HUMAN-ROBOT COLLABORATIVE CONTROL FOR INSPECTION AND MATERIAL HANDLING USING COMPUTER VISION AND JOYSTICK

A Thesis

by

RIYA KHURANA

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Prabhakar R. Pagilla
Committee Members,	Swaroop Darbha
	Dezhen Song
Head of Department,	Andreas A. Polycarpou

December 2020

Major Subject: Mechanical Engineering

ABSTRACT

Human-Robot Collaboration is often required in unstructured, uncertain, and dynamic environments where automation is not ideal. Human operators are needed for their ability to perform complex tasks employing their situational awareness and decision-making capabilities. Collaboration between human and robot is necessary for achieving a higher level of safety and performance in such cases. An effective shared control system will provide humans an intuitive interface for robot control and provide intelligent assistance to improve overall task performance.

In this work, we present a Human-Robot Collaborative Control framework for inspection and material handling tasks by employing computer vision and a general purpose joystick for providing human input to the robot remotely. In order to facilitate the human operator, an intuitive joystick control interface is developed allowing the operator to command the robot in the end-effector frame. This is combined with vision-based motion planning algorithms providing assistance to the operator to complete the task. Human operator controls the robot until the object is detected by vision node and then automatic control takes over. Hue, Saturation and Value (HSV) based OpenCV contour detection algorithms are used for object detection and pose estimation. ROS integrated open-source software, MoveIt has been utilized for motion planning algorithm. For joystick interface, we present a hybrid control law which allows human operators to provide orientation/torque reference in the world-frame and translation/force reference in the robot end-effector frame and automatic control takes care of underlying kinematics and joint level control. A physical platform and a simulation environment consisting of a six Degrees-Of-Freedom UR5 robot, a general purpose joystick, a USB camera, a vacuum gripper, force-torque sensor, and fixed speed Conveyor Belt are employed to develop and test the approach.

DEDICATION

To my mother, my father, and my grandparents.

ACKNOWLEDGMENTS

Before beginning this thesis, I would like to acknowledge and extend gratitude to the many people who have supported me through out my journey at Texas A&M.

My heartfelt thanks to my advisor Dr. Prabhakar R. Pagilla, words cannot explain my gratitude and respect towards you. Your guidance, words of encouragement and valuable feedback towards my work has helped me grow as a researcher. I also extend gratitude towards all the members of my research group: Jie Hu, Daniel Jaeger, Zongyao Jin and Yalun Wen, for always lending a helping hand.

My sincere thanks to the members of my thesis committee: Dr. Swaroop Darbha and Dr. Dezhen Song for giving their time to review my work.

I also extend thanks to entire staff of Mechanical Engineering Department and the Office of Graduate and Professional Studies for their assistance and guidance throughout my studies.

Finally, I thank my family and friends for their unwavering support, love and encouragement.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a committee consisting of Professor Prabhakar R. Pagilla (chair) and Professor Swaroop Darbha of the Department of Mechanical Engineering and Professor Dezhen Song of the Department of Computer Science and Engineering.

The work done for Human-in-Loop control (Chapter 3) was a collaborative effort. The concept of Instantaneous Surface Normal approach for Joystick Control Interface was proposed and formulated by Dr. Zongyao Jin. Experimental developments and formulation for torque based orientation correction were done by the student. All other work conducted for the thesis was completed by the student independently with supervision from Dr. Pagilla.

Funding Sources

Graduate study was generously supported by Mechanical Engineering Department of Texas AM University in forms of Teaching Assistantship and Eddie Joe Mattei'53 Graduate fellowship. Additionally, Research Assistantship opportunities were provided by Dr. Pagilla and research project was supported by the National Science Foundation under grant number 1900704.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION AND LITERATURE REVIEW

The usage of robotics systems have increased across many industries to automate production lines for manufacturing, material handling and assembly tasks. With improvements derived in system's efficiency, precision and reduced labor costs, the benefits of automation are well known and the field is extensively researched and well documented in literature. The factory environments are structured and well defined, tasks are often repetitive, therefore automation, with its ability to perform tasks with high speed, high precision and repeatability, is a go to solution. However automatic control is not ideal for unstructured, uncertain and dynamic environments in which the task could be constantly changing and requires adaptation. Human operators, in such cases, are needed for their ability to perform complex tasks under uncertainty by exhibiting their situational awareness and decision making capabilities. The challenges associated with developing fully autonomous solutions encouraged research in human-machine shared control in recent years [4, 5, 6, 7, 8, 9, 10].

When human presence in the robot space can be unsafe and undesirable and environment is dynamic, remote control of the robot with shared autonomy is required. Such cases include number of applications such as handling of hazardous material, remote repairs, operation at inaccessible sites, underwater operation, etc. Figure 1.1 illustrates example scenarios where shared control paradigm would be useful.

1.1 Shared Control Methods

Shared Control refers to methods where human inputs are directly or indirectly integrated with automatic control input to perform a task [11]. Human inputs and automatic control can be combined in multitude of ways depending upon complexity of applications. General classification of shared control methods includes: collaborative control, traded control, indirect shared control, coordinated control, virtual constraint control and blended shared control. In collaborative control, a subset of control authority lies with human operator and automatic control takes care of the rest. In traded control, the control authority is traded between human and automatic control. In indi-

rect shared control, an additional feedback is provided to aid the human operator with sensory information. Coordinated control is defined as the process where automatic control converts reference human inputs into low level commands. In virtual constraint control, automatic control is programmed to override the human operator commands to overcome system constraints and safe operation. Lastly, in blended shared control method, human operator intent is predicted and automatic control input is generated for the intended command. Human and automatic controls are then blended to complete the task. Each of the shared control methods has its own merits and target applications.

Deciding on what shared control method to be used and appropriate autonomy level in the task typically depends on the application under consideration. The problem is challenging and appropriate answers will vary for different types of tasks and various levels of complexities. An effective shared control system can be said to be the one where humans are provided with an intuitive interface for robot control and automatic control provides appropriate assistance based on the system sensing capabilities to optimize task specific performance. Utilizing this viewpoint, a collaborative control framework is developed for this thesis.

1.2 Background and Relevant Literature

Many methods have been developed to allow human operator to control the robot remotely. The general goal is to allow human operators to specify the motion of the robot end-effector directly, via different interfaces, while having control algorithms to handle robot kinematics, dynamics and complex operations. Various interfaces developed in the literature are: 1) Mechanical hand-held devices like Joysticks, Space balls, Dials, etc., 2) Computer interfaces like simulation environments, touch screen control pendant, augmented and virtual reality devices, 3) Dexterous manipulators which includes mechanical interfaces like exoskeletal devices, instrumental gloves, etc and 4) Tele-presence systems including motion tracking interfaces with sensors and vision.

Methods based on joystick devices can be found in [12, 13, 14, 15, 16]. These methods allow operator to send high level automation commands to the robot using different hand held controllers like joysticks, space balls, etc. Many methods with control through computer interfaces are also

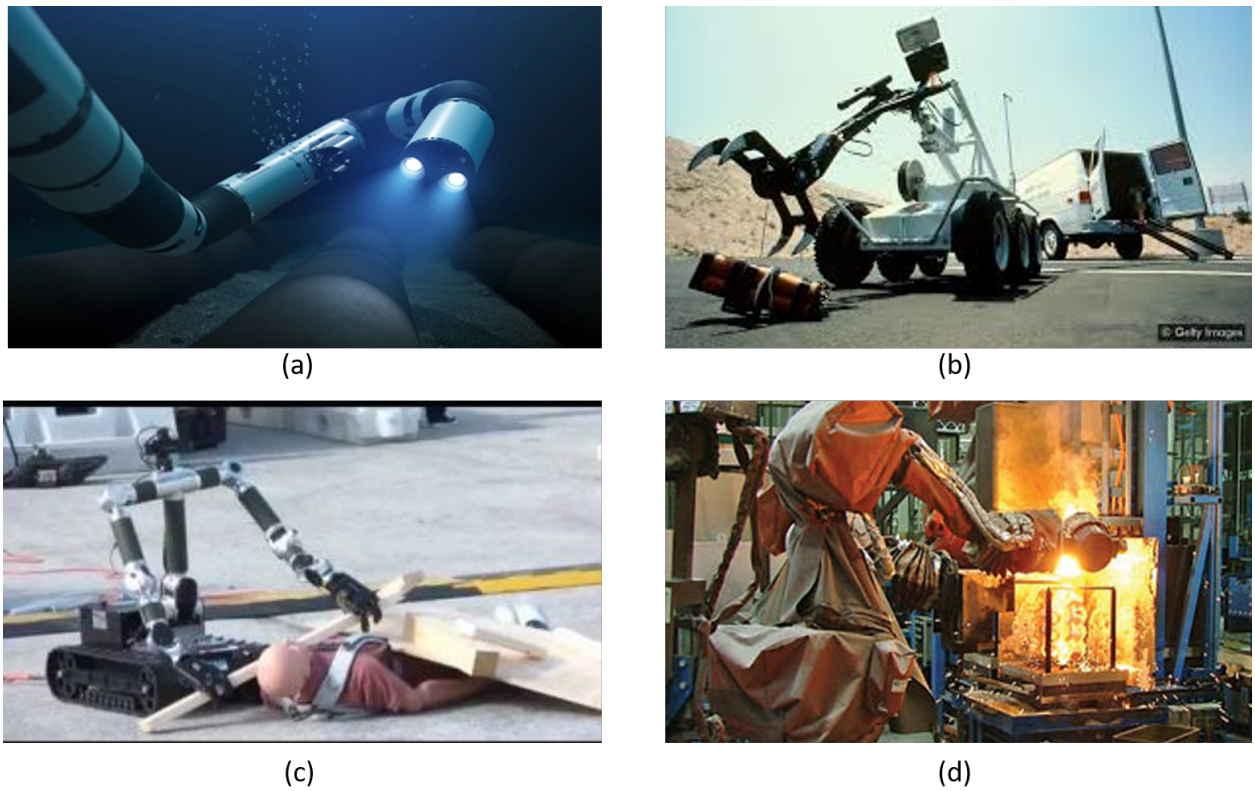


Figure 1.1: Possible Shared Control Applications: a) Underwater Robotics, b) Bomb Diffusal Robots, c) Rescue Operation, d) Operation at hazardous environment

available in literature. In [17, 18], touch-screen interfaces are proposed which provide the human operator a first-person-view from the end-effector camera and the user can directly specify the desired pose and position of the robot by touching and dragging the part on simulated environment on computer. Research has also been done to develop dexterous manipulation techniques where operator commands the robot end-effector to mimic their hand gestures with a wrist tracker or instrumental glove on their hand [19, 20, 21, 22]. Control via mimicking gestures is also done by telepresence methods where the system can track human hand using motion trackers, Kinect depth sensors or vision systems and reference inputs for the robot is generated by mirroring tracking motion [23, 24, 25]. The study described in [26, 27] proposed master-slave mechanism or specialized joysticks for the control of robotic manipulator especially in space and surgery robotics.

Apart from interface to control the robot remotely, providing intelligent assistance to human

operator is also very important in shared control framework. Some systems are developed to offer feedback to the operator to make adjustments and improvements. Haptic interfaces, [28, 29] are developed to provide feedback like collision avoidance, blind spot warnings, etc. [16, 17]

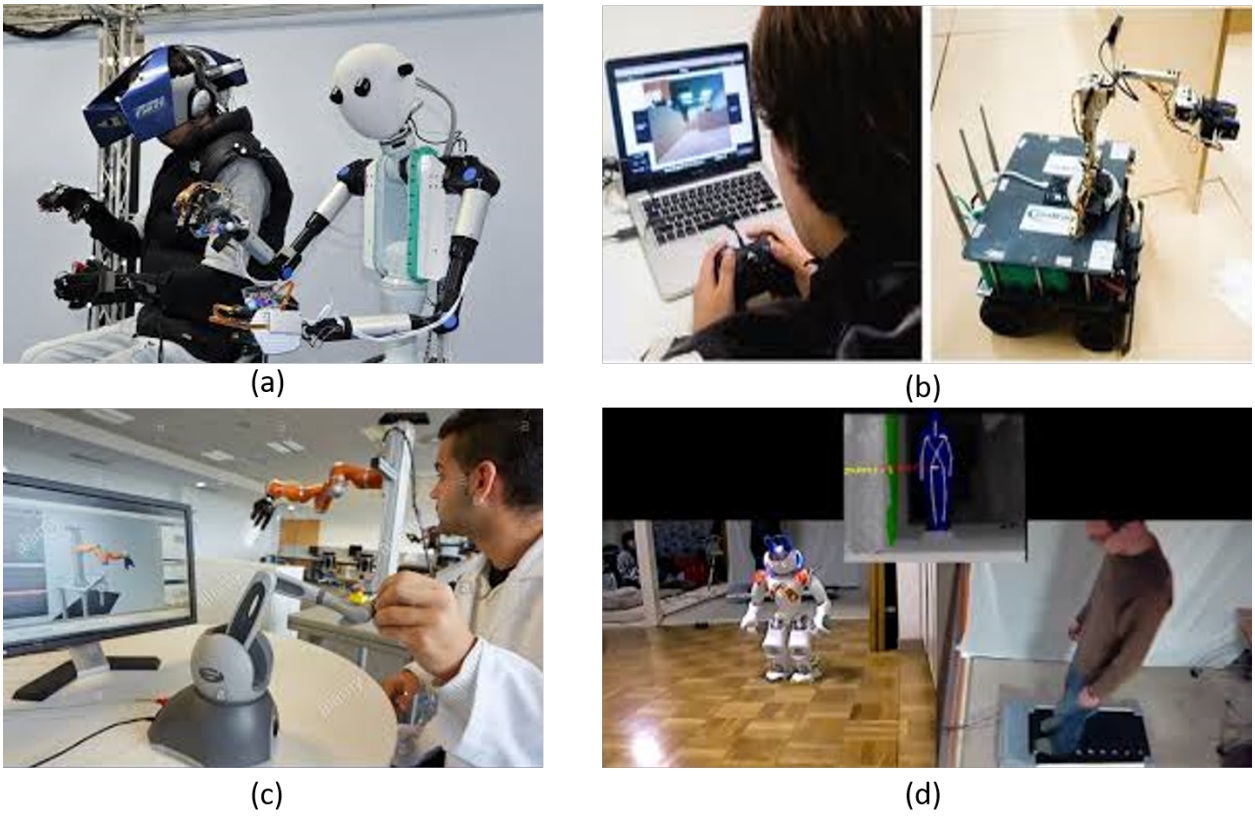


Figure 1.2: Existing teleoperation systems

1.3 Challenges and Problem Formulation

The novel interfaces, like touch-screen or dexterous manipulation techniques or human motion capture systems, allow operators to control robot end-effectors intuitively, however, fast and precise registration of operator's input is difficult and there is a penalty of complexity with the advantages offered by the system. The master-slave systems allow the human operator to specify precisely the position of the robot end-effector, some times even its configuration; however, a drawback is that

another master robot or a specialized joystick is required which substantially increases the cost of the setup.

Cartesian space control with joysticks allows the operator to send commands to move the robot in a fixed/ Cartesian frame, however for material handling tasks in constrained space defining way points and path of the end-effector in cartesian space would not be intuitive for a operator to do remotely. Another drawback of the joystick control is that it is difficult for the human operator to plan around obstacles, singularities, or joint limits as they are not aware of intrinsic robot properties. Therefore, it would be beneficial to design a shared control mechanism which allows humans to operate the robot end-effector intuitively without much training using general purpose joysticks that are readily available and which provide further assistance to the operator to navigate around obstacles and robot limitations.

Material handling and pick-and-place operation are typical tasks that robots handle in a manufacturing line. Many algorithms have been developed in a fully automatic setup. Computer vision techniques for material handling tasks are well established [30, 31, 32, 33]. Therefore, it could be beneficial to utilize the vision based algorithms to provide additional assistance to the operator.

Hence, the objective of this work is set to design a framework which combines computer vision with Human Robot Collaboration (HRC) techniques for remote operation of the robot. In order to develop such a shared control strategy, one can pose the the following questions:

- First, to develop an intuitive control interface for human operators, are there any typical operating features and frames of the task that would allow for easier control for task execution and how to utilize a general purpose joystick to send these commands to the robot?
- Second, how to design an underlying control law to realize the motion commanded by the human operator via a joystick?
- Third, how to incorporate vision sensing in the system to improve task efficiency?
- Lastly, how can assistance be provided for completion of a task while avoiding obstacles and other system constraints such as singularities and joint limits?

In this study, various frameworks were developed to answer these questions.

1.4 Contributions

The two main contributions of the thesis can be summarized as follows:

1. Developed an easy and intuitive way to control the robot remotely utilizing general purpose gaming joysticks. A hybrid control law was designed which allows human operators to provide orientation/torque reference in the world frame and translation/force reference in the instantaneous robot end-effector frame and automatic control takes care of underlying kinematics and joint level control.

2. Developed a method to provide intelligent assistance to the human operator for moving around the task space using vision. Vision based object pose and orientation estimation algorithm and motion planning strategy to grasp a moving object avoiding system constraints and obstacles in the environment are developed.

1.5 Document Outline

This document is organized as follows: Chapter 2 provides brief description of our solution approach, system architecture and each individual components. Chapter 3 explains the Joy Stick Interface, Hybrid Control Law and Kinematics modeling of the robot. The framework for vision based automatic control is discussed in chapter 4. Hardware platform, experiment design and results are presented in Chapter 5. Lastly, the study is summarized and possible future work are presented in Chapter 6.

2. SYSTEM DESCRIPTION

The previous chapter outlined the motivation for the study along with the background on the methods developed in the literature. In this chapter, we present a detailed description of the system architecture and our solution approach. The aim of the study is to perform pick and place tasks with a general purpose joystick with assistance from computer vision.

2.1 Shared Control Architecture

A shared remote operation system with control authority divided between human operator and automatic control for different phases of task completion has been developed as described below:

Phase I, Locating object to be picked: Human operator guides the robot through joystick commands to bring the object into the field of view of the camera

Phase II, Automatic Control with Computer vision: With object in camera's field of view, it can be detected and tracked. Automatic control takes over here to track and grasp the object

Phase III, Dropping the object: After the object has been grasped, human operator can then take over again to decide and maneuver the robot to a desired dropping location. Hybrid shared control algorithm can then assist the operator for orientation correction by torque control.

Fig. 2.1 shows the overall framework of the system developed in this study. There are two components to the system as discussed above, first is joystick control of the robot (Robot Control with Human in Loop) and second is vision based automatic control. Based on environment awareness, human operator first controls the robot by providing reference input using a general purpose joystick in robot end-effector frame. Once the object is in the field of view of camera mounted on the robot automatic control will take over. We now provide details of the two aforementioned modes of operation.

2.2 Joystick Operation

A joystick-based human–robot interface was developed for direct human control of a six-axis robot manipulator which allows the human operators to provide orientation reference in the world

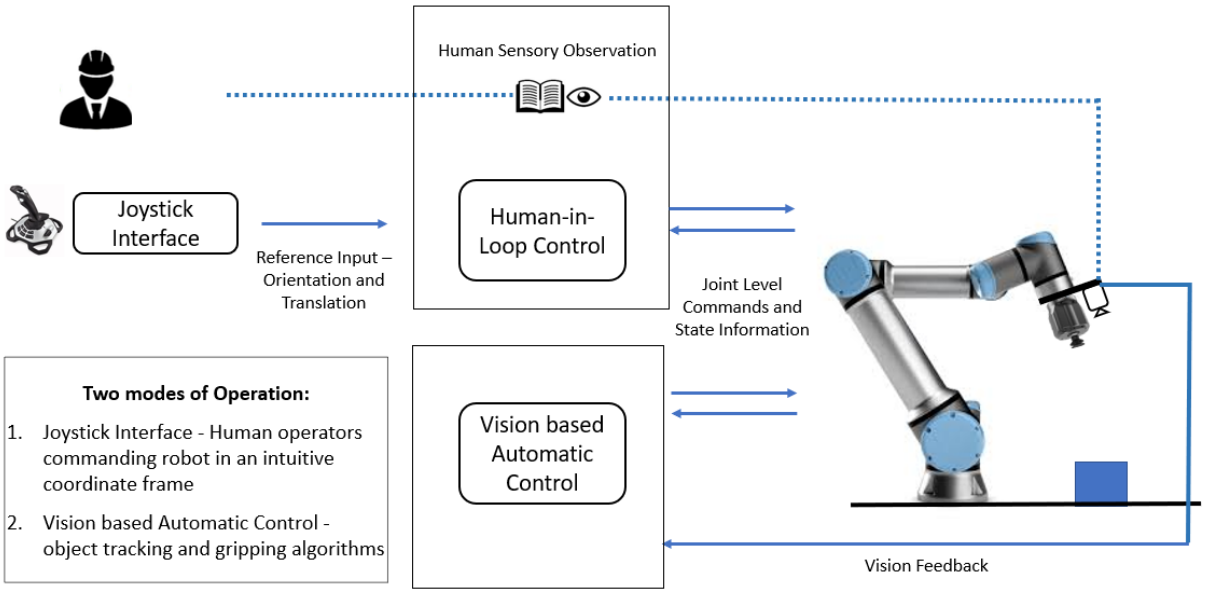


Figure 2.1: Human-Robot Collaborative Control Architecture

frame and translation reference in the instantaneous robot end-effector frame. The human operator commands translation and orientation reference signal to the robot via joystick and automatic control takes care of underlying kinematics and joint-level control. The control schematic is shown in Figure 2.2.

The interface was developed for commanding the robot using general purpose joysticks. Figure 2.3 shows common features on commercially available joystick. Orientation reference were encoded with the 3-axis stick in continuous state range $[-1, 1]$. A 2-axis discrete navigator button, with states {-1,0,1}, were employed to command x and y translation. Throttle slider provides a continuous states and was employed to define magnitude of translation velocity. Discrete programming buttons and a trigger were utilized command end-effector to move along Z-axis and activate gripper.

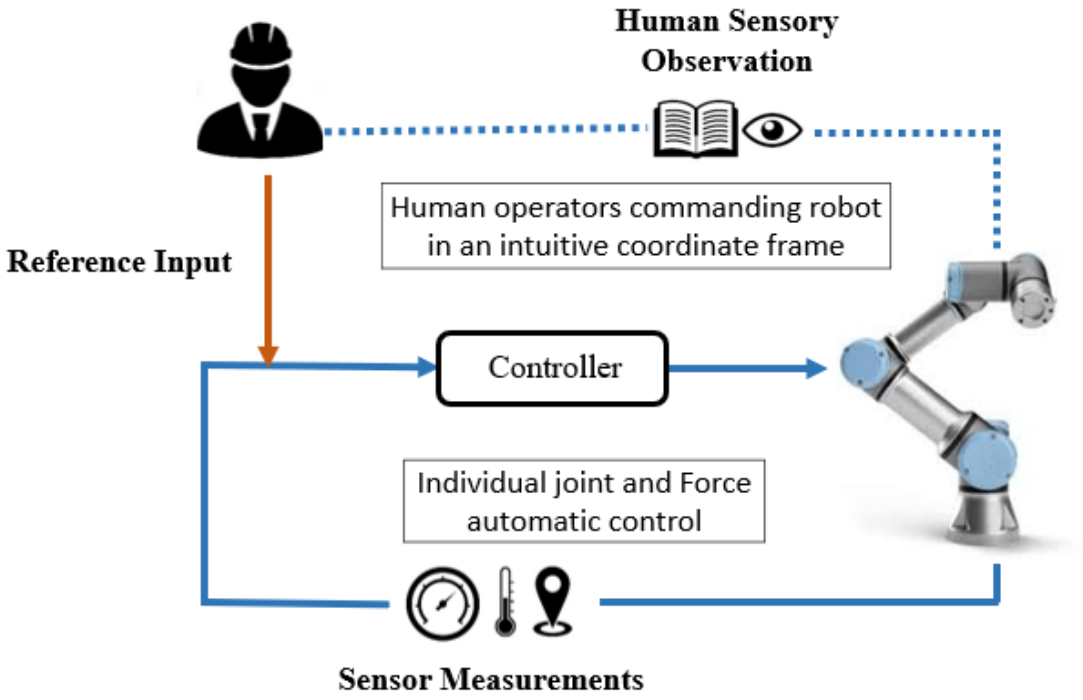


Figure 2.2: Hybrid Shared Control with Joystick

2.3 Vision based Automatic Control

A vision based control approach is developed to detect and track the object automatically. This approach is divided into 2 sub-processes. The first one is work-space analysis, where the object to be grasped is detected and its position and orientation is estimated based on RGB (Red Green Blue) pixel information available from USB camera. The geometric description (position and orientation) of object is obtained from the image data. The vision processing is carried by OpenCV library which is specifically developed for real time computer vision [2].

The second step is motion planning and execution based on information from the vision system. Figure 2.4 shows the system schematic. The target pose is available from the vision module and the motion planning to the target is performed with MoveIt software [3]. Moveit is an open source platform designed for development of solutions for motion planning problems and is widely

Figure 2.3: General Purpose Joysticks and Common Features

used in the robotics community. It has been used on a range of commercial robots in many industrial applications (UR, Kuka, ABB, Motoman, Fanuc, etc robots) to outerspace (Robonaut and Robonaut2 NASA Johnson Space Center), to Humanoids (Valkyrie - NASA Johnson Space Center, Atlas - Boston Dynamics, etc), to mobile robots (PR-2 robots). It provides a wide functionality that covers several features of mobile manipulation, encapsulates a forward/inverse kinematics solver, planning techniques,and collision-detection methods through a plug-in-based mechanism. The details of this framework is presented in Chapter 4.

2.4 System Components

A physical platform consisting of a six Degrees-Of-Freedom UR5 robot, Logitech Extreme Pro joystick, a USB camera, Robotiq E-pick vacuum gripper, ATI axia-80 force-torque sensor and Ultimation Conveyor Belt are employed to develop and test the approach, Figure 2.5. The camera, gripper and force-torque sensor are mounted on the robot end-effector. Objects to be picked and mounting bracket for auxiliaries on robot were 3D printed.

The UR5 allows a user to command its joint velocities independently, and provides feedback measurements of its joint angles and velocities in real-time. The Astra pro camera provides RGB and point cloud data. The ATI force/torque sensor provides force and torque measurements along

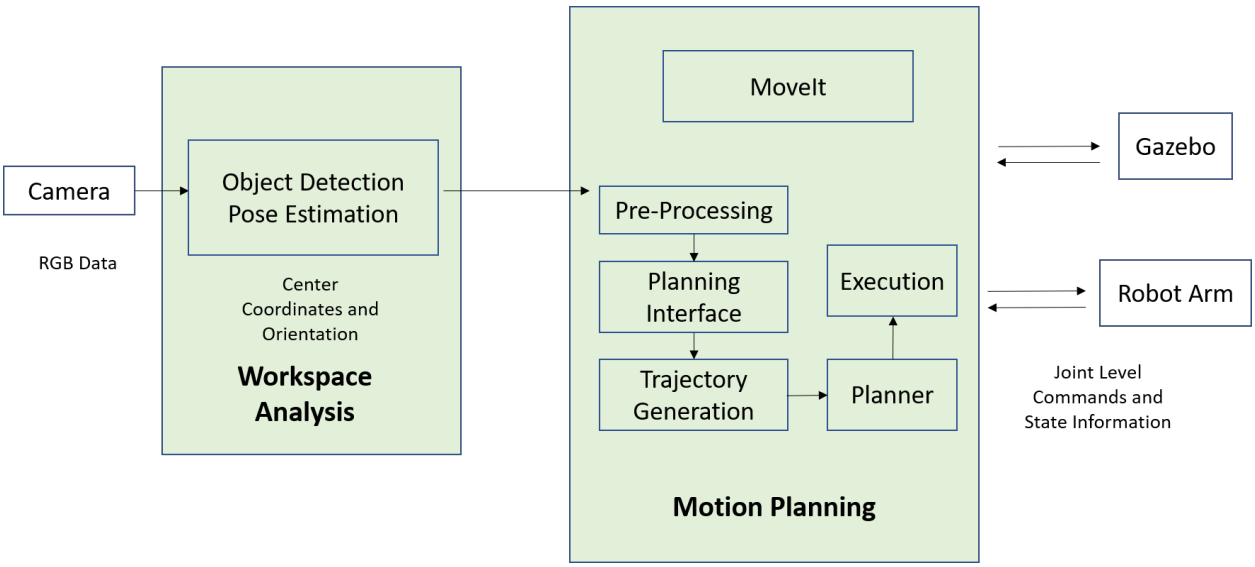


Figure 2.4: Vision based Automatic Control

its x, y, z axes. EPick Vaccum Griper features a built-in electrical vacuum pump. It connects directly to the robot's wrist and can handle a wide range of applications and is ideal for picking up even and uneven workpieces made of different materials such as cardboard, glass, dry sheet metal and plastic.

The entire system was developed in ROS on Ubuntu operating system. All the code was written in Python. We utilized OpenCV library for object detection and pose estimation. For motion planning Moveit platform was used. Motion planning algorithms were tested in simulation first before implementation on a physical system. Gazebo platform was used for simulation.

2.5 Simulation Environment

In order to recreate the system in a simulation environment, ROS Gazebo software is used. A bare-bone version of the robotic arm is created using Universal Robot Description Format (URDF) file. A URDF file describes the description of the robot and environment to ROS. This allows to import the model into simulation environment. The model is built in accordance with the dimensions of the real robot, the exact contours are replaced by cuboids. The cuboids are slightly bigger than the actual dimensions of the hardware, this is useful in having a higher clearance with the

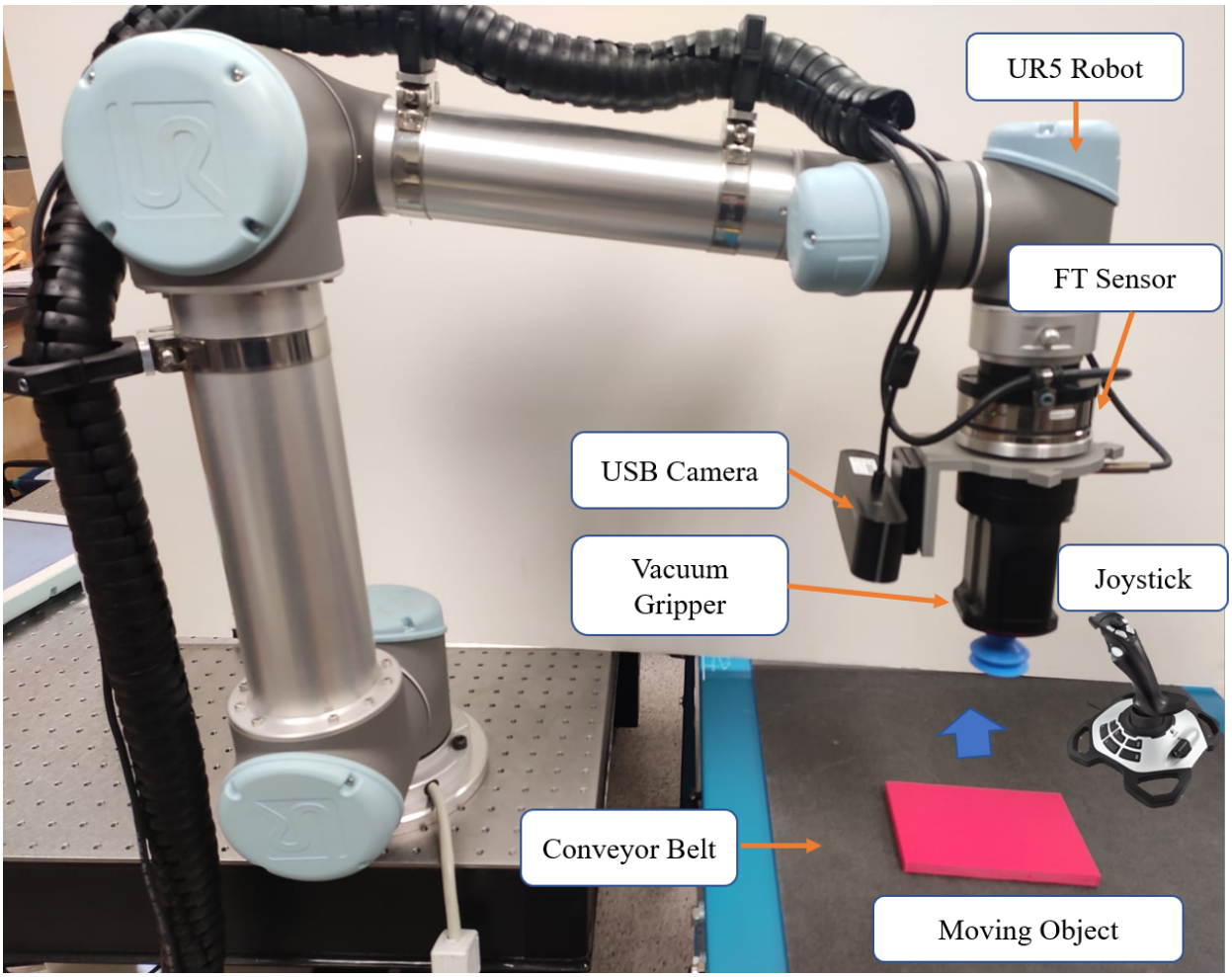


Figure 2.5: System Components

obstacles during motion planning for the actual arm. The conveyor belt is modeled as a long flat box (size $5 \times 1 \times 0.2$ m) for objects to slide on it. Objects were modeled to be red in color, size $0.1 \times 0.075 \times 0.025$ m. The object is moved by applying a constant force on it of 5 N. Bin for collecting the objects is modelled in the environment too, size $0.4 \times 0.4 \times 0.02$ m.

Additionally, the gazebo camera and vaccum gripper plugin were utilized to simulate the experiment. The mounting location of the camera and gripper were replicated in the simulation. The development of simulation environment followed the procedure from [34].

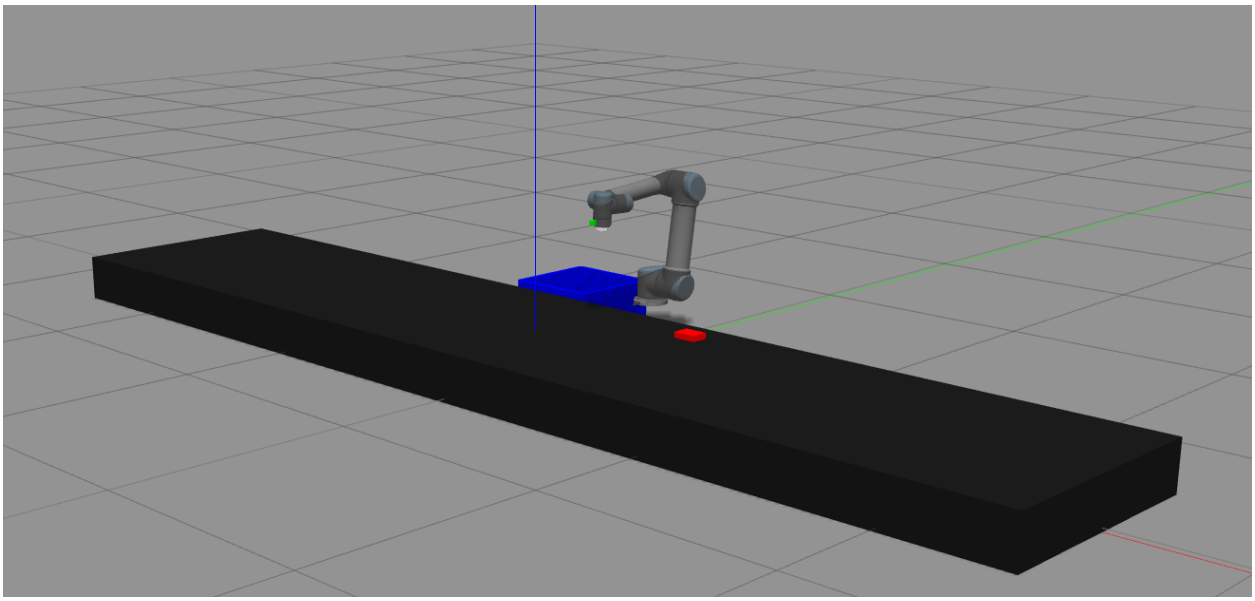


Figure 2.6: ROS Gazebo - Simulation Environment

3. ROBOT CONTROL WITH HUMAN IN LOOP

Chapter 2 defined two modes of operation for the system. First is Human-in-loop control, where human operator commands the robot using a general purpose joystick. Second is vision based automatic control. In this chapter, we would discuss the first mode of operation, i.e., robot control with human in loop using joystick. The goal of this study is to provide human operators intuitive interface to operate robotic manipulators. This chapter will look at development of a human-friendly operating frame and its joystick interface. The formulation and implementation of a hybrid control law for converting human commands to joint level controls are provided.

3.1 Joystick Interface Design

Instantaneous Surface Normal Approach (ISNA) defined in [1] is utilized for developing joystick interface. It uses mixed frame method to command the robot where orientation reference is provided in the world frame and translation reference is provided in robot end-effector frame. This way of generating reference commands allow a human operator to observe and command a robot end-effector from a third-person view. Human operator can intuitively control the robot end-effector to travel along unspecified trajectory in 3D space. Human operator first orient the robot end-effector along the normal plane of the desired trajectory with orientation commands in global frame and trajectory can be followed with translation commands in body frame. Human operator's joystick commands motions are closely mapped to the motions of the robot end-effector from a geometric perspective.

The command interface is encoded as follows for manipulator operation: We first denote worlds frame as $\{s\}$-frame, robot end-effector frame as $\{b\}$-frame and joystick frame as $\{j\}$-frame. A 3-axis stick is employed to encode reference orientation of end-effector, $\{b\}$-frame in $\{s\}$ and roll, pitch and yaw reference are recorded. A 2-axis navigator button is employed to command translation on the X_b-Y_b plane. A trigger and top programming buttons are employed to command end-effector to move along Z_b-axis and activate/deactivate gripper as shown in Figure 3.1.

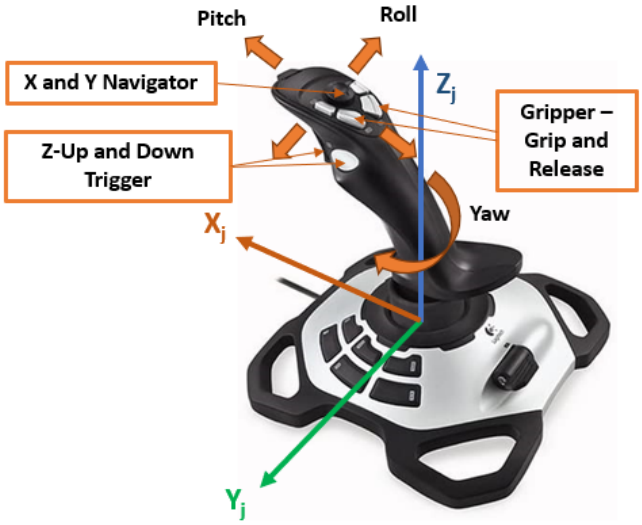


Figure 3.1: Joystick Interface

3.2 Design and Implementation of a Hybrid Control Law in the Mixed Frame

The commands provided by the human operator with joystick are required to be translated into joint velocities command to be sent to the robot. A hybrid control law is needed to track the command reference input - orientation inputs in $\{s\}$-frame and translation inputs in $\{b\}$-frame. Product of Exponential (PoE) methods [35, 36] and screw axis formulation were employed. These methods provide easy access to space and body Jacobians, in contrast to the traditional analytical Jacobian, which are crucial in operating the robot in end-effector frame. A PI controller was developed to convert the human operator commands into joint velocities. In the subsequent section we will provide controller design and define how inputs to the controller are generated.

3.2.1 Orientation Error

Desired Roll, Pitch and Yaw (ψ, θ, ϕ) inputs were available from 3-axis stick states. In the neutral stick position, joystick frame $\{X_j, Y_j, Z_j\}$ is aligned with robot world frame $\{X_s, Y_s, Z_s\}$

and the orientation of $\{X_j, Y_j, Z_j\}$ with respect to $\{X_s, Y_s, Z_s\}$ can be represented as

$$\begin{aligned} R_d &= \underbrace{R_Z(\psi)}_{\text{Yaw}} \times \underbrace{R_Y(\theta)}_{\text{Pitch}} \times \underbrace{R_X(\phi)}_{\text{Roll}} \\ &= \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - c\phi s\psi & s\psi s\phi + c\psi c\phi s\theta \\ c\theta s\psi & c\psi c\phi + s\psi s\theta s\phi & c\phi s\psi s\theta - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \end{aligned} \tag{3.1}$$

where $R_d \in SO(3)$ is the rotation matrix which contains the desired orientation reference. The end-effector orientation of the $R \in SO(3)$ with respect to robot world frame can be extracted from $T(\mathbf{q})$, transformation matrix of robot forward kinematics:

$$T(\mathbf{q}) = \begin{bmatrix} R & \mathbf{L} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3.2}$$

where $\mathbf{q} \in R^n$ is joint angles vector of an n-DOF robot, and $\mathbf{L} \in R^3$ is the position of end-effector in the robot world frame. $T(\mathbf{q}) \in SE(3)$ is computed based on the PoE formulation as

$$T = Me^{[\beta_1]q_1 \cdots [\beta_n]q_n} \tag{3.3}$$

where $[\beta_i] \in se(3)$ is the matrix representation of the i^{th} screw axis $\beta_i \in R^6$ in the body frame. $M \in SE(3)$ represents the end-effector configuration in the world frame when all q_i's are zero. Formulations of M, $[\beta_i]$, β_i are given in Robot Kinematics section ((3.17), (3.18), and (3.19)) for UR5 robot used for the study.

The orientation error vector $\Omega_e \in R^3$, then, can be calculated as

$$\Omega_e = \mathcal{F}[\log(R^T R_d)] \tag{3.4}$$

where $\log(R^T R_d) \in so(3)$ is the matrix logarithm of the matrix representation of orientation error, and $\mathcal{F}(\cdot) : so(3) \mapsto R^3$ is a mapping which takes a 3×3 skew-symmetric matrix to its vector form.

3.2.2 Translation Velocity with Adjustable Magnitude

Translation inputs are provided by the 2-axis navigator for x and y axes and two buttons for z, which provides three discrete signals that can be encoded into a vector $v = [x, y, z]$ where $x, y \in \{-1, 0, 1\}$ and $z \in \{0, 1\}$. The input vector v is normalized to generate a constant translation velocity reference, $V_c \in R^3$ as

$$V_c = \alpha[\frac{x}{||v||}, \frac{y}{||v||}, \frac{z}{||v||}]^T \tag{3.5}$$

where $\alpha \in R^+$ is the scaling factor which could be utilized to set translation velocity magnitude using slider on joystick.

3.2.3 Hybrid Control Law

With the orientation error, Ω_e, and translation error, V_c, at any given time $t \in R^+$, one can construct a control input term $H_e(t) \in R^6$ by combining those decoupled elements in the form of

$$H_e(t) = \begin{bmatrix} \Omega_e & V_c \end{bmatrix}^T \tag{3.6}$$

Then, the hybrid control law to find joint velocities for the control input is given by

$$\dot{\mathbf{q}} = J_b(\mathbf{q})^\dagger \left[K_P H_e(t) + K_I \int_0^t H_e(t) dt \right] \tag{3.7}$$

where $\dot{\mathbf{q}} \in R^6$ is the final output to the robot in the form of individual joint velocities and $J_b(\mathbf{q})^\dagger \in R^{n \times 6}$ is the pseudo-inverse of the body Jacobian for an n-joint robot represented in the $\{b\}$ frame. The body Jacobian $J_b(\mathbf{q})$ is defined as

$$J_b(\mathbf{q}) = \begin{bmatrix} J_{b1} & J_{b2} & \cdots & J_{bn} \end{bmatrix} \tag{3.8}$$

where J_{bi} for $i = 1, 2, \cdots, n$ are columns of the matrix $J_b(\mathbf{q})$ and are given by

$$J_{bi} = \mathrm{Ad}_{e^{-[\beta_n]q_n} e^{-[\beta_{n-1}]q_{n-1}} \ldots e^{-[\beta_{i+1}]q_{i+1}}} (\beta_i) \tag{3.9}$$

for $i = n-1, n-2, \cdots, 1$ with $J_{bn} = \beta_n$, where $e^{(\cdot)} : se(3) \mapsto SE(3)$ is the matrix exponential function and $\mathrm{Ad}_{(\cdot)} : SE(3) \mapsto R^{6\times 6}$ is the adjoint operator of a homogeneous transformation. Taking $T(\mathbf{q})$ in (3.2) as an operand, the result of the adjoint mapping is simply

$$\mathrm{Ad}_{T(\mathbf{q})} = \begin{bmatrix} R & \mathbf{0} \\ \mathbf{L}R & R \end{bmatrix} \tag{3.10}$$

The control gains in (3.7) are designed as follows. $K_P \in R^{6\times 6}$ is given by

$$K_P = \mathrm{diag}(p_\phi, p_\theta, p_\psi, 1, 1, 1) \tag{3.11}$$

and $K_I \in R^{6\times 6}$ is given by

$$K_I = \mathrm{diag}(i_\phi, i_\theta, i_\psi, 0, 0, i_f) \tag{3.12}$$

where scalars p_ϕ, p_θ, p_ψ and $i_\phi, i_\theta, i_\psi, i_f$ are proportional and integral gains for orientation and translation terms, respectively.

3.2.4 Normal Mismatch Correction using Force/Torque sensor

With a robot capable of measuring the moments at contact τ along the X_b-axis, Y_b-axis and Z_b-axis with force/torque sensor, orientation correction law can be defined. For peg-in-hole applications, the moments measured by sensor reflects the normal mismatch. Normal mismatch could be corrected by providing the moment offset as control input with orientation error. We can define the moment compensation term as

$$\tau_e = 0 - \tau \tag{3.13}$$

Note that $\tau_e \in R3$ from (3.14). This compensation term then can be added to orientation error to correct for mismatch:

$$\Omega_e' = \Omega_e + K_t \tau_e \tag{3.14}$$

$K_t \in R^{3\times3}$ converts moment input to orientation reference and is defined as

$$K_t = \text{diag}(k_x, k_y, k_z) \tag{3.15}$$

where, k_x, k_y, k_z are scalars With the new orientation error, Ω'_e, control input term $H_e(t) \in R^6$ can then be constructed as

$$H_e(t) = \begin{bmatrix} \Omega'_e & V_c \end{bmatrix}^T \tag{3.16}$$

Joint velocities for the control input can be calculated following the same procedure as described in section 3.2.3.

3.3 Robot Kinematics

The approach developed in this study is for UR5 robot and can be extended to any open-chain robots. A UR5 robot is shown in Figure 3.2 with all joints at their zero configuration. The axes $\{s\}$ denote robot world frame and axes $\{b\}$ end-effector frame, $s_1, s_2, \cdots, s_6 \in R^3$ to denote the joint rotation axes and $H_1, H_2, W_1, W_2 \in R^+$ denote link lengths.

The geometric information for calculating the robot kinematics is summarized in Table 3.1 where, each joint axis s_i and a selected point a_i on the axis are provided as vectors in the robot end-effector frame. The M matrix in (3.3) is defined as

$$M = \begin{bmatrix} 1 & 0 & 0 & -(L_1 + L_2) \\ 0 & 0 & 1 & -(W_1 + W_2) \\ 0 & -1 & 0 & H_1 - H_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.17}$$

and the i^{th} body screw axis in its matrix form is computed as

$$[\beta_i] = \begin{bmatrix} \mathcal{F}^{-1}(s_i) & -s_i \times a_i \\ \mathbf{0} & 0 \end{bmatrix} \tag{3.18}$$

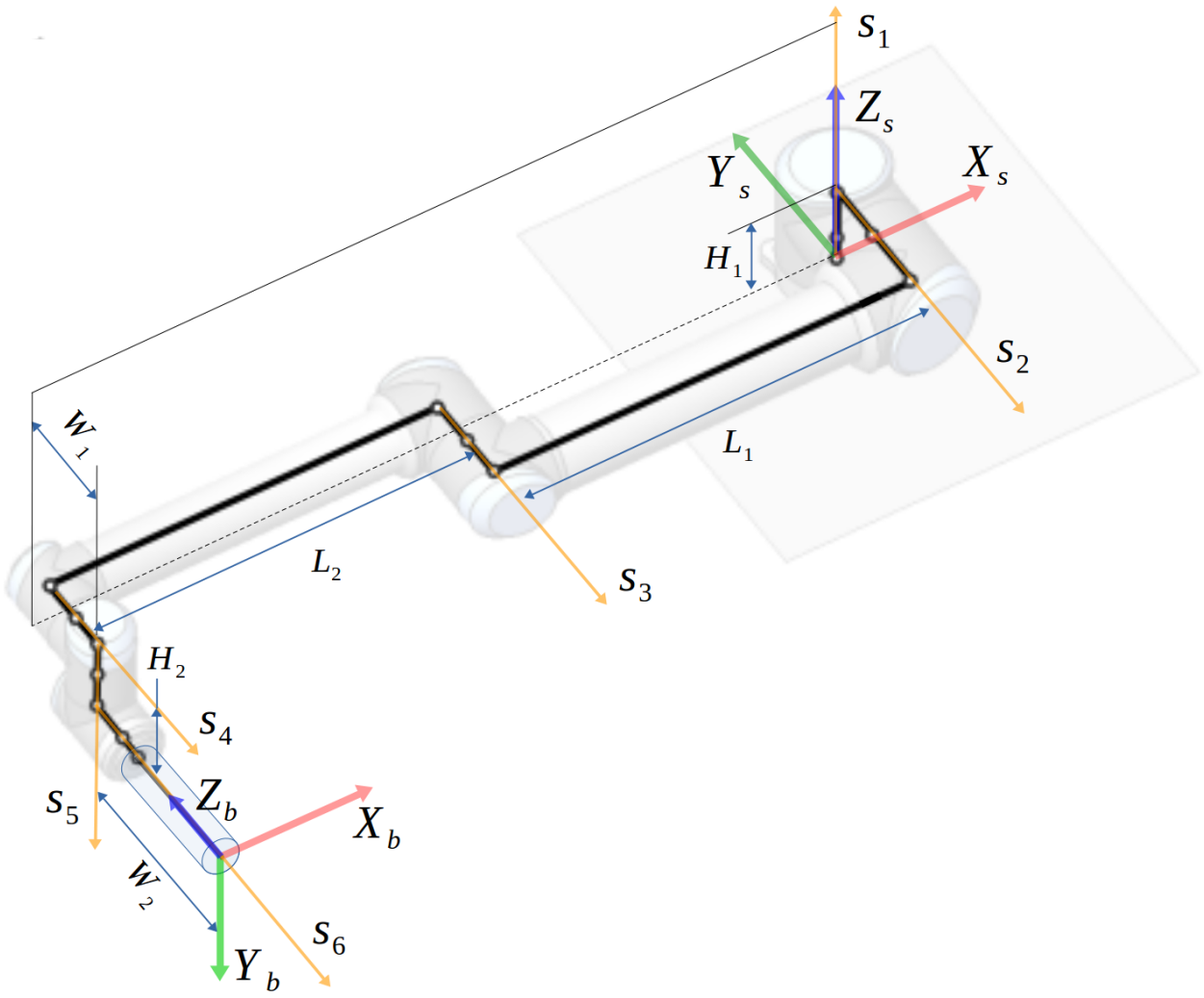


Figure 3.2: Frames and Axes Definition on UR5 [1]

where $\mathcal{F}^{-1}(\cdot)$ is the inverse mapping of $\mathcal{F}(\cdot)$ in (3.4). The corresponding vector form of the same screw axis as in (3.18) is then given by

$$\beta_i = \begin{bmatrix} s_i \\ -s_i \times a_i \end{bmatrix} \tag{3.19}$$

Table 3.1: Robot Geometry

Joint axis s_i in $\{X_b, Y_b, Z_b\}$		Selected point a_i on s_i in $\{X_b, Y_b, Z_b\}$	
s_1	$(0, -1, 0)$	a_1	$(L_1 + L_2, 0, W_1 + W_2)$
s_2	$(0, 0, -1)$	a_2	$(L_1 + L_2, -H_2, 0)$
s_3	$(0, 0, -1)$	a_3	$(L_2, -H_2, 0)$
s_4	$(0, 0, -1)$	a_4	$(0, -H_2, 0)$
s_5	$(0, 1, 0)$	a_5	$(0, 0, W_2)$
s_6	$(0, 0, -1)$	a_6	$(0, 0, 0)$

4. VISION BASED AUTOMATIC CONTROL

As described in Chapter 2, the solution approach presented in this study has two modes of operation. The control of the robot is traded between human-in-loop and automatic control. In Chapter 3, we described the Joystick Control Interface of the robot. An interface was designed to allow the human operator to control the robot intuitively in the end-effector frame. In this chapter we will examine automatic mode of operation using vision. First, work-space analysis (Object Detection and Pose Estimation) techniques are discussed, the location of the moving object was tracked in the robot world frame. The next step is for the robot to follow the moving object, position the end-effector gripper at the center of the object and pick it up. Motion planning and manipulation for the aforementioned two steps are described..

4.1 Object Detection

In order to have the robot pick up the object using vision, the first step is to detect the object from information available from the camera. The object detection process was broken down into three critical steps. The first step is Image Segmentation (determining the regions of interest by distinguishing object from background), the second step is Contours Recognition (identifying shape and boundary of the object), and lastly the third step is Contour Selection (processing and filtering noise in the data). Methodologies used for each of the steps is discussed in the subsequent sections. We have utilized the OpenCV Library [2] to develop the vision module.

4.1.1 Image Segmentation

Determining regions of interest in the image requires picking out shapes, colors, patterns, etc. and therefore, requires robust feature recognition methods. Many methodologies have been developed for object detection based on color, intensity or texture of pixels in the image. The work in [37] summarises different approaches and perspectives to Image segmentation problem. Some of the techniques are:

- Threshold based Segmentation - In this technique, the segmentation is done either by thresh-

olding the image data in color space range or by pixel histogram of the image. The classification of object from background is done with preset limits [38, 39, 40]. The method produces good results when the object and the background have sufficiently large variation in intensity values.

- Region based Segmentation - In this method, an initial seed point is chosen at random and then pixel neighbours are examined iterativly with a preset criteria to determine if they can be added to initial cluster or not [41]. This technique is useful for partitioning the image into regions and local/regional analysis of the image.

- Edge based Segmentation - This method first finds edges in the image (identifying and locating sharp discontinuities) and then intelligently connects the edges to form boundaries around the object. Various methods for edge detection are developed in literature [42].

Numerous other special-theory based algorithms like fuzzy clustering based or neural network based methods have also been developed for segmentation.

The simplest segmentation technique is color space thresholding method and we have utilized the same in this study to develop and test our vision based motion planning framework. We utilize the Hue/ HSV based thresholding to detect the objects and determine its relative pixel position. And with contour detection algorithms, object size and center coordinate are determined.

The image available from the camera is an RGB (Red-Green-Blue) image and one can simply threshold the channel based on this and get the region of interest. However, RGB values are highly sensitive to illumination making this to be an ineffective approach. The shadow on the object can affect the channel values. For this reason, we employ the HSV color space for the detection of objects.

4.1.1.1 HSV color space

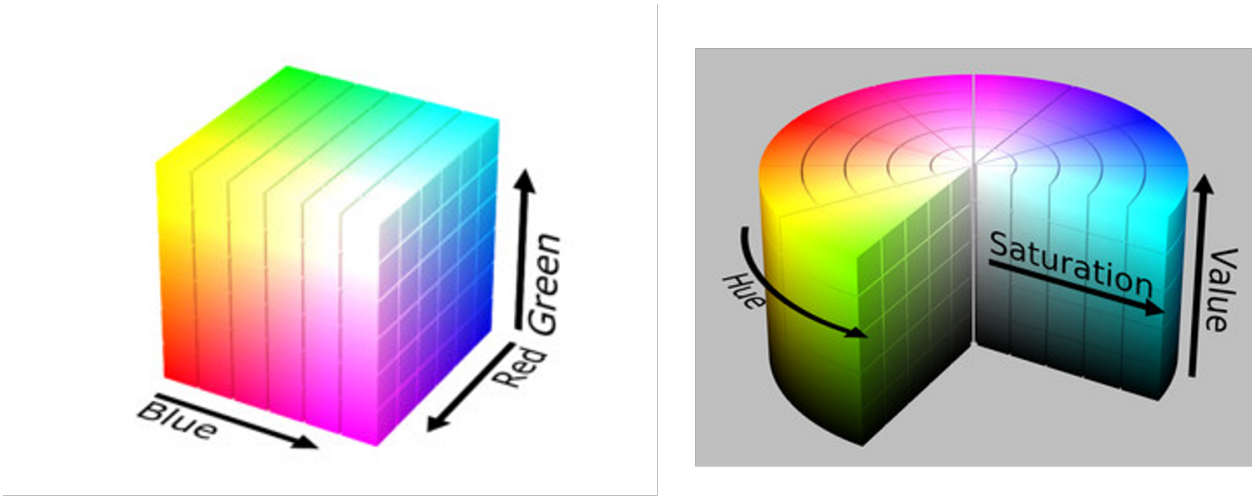
The HSV color space represents colors using three values which are: Hue (H), Saturation (S), and Brightness Value (V).

Hue: Measured by angle encodes color information. The value range is 0° - 360°, starting from red and counting in a counterclockwise direction, red is 0°, green is 120°, and blue is 240°.

Saturation: This channel encodes the intensity/purity of color. The value range is 0.0 - 1.0, the larger the value, the more saturated the color.

Value: The value ranges between 0 (black) to 255 (white) and encodes brightness of the color.

Figure 4.1 explains hsv distribution. Unlike RGB which is defined in relation to primary colors, HSV is defined in a way that is similar to how humans perceive color and the color/tint/wavelength is represented by just the Hue component. The



(a) (b)

Figure 4.1: RGB and HSV Color Space [2]

4.1.1.2 RGB to HSV Conversion

The RGB values are converted to HSV values using the formulation below: The color components are first normalized between 0-1.

$$\begin{aligned} R' &= \frac{R}{255} \\ G' &= \frac{G}{255} \\ B' &= \frac{B}{255} \end{aligned} \tag{4.1}$$

Maximum, minimum and chroma components of color are defined for ease of writing further definitions.

$$\begin{aligned} C_{max} &= max(R', G', B') \\ C_{min} &= min(R', G', B') \\ \Delta &= C_{max} - C_{min} \end{aligned} \tag{4.2}$$

Hue component is calculated as:

$$H = \begin{cases} 0, & \text{if } \Delta = 0 \\ 60(\frac{G'-B'}{\Delta}), & \text{if } C_{max} = R' \\ 60(\frac{B'-R'}{\Delta} + 2), & \text{if } C_{max} = G' \\ 60(\frac{R'-G'}{\Delta} + 4), & \text{if } C_{max} = B' \end{cases} \tag{4.3}$$

Saturation component is calculated as:

$$S = \begin{cases} 0, & \text{if } C_{max} = 0 \\ \frac{\Delta}{C_{max}}, & \text{if } otherwise \end{cases} \tag{4.4}$$

And lastly, Value component is calculated as:

$$V = C_{max} \tag{4.5}$$

We employed an OpenCV library function that can directly convert RGB model to HSV model. Now, for object detection, the HSV values are masked between minimum and maximum range for the color. The range identified and verified experimentally for this study is H from 0 to 20, and both S and V are between 100 and 255.

4.1.2 Contour Recognition

Next in order to recognize the contours, the masked image is converted full scale color to black and white (Image Binarization). The results of the binarization process can be seen in Figure 4.2. Gaussian blurring is used before the binarization in order to remove noise and enhance results from the binarization process. After the binarization of the image, contours are found using simple opencv contour recognition functions to find the outline of the positive areas returned from the binarization. The OpenCV function retrieves contours from the binary image using the algorithm [43].

4.1.3 Contour Selection

The final step of the panel recognition process is to sort through the noise left behind from the binarization of the image. There can be many contours in the field of view. The required contour will form a bigger contour compared to others because of the narrow color range. Areas of the contours are calculated, and thresholded such that contours within 90% of the largest contour found were considered to be object. Figure 4.3 illustrates all steps of the object detection process.

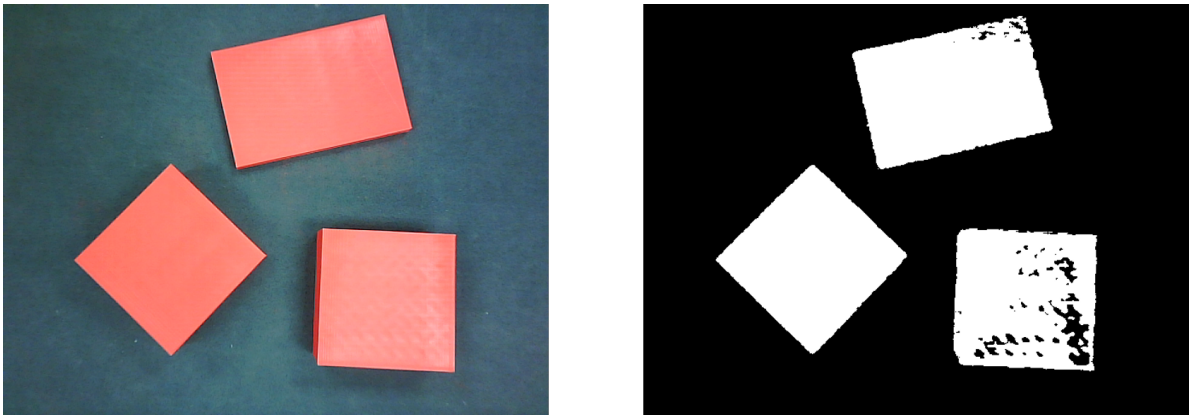


Figure 4.2: Image after Binarization Process

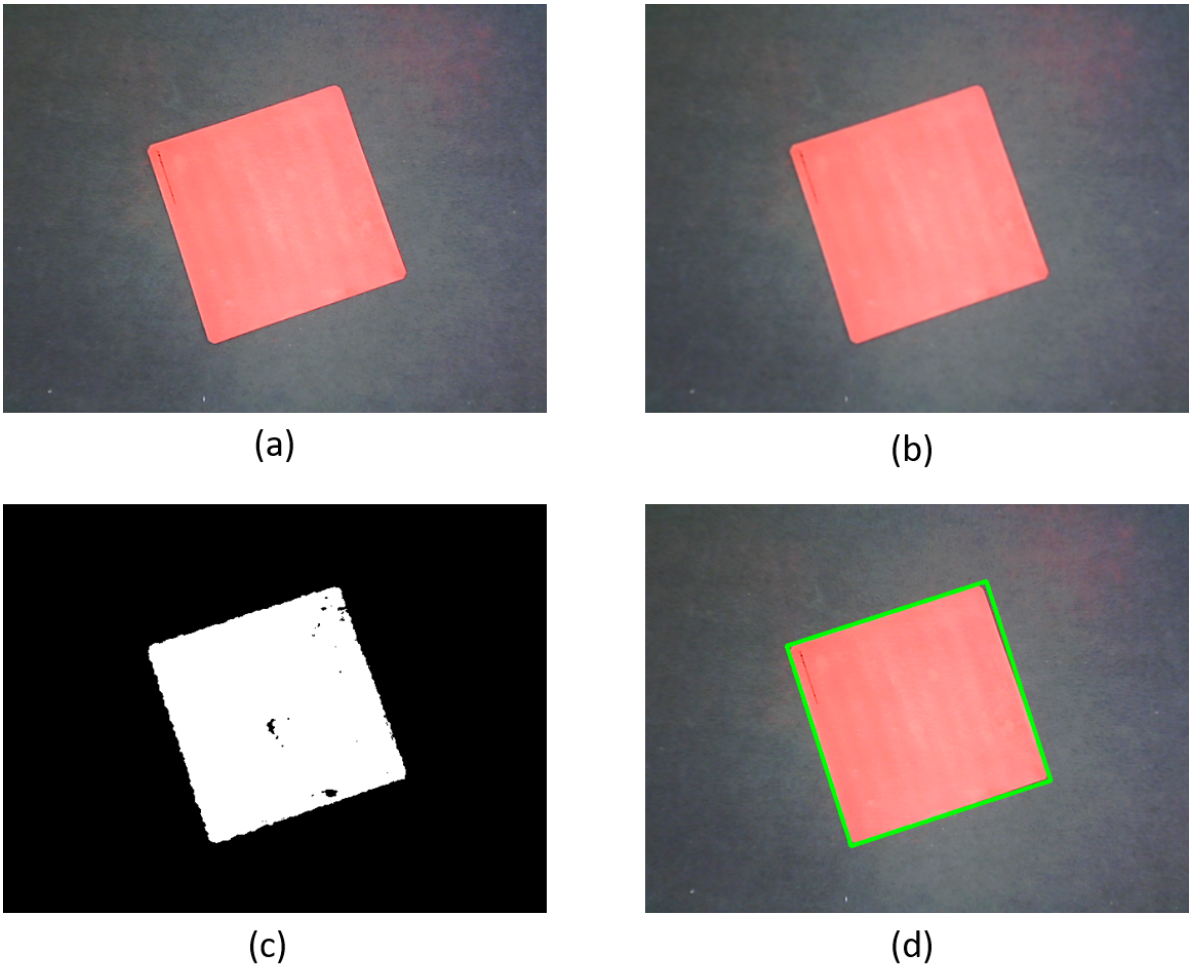


Figure 4.3: a)Image Captured by Camera; b)Image after noise removal with Gaussian Blur Filter; c) Image after binarization; d) Contour Detected

4.2 Pose and Orientation Estimation

After a contour was detected and selected, the coordinates of corners in pixel coordinates and area of the contour are available. Centroid coordinates in the plane (mid-point from corners coordinates) and orientation of the shape can be calculated as shown in Figure 4.4. Z-coordinates of the object is estimated by contour area ratio to actual object area. The scaling coefficient for Z-estimation is determined experimentally. A moving average filter was employed on data to smooth out short-term fluctuations and noise. This data can then be used for the robotic arm to move the end-effector to the necessary locations after information to relate pixel count to physical locations is available.

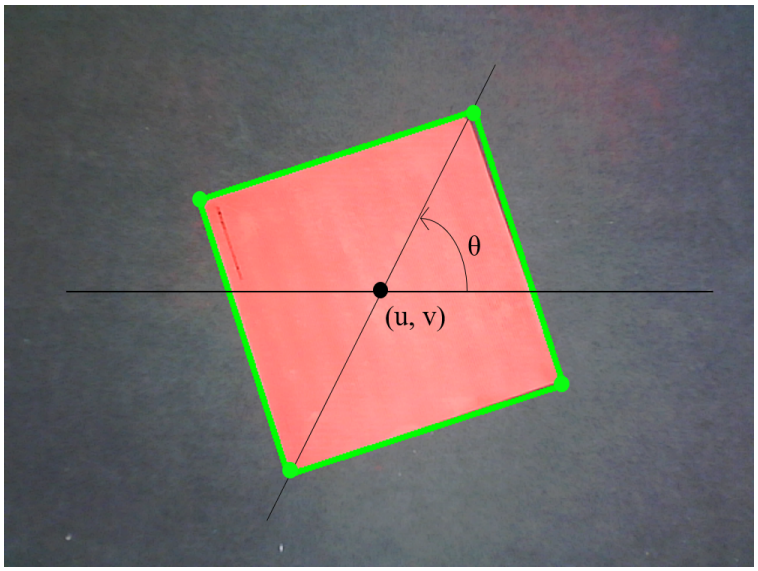


Figure 4.4: Estimated Position and Orientation in Image Frame

4.2.1 Transformation from Image Frame to Cartesian Frame

We have employed a pin-hole camera model, a simple mathematical model, for conversion from pixel coordinates to world coordinates. In this model, a scene view is formed by projecting 3D points into the image plane using a perspective transformation. Figure 4.5 illustrates the model.

$$x = z\frac{(u - c_x)}{f_x} \tag{4.6}$$

$$y = z\frac{(v - c_y)}{f_y} \tag{4.7}$$

where:

- (x, y, z) are the projected coordinates in the optical world frame
- (u, v) are coordinates of the projection points in pixel coordinates
- (c_x, c_y) is center coordinates of image frame
- f_x, f_y are the focal lengths of the camera

The coordinates (x, y, z) from pin-hole model are now needed to be transformed to robot world frame (X, Y, Z), and is done as:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = R|t \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{4.8}$$

$R|t$ is a joint rotation-translation matrix which is the transform from optical world frame to robot space/ world frame. The camera focal lengths f_x and f_y are estimated experimentally.

The coordinates (X, Y, Z, θ) obtained from here are now sent to the motion planning module to plan approach and retreat strategy. We have assumed that the camera lens introduces minimal tangential and radial distortions to the images, and thus, these effects are neglected.

4.3 Motion Planning with MoveIt!

The motion planning strategy constitutes of finding a path to the target pose while following system constraints, like avoiding singularities, joint limits, and obstacles in the environment. The

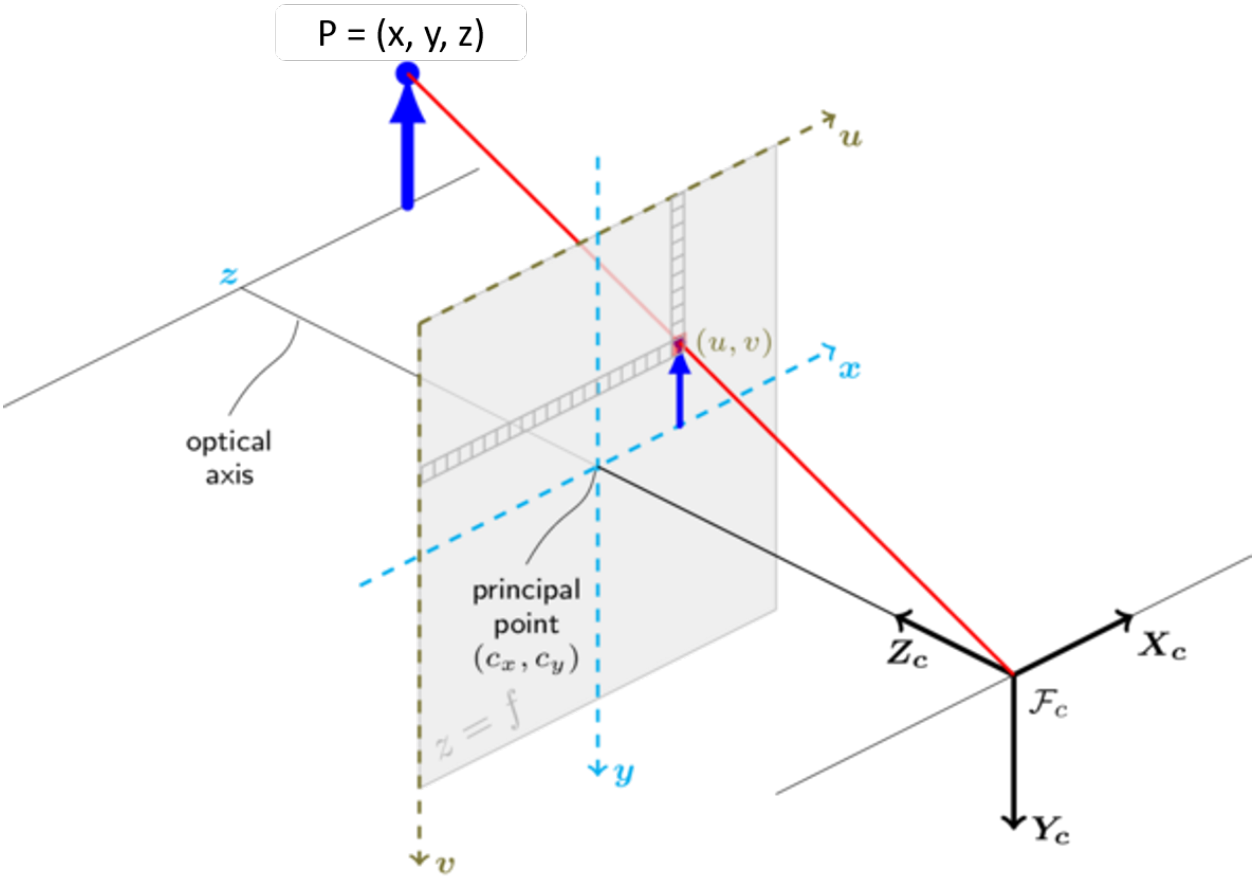


Figure 4.5: Pinhole Camera Model [2]

target pose is available from the vision module and the motion planning module must find smooth robot trajectory to the object. We employed the MoveIt! platform to develop the motion planning algorithm.

4.3.1 MoveIt! Interface

MoveIt! is a software framework widely used in the robotics community which provides wide range of capabilities for manipulation, motion planning, control and mobile manipulation. Various motion planning libraries are already integrated in it, including the Kinematics and Dynamics Library (KDL), the Open Motion Planning Library (OMPL) and the Fast Collision Library (FCL).

The architecture of MoveIt! can be seen in Figure 4.6. The move_group is the central node that connects and integrates all individual nodes for transmission of information and provides a set

of ROS actions, services and topics for users. The user can interface with the move_group node in three ways to access actions or services and modify environment:

- C++ : move_group_interface package
- Python : moveit_commander package
- GUI : Rviz (ROS Visualizer) plugin

The robot description is provided via URDF file which describe a robot's and environments physical description to ROS. The Semantic Robot Description Format (SRDF) file complement the URDF and specifies joint groups, default robot configurations, additional collision checking information, and additional transforms that may be needed to completely specify the robot's pose. The move_group node accesses all this information and is also configured through ROS param server.

The node interacts with robot and sensors in system via various ROS topics and command the robot via action server protocol. The current position of each joint/ Joint State Information is read from /joint_states topic. Transform information is monitored using ROS TF library which provides robot's pose information. The node sends commands to controllers on the robot using JointTrajectoryAction interface. Planning scene monitor is where information about the robot and environment is updated.

MoveIt! interacts with different motion planners from various libraries using a plugin interface and default planner is configured using OMPL. OMPL computes motion plans using sampling-based algorithms. The complete list of motion planners in OMPL is present in [44]. Planners are configured using MoveIt! Setup Assistant and one can add kinematic constraints for planner to work with. Collisions and constraints are checked while planning. Allowed kinematic constraints setting include:

- Position Constraints - To restrict position of robot arm in a defined space.
- Orientation Constraints - To specify roll, pitch or yaw limits for the arm.

- Visibility Constraints - To restrict a point on the arm to lie within visible region for some sensor.
- Joint Constraints - To set joint limits.
- User-specified constraints - Any additional application specific constraints can be defined with user callback.

Th Joint constraints are specified in URDF file.

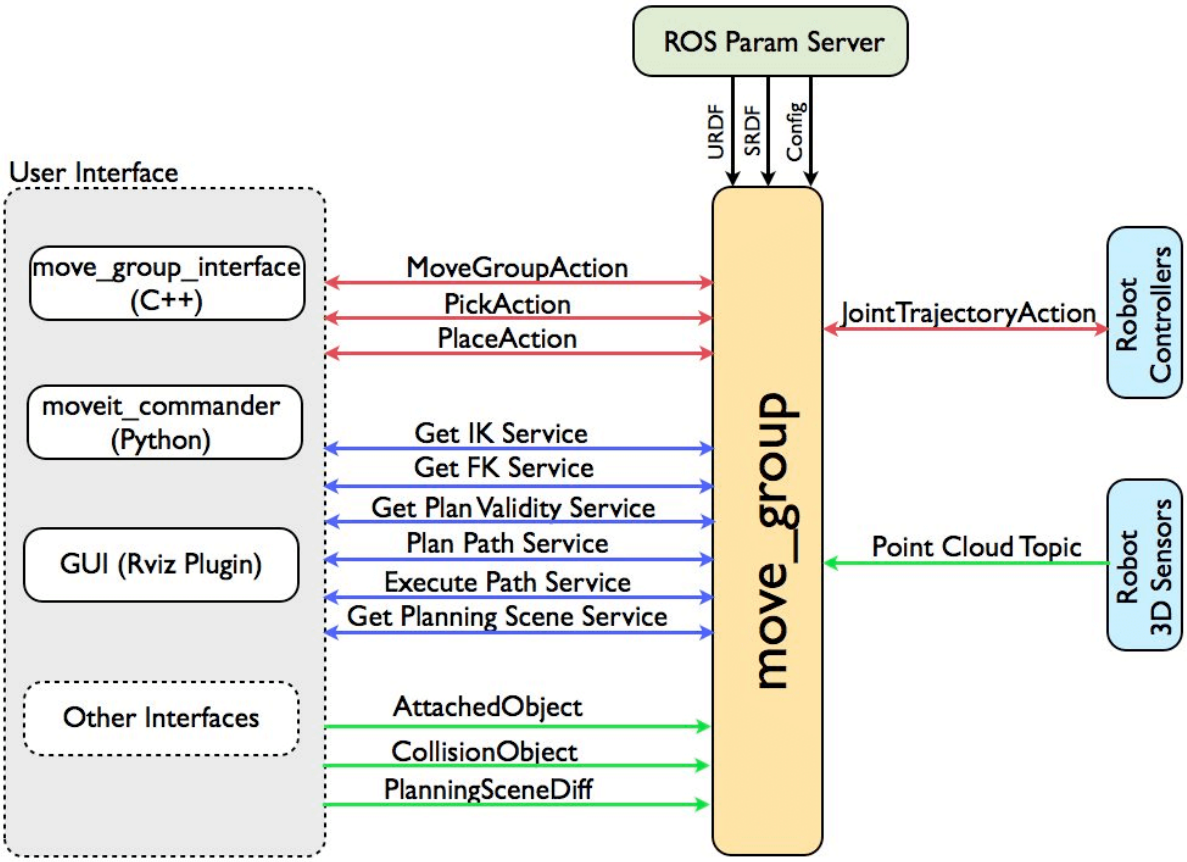


Figure 4.6: Move Group Architecture [3]

In this study, we developed the planning interface with moveit_commander python package. UR5 robot and conveyor belt description were added via URDF file as with details as described in

Chapter 2 section 2.5. Gazebo camera and vaccum gripper plugin were added into the simulation. A dimensional representation of camera and gripper were added to MoveIt! Planning scene for collision avoidance during motion planning. Rapidly Exploring Random Tree (RRT) algorithm [45] from OMPL library is the default motion planning library that MoveIt! uses and same has been utilized for this study. A tree based planning algorithm which uses random sampling to construct a path. It grows trees of path rooted at start configuration using random samples in the space. A greedy heuristic is utilized to find solution to move towards the goal pose.

4.3.2 Motion Planning

The python interface of MoveIt!, moveit_commander, comes integrated with the following APIs: MoveGroupCommander class, a PlanningSceneInterface class, and a RobotCommander class, which were utilized in this study.

MoveGroupCommander class allows to use existing libraries to set parameters for path planning, like target goal tolerances, reference frames, trajectory constraints, acceleration and velocity limits etc. Path start position, goal and planning time are also set/reset via MoveGroupCommander. Planning Scene Interface allows to add objects, constraint planes into the environment. Both MoveGroupCommander and RobotCommader allows to read current state of the robot.

MoveGroupCommander Class allows motion planning request to planner to move the arm to target Joint Goal Pose or the end-effector target pose. The planner generates path between target and current location while checking for collisions and constraints in the environment. The planners will generate kinematic paths they may not obey velocity and acceleration constraints or the arm can be out of bound at the start of planning, etc. The path generated by planner needs to be converted to time-parameterized trajectories. Various motion planning adaptors are set to be used for pre-processing planning requests and post processing of the planning response:

- FixStartStateBounds - This adaptor fixes for when the start robot configuration is with joints outside joint limits specified in MoveIt Environment. This is important if joint limits on physical robot and for planner are different, which may be done to add safety measures. In

our application, it is useful as transfer of control from Human-in-loop control to automatic control happens when object is detected by camera. The robot configuration at which control is traded may have joints in out of bound joint limits for planning library leaving planner unable to plan the trajectories. The FixStartStateBounds adapter corrects the robot configuration by moving in with joint limits. However, only small corrections like this are advisable and a parameter for the adapter specifies the fixable limits.

- FixWorkspaceBounds - This adaptor defines default planning workspace. The values preset in the interface are a cube of size 10m X 10m X 10m, same has been considered for this study.

- FixStartStateCollision - In case the robot links are in collision state at the start, this adapter reconfigures the arm to a no coliision state within a preset allowable range.

- FixStartStatePathConstraints - This adapter corrects for violation of path constraints for the start position. The current start configuration of the arm is moved to new location where path constraints are followed.

- AddTimeParameterization - This adapter allows for generating time-parameterized trajectories by applying velocity and acceleration constraints. Move_group will use maximum velocities and accelerations to generate trajectories which will also obey the joint level limits preset for the system.

The motion planner needs to specified of target position to generate trajectories. To pick-up a stationary object one just need to estimate target pose of the object and planner takes care of trajectory generation while taking care of all the constraints of the system. However, picking up a moving object is a difficult problem. The planner needs to generate dynamically feasible grasp trajectory for the object. The timing of trajectory and time of object grasp during the trajectory are to be considered. Other problem to be addressed is smooth pickup of the moving object. The end-effector is needed to track the object while grasping to avoid mishandling of the object. To

address the challenges described above, we have developed four planning modules for different phases of operation:

- Pre-grasp phase: At first when the object is detected, the end-effector is not close to the object. The first goal of motion planning is to align the robot end-effector with the moving object. In this phase, the end-effector is aligned with the moving axis and orientation of the object.

- Object Tracking phase: For smooth grasping of the object, the end effector is needed to track the object. In this phase, the speed of the object is estimated and planner tracks the object.

- Grasp Acquisition phase: Tracking the object, the end-effector moves closer to the object and secures it's hold.

- Post-grasp transport: Once the end-effector has achieved a stable grasp of the object, the robot now moves to a pre-determined target drop location. In this phase object is added in the planning scene and is considered rigidly attached to the robot.

We utilized Go To Goal Pose planner for setting start and drop location. A prior default start and drop position of robot can be defined. The start pose is default state of the robot and drop location can either be defined as bin location where the object is dropped or the location where control will be transferred to human. Cartesian Path planner was utilized for rest of the path planning sequence. This planner generates a straight line time parameterized trajectories that follow the path specified by waypoints inputed to it. Waypoints between the object and end-effector are generated using proportional controllers for different phases of operations. The subsequent sections will describe each module in details. One can specify the step size after which configurations are computed (eef_step) and the jump_threshold which specifies the maximum distance in configuration space between consecutive points in the resulting path. The planner returns the actual Robot Trajectory in joint space and a performance measure of how much of the path planned is similar to desired path with respect to waypoints inputed. We have specified step size to be 0.01 meters for Pre-Grasp Manipulation and 0.02 meters for Object Tracking. The acceptable plan was thresholded to

be at least matching 80% the desired trajectory as specified by waypoints. The subsequent sections will present the design strategy employed to define the waypoints for each phase of planning and manipulation.

4.3.3 Waypoints Generation

As discussed above, the move_commander interface has been utilized for this study, and one needs to specify waypoints to generate trajectories. Proportional controllers are employed to generate waypoints between robot and target pose and MoveIt! planning algorithms plan the path connecting the waypoints. We will discuss the strategy employed for waypoint generation in different phases of motion planning below.

4.3.3.1 Pre-Grasp Manipulation Phase

To approach the detected object, at any time instant the target pose is available from vision module (x_t,y_t,z_t,θ_t) and current end-effector pose is available from Joint State measurements and robot kinematics model (x_i,y_i,z_i,θ_i). The goal, in this phase, is to align the end-effector with the object, therefore at every moment error between ee pose and target pose is measured and next waypoint is defined proportional to the error value. The time-parameterized trajectory in joint space is generated with Cartesian Planner Library, which generates a straight line segment path between the current pose and target pose. The plan is executed on the robot only when more than 80% of the trajectory planned matches with trajectory extrapolted between waypoints. The phase ends when the object is aligned with moving axis and orientation of the object within predefined tolerance limits as shown in Algorithm 1.

Algorithm 1: Pre-Grasping Manipulation

Input: $x_t, y_t, z_t, \theta_t, x_i, y_i, z_i, \theta_i, K_{x1}, K_{y1}$

initialization;

while $y_t - y_i$ > *Goal Position Tolerance or* $\theta_t - \theta_i$ > *Goal Orientation Tolerance* **do**

Generate Pose_waypoints;

$Wpose\_x = x_i + K_{x1}(x_t - x_i)$;

$Wpose\_y = y_i + K_{y1}(y_t - y_i)$;

Orientation Goal;

Orientation_goal = quaternions(θ_t)

$fraction, plan =$

$compute\_cartesian\_path((Pose\_waypoints, Orientation\_goal), eef\_steps, jump\_threshold)$;

if *fraction > 0.8* **then**

execute(plan);

end

end

4.3.3.2 Object Tracking

The robot once aligned with the object needs to track the moving object to have zero relative velocity between the end-effector and object to activate smooth grasp. The waypoints, therefore in this phase of operation, are then computed proportional to the estimated speed of object, as explained in Algorithm 2. We estimate speed of the object from the target pose data available from vision module. A moving average filter was employed to smooth out noise fluctuations in measurements. Once again the trajectories planned are executed when more than 80% of path matches with inputs to the planner.

Algorithm 2: Object Tracking

Input: $x_t, y_t, z_t, \theta_t, x_i, y_i, z_i, \theta_i, K_{x2}, K_s$

initialization;

$EstimatedSpeed, x_{speed} = K_s Moving\_Average(x_{ti} - x_{t(i-1)})$;

Generate Pose_waypoints;

$Wpose\_x = x_i + x_{speed} + K_{x2}(x_t - x_i)$;

$fraction, plan =$

$compute\_cartesian\_path(Pose\_waypoints, eef\_steps, jump\_threshold)$;

if *fraction > 0.8* **then**

    execute(plan);

end

4.3.3.3 Grasp Acquisition

Once the end-effector is tracking the moving object, it is moved closer to to make contact and activate grasp. The waypoints are computed proportional to error between z_t from vision module and z_i, current end-effector position. The vision module is develped with a 2D camare and Z-coordinates of object is being estimated with area of the contours. Therefore, additionally, force-torque sensor mounted on the robot is utilized to establish contact with the object. The gripper is activated while moving at a constant speed when a desired contact force has been recorded by the sensor.

4.3.3.4 Post-Grasp Manipulation

After the object is grasped, the robot now moves to a pre-determined target drop location. Go To Goal Pose MoveIt library is utilized for planning the motion. The object grasped is now added to the planning scene as a rigid attachment to the robot end-effector to allow for collision free planning.

4.3.4 Obstacle Avoidance

MoveIt! is integrated with Fast Collision Library (FCL) library [46] to plan collision-free trajectories. The same has been utilized in this study. This library allows to plan robot path around different types of obstacles:

- Meshes - Mounting plane of the robot
- Primitive shapes - Defined Objects in Environment like conveyor belt and dropping bin
- Octomap - Unknown dynamic objects cab be integrated with MoveIt! using external sensor data. Octomaps [47] allow to include world representation with an octotree map generated with external sensors.

In this study, we specified prior known obstacles in the environment. However, integration of dynamic obstacles using Octomap is not considered in the scope of study.

5. EXPERIMENTAL DESIGN AND RESULTS

This chapter presents the hardware platforms, design of experiments and test results for the methods developed in this study. The framework proposed has two modes of operation; control with human-in-loop (Joystick Control) and automatic control using computer vision. The two methods were first independently tested and then combined for testing of collaborative control. Therefore, we will first illustrate testing and results of joystick control, followed by demonstration of vision based automatic control. Lastly, integrated results on the overall strategy are presented.

5.1 Demonstration of Human-in-Loop Control

A framework is developed which allows human operator to command the robot intuitively in the end-effector frame [1]. To test the effectiveness of the developed method, an experiment was conducted simulating pick-and-place type applications. A physical platform is employed for the experiment which includes a 6-DOF UR5 robot, a Logitech Freedom 2.4 GHz wireless joystick, an ATI AXIS-80 force/torque sensor, 3D printed end-effectors and operation platforms. The ATI force/torque sensor provides force and torque measurements along its x, y, z axes.

The setup for the experiment is shown in Fig. 5.1 where a block is attached to the robot end-effector, and there are two 3D printed structures with the corresponding cavities that will fit the block. The experiment requires the human operator to command the robot to move the block from one platform to another. The cavities are designed such that the block can only fit when it is correctly oriented with respect to the opening space. There are no additional sensors except human observation.

The system software is built in ROS [48] with different nodes (programing unit in ROS). A detailed network structure and control flow is shown in Figure 5.2. Joy reader node registers joystick inputs, command reference is generated in reference generator node and Hybrid Controller converts the joystick commands to joint level inputs to the robot. The software communicate with

[1] As specified in Contributors and Funding Sources section, a part of section 5.1 is adapted from joint work done by the student with Dr. Zongyao Jin [1].

the robot via joint_state and ur_script topics and with sensor via netft_utils topic.

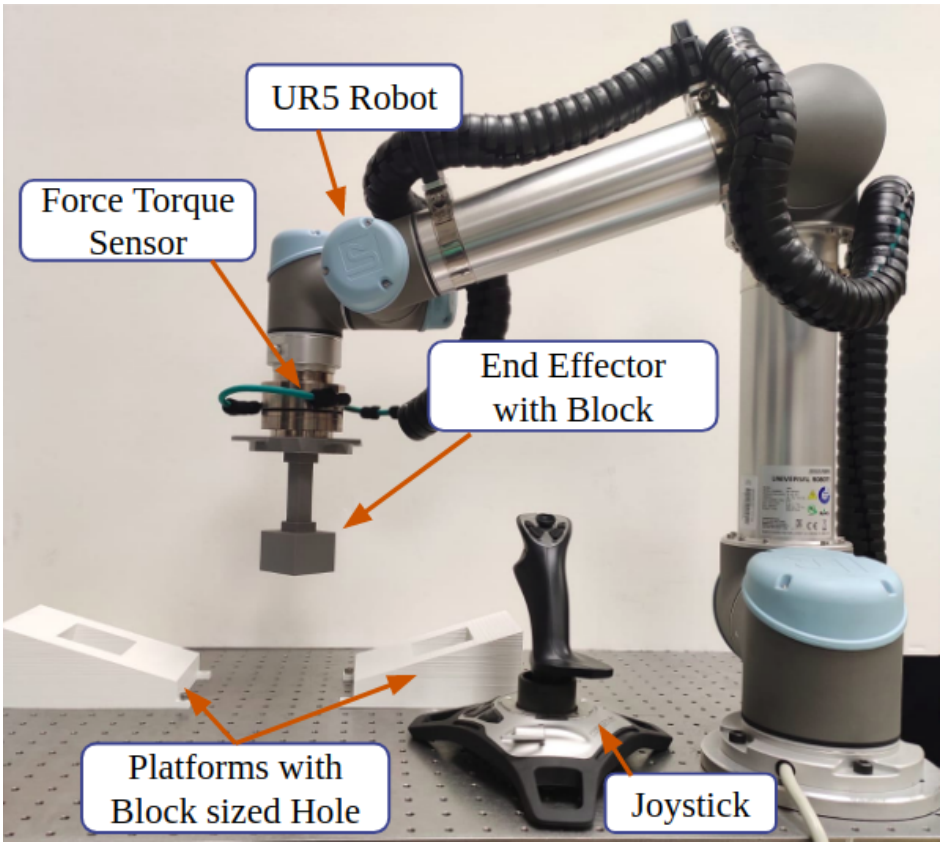


Figure 5.1: Setup for Material Handling Experiment

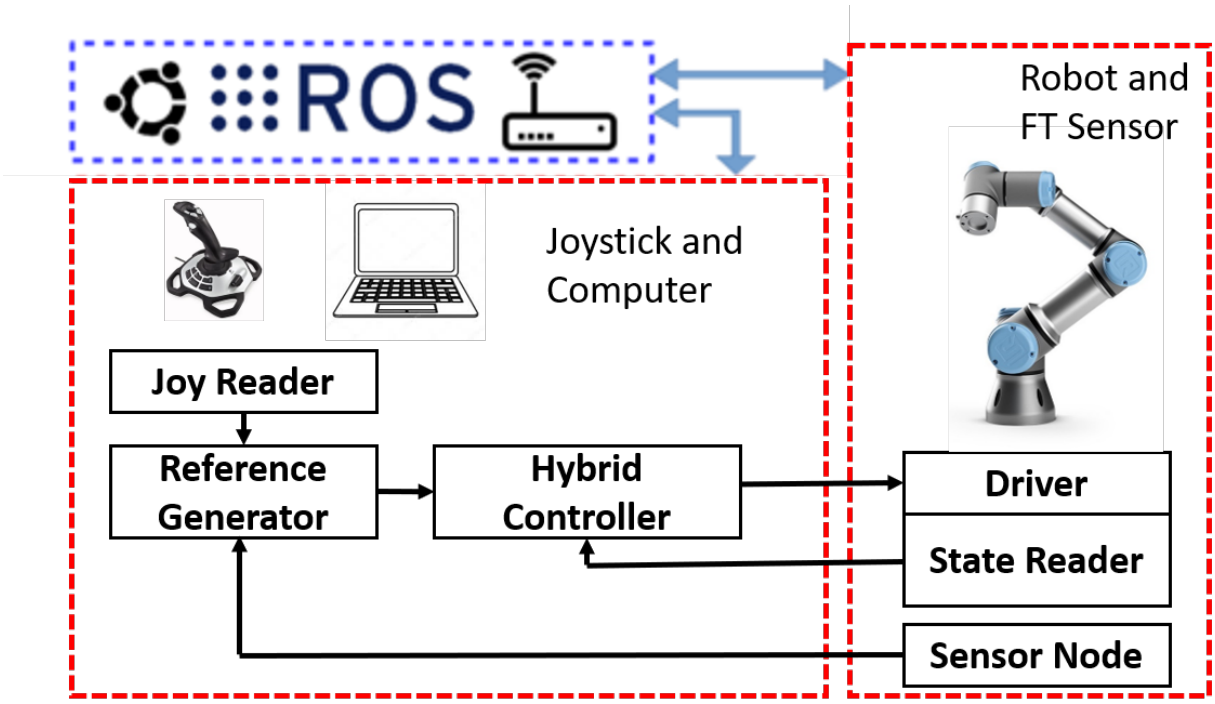


Figure 5.2: ROS based Software Structure and Control Flow

The controller performance was recorded by recording end-effector orientation with respect to the reference commands sent by the joystick. Fig. 5.3 shows the plots of robot performance against roll, pitch and yaw reference inputs provided via the joystick. We observe that the end-effector tracks the desired inputs effectively. Sequence of robot configurations during the operation are presented in Fig. 5.4. The robot is shown to be picking up the block from the left platform and dropping the block in the right platform. Figure 5.5 provides a wire-frame diagram of robot configurations. In subplot (a) the robot picks up the object and subplot (b) shows the placing the object in the right platform. The wire-frame color in each of Figure. 5.5 (a) and (b) is shaded from light to dark to highlight passage of time as the task progresses.

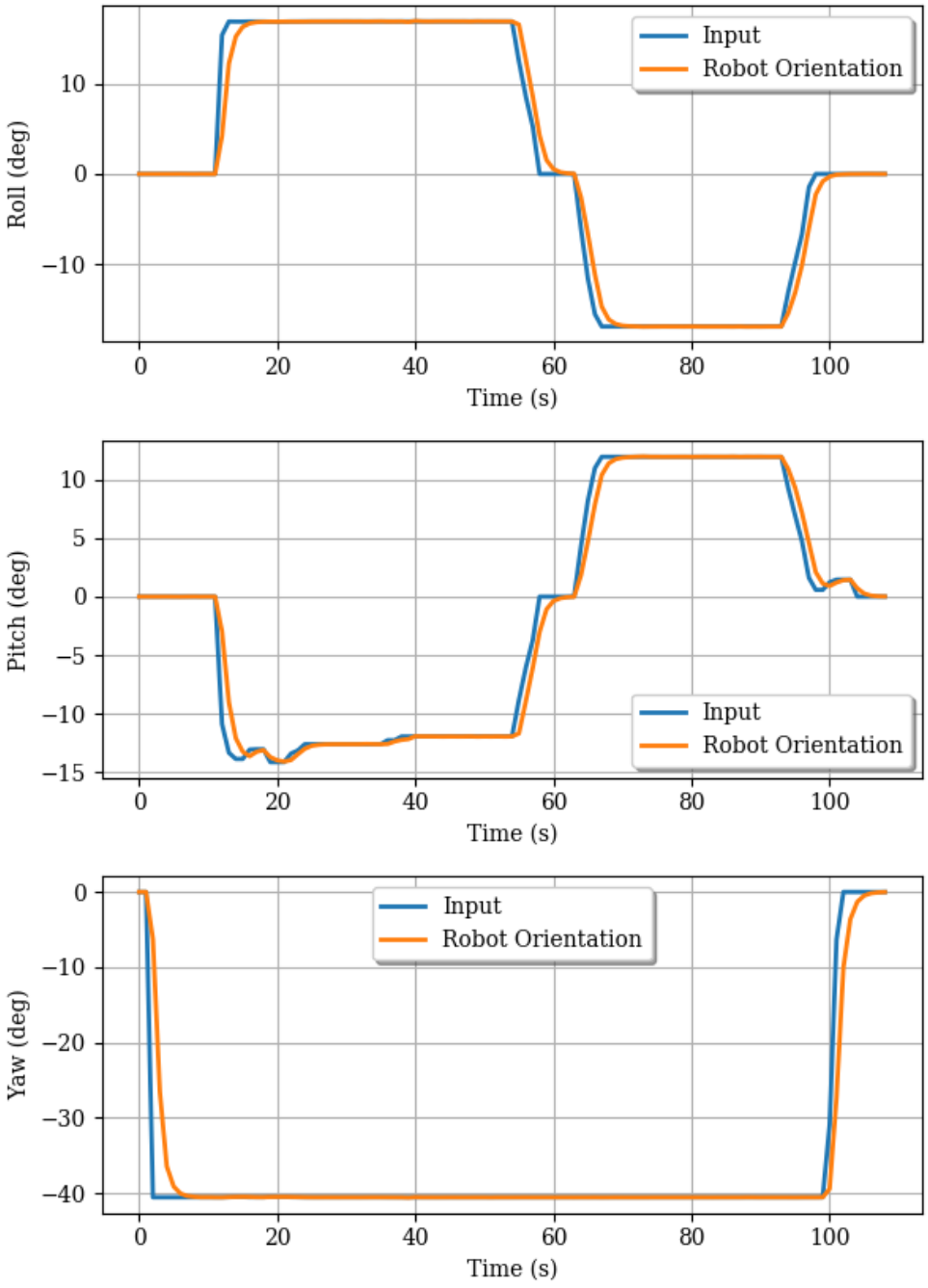


Figure 5.3: End-Effector Orientation Tracking with Joystick Input

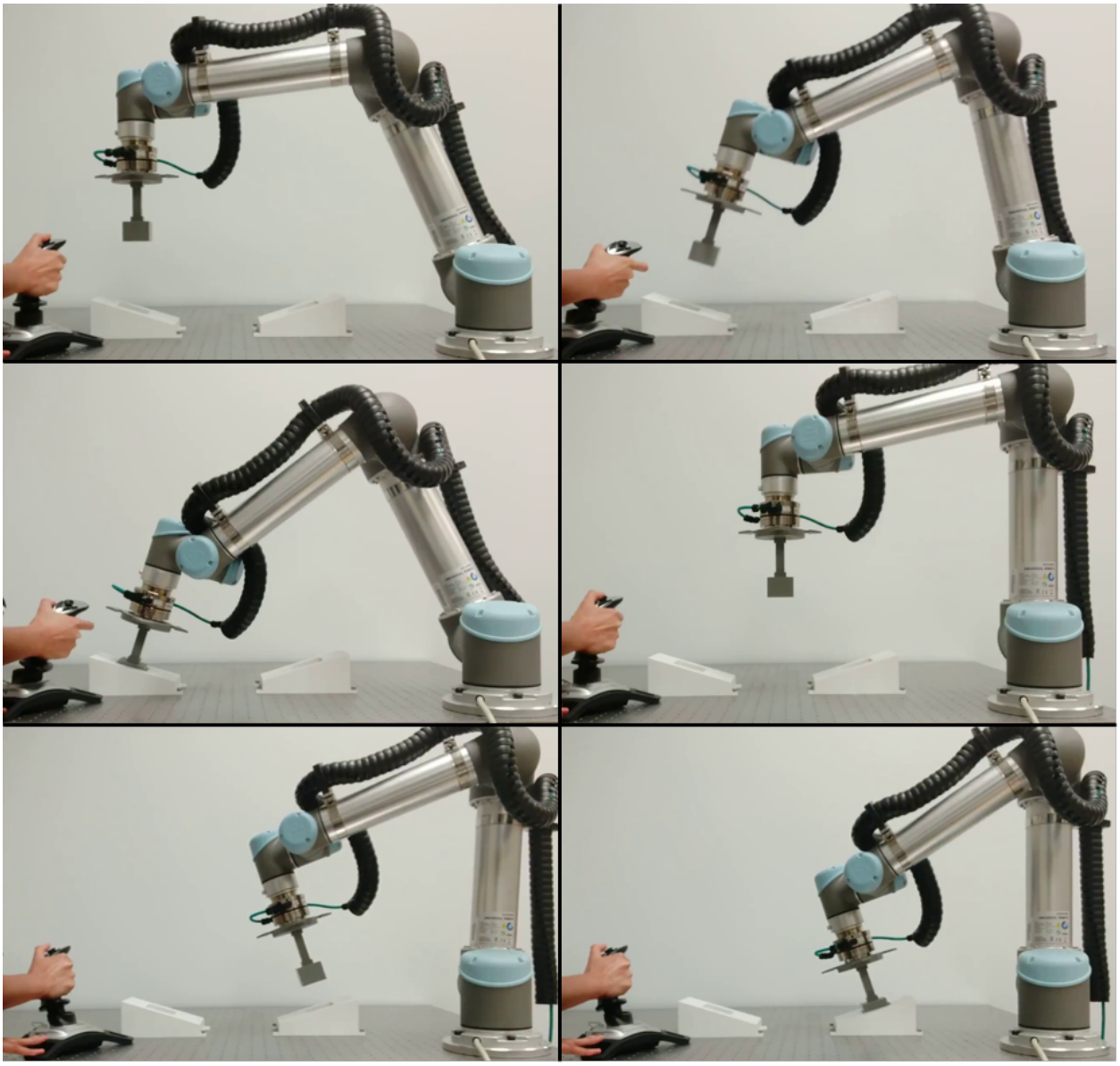
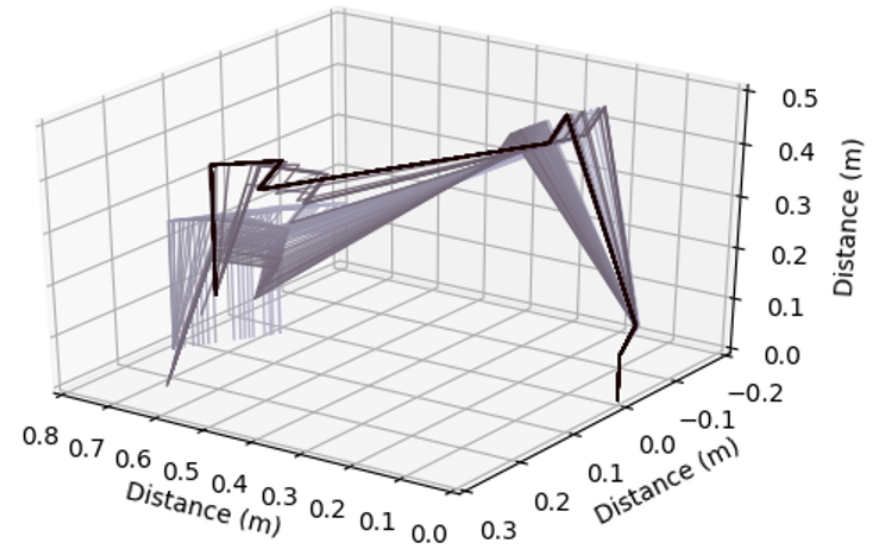
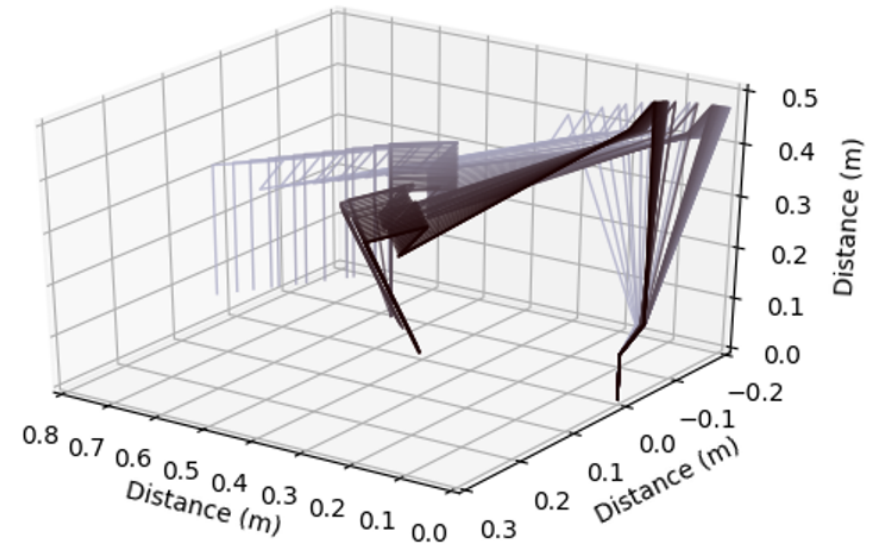


Figure 5.4: Sampled Robot Configurations during Joystick Interface Experiment



(a)



(b)

Figure 5.5: Robot Wire-frame Diagram during Operation (wire frame is lighter shade at the start of the task and gradually becomes darker as the task progresses in time)

5.2 Demonstration of Vision based Automatic Control

A framework has been developed to provide assistance to operator where automatic control based on vision takes over the control of the robot once the object is detected by the vision system. The vision based motion planning algorithms were developed in MoveIt! interface. The simulations were first performed in ROS Gazebo Environment and then the integrated system is validated with Human-in-Loop module on a physical platform for testing effectiveness of collaborative system (presented in section 5.3).

The working space in Gazebo included a UR5 Robot, Vaccum Gripper, USB Camera and Conveyor Belt. The conveyor belt carried workpiece (a box of size 0.1 m x 0.075 m X 0.1 m) over it with randomized initial y-location and moving with a speed of 7.5 cm/s. The boxes were designed to be red in color and the HSV range from (0,255,255) to (20,255,255) were utilized for detection by vision algorithms. The siulation setup is shown in Figure 5.6.

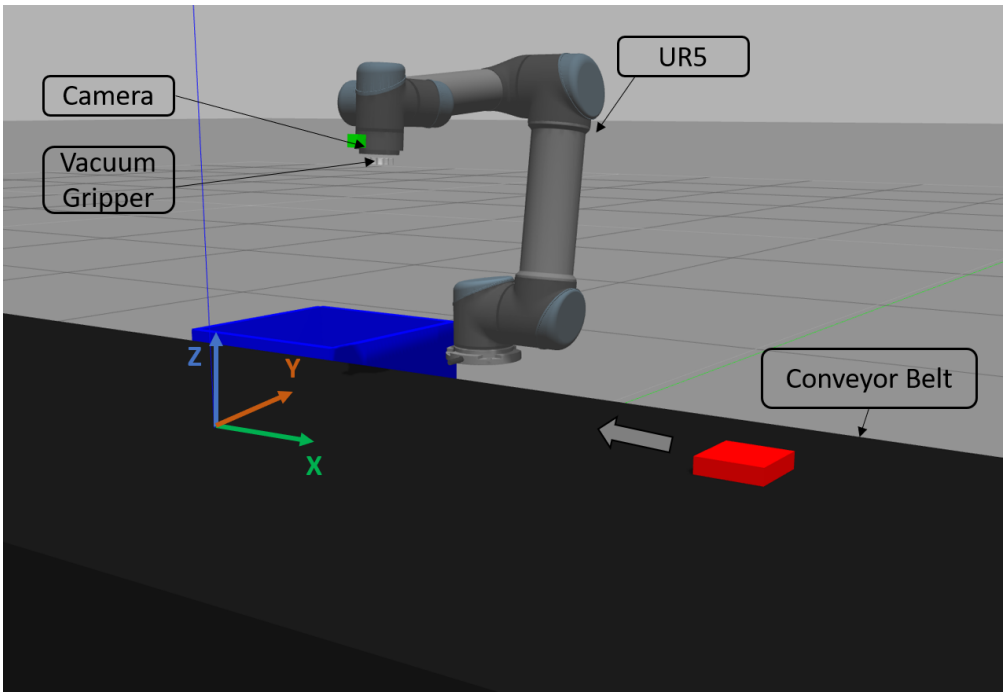


Figure 5.6: ROS Gazebo Environment for Vision based Automatic Control Testing

The motion planning in this case included tracking and grasping the moving object while avoiding collision with the conveyor belt. A preset tolerance of 0.01 m in position and 0.1 rad in orientation were permitted. Figure 5.7 shows the sequence of the robot position while grasping the object. The results of EE pose planned versus actual are presented in figure 5.8. As described in Chapter 5, the motion planning phases; Pre-Grasp Transport, Object tracking, and Grasp Manipulation can be identified. The effectiveness of the method can be observed from end-effector closely following the planned path.

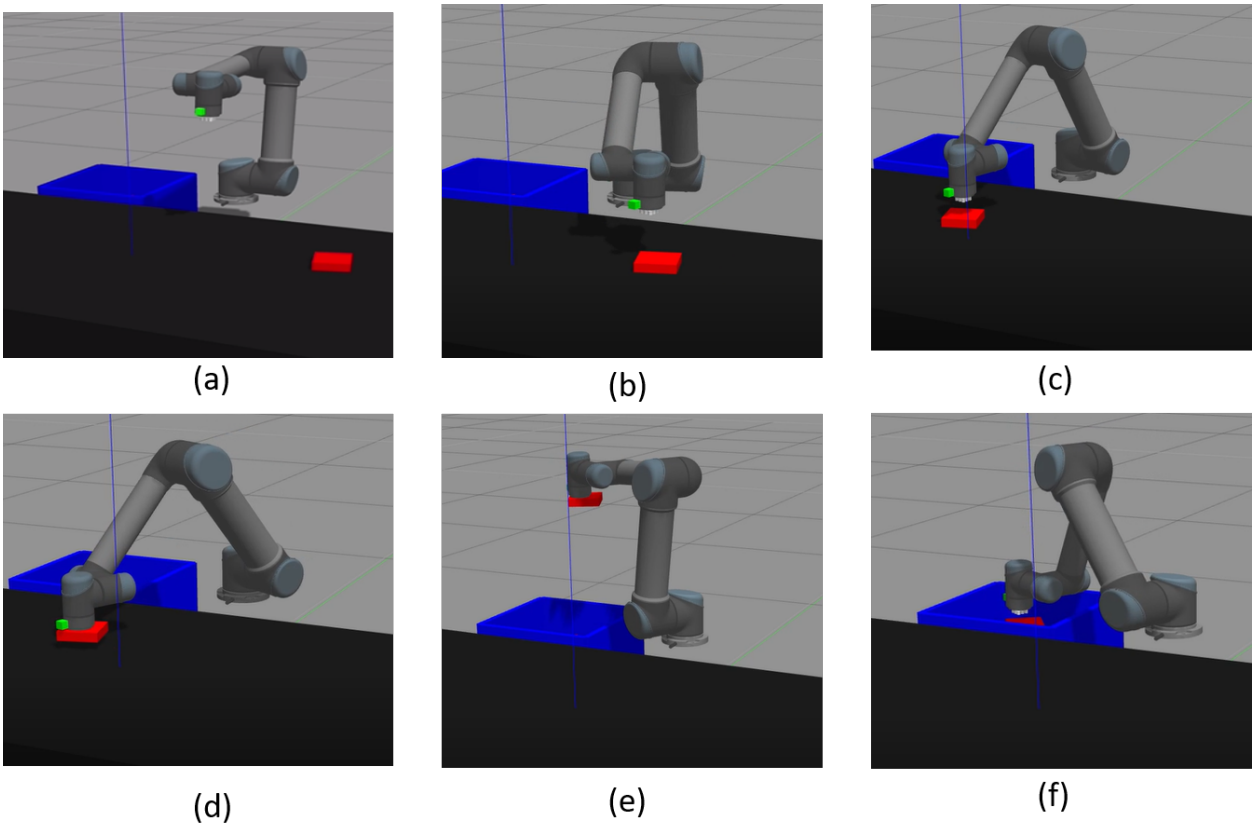


Figure 5.7: Sampled Robot Configurations during Vision based Automatic Control

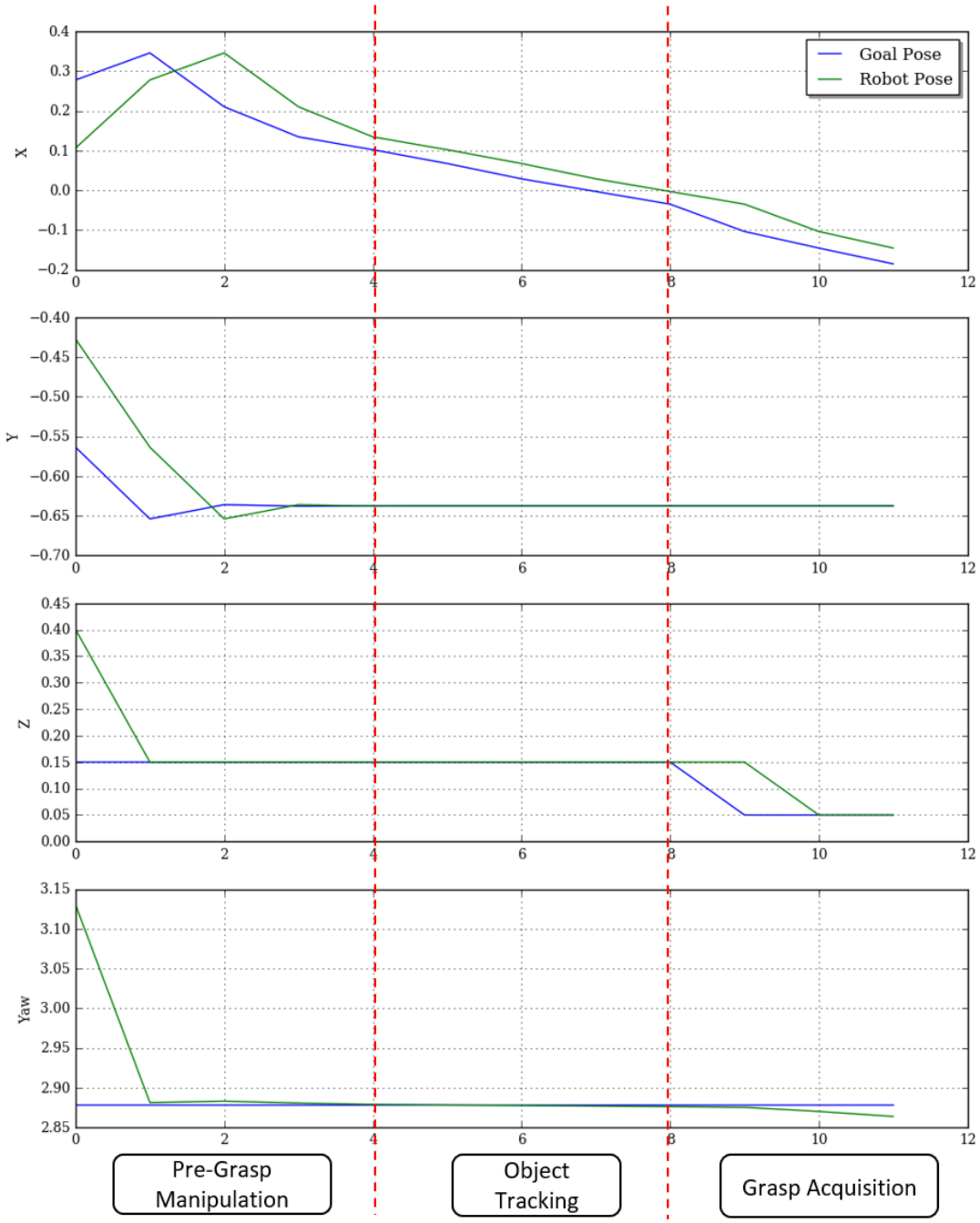


Figure 5.8: Planned versus Actual EE Trajectory in Gazebo

5.3 Validation of Integrated System

Implementation of Human-Robot Collaborative system was again done on UR5 robot. The platform consisted of ATI Force-Torque sensor, Astra Pro Orbec camera, Robotiq vaccum gripper, which are mounted on the robot end-effector. Logitech Extreme Pro joystick was utilized for Human-in-Loop Control. The experimental setup is shown in Figure 5.9. In order to simulate changing environment where the location of the object may not be known a-priori, the task is defined in the following manner: Human operator is asked to grasp the moving object on conveyor belt and drop it at the designated dropping location. Initial configuration of the robot is randomly set and it may not have the camera on robot end-effector facing the conveyor belt to have object in camera field of view for vision module to pickup. To complete the task, first, the human operator brings the robot over conveyor belt using the joystick interface and positions the robot to have object identified by vision module. Once the object is detected by the vision algorithm, automatic control will take over to grasp the object. The object once grasped will be moved to a pre-determined pose and the human then takes over to navigate the robot to place the object at dropping bin.

The system software is again built in ROS for vision based control as well. A detailed network structure and control flow is shown in Figure 5.10. Image reader node registers camera inputs and vision node detects object and target pose is estimated. Waypoint Generator generates valid intermediary goals between current pose and target pose depending on different phases of Grasp Manipulation. The waypoints generated by this node is then sent to MoveIt! planner to generate time-parameterized trajectories in joint space for execution. MoveIt! planner communicates the planned trajectory to the robot using JointTrajectoryAction action server and joint state information is read via joint_states topic. FT Sensor data is again read from netft_utils topic and camera data is read from usb_cam/raw_image topic. Tracker and Gripper topics were created to access target pose from vision node and to send activation commands to the gripper respectively.

Figure 5.11 shows the sequence of robot configuration during the trial. The 3D trajectory of the complete EE movement is plotted in Figure 5.12. The segments of the plot are color coded

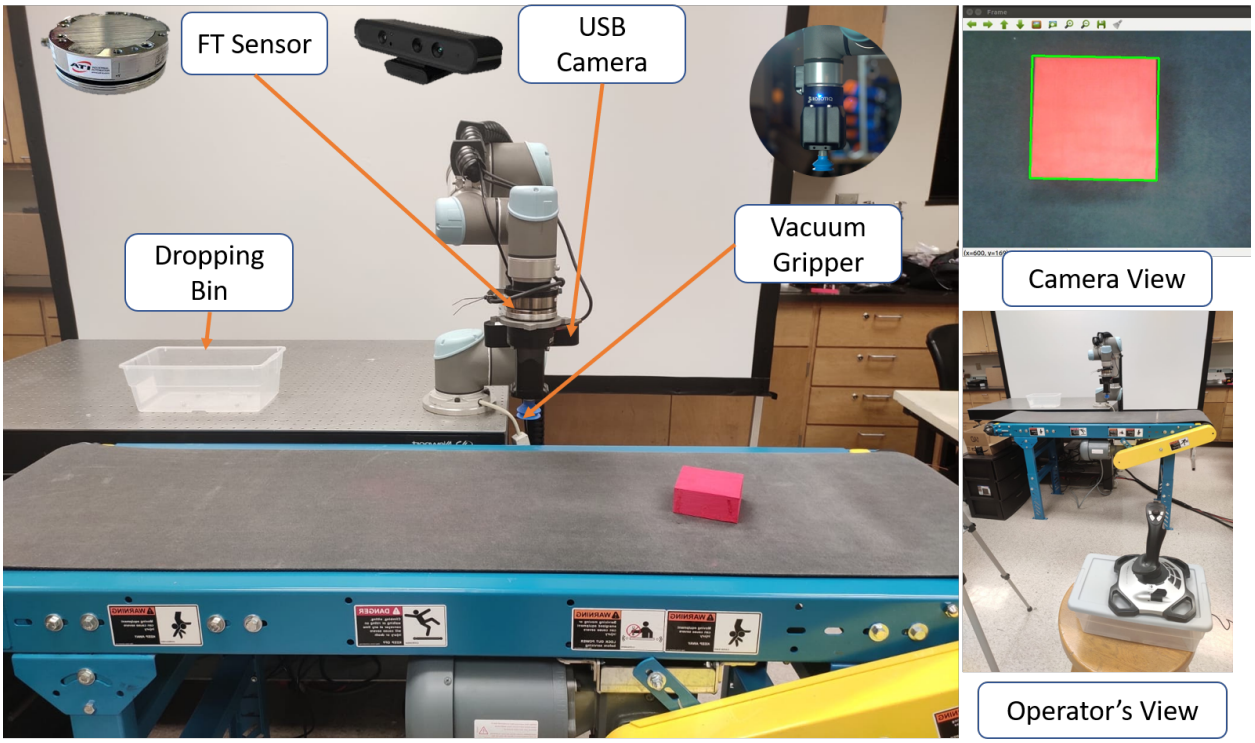


Figure 5.9: Experiment Setup for Collaborative System Testing

to indicate the different modes of operation at different stages of the experiment.The trajectories where the human is in loop is in shown red (intialize) and black (dropping the object), automatic grasping and post grasp transport trajectories are shown in blue and green in color respectively. We also plotted Planned Pose versus Actual Pose for vision based automatic motion planning operation. The results are shown in figure 5.13. Different phases of grasp manipulation can again be identified. The experiment again showed effectiveness of both Joystick Control framework and Vision based Motion Planning on a physical system. However, can note that the target path and actual path during automatic phase of operation on physical system did not follow each other as closely as in the simulated system. This is attributed to shortcomings of vision detection mechanisms under environmental conditions. HSV based detection algorithm used in this study was observed to be affected by the lighting in the room leading to errors in pose estimation.

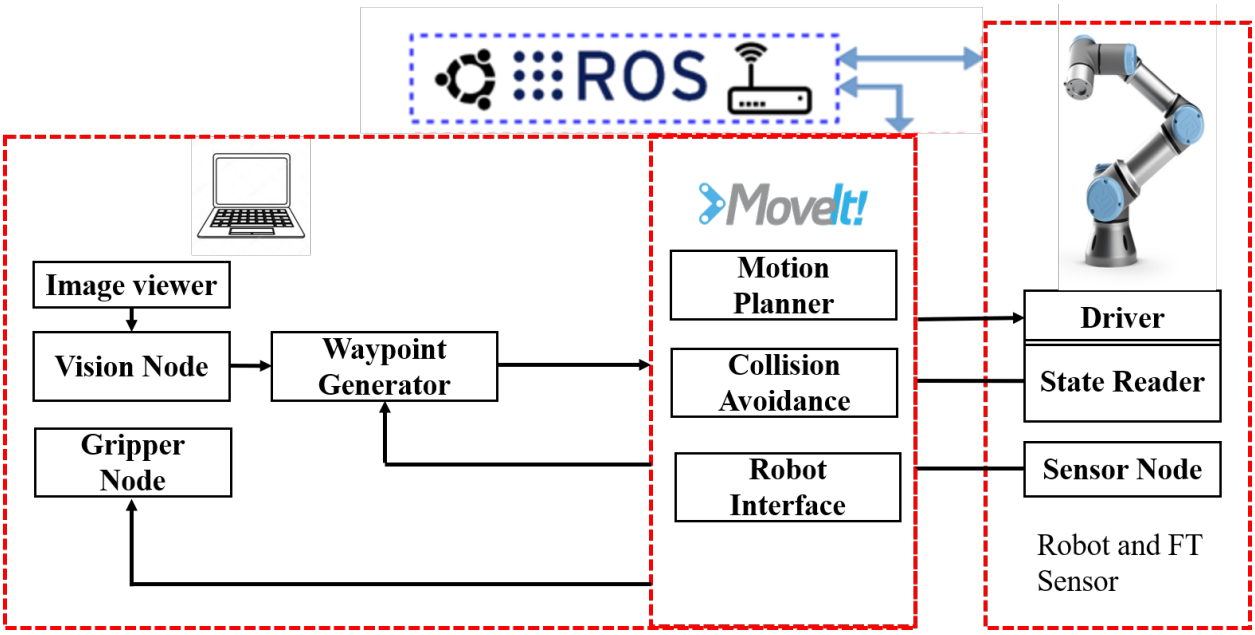


Figure 5.10: ROS based Software Structure and Control Flow

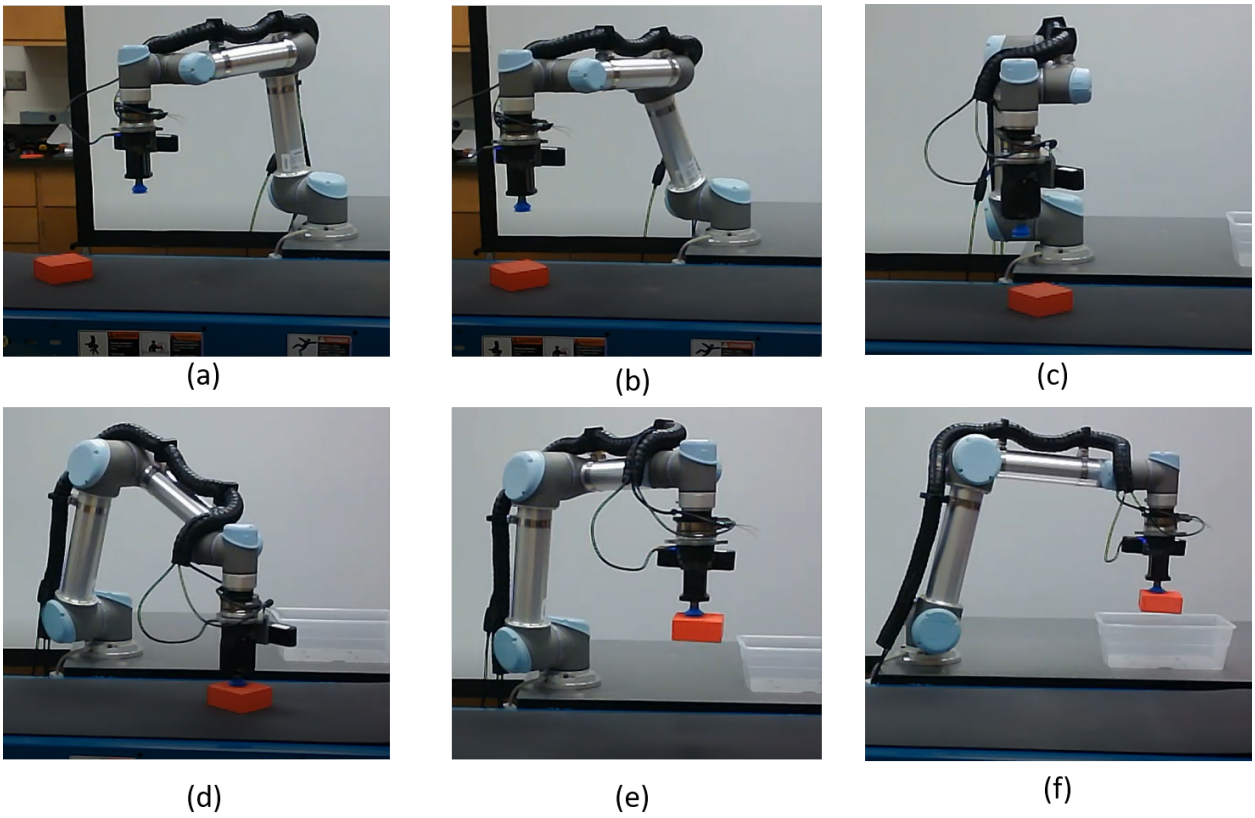


Figure 5.11: Sampled Robot Configurations during Collaborative System Testing: (a) Initial Pose of robot where object not in camere FOV; (b) Human operator bringing EE over the object; (c) Vision based motion planning; (d) Object getting picked up; (e) Post Grasp Transport and Orientation Correction of the object; (f) Object dropped into the bin

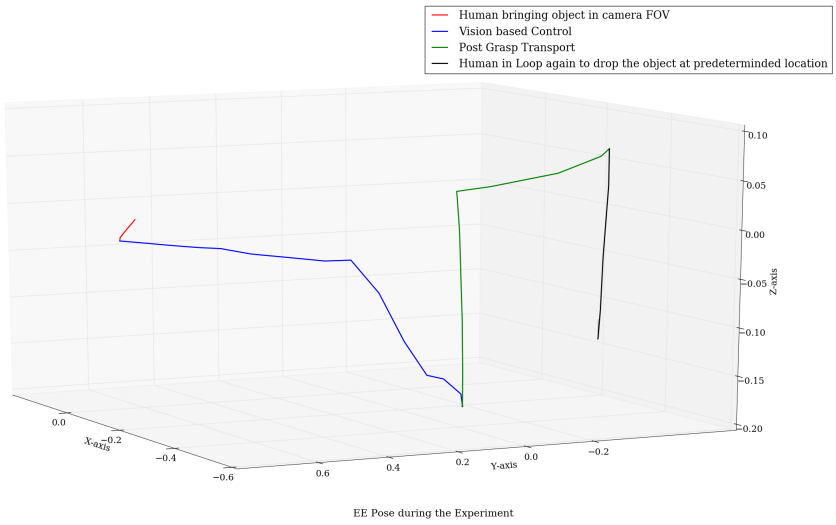


Figure 5.12: EE Pose during the Full Experiment

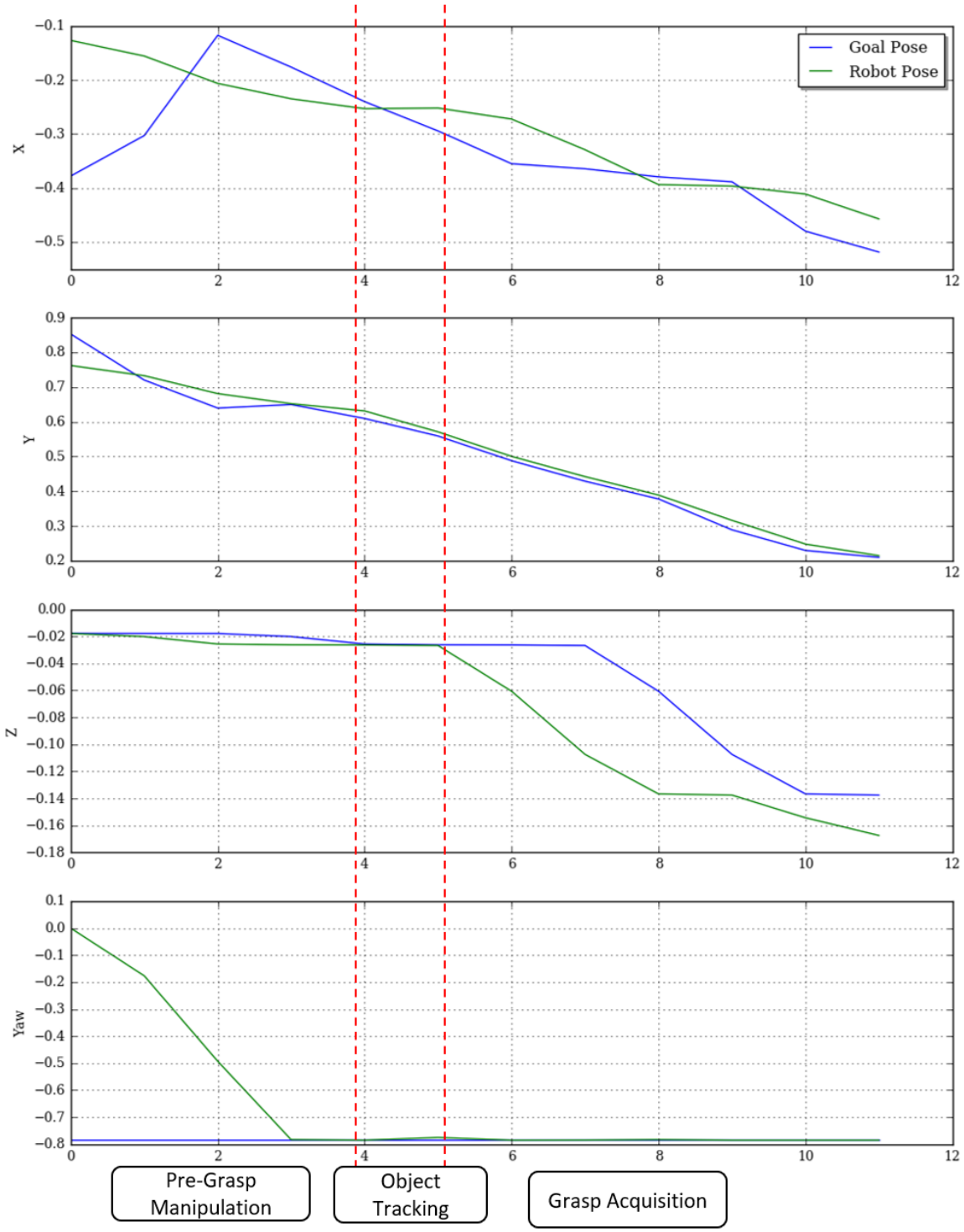


Figure 5.13: Vision Module Planned versus Actual EE Trajectory on UR5

6. SUMMARY AND FUTURE WORK

In this work, a framework was presented for Human-Robot Collaborative Control for Inspection and Material Handling tasks with Computer Vision and Joystick. First, a novel method for human operators to control a robot end-effector intuitively in mixed frame (orientation reference in robot world frame and translation reference in instantaneous end-effector frame) with human-machine shared control is presented. To facilitate effective and intuitive manipulator operation by human operators using a general purpose joystick, the Instantaneous Surface Normal Approach is utilized. Based on the operational characteristics of this approach, we have designed an appropriate joystick command interface. We have also introduced the corresponding method to generate orientation error, constant translation velocity, and torque terms based on the reference provided by the human operator via joystick. The control terms were decoupled and simultaneously mapped to robot joint velocities through the proposed hybrid control law.

Second, vision based motion planning method is developed. HSV based object detection and pose estimation method is presented for localization of the object. Motion planning algorithm to track and grasp a moving object identified by vision module were developed in MoveIt! software. We described how the robot and environment can be integrated with MoveIt! to generate collision free trajectories. Rapidly exploring random tree, RRT planning algorithm is utilized for generating path trajectories. Proportional controllers are developed to generate valid waypoints for path planner between dynamically changing target pose from vision module and current pose of the end-efffector to align, track and grasp the moving object. Fast Collision Check MoveIt! Library is utilized for planning of collision-free paths.

Experiments are conducted to test both modes of operation and integrated collaborative system on physical platform and simulation environment. Experimental results indicate that by utilizing the method proposed in this study, the human operator can intuitively and effectively command the robot to execute a pick-and-place task which is shown utilizing a peg-in-hole insertion example. Data also corroborate that the hybrid controller is capable of tracking human operator's reference

input effectively. Vision based automatic control experiments validated the effectiveness of motion planning algorithms developed. Results showed the robot end-effector to be closely following the path generated by the algorithm and successfully grasping the moving object. Finally, experiments are conducted integrating both joystick and vision based automatic operation to validate proposed Human-Robot Collaborative testing. Successful implementation on physical hardware is presented.

The results from the simulations and experiments conducted in this thesis showed the effectiveness of the methods developed. The tests were conducted for a moving object with speed of 7.5 cm/sec. However, it was observed with further experiments that the algorithms fails to effectively work for objects moving at higher speeds (>10 cms/sec). A simple technique based on color filtering has been employed for object detection in this study which made the module sensitive to light and reflection from the object surface. The sensitivity to light and reflection added noise in the data and the effects were amplified when lesser frames were captured at higher speeds leading to failure in estimating correct target pose. Better vision algorithms like edge detection method could be employed in the future. These methods would also help generalize the framework developed despite the color of the object. Another limitation to the study is that we utilized a 2D camera and Z coordinates of object were roughly estimated based on area of object detected, however a 3D camera could be used for precise localization of the object.

Also with MoveIt!, we utilized one motion planning algorithm to generate trajectories by generating valid waypoints using a local controller. The parameters of the local controller to generate the waypoints were tuned to get optimized performance, however, tuning the parameters of global planner with MoveIt were not explored in this study. There are also multitude of other planning algorithms available with MoveIt! to be explored for optimizing performance. Also, collision avoidance library allows for integration of dynamic obstacles using octomaps in the environment and this was not considered in the scope of this study. Further explorations with these MoveIt! capabilities remains a topic of future research.

REFERENCES

[1] Z. Jin, *Novel Methods for Human-Robot Shared Control in Collaborative Robotics*. PhD thesis, Texas AM University, Aug. 2020.

[2] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[3] S. Chitta, *MoveIt!: An Introduction*, pp. 3–27. Cham: Springer International Publishing, 2016.

[4] D. Abbink, T. Carlson, M. Mulder, J. de Winter, F. Aminravan, T. Gibo, and E. Boer, "A topology of shared control systems: Finding common ground in diversity," *IEEE Transactions on Human-Machine Systems*, vol. 48, pp. 509–525, 2018.

[5] Z. Jin, P. R. Pagilla, H. Maske, and G. Chowdhary, "Methods for blended shared control of hydraulic excavators with learning and prediction," in *57th IEEE Conference on Decision and Control*, pp. 1973–1978, 2018.

[6] S. Jain, A. Farshchian, A. Broad, F. Abdollahi, F. Mussa-Ivaldi, and B. Argall, "Assistive robotic manipulation through shared autonomy and a body-machine interface," in *IEEE International Conference on Rehabilitation Robotics*, pp. 526–531, 2015.

[7] J. Storms, K. Chen, and D. Tilbury, "A shared control method for obstacle avoidance with mobile robots and its interaction with communication delay," *The International Journal of Robotics Research*, vol. 36, pp. 820–839, 2017.

[8] S. Lichiardopol, *A survey on teleoperation*. DCT rapporten, Technische Universiteit Eindhoven, 2007. DCT 2007.155.

[9] Z. Jin and P. R. Pagilla, "Collaborative operation of robotic manipulators with human intent prediction and shared control," in *2020 IEEE International Conference on Human-Machine Systems (ICHMS)*, pp. 1–6, 2020.

[10] Z. Jin and P. R. Pagilla, "Human-robot teaming with human intent prediction and shared control," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications II* (T. Pham, L. Solomon, and K. Rainey, eds.), vol. 11413, pp. 295 – 303, International Society for Optics and Photonics, SPIE, 2020.

[11] F. Flemisch, D. Abbink, M. Itoh, M.-P. Pacaux-Lemoine, and G. Weßel, "Shared control is the sharp end of cooperation: Towards a common framework of joint action, shared control and human machine cooperation," *IFAC-PapersOnLine*, vol. 49, no. 19, pp. 72 – 77, 2016. 13th IFAC Symposium on Analysis, Design, and Evaluation of Human-Machine Systems HMS 2016.

[12] A. Hansson and M. Servin, "Semi-autonomous shared control of large-scale manipulator arms," *Control Engineering Practice*, vol. 18, pp. 1069–1076, 2010.

[13] T. Kot, V. Krys, V. Mostýn, and P. Novák, "Control system of a mobile robot manipulator," in *15th International Carpathian Control Conference*, pp. 258–263, 2014.

[14] Y. Zhang, L. Xie, Z. Zhang, K. Li, and L. Xiao, "Real-time joystick control and experiments of redundant manipulators using cosine-based velocity mapping," in *IEEE International Conference on Automation and Logistics*, pp. 345–350, 2011.

[15] H. Arioui, L. Temzi, and P. Hoppenot, "Force feedback stabilization for remote control of an assistive mobile robot," in *Proceedings of the American Control Conference*, pp. 4898–4903, 2011.

[16] Z. Jin and P. R. Pagilla, "Human-robot teaming with human intent prediction and shared control," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications II* (T. Pham, L. Solomon, and K. Rainey, eds.), vol. 11413, pp. 295 – 303, International Society for Optics and Photonics, SPIE, 2020.

[17] A. Singh, S. Seo, Y. Hashish, M. Nakane, J. Young, and A. Bunt, "An interface for remote robotic manipulator control that reduces task load and fatigue," *IEEE Conference on Robot and Human Interactive Communication*, pp. 738–743, 2013.

[18] S. Hashimoto, A. Ishida, M. Inami, and T. Igarashi, "Touchme: An augmented reality interface for remote robot control," *J. Robotics Mechatronics*, vol. 25, pp. 529–537, 2013.

[19] L. Li, B. Cox, M. Diftler, S. Shelton, and B. Rogers, "Development of a telepresence controlled ambidextrous robot for space applications," *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, pp. 58–63 vol.1, 1996.

[20] M. A. Diftler, C. J. Culbert, R. O. Ambrose, R. Platt, and W. J. Bluethmann, "Evolution of the nasa/darpa robonaut control system," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 2, pp. 2543–2548 vol.2, 2003.

[21] M. Fischer, P. van der Smagt, and G. Hirzinger, "Learning techniques in a dataglove based telemanipulation system for the dlr hand," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 2, pp. 1603–1608 vol.2, 1998.

[22] Haiying Hu, Xiaohui Gao, Jiawei Li, Jie Wang, and Hong Liu, "Calibrating human hand for teleoperating the hit/dlr hand," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 5, pp. 4571–4576 Vol.5, 2004.

[23] M. Marinho, A. Geraldes, A. Bó, and G. Borges, "Manipulator control based on the dual quaternion framework for intuitive teleoperation using kinect," in *Brazilian Robotics Symposium and Latin American Robotics Symposium*, pp. 319–324, 2012.

[24] J. Kofman, S. Verma, X. Wu, and T. Luu, "Teleoperation of a robot manipulator from 3d human hand-arm motion," *Proceedings of SPIE - The International Society for Optical Engineering*, 10 2003.

[25] M. Syakir, E. S. Ningrum, and I. Adji Sulistijono, "Teleoperation robot arm using depth sensor," in *2019 International Electronics Symposium (IES)*, pp. 394–399, 2019.

[26] F. Kulakov, G. Alferov, and P. Efimova, "Methods of remote control over space robots - seventh polyakhov's reading," in *International Conference on Mechanics*, pp. 1–6, 2015.

[27] J. Webb, S. Li, and X. Zhang, "Using visuomotor tendencies to increase control performance in teleoperation," in *American Control Conference*, pp. 7110–7116, 2016.

[28] M. O'Malley and R. Ambrose, "Haptic feedback applications for robonaut," *Industrial Robot: An International Journal*, vol. 30, pp. 531–542, 12 2003.

[29] G. C. Burdea, *Force and Touch Feedback for Virtual Reality*. USA: John Wiley Sons, Inc., 1996.

[30] K. Harada, T. Foissotte, T. Tsuji, K. Nagata, N. Yamanobe, A. Nakamura, and Y. Kawai, "Pick and place planning for dual-arm manipulators," in *2012 IEEE International Conference on Robotics and Automation*, pp. 2281–2286, 2012.

[31] C. Rennie, R. Shome, K. Bekris, and A. De Souza, "A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place," *IEEE Robotics and Automation Letters*, vol. 1, 09 2015.

[32] A. J. Sbnchez and J. M. Martinez, "Robot-arm pick and place behavior programming system using visual perception," in *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, vol. 4, pp. 507–510 vol.4, 2000.

[33] A. Tellaeche, I. Maurtua, and A. Ibarguren, "Use of machine vision in collaborative robotics: An industrial case.," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–6, 2016.

[34] Z.-H. Huang, L., "Implementation of ur5 pick and place in ros-gazebo with a usb cam and vacuum grippers." https://github.com/lihuang3/ur5_ROS-Gazebo.git, 2018.

[35] F. Park, "Computational aspects of the product-of-exponentials formula for robot kinematics," *IEEE Transactions on Automatic Control*, vol. 39, pp. 643–647, 1994.

[36] K. Okamura and F. Park, "Kinematic calibration using the product of exponentials formula," *Robotica*, vol. 14, pp. 415–421, 1996.

[37] N. R. Pal and S. K. Pal, "A review on image segmentation techniques," *Pattern Recognition*, vol. 26, no. 9, pp. 1277 – 1294, 1993.

[38] Y. Zhang, "A survey on evaluation methods for image segmentation," *Pattern Recognition*, vol. 29, no. 8, pp. 1335 – 1346, 1996.

[39] R. M. Haralick and L. G. Shapiro, "Image segmentation techniques," *Computer Vision, Graphics, and Image Processing*, vol. 29, no. 1, pp. 100 – 132, 1985.

[40] M. Cheriet, J. N. Said, and C. Y. Suen, "A recursive thresholding technique for image segmentation," *IEEE Transactions on Image Processing*, vol. 7, no. 6, pp. 918–921, 1998.

[41] V. Kumar, T. Lal, P. Dhuliya, and D. Pant, "A study and comparison of different image segmentation algorithms," in *2016 2nd International Conference on Advances in Computing, Communication, Automation (ICACCA) (Fall)*, pp. 1–6, 2016.

[42] G. Shrivakshan and C. Chandrasekar, "A comparison of various edge detection techniques used in image processing," *International Journal of Computer Science Issues*, vol. 9, pp. 269–276, 09 2012.

[43] S. Suzuki and K. be, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32 – 46, 1985.

[44] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, pp. 72–82, December 2012. https://ompl.kavrakilab.org.

[45] S. LaValle, "Rapidly-exploring random trees : A new tool for path planning," *The annual research report*, 1998.

[46] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3859–3866, 2012.

[47] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Auton. Robots*, vol. 34, p. 189–206, Apr. 2013.

[48] M. Quigley, “Ros: an open-source robot operating system,” in *ICRA Workshop on Open Source Software,*, 2009.