Cognetti, Alexander Oliva, Paolo Robuffo Giordano, and Alessandro De Luca. Dynamic identification of the franka emika panda robot with retrieval of feasible parameters using penalty-based optimization. *IEEE Robotics and Automation Letters*, 4(4):4147–4154, 2019.

[50] Lentin Joseph and Aleena Johny. *Robot Operating System (ROS) for absolute beginners*, 2022.

[51] Jinjun Duan, Yahui Gan, Ming Chen, and Xianzhong Dai. Adaptive variable impedance control for dynamic contact force tracking in uncertain environment. *Robotics and Autonomous Systems*, 102:54–65, 2018.

Appendix A

Publication List

QiGuang Chen, Lucas Wan and Ya-Jun Pan, "Object Recognition and Localization for Pick-and-Place Task using Difference-based Dynamic Movement Primitives", In Proceedings of the 2023 IFAC International Federation of Automatic Control, July 2023 Yokohama, Japan, Accepted.

QiGuang Chen, Lucas Wan and Ya-Jun Pan, "Robotic Pick-and-Handover Maneuvers with Camera-based Intelligent Object Detection and Impedance Control",Transactions of the Canadian Society for Mechanical Engineering (TCSME), Revised, Apr 2023.

QiGuang Chen, Lucas Wan, Prabahar Ravichadran, Ya-Jun Pan and Young Ki Chang, "Vision-based Control of 7-DoF Robotic Manipulator for Pick-and-Place Tasks in Grasping Fruits", In Proceedings of the 2022 CSME International Congress of Canadian Mechanical Engineering, June 2022, Edmonton AB, Canada.

Lucas Wan, **QiGuang Chen** and Ya-Jun Pan, "Admittance-Based Non-Singular Terminal Sliding Mode Control of Multiple Cooperative Manipulators", In Proceedings of the 2023 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), Seattle, USA, June, 2023, Accepted.

Nuo Chen, Lucas Wan, **QiGuang Chen** and Ya-Jun Pan, "Real Time Vision-based Human Hand Motion Tracking and Grasping for a Robotic Manipulator with Soft Hand", In Proceedings of the 2023 CSME International Congress of Canadian Mechanical Engineering, May, 2023, Sherbrooke, QC, Canada, Accepted.