

Santa Clara University

Scholar Commons

Mechanical Engineering Master's Theses

Engineering Master's Theses

Winter 2024

Object-Centric Spatially-Aware Gesture-Based Motion Specification of Robotic Manipulation Systems

Rehan Nazir

Follow this and additional works at: https://scholarcommons.scu.edu/mech_mstr

SANTA CLARA UNIVERSITY

School of Engineering

**I HEREBY RECOMMEND THAT THE THESIS
PREPARED UNDER MY SUPERVISION BY**

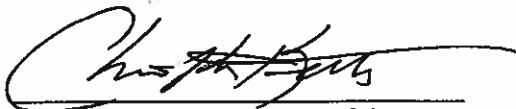
REHAN NAZIR

ENTITLED

Object-Centric Spatially-Aware Gesture-Based Motion Specification of Robotic Manipulation Systems

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

**MASTER OF SCIENCE IN
ROBOTICS AND AUTOMATION**



**Thesis Advisor & Director of the
Robotics and Automation Program
Dr. Christopher Kitts**



**Thesis Reader
Dr. Michael Neumann**



**Thesis Reader
Dr. Manoj Sharma**

**Object-Centric Spatially-Aware Gesture-Based Motion Specification of Robotic
Manipulation Systems**

**By
Rehan Nazir**

GRADUATE MASTER'S THESIS

**Submitted to
the School of Engineering**

of

SANTA CLARA UNIVERSITY

**in Partial Fulfillment of the Requirements
for the degree of
Master of Science in Robotics and Automation Engineering**

Santa Clara, California

Winter 2024

Abstract

As collaborative robots (cobots for short) become more prominent in industrial settings, intuitive ways of interaction between humans and cobots are essential for their success. In this Thesis, a novel framework that enables a human to easily guide a cobot to co-manipulate an extended object is presented. The developed framework enables a human to guide the object grasped by the cobot in all dimensions of motion (translation, rotation, and gripper actuation). To achieve this, a novel control has been developed that uses hand gestures and spatially-aware features of the operator (e.g. operator's location in space) with respect to the manipulated object and generates object-centric control commands. The generated object-centric control commands are then sent to an impedance controller for smooth object co-manipulation. Hand gestures are recognized by a deep learning model using vision data and human position estimation is calculated using stereocamera. The proposed framework is implemented on a 7-degrees of freedom cobot equipped with a camera sensor which is used to estimate the operator's position with respect to the robot. Additionally, a wearable camera is worn by the human teammate that monitors hand gestures. To evaluate the performance and usability of the proposed framework, an experimental scenario is developed where a human is required to co-manipulate an extended object by guiding the robot to pick it up from the table and place it into four tubes with different orientations in different locations of the workspace. A user study of 16 participants was conducted and the results are presented in this Thesis. The developed system was compared to the state-of-the-art motion capture system and had an average error of 0.01 m which is acceptable for our application. Moreover, the system was evaluated positively by the participants in the study.

Acknowledgments

I would like to express my gratitude to Prof. Christopher Kitts for his guidance and support throughout the completion of my Master's Thesis. His expertise and insightful feedback played an important role in shaping this Thesis. Additionally, I would like to thank Dr. Manoj Sharma and Dr. Michael Neumann for their support in the project and for being readers of this Thesis. I am also grateful to all the participants who took part in the evaluation study of my Thesis.

Contents

1	Introduction	1
1.1	Background and Motivation	1
1.2	Related Work	2
1.2.1	Related Work on Hand Gesture-based Cobot Control using Vision Sensors	2
1.2.2	Related Work on Human-Robot Co-Manipulation	6
1.3	Problem Statement	10
1.4	Thesis Contributions	11
1.5	Reader's Guide	12
2	Object-Centric Spatially-Aware Gesture-Based Motion Specification of Robotic Manipulation Systems	13
2.1	Dual Hand Gesture Recognition	14
2.2	Relative Operator-Object Orientation	16
2.3	Pivot Point Position Sensing	17
2.4	Mathematical Formulation	18
2.5	Summary	22
3	Implementation of the Proposed System	24
3.1	Multi-Camera and Robotic System	27
3.2	Vision-Based Hand Gesture Recognition	29

3.3	Human and Hand Position Estimation	32
3.4	Object-Centric Spatially-Aware Gesture-based (OSAG) Control	35
3.5	Cartesian Impedance Controller	40
3.6	ROS Implementation	42
4	Experimental Results	45
4.1	Experimental Setup & Study Protocol	45
4.2	User Study and Results	49
4.3	System Evaluation	52
4.4	Discussion: Ethical Considerations and Privacy	59
5	Conclusion and Future Work	61

List of Figures

1	Possible Motion commands in Object frame relative to operator (not to scale) . . .	15
2	Motion generation for several “push/pull” command scenarios based on operators position	17
3	Yaw command in operator frame and resulting End-effector motion	18
4	Frames associated with the transformation process	20
5	Hardware Setup: ZED Mini (stereo) camera; Headband Webcam; Franka Emika Panda Cobot.	24
6	Overview of the proposed framework for human-robot co-manipulation.	25
7	Dual Hand Gestures	26
8	Left and Right Hand Landmarks.	30
9	Flowchart of the proposed vision-based hand gesture recognition	31
10	Bounding box of human standing on the left (left image) and on the right (right image) of the cobot.	32
11	Bounding box of human standing in front of the cobot and the defined x and y axis of the image.	33
12	Human and Hand Position Estimation	35
13	Franka Emika Panda robot with end-effector and robot base frames. Red, Green and Blue represent the X, Y and Z-axis, respectively.	36
14	Calculation of translation and rotation step based on user input.	39

15	Control diagram of the Cartesian Impedance Controller	41
16	ROS Architecture	43
17	Robot Setup for the User Study	45
18	OptiTrack View in Motive Software(OptiTrack cameras in orange and robot end-effector in green)	53
19	Robot's End-Effector Position in X-axis; Desired, measured by Optitrack and calculated by robot's forward kinematics (left). Absolute error in X-axis (right)	55
20	Robot's End-Effector Position in Y-axis; Desired, measured by Optitrack and calculated by robot's forward kinematics (left). Absolute error in Y-axis (right)	57
21	Robot's End-Effector Position in Z-axis; Desired, measured by Optitrack and calculated by robot's forward kinematics (left). Absolute error in Z-axis (right)	58

List of Tables

1	Gestures and postures of the TV assembly dataset collected in [1].	4
2	Robot command generation based on gesture recognition and human position . . .	16
3	Velocity Commands in Operator frame	20
4	Specifications of the Logitech C615 Webcam (adapted from [2])	28
5	Specifications of the ZED Mini Camera (adapted from [3])	28
6	Basic Questionnaire	46
7	NASA Task Load Index Questionnaire	46
8	Human Robot Co-manipulation Questionnaire	47
9	System Usability Scale (SUS) Questionnaire	48
10	Age distribution of the study participants	49
11	Completion time for each participant, including average and standard deviation. . .	50
12	SUS Score for each participant, including average and standard deviation	51
13	Specifications for Flex V100 camera used in OptiTrack	53
14	Minimum, Maximum and Average Absolute Error for Robot-OptiTrack, Desired- OptiTrack and Desired-Robot in X-axis	55
15	Minimum, Maximum and Average Absolute Error for Robot-OptiTrack, Desired- OptiTrack and Desired-Robot in Y-axis	56
16	Minimum, Maximum and Average Absolute Error for Robot-OptiTrack, Desired- OptiTrack and Desired-Robot in Z -axis	59

1 Introduction

1.1 Background and Motivation

For decades, robotic manipulators in industry have been operated in cages, independent of any direct human interaction, with the robots performing repetitive tasks requiring high speed, accuracy, and power and performing tasks that are hazardous or challenging for humans [4, 5]. These types of tasks were unsafe for humans and, therefore, the robots stayed in well-defined workspaces with fences ensuring human safety. As robotic technology and Artificial Intelligence (AI) advance, robotic manipulation has started to move outside the cages and to become “collaborative” [6]. Collaborative robotic arms (cobots for short) are transforming the fields of automation and manufacturing as they are easy to set up and use and can work alongside humans [7]. To ensure human safety and enable humans to work within their workspace, cobots work at lower speeds, are lightweight, and have built-in collision sensing.

One important characteristic of cobots is that they can work alongside humans. Because cobots are easily set up, flexible, safe, and user-friendly, industry and manufacturing have started using cobots for applications, such as assembly, inspections, and cooperative manipulation (co-manipulation) [8]. This enables the cobots to assist their human teammates with demanding tasks, such as lifting and manipulating heavy objects [9, 10]. Therefore, seeing the advantages of cobots and their potential to support naive users, this Thesis focuses on developing an intuitive way for humans to guide an object through a collaborative manipulation task.

However, as robotic technology advances, finding more human-like ways of interacting with cobots is important. For example, Ionescu & Schlund [11] used voice commands as a method for instructing cobots for manipulation tasks. Another human-like way of instructing robots is by using

hand gestures. In the recent review paper on hand gesture recognition, Guo et al. [12] summarized different technologies that enable hand gesture recognition for human-machine interaction tasks. The authors identified different sensing technologies that can be used to identify hand gestures, such as vision sensors, ultrasound sensors, joysticks (e.g. Wii remotes), wearable gloves, and surface electromyography (sEMG) sensors. Wearable sensors typically require setup time and must be adjustable so that they can have the right fit for different users. On the other hand, vision sensors, such as cameras, do not require any setup from the user's perspective; however, they can lead to human data bias [13]. Moreover, with the recent advances in Artificial Intelligence (AI) and Deep Learning (DL), computer vision systems have become popular due to the easiness of the setup and their high accuracy [14–16].

1.2 Related Work

This section provides details on the related work on hand gesture-based cobot control using vision sensors and on human-robot co-manipulation. In this Thesis, the term cobot describes a collaborative robotic arm.

1.2.1 Related Work on Hand Gesture-based Cobot Control using Vision Sensors

Hand gesture-based robot control is a method that maps hand gestures to robot motions. There is significant work on hand gesture-based control of mobile bases, as mobile bases move in two dimensions and require very few gestures (move forward, backward, left, right, rotate, and stop) [12, 17, 18]. However, this Thesis focuses on the control of collaborative robotic arms (cobots) that have higher dimensionality for their motion (motion in 3D space, including translations and rotations) taking into consideration the user's position in the workspace.

Nuzzi et al. [19] developed a vision-based hand gesture recognition system, to ensure flexibility in the setup and mapped four hand gestures to specific commands of a robot arm (cobot), such as start, move left or right, and stop. The authors used a Faster Region Proposal Convolutional Neural Network (F-RNN) to classify the four gestures from images and achieved accuracy in the range of 92.12 to 95.01%. However, the four gestures provide a limited number of robot control actions, not enabling full control of a robotic arm with 6- or 7- Degrees of Freedom (DoF). Moreover, the authors did not integrate their gesture-based recognition with a real robot to evaluate the complete framework and did not enable object manipulation. Additionally, the proposed system did not take into account the spatial information of the operator.

Papanagiotou et al. [1] used first-person camera data (i.e. egocentric view) to recognize hand gestures for a TV assembly. The authors first collected data from 13 users on gestures related to TV assembly. The dataset includes eleven classes (six gestures and five postures) captured in an egocentric view in front of a green background. For example, if the operator's left palm faces the camera, it means the assembly starts, if the operator holds a card with both hands, it means the card must be placed, and if the operator holds the card with the left hand and extended the index finger of the right hand, it means the card functions. The list of the classes is explained in Table 1. Subsequently, the authors trained a 3D Convolutional Neural Network (3DCNN) by splitting the data into 80% for training and 20% for testing. The accuracy of the testing set for the hand gesture recognition was 98.5%. The developed framework was evaluated with a cobot from Universal Robots (UR), the UR3 with 14 operators (ten men and four women, aged from 23 to 44 with little and medium experience in TV assembly). For the evaluation, a defined sequence of actions had to be followed by the cobot and the operator. The cobot was pre-programmed to fetch the cards, while the operator was responsible to check their functionality and inform the cobot if the cards function or not by using gestures. If the cards were functioning, the operator would place them

Table 1: Gestures and postures of the TV assembly dataset collected in [1].

Gesture	Gesture Explanation	Posture	Posture Explanation
G1 - Left palm facing the camera	Start of the assembly	G3 - Hold the green card with both hands	Place green card
G2 - Hold the green card with the left hand and extend the index finger of the right hand	Green card functioning	G4 - Hold the screw tool with the right hand on top of the green card	Screw green card
G6 - Right palm facing the camera	End of the assembly	G5 - Hold the screw tool with the right hand on top of the gold card	Screw gold card
G8 - Hold the gold card with the left hand and extend the index finger of the right hand	Gold card functioning	G7 - Hands next to the body	Waiting
G10 - Hold the green card with the left hand and extend the index and middle fingers of the right hand	Green card not functioning	G9 - Hold the gold card with both hands	Place gold card
G11 - Hold the gold card with the left hand and extend the index and middle fingers of the right hand	Gold card not functioning		

and screw them in the required location. The cobot would monitor the process by recognizing the gestures and then would provide the next card. The study showed that the assembly process was accelerated with gesture recognition and the cobot's assistance by 20% in comparison to the TV assembly without assistance. However, the framework was developed for the particular TV assembly and the cobot had a limited role that included fetching cards from specific locations to the human operator and monitoring what the operator was doing. For every new scenario or change to the existing TV assembly scenario (e.g. green card changes to a different color), new data collection, data labeling, and training of the 3DCNN will be required. Collecting data from different humans, getting the data labeled, and then training the network is a time-consuming process [20]. Additionally, the cobot is required to be programmed for new scenarios or changes in the current scenario, which does not provide the necessary flexibility. This framework does not connect the hand gestures to robot actions but rather uses the hand gestures as a monitoring method for the cobot to understand the operator's actions.

Dynamic gesture commands were used by Chen et al. [21] for cobot control to enable flexibility

and fluidity in the interaction. Dynamic gestures are defined by Shi et al. [22] as hand motions captured over a continuous period of time. The hand positions and gestures are not required to stay in a particular pose but move over time. Chen et al. had to answer the following critical question: *“how to make cobots understand the dynamic gestures from different human users, and react with handling operations that meet humans’ expectation?”* [21]. To address this question, the authors developed a Conditional Collaborative Handling Process (CCHP) to model a context-aware cobot handling policy for a furniture assembly task. The policy was learned from a dataset of human-human (i.e. user-handler) collaboration from 15 users with their handlers. After the model’s training was completed another study was conducted with 10 participants for the furniture assembly task. At the beginning of the study, five-minute demonstration data were collected from each participant. Then, the participants familiarized themselves with the procedure. When they were ready, the participants assembled the furniture once with two cobot policies, one encoded with CCHP and one with a fixed rule-based policy. The cobot was responsible for fetching tools to the operator when the operator would point at the tool or for holding a part of the furniture in place. The study used a Wilcoxon signed-rank test to show that a cobot is a significantly better helper using the CCHP policy rather than the fixed rule-based policy. The results of the Wilcoxon signed-rank test showed that the CCHP-enabled robot understands better the user input (test statistic $W=2$, $p\text{-value}=0.006$) and enables better collaboration ($W=3$, $p\text{-value}=0.010$). Therefore, since the p -values are less than 0.05 and the test statistic is less than 6, the hypothesis that CCHP-enabled cobots help significantly better than the fixed rule-based cobots is valid. Therefore, dynamic gesture commands have the potential to improve context-aware cobots; however, the proposed method relied on the human-human collaboration dataset. Additionally, the operator was expected to sit in a chair in front of the cobot and the workstation was a desk that was shared by the cobot and the operator. The cobot had only two tasks, either fetching tools or holding a part of the furniture in place, which

is not easily extendable to other scenarios/tasks and also limits the cobot's capabilities.

It is clear that the methods discussed in this section are mainly focused on a particular application, such as assembly, it is not easily transferable to other applications without a data collection process and additional training process. Furthermore, the cobots are used for tasks that are specific to the applications they are deployed (e.g. fetching objects or holding objects), which means that it is application specific. Therefore, there is a need for a hand gesture-based interface that enables the operator to manipulate an object with the cobot's help in all dimensions, providing better flexibility and being able to get deployed in a variety of tasks. Enabling the operator to freely move around the cobot's workspace is also an important aspect as not all tasks can be completed on a desk.

1.2.2 Related Work on Human-Robot Co-Manipulation

Physical Human-Robot Interaction is defined by Selvaggio et al. [23] as proximal when the human and the robot are in direct contact, which may be mediated by another object (e.g. a human and a robot grasping the same object). Selvaggio et al. [23] identified shared control schemes as one approach to human-machine co-manipulation. Shared control is the control architecture in which the robot is controlled by a combination of autonomy algorithms and direct user commands [24]. Another approach for human-robot co-manipulation is admittance control [25, 26], which is a control scheme that automatically calculates robotic parameters, such as position, velocity, etc. based on applied external forces. For this method, accurate measurements of external forces are needed by using force/torque sensors on the robotic system.

Shared control has been extended to shared autonomy, which consists of additional information from the user, such as human intent, human skills, or human muscle activity [23]. For example, Pernel et al. [27] used electromyography (EMG) sensors to monitor human muscle activity during

human-robot collaborative sawing and polishing. The authors used a KUKA Lightweight Robot equipped with a Pisa/IIT SoftHand, which provides position and force feedback. The authors proposed a framework based on hybrid impedance control that adapts the robot's behavior based on the muscle activity from EMG data and the position and force feedback from the robot. To evaluate the framework, four male participants collaborated with the robot in the two collaborative tasks (sawing and polishing). For the sawing task, the average muscle activity for the four male participants without the robot was $19.2 \pm 11.2\%$ and after the robot took over the average muscle activity was $6.0 \pm 2.7\%$. Similarly, for the polishing task, the average muscle activity without and with the robot was $17.2 \pm 1.7\%$ and $4.6 \pm 1.1\%$, respectively. Therefore, it is clear that the robot collaboration reduced the muscle activity required for the tasks. However, additional evaluation with more participants is required and the framework assumes that the human and the robot will always be in contact.

Van der Spaa et al. [28] focuses on the co-manipulation of a large object between a human and a bimanual mobile robot (a mobile base equipped with two Kuka robotic arms). The human wore an XSENS motion capture suit which is equipped with inertial measurement units (IMUs) and the Rapid Entire Body Assessment (REBA) scores were calculated to assess the ergonomic posture of the human. According to Hignett and McAtamney [29] REBA is defined as "*a postural analysis system sensitive to musculoskeletal risks in a variety of tasks*". Van der Spaa et al. conducted a study of four participants for the task of collaboratively rotating the box 180 degrees, clockwise and counterclockwise. Based on the ergonomics scores, the robot's planner adapts to improve the REBA score. However, a full wearable suit requires setup time and it requires to be adjustable to different body types and heights.

Co-manipulation of extended objects is also an interesting problem in human-robot collaboration. Mielke et al. [30,31] collected data from human-human interactions of moving an extended

planar object and then developed an algorithm that enables human-robot co-manipulation of the same object. The collected data were from sleeves with markers worn by the humans (e.g. markers to track human pose), handles with force/torque sensors on the object, and infrared markers mounted on the object. The robotic platform was the Baxter, a dual-arm robot, mounted on a mobile base. The proposed method was an extended variable impedance control that generated the robot's actions based on the sensors on the object and the sleeves on the operator, and it is called Neural Network Prediction Control (NNPC). A study with 16 participants was conducted to evaluate the feasibility of the proposed solution. Metrics such as minimum jerk, minimum torque change, and completion time were measured. The pair of a human and the NNPC-enabled robot showed better results in almost all metrics in comparison to human-human interaction. However, this approach requires handles with force/torque sensors to be added to the manipulated planar object, and the NNPC method is limited to performing translation or rotation in the horizontal axis only, while the object is parallel to the horizontal axis. Therefore, this approach is difficult to implement in the real world and is not extendable to translations or rotations in all the dimensions.

Another aspect of human-robot co-manipulation is to resolve conflicts between who leads, the robot or the human [32]. The authors built an experimental setup with a Universal Robot 5 (UR5) that grasps a box-shaped object made of plexiglass equipped with two force/torque sensors. The users are required to move the box-shaped object to different locations on a table based on the visual feedback that they received from a computer screen. Machine learning (ML) methods were used to detect if the human and the robot work in harmony or if they are in conflict (e.g. the robot has a different goal position than the user) based on the force/torque data. The ML classifies if the human and the robot work in harmony (WH), if there is a conflict in movement direction (CD) (e.g. the new target location requires a change in the moving direction), and if there is a conflict in parking location (CP) (e.g. the target location changes from point A to point B). If conflict

is detected by the ML models, then the robot will follow the user. Nevertheless, having sensors mounted on the objects is not realistic.

Solanes et al. [33] developed a hybrid framework that uses visual and force sensing in order to enable a robot and a human to co-manipulate objects. A camera was used to estimate the motion of the human's hand and a force sensor mounted on the arm of a Sawyer cobot was used to estimate the external force. The operator physically grasps and manipulates the cobot's end-effector that has grasped the object and does not get in contact with the manipulated object. Based on these inputs, the hybrid framework was able to generate the desired robot motion for controlling the cobot's end-effector that manipulates the object. To demonstrate the framework in the real world, a glass with liquid was positioned on a flat surface that was mounted on the robot's end-effector and a user moved, pushed, and pulled the end-effector of the robot in the workspace. The liquid remained in the glass through the demonstration. However, the framework was not evaluated in a study and no metrics were defined for its validation. Additionally, the operator's position in space or the manipulated object information (e.g. size, dimensions, etc.) are not taken into account as only their hand and applied force on the end-effector are considered.

Ambauen [34] developed a robot-assisted manipulation of extended objects utilizing tactile sensing and spatial hand tracking to recognize user intent. The developed framework consists of a pair of force-sensing gloves, an industrial robotic manipulator SCARA, and an external motion capture system to track the positions of the robot, the human, and the extended object. The forces measured from the glove were classified into four gestures per hand (i.e. push, pull, grip, and squeeze) and a robot controller was developed that translated the gestures to translations in X- and Y-directions and rotation around the Z-axis taking into account the hand position from the motion capture system. The proposed framework with gloves was evaluated with a user study of five participants and compared to a joystick-based control. The user study demonstrated that the

glove-based interface received higher scores in the adjective ratings of usable, repeatable, intuitive, natural, efficient, and accurate interface compared to the scores from the joystick-based interface. This Thesis was inspired by this work and extended to enable robot motion in all three dimensions (X, Y, and Z) and three rotations (roll, pitch, yaw).

As it is clear from the prior work presented in this section, accurate force sensing and accurate gesture recognition are critical components in the development of a natural way of interaction between a human and a robot for co-manipulation tasks. Force sensing requires additional sensors that have to be mounted either on the object or on the cobot's end-effector may not be realistic or add additional cost. Additionally, none of the above methods focus their control scheme on the manipulated objects but mainly on the cobot's end-effector. Therefore, in this Thesis, we propose a framework that inherently enables human-robot co-manipulation by using as inputs the operator's hand gestures and the operator's spatial information (e.g. pose) with respect to the manipulated object without the use of force/torque sensors.

1.3 Problem Statement

In human-robot co-manipulation, a cobot assists its human teammate by bearing the weight of the manipulated object while the human guides the cobot through the task. In cases of complex tasks, the human may be required to be in contact with the manipulated object or to freely move around the workspace to provide better guidance to the cobot. Human-robot co-manipulation can improve working conditions for workers who are required to lift, transport, and manipulate objects, especially heavy or extended objects. By augmenting human skills, the cobots can improve safety and enhance productivity at the workplace.

For the cobots to be accepted in the workplace, an intuitive way of interaction for object co-

manipulation is essential. Several researchers have proposed hand gesture-based interaction with a cobot for human-robot co-manipulation of objects, as presented in detail in Section 1.2). Hand gestures require very little training by humans to be able to control the robot, making them an intuitive and easy-to-use interface. Moreover, recognizing gestures via vision sensors provides a contactless interface for robot interaction. However, in the available hand gesture-based interfaces, gestures are mapped to predetermined robot motions for a defined application, without providing the flexibility of manipulating an object in all dimensions of motion (translation, rotation, and gripper actuation).

Moreover, most available interfaces do not consider that a human teammate may be required to move within the workspace and may require the gestures to have an adjustable mapping to cobot motions based on the human position. For example, if a human teammate stands on the left side of the cobot and performs a pull gesture so that the object moves toward them, the cobot will be required to move the object to its left. However, if the human stands on the right side of the cobot and performs the same gesture as before, the robot should not move to the left but to the right, taking into account where the human stands. Therefore, there is a need for an intuitive and easy-to-use robotic framework that enables naïve users to guide a cobot through object co-manipulation tasks and includes adaptive mapping of robot motion using gestures and human position as inputs.

1.4 Thesis Contributions

This Thesis proposes a novel object-centric spatially-aware gesture-based motion specification technique for robot manipulation systems and has the following contributions:

- Development of a framework that enables the human to guide an object that is grasped by a cobot in all dimensions of motion (translation, rotation, and gripper actuation),

- Development of an adaptive mapping algorithm, which is called Object-centric Spatially-Aware Gesture-based control (OSAG for short), that takes into account the human's position in the workspace and the performed hand gestures with respect to the manipulated object,
- Implementation of impedance control for smooth object co-manipulation.

To the best of the Author's knowledge, this is the first control method that uses hand gestures and spatially-aware features of the operator (e.g. operator's location in space) with respect to the manipulated object and generates object-centric control commands. The developed method is generic and does not depend on a specific application. Additionally, as it is object-centric, it is not specific to a robotic arm and it can be easily implemented for any collaborative robotic arm. To evaluate the performance and usability of the proposed framework, an experimental scenario is developed where a human is required to co-manipulate an extended object by guiding the robot to pick it up from the table and place it into four tubes with different orientations in different locations of the workspace. A user study of 16 participants is conducted and the results are presented in this Thesis.

1.5 Reader's Guide

This Thesis is organized as follows. Section 2 provides an overview of the proposed object-centric spatially-aware gesture-based motion for robot manipulators and Section 3 provides a detailed explanation of the implementation of the proposed vision-based framework for human-robot co-manipulation and its components. Section 4 described the system's validation and the experimental results from the user study for evaluating the framework. Last, Section 5 concludes the Thesis and discusses future directions of research.

2 Object-Centric Spatially-Aware Gesture-Based Motion Specification of Robotic Manipulation Systems

Disclaimer: This chapter was mainly written by the advisor, Dr. Christopher Kitts, and adapted by the author.

This Thesis explores a novel approach to object manipulation by a cobot using a gesture-based “object guiding” approach. Cobots can be programmed by kinesthetic teaching, which is a method that enables an operator to physically move the cobot’s end effector to specific waypoints that becomes the part of a learned trajectory. Kinesthetic teaching is considered an interactive, simple, and natural way for operators to “program” the cobot without requiring writing code. This Thesis advances the current research in the field of object manipulation by proposing an interactive operator-based “object guiding” which is referenced to the object being manipulated by the cobot. Therefore, the operator is not required to consider the cobot’s movements but is solely focused on the object’s motion in space.

The developed approach extends prior work, which is discussed in Section 1.2), in the following two ways. First, the operator uses dual hand gestures to define the object’s motion instead of the cobot’s end-effector motion. Second, the spatial knowledge related to the relative hand positions with respect to the object is used to develop an object-centric motion command. The important benefit of the developed approach in this Thesis is that it allows the operator to naturally control the object in all dimensions (translations and rotations) without requiring them to consider the cobot’s motion. Another benefit of the proposed approach is that it does not require additional retraining for extended objects of different dimensions that need to be co-manipulated and is independent of the application.

While the developed approach is generic, as a proof of concept, a long beam-like object that is grasped at one end by a robotic arm is selected, as is shown in Figure 1. A Six degree-of-freedom motion control for the object is implemented that enables pure translational (along the x-, y-, or z-axis) or rotational commands (about the object's pitch, roll, or yaw axis) in the object frame. In order to achieve all the rotations, the operator is required to move in space. For example, if the operator desires to move the object in the roll axis with respect to the object frame, they are required to stand in front of the object (center position as shown in Figure 2). However, if the operator is positioned on the left or right of the extended object, then the roll rotation with respect to the object frame is not possible.

Finally, it is worth noting that the proposed approach is limited within the cobot's workspace and the range and quality of the perception system for which it was developed, as described in Chapter 2. While various implementations of the overall strategy are possible, the spatial gesturing command specification capability that has been designed and implemented in this project relies on a three-step perception process, as described in the next sections.

2.1 Dual Hand Gesture Recognition

The first element of the proposed method is to recognize dual hand gestures by observing the operator's two hands. To do this, an operator-referenced head-mounted camera is used, and a gesture recognition algorithm is employed, as described in Section 3.2. At the start of the interaction, the operator performs a gesture that defines the velocity of the manipulated object (slow or fast). Next, the operator performs the gestures necessary to move the object to the desired location. These gestures and associated commands are illustrated in Table 2. Three pairs of gestures specify object translational motion in each of the three operator-specific reference frame directions as shown in

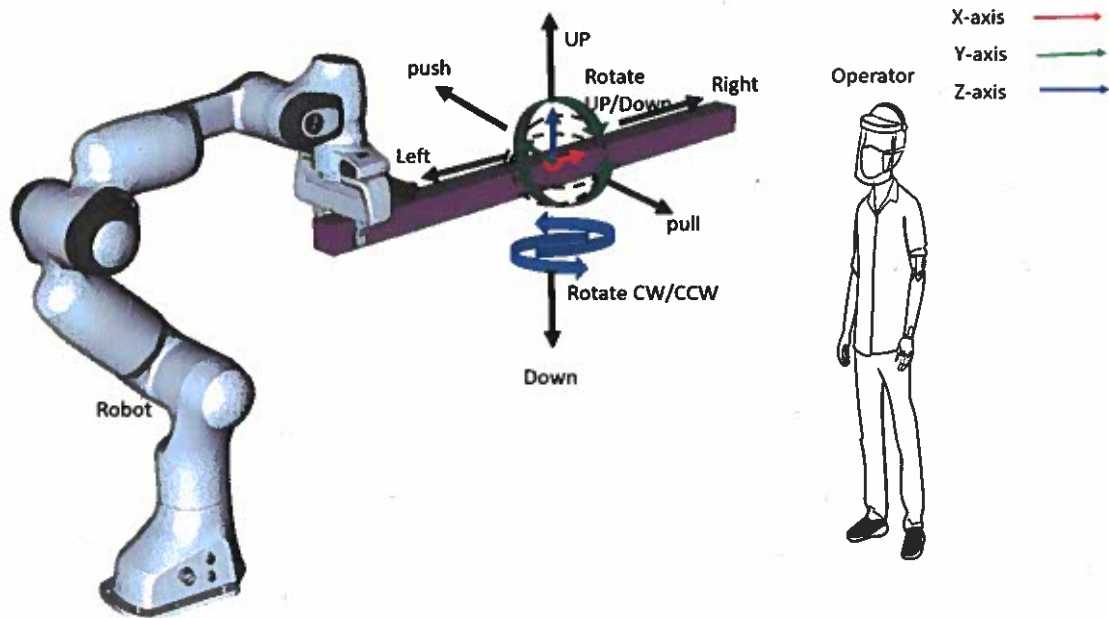


Figure 1: Possible Motion commands in Object frame relative to operator (not to scale)

Figure 1: Push/Pull (+/- x-axis motion), Right/Left (+/- y-axis motion, and Down/Up (+/- z-axis motion). Two pairs of commands specify rotational motion in the form of a pivot point and the sense of rotational direction: Rotate Up/Down (+/- roll motion) and Rotate Counter-ClockWise (CCW) / ClockWise (CW) (+/- yaw motion).

Given these gesture commands, it should be noted from Figure 2 that converting the command to object motion requires an understanding of the relative position and orientation of the operator with respect to the object. For example, in the scenario shown in Figure 2, the “Push” command would lead to object motion in the opposite direction if the operator was issuing that command from the other side of the object. Similarly, the pivot point of an object Pitch, Roll, or Yaw command depends on the location of the hand along the object’s x-axis. Because of this, spatial sensing is required, thereby motivating the tasks that are implemented in steps two and three of the perception process.

Table 2: Robot command generation based on gesture recognition and human position

Gestures	Position		
	Left	Right	Center
	Motion Command axis (Object Frame)		
Push	+Y-axis	-Y-axis	-X-axis
Pull	-Y-axis	+Y-axis	+X-axis
left	+X-axis	-X-axis	+Y-axis
Right	-X-axis	+X-axis	-Y-axis
Rotate Up	Y-axis-Counterclockwise	Y-axis-Clockwise	X-axis-Counterclockwise
Rotate Down	Y-axis-Clockwise	Y-axis-Counterclockwise	X-axis-Clockwise
Rotate Clockwise	Z-axis-Clockwise		
Rotate Counterclockwise	Z-axis-Counterclockwise		
Up	+Z-axis		
Down	-Z-axis		
Ok	Gripper Operation(Open/Close)		
Fast	Fast Speed mode activated		
Slow	Slow Speed mode activated		
Stop	Motion Stopped		

2.2 Relative Operator-Object Orientation

The second element of the perception process establishes the relative orientation of the operator with respect to the object. The developed system determines the correspondence of the operator frame with the object frame (see Figure 4 for frame references). So, for example, this knowledge allows a “push” gesture, which specifies motion away from the operator along the operator’s X-axis, to be interpreted as a motion command for the object to move along its appropriate axis in the indicated direction. Figure 2 describes this process for several “push/pull” command scenarios. To achieve this adaptive capability for our system, a camera mounted on the cobot’s end-effector and a human detection and tracking algorithm are used to determine the relative position of the operator with respect to the object, from the point of view of the cobot-mounted camera. The following assumptions are made:

- The relative poses between the object, the cobot’s end-effector, and the cobot-mounted cam-

era are known.

- The operator faces the object, which allows the relative alignment of the operator and object axes to be computed. This process is described in Section 3.3.

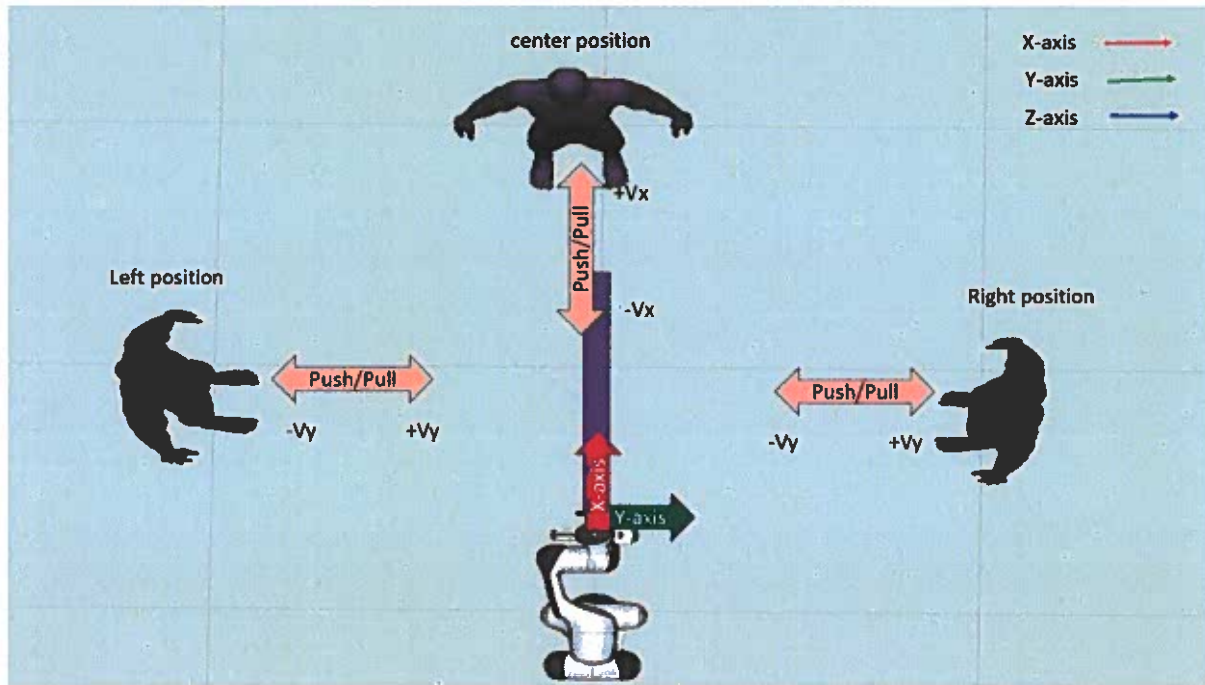


Figure 2: Motion generation for several “push/pull” command scenarios based on operators position

2.3 Pivot Point Position Sensing

The third step of the perception process calculates the relative position of the operator’s hands to be computed in the object’s frame. This is critical for object rotation commands given that rotation pivot points may be specified at arbitrary locations along the object’s length. To do this, the manipulator-mounted camera is equipped with a depth sensor that is used to measure the distance

of the operator's hand which is close to the camera along the object's long axis. Given the knowledge of the relative poses of the object, the cobot's end-effector, and the cobot-mounted camera, the location of the pivot point along the object's length can be computed geometrically. For example, Figure 3 illustrates an object yaw command using the calculations described in this section, and it indicates how this relates to the cobot's motion that must be implemented to produce the desired object motion. The overall process for the object motion control is detailed in Section 3.4.

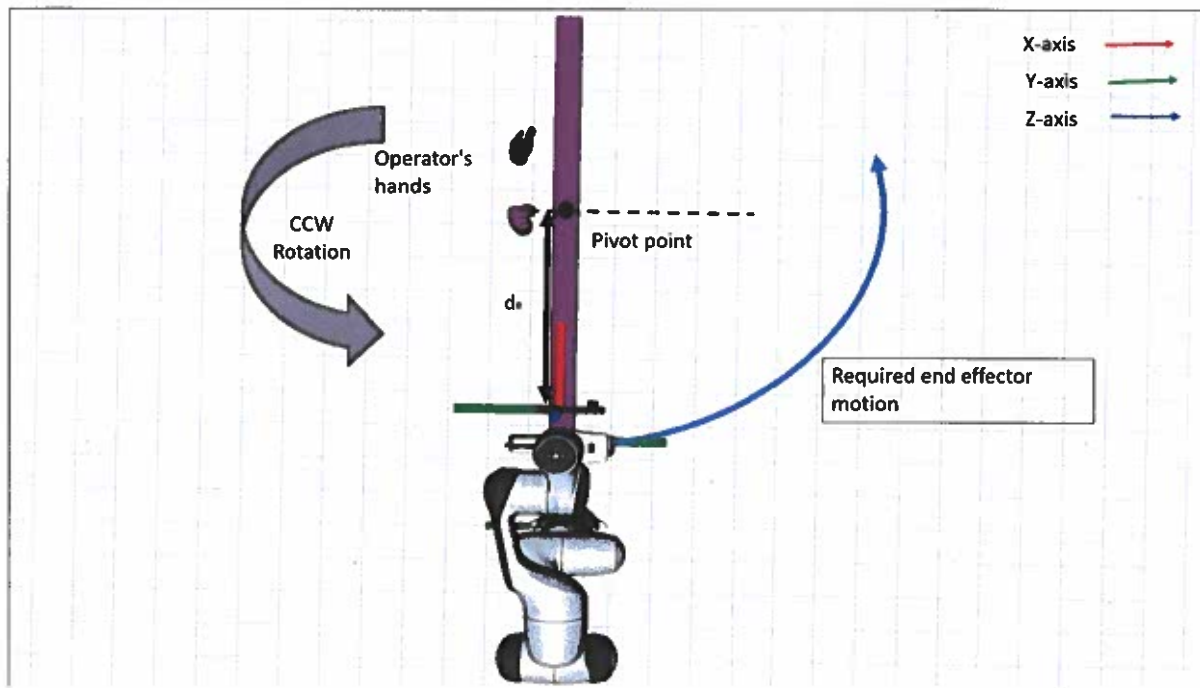


Figure 3: Yaw command in operator frame and resulting End-effector motion

2.4 Mathematical Formulation

This section outlines the underlying mathematics supporting the computation of object translational and rotational velocity commands as a function of gesture-based commands. This transformation is based on a number of assumptions, as follows:

- ${}^E_{Obj}T$: The relative transformation of the object frame $\{Obj\}$ with respect to the cobot's end-effector frame $\{E\}$ is known.
- ${}^E_C T$: The relative transformation of the cobot-mounted camera $\{C\}$ with respect to the cobot's end-effector frame $\{E\}$ is known.
- The relative orientation of the operator frame will be discretized such that its frame axes are assumed to align with the frame axes of the object; however, which operator axis is aligned with which object axis must be determined.
- For rotational commands, one hand of the gesture command specifies a pivot point, which is assumed to be at a point along the long axis of the object that is the closest distance to that hand.

Figure 4 depicts an example of the various frames associated with the transformation process. In particular, there are frames for the operator, the object, the cobot's end-effector, and the cobot-mounted camera.

In the first step of the 3-step perception process (Sections 2.1, 2.2, and 2.3), recognized gestures correspond to different motion commands, referenced to the operator's frame. The proposed work assumes constant jog velocities for the object to move (2-speed setpoints are supported in the current implementation). Therefore, the critical information produced by the gesture recognition process is the direction of translation or rotation with respect to operator frame axes $\{Op\}$. The output of this process for a recognized gesture is mapped to one of the velocity jog commands as shown in Table 3.

The second step of the perception process calculates the relative frame rotation of the operator frame with respect to the object frame, ${}^{Obj}_{Op}R$. This is determined by sensing the relative frame

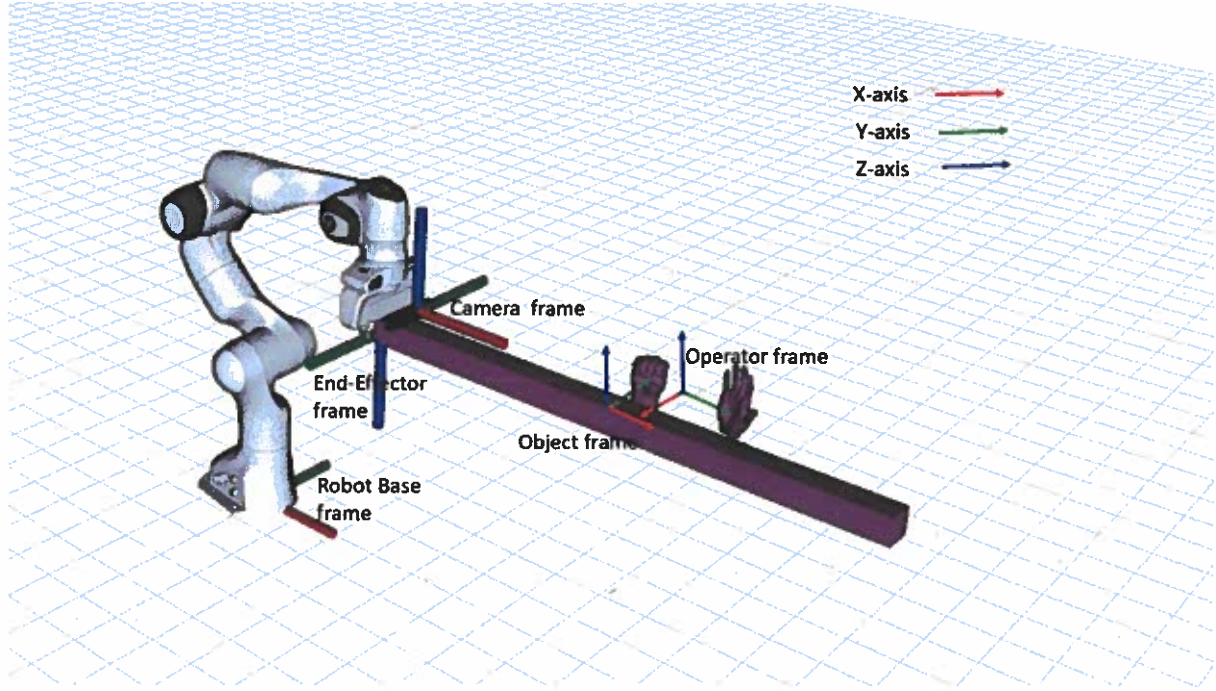


Figure 4: Frames associated with the transformation process

Table 3: Velocity Commands in Operator frame

	push	pull	right	left	up	down
${}^{op}\vec{v}$	$[+v, 0, 0]^T$	$[-v, 0, 0]^T$	$[0, -v, 0]^T$	$[0, +v, 0]^T$	$[0, 0, +v]^T$	$[0, 0, -v]^T$
	Rotation Down	Rotation Up	Rotation CCW	Rotation CW		
${}^{op}\vec{\omega}$	$[+\omega, 0, 0]^T$	$[-\omega, 0, 0]^T$	$[0, 0, +\omega]^T$	$[0, 0, -\omega]^T$		

rotation of the operator frame via the cobot-mounted camera frame, ${}^C_{Op}\mathbf{R}$, and then using the known rotations ${}^C_E\mathbf{R}$ and ${}^E_{Obj}\mathbf{R}$. Given that the perception process established ${}^C_{Op}\mathbf{R}$, ${}^{Obj}_{Op}\mathbf{R}$ is computed using the following equation:

$${}^{Obj}_{Op}\mathbf{R} = {}^{Obj}_E\mathbf{R} \cdot {}^E_C\mathbf{R} \cdot {}^C_{Op}\mathbf{R} = {}^E_{Obj}\mathbf{R}^{-1} \cdot {}^E_C\mathbf{R} \cdot {}^C_{Op}\mathbf{R} \quad (1)$$

With the computation of the ${}^{Obj}_{Op}\mathbf{R}$, a gesture-based translational velocity command in the oper-

ator frame can be transformed into a velocity command in the object frame based on the following equation:

$${}^{Obj}v = {}^{Obj}_{Op}\mathbf{R} \cdot {}^{Op}v \quad (2)$$

Given this, for translational velocity commands, the cobot's end-effector velocity command can be computed as a Cartesian velocity command based on the following equation:

$${}^E v = {}^E_{Obj}\mathbf{R} \cdot {}^{Obj}v \quad (3)$$

For rotational commands, the sense of object rotation can be computed by using the following equation:

$${}^{Obj}\omega = {}^{Obj}_{Op}\mathbf{R} \cdot {}^{Op}\omega \quad (4)$$

However, the cobot's motion command requires knowledge of the pivot point for the rotation, which is determined in step 3 of the perception process.

The third step of the perception process determines the location of the hand closest to the cobot-mounted camera in object frame, ${}^{Obj}\mathbf{p}_p$. The location of this point relative the object is calculated as follows:

$${}^{Obj}\mathbf{p}_p = {}^{Obj}_C\mathbf{T} \cdot {}^C\mathbf{p}_p = {}^{Obj}_E\mathbf{T} \cdot {}^E_C\mathbf{T} \cdot {}^C\mathbf{p}_p = {}^E_{Obj}\mathbf{T}^{-1} \cdot {}^E_C\mathbf{T} \cdot {}^C\mathbf{p}_p \quad (5)$$

where ${}^E_{Obj}\mathbf{T}$ represents the relative transformation of the end-effector frame $\{E\}$ with respect to

the object frame $\{Obj\}$, ${}^C\mathbf{p}_p$ is the position of the hand relative to the cobot-mounted camera and ${}^{Obj}_C\mathbf{T}$ represents the relative transformation of the cobot-mounted camera $\{C\}$ with respect to the object frame $\{Obj\}$. We only consider the distance of the pivot point along the length of the object, which is along the x-axis in the object frame. Therefore, to compute the pivot point in the object frame ${}^{Obj}\mathbf{p}_{pivot}$ we use the following equation:

$${}^{Obj}\mathbf{p}_{pivot} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot {}^{Obj}\mathbf{p}_p \quad (6)$$

Next, to compute the required cobot arm motion, the distance of the pivot point from the end effector is required, in addition to the sense and axis of the commanded rotation. Therefore, ${}^E\mathbf{p}_{pivot}$ is required, which is computed by the following equation:

$${}^E\mathbf{p}_{pivot} = {}^E_{Obj}\mathbf{T} \cdot {}^{Obj}\mathbf{p}_{pivot} \quad (7)$$

Finally, the required robot Cartesian commands for the commanded pivot are as follows:

$${}^E\boldsymbol{\omega} = {}^E_{Obj}\mathbf{R} \cdot {}^{Obj}\boldsymbol{\omega} \quad (8)$$

$${}^E\mathbf{v} = {}^E\mathbf{p}_{pivot} \times {}^E\boldsymbol{\omega} \quad (9)$$

2.5 Summary

This research proposes a novel gesture-based approach for controlling a cobot manipulator to perform object manipulation tasks. The system takes advantage of human intuition by allowing the

user to directly specify the desired object motion through hand gestures, instead of specifying the manipulator's endpoint motions. Spatial information regarding the relative positions of the hands and object are incorporated to translate the gestures into appropriate manipulator motions. This system simplifies the programming process for the user and enables more natural control of the object.

3 Implementation of the Proposed System

This Thesis proposes a novel vision-based framework that enables human-robot co-manipulation of extended objects, such as pipes, iron bars, long sticks, etc. As the goal of this Thesis is to develop a robotic framework that adapts to different human positions in the workspace and enables control of a co-manipulated object by a cobot in three-dimensional space (translation, rotation, and gripper actuation), a complete system is developed.

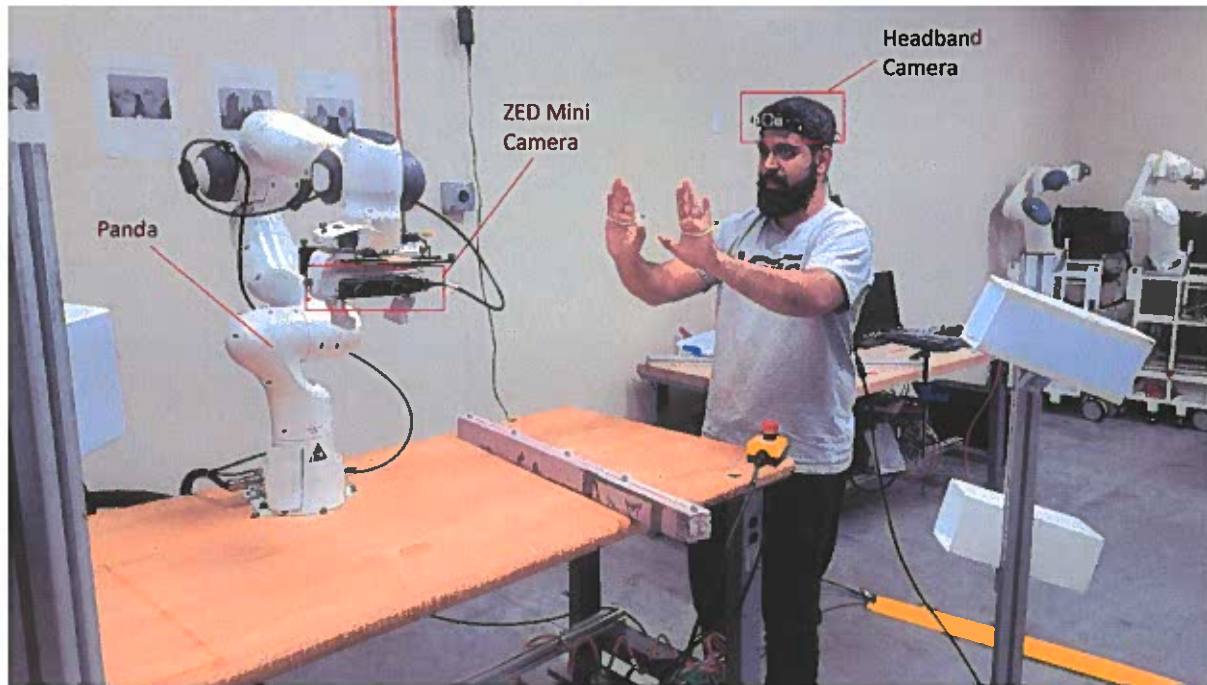


Figure 5: Hardware Setup: ZED Mini (stereo) camera; Headband Webcam; Franka Emika Panda Cobot.

The hardware of the system (see Figure 5) consists of a collaborative robotic arm (cobot), a computer that runs the developed framework and two cameras; one mounted on the cobot's end-effector to calculate the operator's position and the hand distance from the end-effector and one worn on the human's head to detect the operator's hand gestures. On the software side, a

framework using Robot Operating System (ROS) [35] was developed. ROS is a widely used open-source middleware suite for robotics. The developed framework, which is shown in Figure 6, has the following components; vision-based hand gesture recognition, human and hand position detection, object-centric spatially-aware gesture-based (OSAG) control, and cartesian impedance controller. The system works as follows.

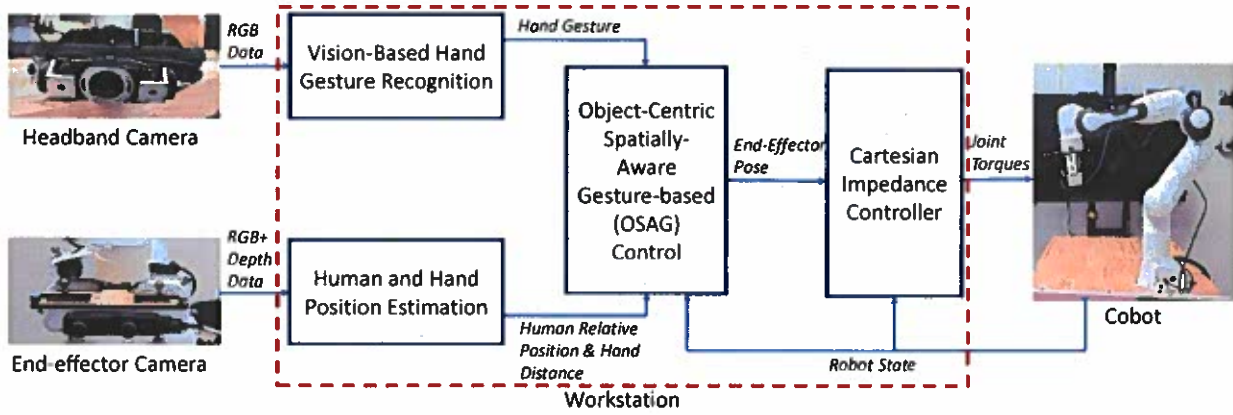


Figure 6: Overview of the proposed framework for human-robot co-manipulation.

- The camera mounted on the user's headband takes images of the user's hands at a rate of 30 frames per second and using the vision-based hand gesture recognition module, the recognized gesture is outputted. In order to recognize the gesture, Google Hand Detection Mediapipe is used which outputs 21 distinct landmarks for each hand, which are then converted into features. The features are then fed into a deep learning model that predicts the user's gesture command by using the current hand features and the stored trained model. In total, 14 gestures that require both hands are selected, as shown in Figure 7, which enables the user to guide the robot in all dimensions (rotation, translation, and gripper actuation) and also toggles its speed (slow or fast). Section 3.2 provides details about hand gesture recognition.

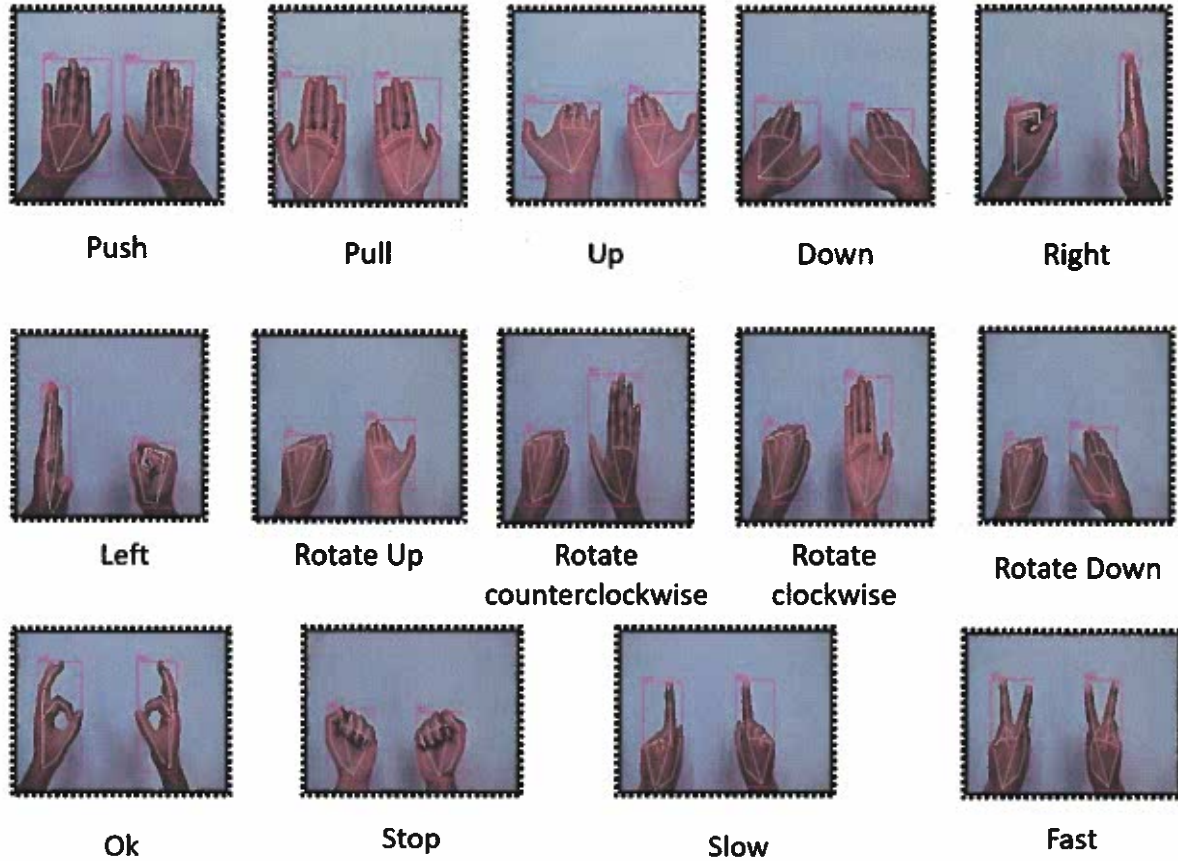


Figure 7: Dual Hand Gestures

- The camera that is mounted on the cobot's end-effector is used to detect the human's position in the workspace with respect to the robot's end-effector, i.e. if the human is on the left, right, or in front of the end-effector. Additionally, the stereoimages are used to compute the distance of the hand with respect to the camera, which is considered the hand position. Section 3.3 provides a detailed description of the human and hand pose estimation.
- The recognized gesture together with the human and hand position estimation are used as inputs by the OSAG control module, which is a novel algorithm that generates a new Carte-

sian pose for the manipulated object based on the inputs. This algorithm provides an adaptive mapping between the performed hand gestures and the robot's motion based on the human's position. For example, if the human performs the "pull" gesture based on where the human stands with respect to the robot (i.e. left, right, or center), the algorithm will generate different motions for the manipulated object. If the user stands to the left or the right side of the manipulated object and has selected a gesture that requires robot rotation, then the closest hand to the end-effector's camera acts as the pivot point of the rotation. To the best of the author's knowledge, this is the first system that adapts the mapping of manipulated object motion to the human position in the workspace. Complete mapping of motion based on the gestures and position is described in Section 3.4.

- The new cartesian pose that is desired for the manipulated object that is generated from the OSAG control is then sent to the cartesian impedance controller that regulates the joint torques of the robotic arm to achieve the desired pose for the manipulated object. The Cartesian controller runs as an attractor with constant damping and stiffness and has a frequency of 100 Hz. The controller is described in detail in Section 3.5.

The following sections discuss in detail each component of the framework.

3.1 Multi-Camera and Robotic System

The multi-camera system consists of two cameras; one RGB camera from Logitech and one ZED Mini Stereo camera from Stereolabs. The Logitech is a USB Webcam C615 Full High Definition (HD) camera with a resolution of 1080p and provides 30 frames per second (fps). Table 4 summarizes the camera's characteristics. It is mounted on an adjustable headband, which enables the camera to be easily worn and adjusted by adults. Figure 5 shows a user wearing the web camera.

Table 4: Specifications of the Logitech C615 Webcam (adapted from [2])

Resolution & frames per second (fps)	Full HD 1080p/30fps; HD 720p/30fps
Diagonal Field of View	78°
Autofocus	Yes
Auto Light Correction	RightLight 2
Connection	USB

The ZED mini stereo camera outputs RGB and depth information and is a popular sensor for robotic applications that require images and depth information [36–38]. The ZED mini is mounted on the end-effector of the robotic arm, as shown in Figure 5. During the collaborative tasks, the ZED mini camera is moved by the robotic arm in a way that can observe the human. Table 5 provides the specifications of the ZED mini camera.

The robotic system is a collaborative robotic arm (cobot) from the Franka Emika company. The cobot is called Panda and is equipped with a 2-finger gripper, as can be seen in Figure 5. Panda has 7-Degrees of Freedom (DoF) and is equipped with torque sensors in all 7-axis. According to the datasheet [39], the maximum Cartesian velocity of the end-effector is 2 m/s and the maximum payload is 3 kilograms. The 2-finger gripper can apply a continuous force of 70 N and a maximum force of 140 N.

Table 5: Specifications of the ZED Mini Camera (adapted from [3])

RGB Output	
Resolution	Side by Side 2x (2208x1242) @15fps; 2x (1920x1080) @30fps; 2x (1280x720) @60fps; 2x (672x376) @100fps
Field of View	Max. 90° (H) x 60° (V) x 100° (D)
Connection	USB
Depth Output	
Depth Range	0.10 m to 15 m (0.3 to 49 ft)
Depth Accuracy	<1.5% up to 3m <7% up to 15m

3.2 Vision-Based Hand Gesture Recognition

This section explains the vision-based hand gesture recognition that is developed in this Thesis. As the goal of this Thesis is to develop an intuitive interactive method for extended object co-manipulation, the user may wish to translate or rotate the object, adjust the speed (slow or fast), and actuate the gripper. Therefore, 14 gestures that require both hands are defined, as shown in Figure 7.

In order to recognize the 14 gestures, a deep learning-based framework is developed. As input, the image (RGB) is used from the headband web camera. The image is given to the Google Mediapipe library [40] for detecting hand landmarks [41]. The hand landmark detection model is a two-step Convolutional Neural Network (CNN) and was evaluated on different skin tones, gender, and geographical regions to ensure its accuracy [42]. The hand landmark detection model outputs and tracks 21 landmarks per hand, as shown in Figure 8. Each landmark has 3 values, x, y, and z (dimensions in 3D with respect to the camera). Let us consider the left-hand landmarks as $L_{i,k}$ and the right-hand landmarks as $R_{i,k}$ where $i = 0, 1, 2, \dots, 20$ and k represents x, y, and z dimensions.

Subsequently, the hand landmarks are used to extract the following features:

- the Euclidean distance DLL_i of each left-hand landmark i from the left-hand wrist point 0 calculated by Equation (10).

$$DLL_i = \sqrt{(L_{i,x} - L_{0,x})^2 + (L_{i,y} - L_{0,y})^2 + (L_{i,z} - L_{0,z})^2} \quad (10)$$

- the Euclidean distance DRR_i of each right-hand landmark i from the right-hand wrist point

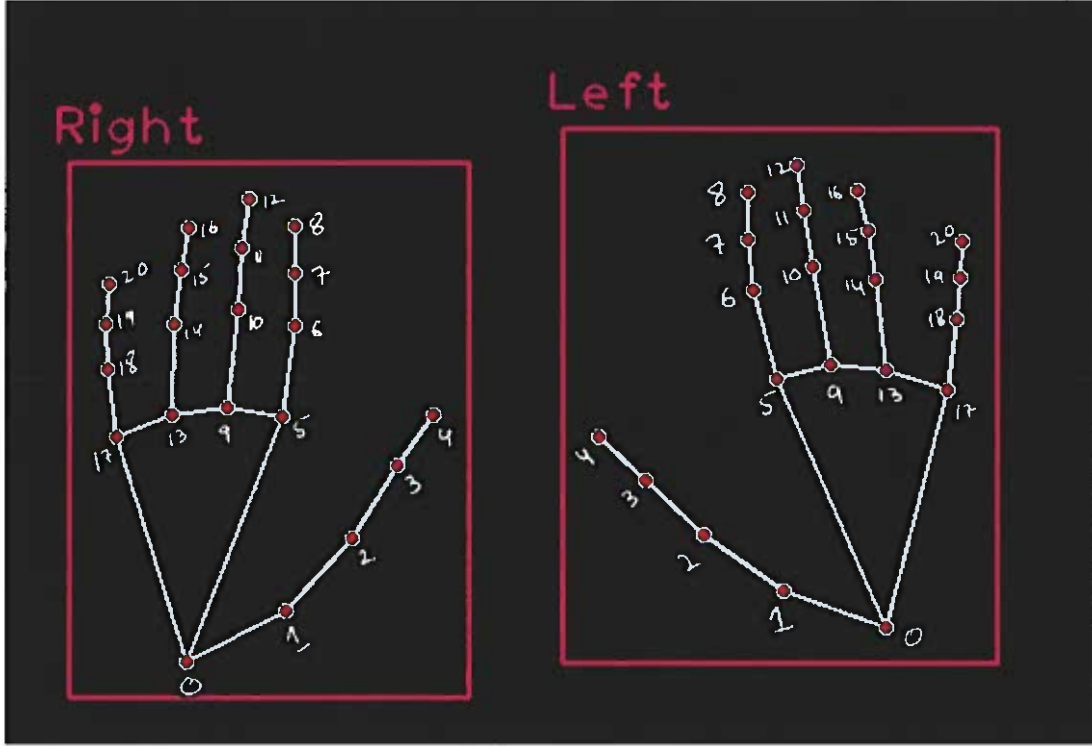


Figure 8: Left and Right Hand Landmarks.

0 calculated by Equation (11).

$$DRR_i = \sqrt{(R_{i,x} - R_{0,x})^2 + (R_{i,y} - R_{0,y})^2 + (R_{i,z} - R_{0,z})^2} \quad (11)$$

- the Euclidean distance DRL_i of each right-hand landmark i from the left-hand wrist point 0 calculated by Equation (12).

$$DRL_i = \sqrt{(R_{i,x} - L_{0,x})^2 + (R_{i,y} - L_{0,y})^2 + (R_{i,z} - L_{0,z})^2} \quad (12)$$

- the Euclidean distance DLR_i of each left-hand landmark i from the right-hand wrist point 0

calculated by Equation (13).

$$DLR_i = \sqrt{(L_{i,x} - R_{0,x})^2 + (L_{i,y} - R_{0,y})^2 + (L_{i,z} - R_{0,z})^2} \quad (13)$$

The extracted features together with the hand landmarks are normalized from 0 to 1 using min-max scaling. The normalized features are used to train a 3-layer Recurrent Neural Network (RNN) to classify the gestures. For the training, 2000 samples per gesture (class) are used. Additionally, a null class with random gestures is created with 4000 samples and the stop gesture class requires 4000 samples for improving accuracy. It is worth noting that all samples are collected by one person. In the Section 4, the accuracy of the system is discussed in a user study of 16 participants. After the RNN model is trained, it is stored in the TensorFlow Keras model so it can detect the

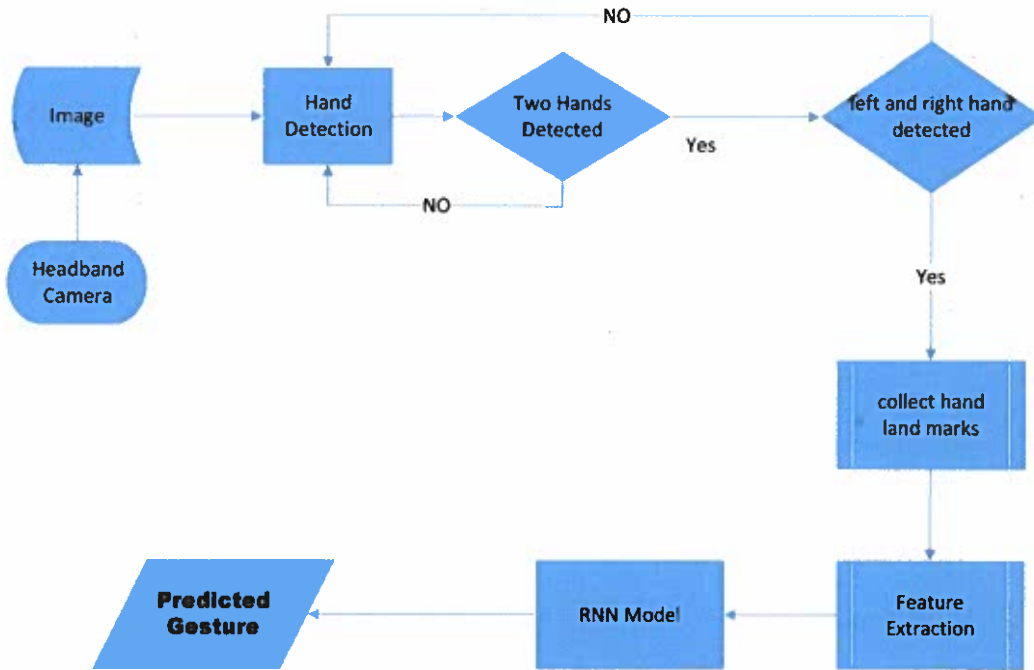


Figure 9: Flowchart of the proposed vision-based hand gesture recognition

performed gestures. Figure 9 presents the flowchart of the vision-based hand gesture recognition.

The detected (predicted) gestures are used by the OSAG control.

3.3 Human and Hand Position Estimation

This section presents the human and hand position estimation by using the RGB and Depth data from the ZED mini camera that is mounted on the cobot's end-effector. The human's position with respect to the robot is important to ensure that the robot has a better understanding of the human's input for the co-manipulation.

Human Position Estimation. The ZED mini camera has an API (Application Programming Interface) that enables 3D object detection [43] including humans. The API outputs the bounding box of the detected objects and their ID (person, vehicle, etc.). In our application, the interest is on the person who stands in front of the robot; therefore, when a human is detected within the workspace of the robot, the detected person's bounding box is used for further processing. The bounding box consists of the following four points; top left point p_{tl} , top right point p_{tr} , bottom left point p_{bl} , and bottom right point p_{br} , as shown in Figures 10 and 11. It is important to note that the developed framework assumes that only one person collaborates with the cobot for a co-manipulation task.



Figure 10: Bounding box of human standing on the left (left image) and on the right (right image) of the cobot.

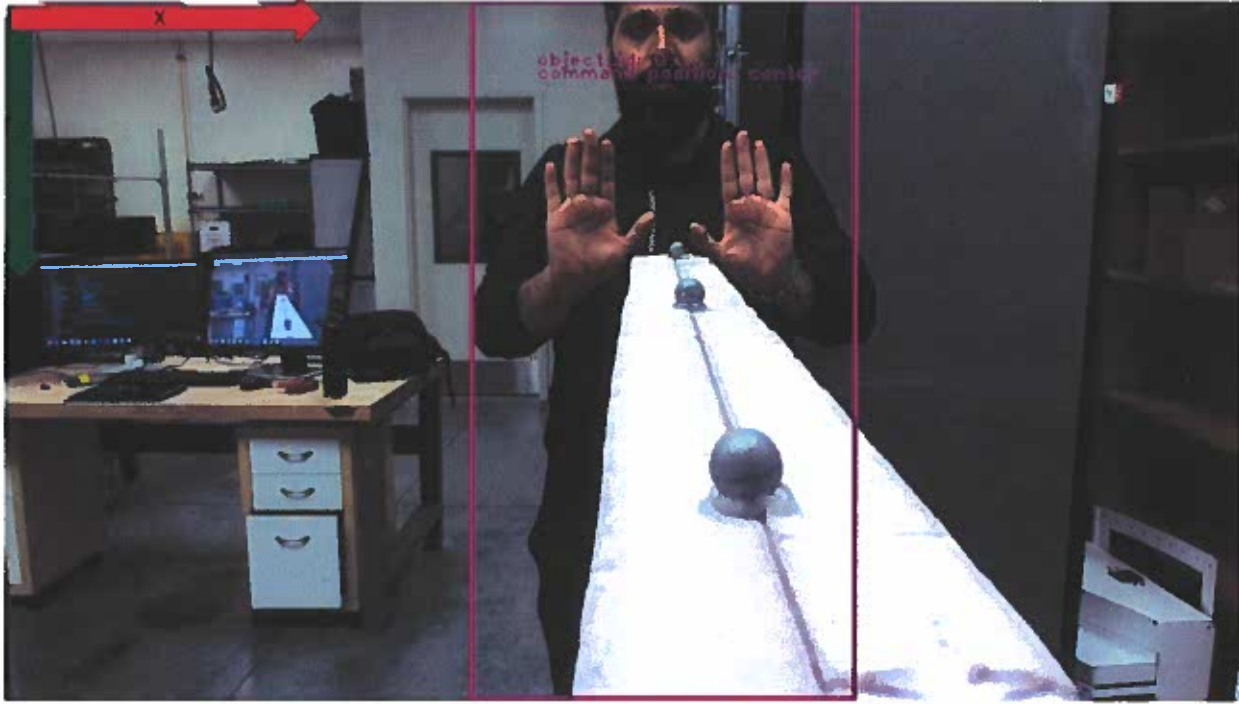


Figure 11: Bounding box of human standing in front of the cobot and the defined x and y axis of the image.

The next step is to determine the relative position of the person with respect to the robot, i.e. if the person stands on the right/left or in front of the cobot's end-effector (as shown in Figures 10 and 11). As the ZED mini camera is mounted on the cobot's end-effector, the relative position of the human can be estimated with respect to the camera coordinate system. The RGB image consists of pixel points that each have an assigned x and y value that represents the position in the image. The top left pixel of the image has x and y values equal to 0, as shown in Figure 11. The most right pixels of the image have x value equal to N , where N represents the total number of pixels in the x direction.

To estimate if the person stands on the left or right or in front of the cobot's end-effector, it is sufficient to estimate if the person's bounding box is in the center of the image, or at the left or right. Therefore, the pixel distance PXL in the x direction between the left side of the bounding

box and the left side of the image is calculated based on Equation (14). Similarly, the pixel distance PXR in the x direction between the right side of the bounding box and the right side of the image is calculated based on Equation (15).

$$PXL = p_{tl,x} - 0 \quad (14)$$

$$PXR = N - p_{tr,x} \quad (15)$$

If PXL is less than PXR and PXL value is less than a threshold value α then the person stands on the left of the cobot's end-effector. Similarly, if PXL is greater than PXR and PXR value is less than α then the person stands on the right of the cobot's end-effector. If the PXL and PXR values are greater than α , then the person stands in front of the end-effector (center of the image). The threshold value α is selected empirically as 50 cm. The estimated position of the person (Left, Right, or Center) is sent to the OSAG control (Section 3.4).

Hand Position Estimation. It is also important to estimate the hand position with respect to the cobot's end-effector in order to recognize the point at which the user would like to rotate the object. The images of ZED mini are processed by the Google Mediapipe library [40] for detecting hand landmarks [41]. Figure 8 shows the hand landmarks. ZED mini camera also provides depth data, and each pixel of the RGB image has a depth value that represents the distance from the camera. Therefore, to estimate the distance of the hand that is closer to the ZED mini camera, the average depth value of the landmark 9 for the hand and its 8 neighboring pixels is calculated. The hand position estimation is then sent to the OSAG control (Section 3.4). Figure 12 provides the flowchart of the human and hand position estimation process described in this section.

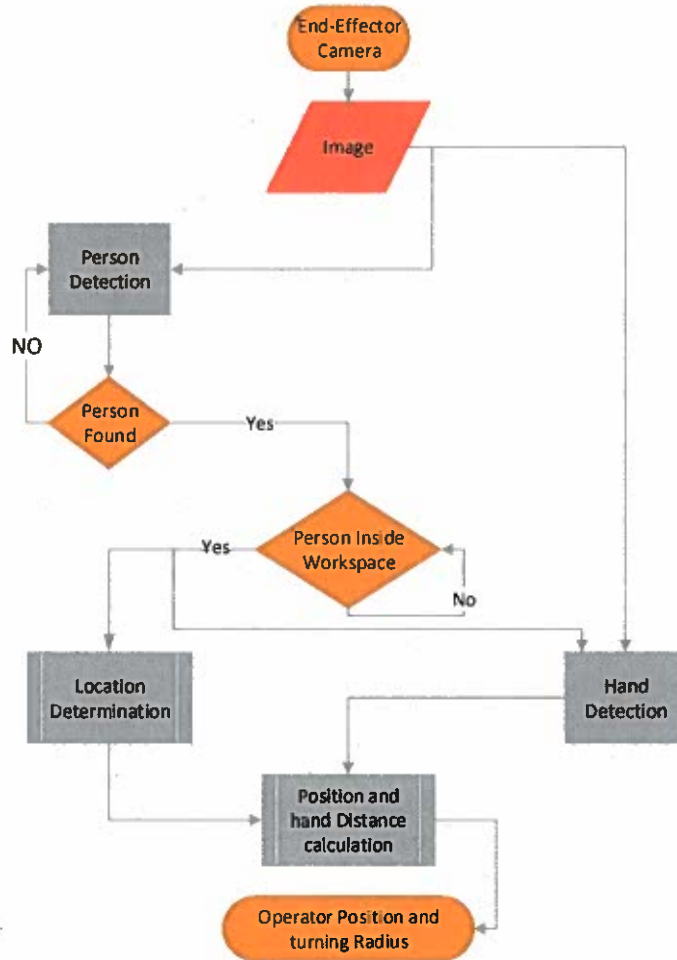


Figure 12: Human and Hand Position Estimation

3.4 Object-Centric Spatially-Aware Gesture-based (OSAG) Control

This section describes the Object-Centric Spatially-Aware Gesture-based (OSAG) control (OSAG control for short) that is developed in this Thesis. The OSAG control uses four inputs: gesture, representing the detected hand gesture (described in Section 3.2); position, indicating the current human position (“left”, “right”, “center”) (described in Section 3.3); distance, providing information about the user’s hand distance to the robot’s end-effector (described in Section 3.3), and the

current robot pose published by ROS. Table 2 presents the mapping between the robot commands and the inputs from the human (gesture recognition and human position).

The OSAG control algorithm uses the current robot pose as one of its inputs. The current robot pose, which contains the 3-dimensional position of the end-effector and the four quaternion values that represent the rotation, is separated into two 3-dimensional vectors, one is denoted as ${}^B\mathbf{v}$ that represents the translation of the robot's end-effector with respect to the robot base B and one is denoted as ${}^B\mathbf{r}$ that represents the rotation as Euler angles (i.e. roll, pitch and yaw) of the robot's end-effector with respect to the robot base B . The robot base frame is shown in Figure 13. The Euler angles are calculated using the four quaternion values using the equations proposed in [44]. Next, based on the human's inputs, the human's position with respect to the robot, and the human's distance to the robot, the OSAG control algorithm takes the following action:

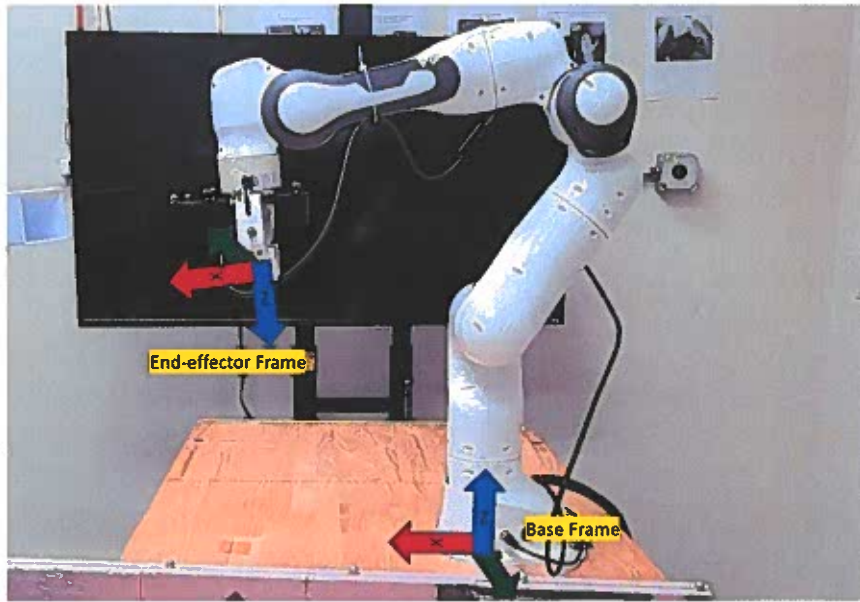


Figure 13: Franka Emika Panda robot with end-effector and robot base frames. Red, Green and Blue represent the X, Y and Z-axis, respectively.

Speed Control: The algorithm selects between two speeds (slow or fast) based on the user's

gesture (gestures slow and fast are shown in Figure 7). The default speed if the user does not select a speed is slow. The slow and fast speeds for translation are equal to 0.02 m/s and 0.04 m/s, respectively. The slow and fast speeds for rotation are equal to 2° per second and 4° per second, respectively. The user can change the speed at any time by performing one of the two gestures (i.e. slow or fast). Additionally, there is the stop gesture, which stops the robot's motion.

Control in Translation: When the user performs one of the following gestures, push, pull, left, right, up, and down, the algorithm will take into account the user's position (left, right, or center) to define in which direction the robot moves. For example, if the user makes the gesture "push" and stands on the right of the robot's end-effector, the robot will move in the negative direction of the Y-axis of the end-effector. However, if the user stands at the center, then the end-effector will move in the negative direction of the X-axis. Table 2 represents all the robot commands based on the gesture and the position of the human with respect to the robot's end-effector. Before the robot starts moving, the end-effector is at position ${}^E\mathbf{v} = [0, 0, 0]^T$ in the end-effector's coordinate system E (shown in Figure 13). The new position of the end-effector is defined based on the user's gesture and position and the selected speed, e.g. if the user's gesture is right and stands at the left of the end-effector and speed is selected as 0.02 m/s, then the algorithm adds a step of 0.002 m every 1/10 of the second in the positive X-axis and therefore the new end-effector position is $[0.002, 0, 0]^T$ with respect to the end-effector's original position. In another example, if the user stands on the center of the end-effector and performs the right gesture, then the algorithm will subtract 0.002 m every 1/10 of the second in the Y-axis and therefore the new end-effector position is $[0, -0.002, 0]^T$ with respect to the end-effector's original position.

Let ${}^E\mathbf{v}_{new}$ represent the new end-effector position with respect to the end-effector frame E . In order to control the robot, the OSAG controller is required to output the desired robot translation with respect to the robot base not with respect to the robot's end-effector. Therefore, the manipula-

tor's kinematics ${}^B_E\mathbf{T}$ is used, which represents the transformation of the robot's end-effector frame E with respect to the robot's base frame B . The new end-effector pose ${}^B\mathbf{v}_{new}$ with respect to the robot base will be calculated by the following equation:

$${}^B\mathbf{v}_{new} = {}^B_E\mathbf{T} \cdot {}^E\mathbf{v}_{new} \quad (16)$$

The output of the OSAG controller is denoted as x_d and calculated by Equation 17. The x_d is then sent to the Cartesian Impedance controller (Section 3.5).

$$x_d = \begin{bmatrix} {}^B\mathbf{v}_{new} \\ {}^B\mathbf{r} \end{bmatrix} = {}^B_E\mathbf{T} \cdot \begin{bmatrix} {}^E\mathbf{v}_{new} \\ {}^E\mathbf{r} \end{bmatrix} \quad (17)$$

Control in Rotation: When the user performs one of the following gestures, rotate up, rotate down, rotate clockwise, and rotate counterclockwise, the algorithm will consider the user's position (Left, Right, or Center) and the position of the hand that is closer to the camera (as explained in Section 3.3) in order to calculate the desired robot movement. First, the position of the hand relative to the cobot mounted camera is projected onto the object to location of the desired pivot point and the distance from the pivot point to the current end-effector position, d_E . Then, the algorithm calculates the desired step using the following equation:

$$step = 2 * d_E * \sin(\phi/2) \quad (18)$$

where ϕ is equal to 0.2 degrees if slow speed is selected or 0.4 degrees if fast speed is selected (as explained in Section 3.3).

Next, the algorithm has to calculate the new pose of the end-effector. The starting position

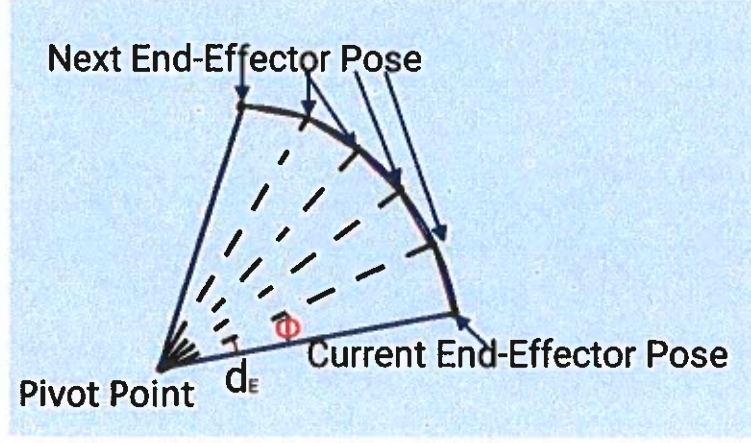


Figure 14: Calculation of translation and rotation step based on user input.

of the end-effector is considered as ${}^E\mathbf{v} = [0, 0, 0]^T$ and the starting rotation of the end-effector is ${}^E\mathbf{r} = [0, 0, 0]^T$. Based on the user's selection of rotation (either around the X, Y, or Z-axis), the new pose of the end-effector is calculated as follows:

- If rotation is around the X-axis: ${}^E\mathbf{v}_{new} = [0, 0, 0]^T$ and ${}^E\mathbf{r}_{new} = [\pm\phi, 0, 0]^T$
- If rotation is around the Y-axis: ${}^E\mathbf{v}_{new} = [0, 0, step]^T$ and ${}^E\mathbf{r}_{new} = [0, \pm\phi, 0]^T$
- If rotation is around the Z-axis: ${}^E\mathbf{v}_{new} = [0, step, 0]^T$ and ${}^E\mathbf{r}_{new} = [0, 0, \pm\phi]^T$

where \pm denotes that if rotation is in the positive direction, it will be plus (+), and if it is in the negative direction, it will be minus (-). The output of the OSAG controller x_d is calculated by the equation (19) and it is sent to the Cartesian Impedance controller (Section 3.5).

$$x_d = \begin{bmatrix} {}^B\mathbf{v}_{new} \\ {}^B\mathbf{r}_{new} \end{bmatrix} = {}^B_E\mathbf{T} \cdot \begin{bmatrix} {}^E\mathbf{v}_{new} \\ {}^E\mathbf{r}_{new} \end{bmatrix} \quad (19)$$

Gripper Operation: When a robot has to manipulate objects, it is also important to actuate the gripper. Therefore, the gesture “OK” shown in Figure 7 is used to enable the gripper to grasp

or release an object. As the gripper has only two states (open and close), the gesture “OK” is used for the transition from the open to the close state and vice versa.

3.5 Cartesian Impedance Controller

The OSAG control outputs the desired end-effector pose. The Cartesian impedance controller takes as input the desired end-effector pose and the current state of the joint space of the robot and calculates the required torques in the joint space to enable the robot to move to the desired end-effector pose.

The Cartesian impedance controller developed in this Thesis was inspired by Mayr & Salt-Ducaju [45]. The Panda robot is a 7-DoF robotic manipulator and its joint space is denoted by $\theta \in \mathbb{R}^7$. The gravity-compensated dynamics of the Panda robot can be described by the following equation [45, 46]:

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} = \tau(\theta) \quad (20)$$

where $M(\theta) \in \mathbb{R}^{7 \times 7}$ represents the generalized inertia matrix, $C(\theta, \dot{\theta}) \in \mathbb{R}^{7 \times 7}$ represents the effects of Coriolis and centripetal forces and $\tau(\theta) \in \mathbb{R}^7$ is the vector of the actuator torques. The torque $\tau(\theta)$ is calculated by adding two joint-torque signals, as shown in the following equation:

$$\tau(\theta) = \tau_{ca}(\theta) + \tau_{ns}(\theta) \quad (21)$$

The $\tau^{ca}(\theta)$ is the torque required to achieve a Cartesian impedance behavior based on the error

$e_{ca} \in \mathbb{R}^6$ in Cartesian space in the frame of the robot's end-effector [45, 47]:

$$\tau_{ca}(\theta) = J^T(\theta)[-K_{ca}e_{ca} - K_{ca}J(\theta)\dot{\theta}] \quad (22)$$

with $J(\theta) \in \mathbb{R}^{6 \times 7}$ is the Jacobian matrix relative to the robot's end-effector, $K_{ca} \in \mathbb{R}^{6 \times 6}$ is the virtual Cartesian stiffness matrix, $D_{ca} \in \mathbb{R}^{6 \times 6}$ is the virtual Cartesian damping matrix and error e_{ca} is the difference between the desired Cartesian pose x_d and the current Cartesian pose x_c , i.e. $e_{ca} = x_d - x_c$.

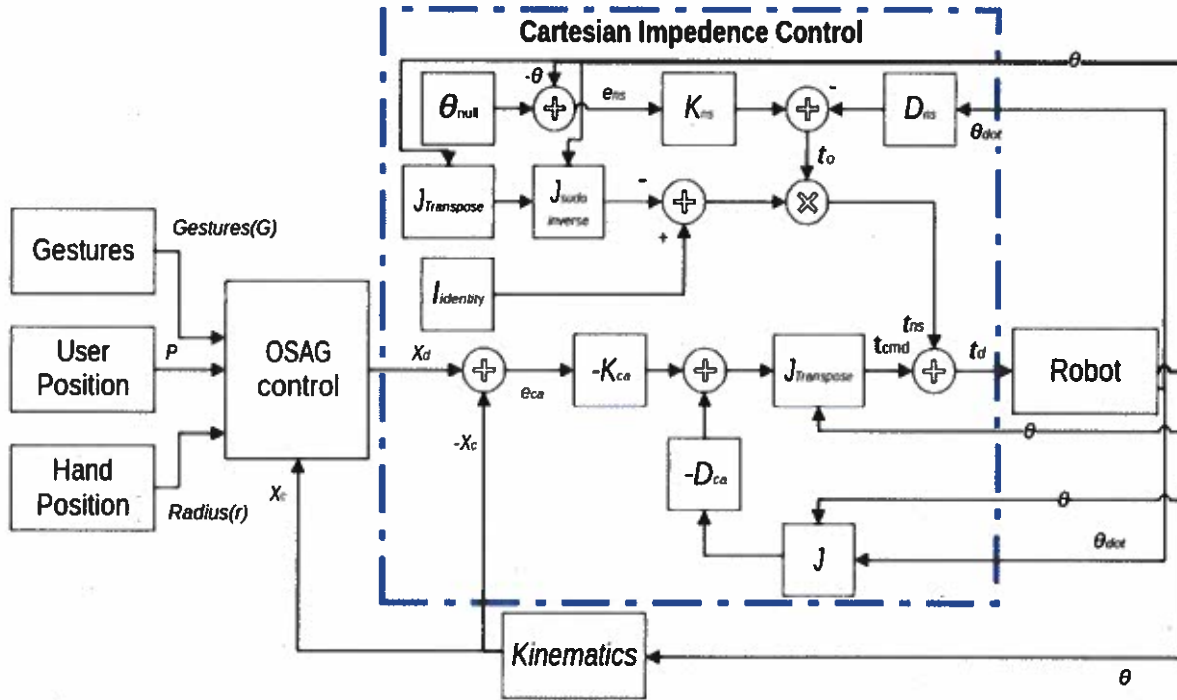


Figure 15: Control diagram of the Cartesian Impedance Controller

The $\tau_{ns}(\theta)$ is the torque required to achieve a joint impedance behavior with respect to a desired configuration and projected in the null space of the robot's Jacobian in order to have a smooth

Cartesian motion of the robot's end-effector [45, 48]:

$$\tau_{ns}(\theta) = [I_7 - J^T(\theta)(J^T(\theta))^{\dagger}] [-K_{ns}e_{ns} - D_{ns}\dot{\theta}] \quad (23)$$

where $K_{ns} \in \mathbb{R}^{7 \times 7}$ is the virtual joint stiffness matrix, $D_{ns} \in \mathbb{R}^{7 \times 7}$ is the virtual joint damping matrix, error e_{ns} is the difference between the desired joint space configuration θ_{null} and the current joint space configuration θ , i.e. $e_{ns} = \theta_{null} - \theta$, and $J^T(\theta)^{\dagger}$ is the Moore-Penrose pseudoinverse matrix [45, 49, 50] which is calculated by the following equation:

$$J(\theta)^{\dagger} = (J^T(\theta) \cdot J(\theta))^{-1} \cdot J^T(\theta) \quad (24)$$

The calculated torque from Equations (21), (22) and (23) is then sent to the robot joints using ROS. Figure 15 shows the Cartesian Impedance control diagram.

3.6 ROS Implementation

For the implementation of the developed system, the Robot Operating System (ROS) is selected as it is a middle-ware that enables communication between different hardware and software components. Figure 16 shows an overview of the ROS implementation for the developed system. ROS acts as a middle-ware framework, streamlining how different parts of the robotic system interact. ROS has several components, such as nodes, topics, messages, services, etc. The proposed system uses nodes and topics. Nodes are Python or C++ scripts of specific functionality and topics are used by nodes to transmit data.

Several ROS nodes are depicted in the block diagram shown in Figure 16. One such node, the Hand Gesture Node, is responsible for capturing and interpreting hand gestures from the Operator.

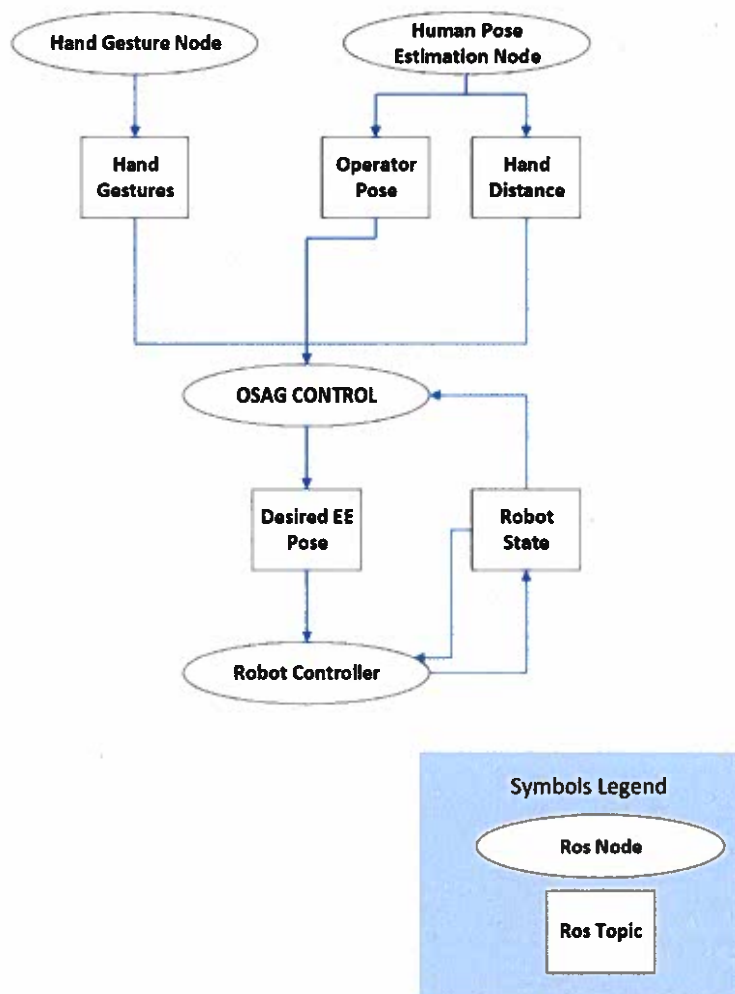


Figure 16: ROS Architecture

Another node, the Human Pose Estimation Node, focuses on estimating a human's pose (Left, Right, or Center) based on visual input from a camera and also estimates the distance of the closest gesturing hand to the camera.

The OSAG Control node plays a crucial role in controlling the cobot and the gripper. It achieves this by subscribing to topics published by the Hand Gesture and Human Pose Estimation nodes. By interpreting the information from these topics, the OSAG Control node determines the desired

pose for the cobot's end effector. This desired pose information is then published on another ROS topic.

The Robot State node keeps the OSAG node updated by publishing data about the cobot's current state, such as the position of its various joints and end-effector position. This information is crucial for calculating the next desired end-effector Pose. The Robot Controller subscribes to two topics: one that publishes the desired end-effector pose from the OSAG Control node and another that publishes the robot's state information from the Robot State node. By combining these inputs, the Robot Controller calculates the control commands necessary to move the robot toward the desired pose. These calculated commands are then sent to the robot's actuators.

Figure 16 showcases how ROS facilitates communication and coordination between various software components for the developed system. This allows the cobot to respond to human gestures, interpret human poses, and ultimately perform actions based on the perceived information. As it is ROS-based architecture, it can easily be implemented in other cobots that are ROS-enabled.

The software code for the complete Object-Centric Spatially-Aware Gesture-based motion control in ROS can be found in the following GitHub repositories:

https://github.com/Rehan080374/thesis_code.git

4 Experimental Results

Cobots are designed to work alongside humans; therefore, new methods of interaction with cobots are important to be evaluated by humans. To evaluate the proposed vision-based framework for human-robot co-manipulation, a user study is conducted. The study was approved by the Institutional Review Board (IRB) at Santa Clara University with the IRB protocol #23-09-2017. This Section provides details on the experimental setup and protocol, user study, and system evaluation.

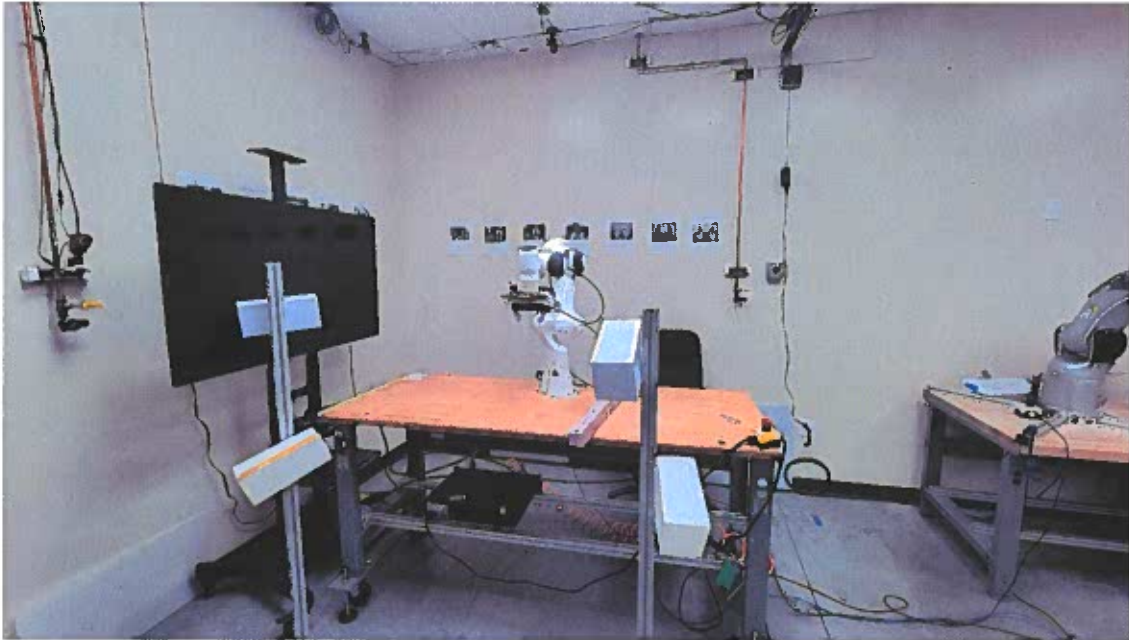


Figure 17: Robot Setup for the User Study

4.1 Experimental Setup & Study Protocol

This section explains the experimental setup that is used to evaluate the proposed framework and the study protocol approved by the IRB (Study protocol #23-09-2017).

Experimental Setup. The experimental setup is shown in Figure 17. The Panda cobot is

mounted on a table and four white tubes (boxes) with different orientations are mounted on two stands, as shown in Figure 17. An extended object, which is positioned on the table is the object that requires to be moved to each of the white tubes. The extended object is a meter-long light object (foam). The user has to control the robot using hand gestures while having the freedom to move around the cobot's workspace in order to complete the co-manipulation task.

Table 6: Basic Questionnaire

Questions	Expected answers
1. What is your age group?	age
2. Please indicate your gender(s).	gender
3. Please indicate your race-ethnicity(ies).	ethnicity
4. Are you Right-Handed, Left-Handed, or Ambidextrous?	hand preference
5. Can you move your hands and arms without any difficulties?	arm mobility
6. Can you move your body from the waist up without any difficulties?	upper body mobility
7. Do you use any assistive mobility devices?	assistive device use
8. Have you ever collaborated with a robot before to complete a task?	robot collaboration history (1-5)
9. I am familiar with Gesture Control	gesture control history (1-5)
10. I am comfortable collaborating with robots	comfort level(1-5)

Table 7: NASA Task Load Index Questionnaire

Questions	Average Score(0-10)	Standard Deviation
How mentally demanding was the task?	4.875	2.78
How physically demanding was the task?	4.75	1.95
How hurried or rushed was the pace of the task?	2.9375	1.88
How successful were you in accomplishing what you were asked to do?	8.25	1.34
How hard did you have to work to accomplish your level of performance?	5.5	2.45
How insecure, discouraged, irritated, stressed, and annoyed were you?	2.6875	2.65

Table 8: Human Robot Co-manipulation Questionnaire

Questions	Average score (0-10)	Standard Deviation
Perceived Usefulness		
I accomplished the given tasks in reasonable time.	7.625	1.89
I accomplished the given tasks successfully.	8.5	1.62
Perceived Safety and trust		
The robot's actions were predictable.	7.06	2.63
I felt safe using the robot.	9.18	1.33
I felt comfortable using the robot at low speeds.	8.5	2.39
I felt comfortable using the robot at high speeds.	7.25	2.19
Perceived ease of use		
I found the robot easy to use.	7.68	1.48
Being able to move around the robot made it easier to manipulate the object.	7.37	2.49
Gesture commands were intuitive.	7.93	1.67
The robot met my expectations.	8.37	1.36
Perceived Interaction		
I had to learn more about robots in order to be able to interact with the system.	2.12	2.31
My gestures were easily recognized by the system.	6.5	2.0
It was easy to perform the gestures while looking at the object that was manipulated.	7.5	1.98
I was able to switch between different commands rapidly.	7.82	2.50
Ethical Consideration		
I am concerned about my privacy when using the robot.	1	1.41
Open-ended Questions		
What additional functionalities should the robot have?	n/a	n/a
What did you like about the robotic system?	n/a	n/a
What frustrated you about the robotic system?	n/a	n/a
Please provide any additional comments/feedback for the robotic system.	n/a	n/a

Table 9: System Usability Scale (SUS) Questionnaire

Questionnaire	Average Score (1-5)	Standard Deviation
I think I would like to use this system frequently.	3.20	0.86
I found the system unnecessarily complex.	2.00	1.07
I thought the system was easy to use.	3.53	0.83
I think that I would need the support of a technical person to be able to use this system.	2.07	1.33
I found the various functions in this system were well integrated.	3.67	1.11
I thought there was too much inconsistency in this system.	2.13	0.83
I would imagine that most people would learn to use this system very quickly.	4.07	0.70
I found the system very cumbersome to use.	2.27	0.96
I felt very confident using the system.	4.00	0.53
I needed to learn a lot of things before I could	1.93	1.03

Study Protocol. To evaluate our framework, a user study was conducted. The study took place in one visit and the following procedure was followed:

1. The participant read and signed the consent form approved by the IRB.
2. The participant filled out a basic questionnaire that included questions about demographics, and experience with robots, and also confirmed that the participants have upper-body mobility. The basic questionnaire can be found in Table 6.
3. The participant wore the headband with the camera. The headband was sanitized before and after every use. The participant also wore safety glasses.
4. The participant had 15 minutes of training on the hand gestures and how the hand gestures are mapped to robot control. Then the participant took a 5-minute break and then he/she/they completed a task. The task required the user to control the robot using gestures to pick up a

meter-long light object (foam) and place it into 4 different tubes. This took approximately 30 minutes.

5. After the task is completed, the participant filled out the following 3 questionnaires: System Usability Scale (SUS) [51] (see Table 9), NASA Task Load Index [52] (see Table 7) and Human-Robot Collaboration Questionnaire (see Table 8).

4.2 User Study and Results

Sixteen adult participants were recruited as volunteers from faculty, students, and staff from the School of Engineering at Santa Clara University (SCU). The advertisement of the study was conducted via e-mail. Of the 16 participants, twelve were male, two were female and two declined to answer. The age distribution of the participants can be found in Table 10. Half of the participants (8) had no prior experience collaborating with a robot to complete a task before, while six had prior experience and two declined to answer.

Table 10: Age distribution of the study participants

Age group	No. of Participants
18-24 years old	8
25-34 years old	5
55-64 years old	1
Decline to answer	2

All the participants were able to complete the task, which was to control the robot to move an extended object from the table to three of four white tubes (boxes) (see Figure 17). The completion time for each participant can be found in Table 11. The minimum completion time for the task was 270 seconds (~ 4.5 min) while the maximum completion time was 1070 seconds (~ 18 min). The average time was 715.31 seconds (~ 12 min).

Table 11: Completion time for each participant, including average and standard deviation.

Participant No	Completion Time (sec)	Participant No	Completion Time (sec)
1	1025	9	795
2	680	10	730
3	890	11	570
4	1070	12	530
5	840	13	620
6	880	14	595
7	550	15	710
8	690	16	270
Average	715.31	Standard Deviation	201.33

One of the metrics to evaluate a new system is the System Usability Scale (SUS) [51]. SUS requires the participants to rate 10 sentences from 1 to 5, where 1 means strongly disagree and 5 means strongly agree. The sentences can be seen in Table 9 and also the average score and standard deviation for each answer by the participants. The SUS score is then calculated based on the following formula [51]:

$$SUS = 2.5 * (Q_1 + Q_3 + Q_5 + Q_7 + Q_9 - 5 + 25 - Q_2 - Q_4 - Q_6 - Q_8 - Q_{10}) \quad (25)$$

where Q_i represents the rate for the i^{th} sentence of SUS and $i = 1, 2, \dots, 10$. The SUS score is then a number between 0 and 100. If the score is above 68, the perceived usability will be considered above average and if it is below 68, the perceived usability will be considered below average [53]. Table 12 shows the SUS score for each participant and also the average and standard deviation. Seven participants' SUS score is above 68 and eight participants' SUS score is below 68. The average score is 67.67 which is very close to the 68 threshold value. These are promising results but it would be interesting to evaluate our system with workers in industry. Moreover, the NASA Task Load Index [52], which is presented in Table 7), required the participants to answer

six questions by rating from 0 to 10, where 0 means low and 10 means high. Table 7 shows that the average score for how mentally and physically demanding the task is 4.875 and 4.75, respectively. This means that the task was in a medium level of mentally and physically demanding. Similarly, the participants' responses were also in the middle level regarding how hard they had to work to accomplish their level of performance. Additionally, the participants did not find that the pace of the task was hurried or rushed (average score is ~ 2.94) and they were not insecure, discouraged, irritated, stressed, or annoyed by what they had to do (average score is ~ 2.69). The participants felt they were successful in accomplishing what they were asked to do (average score is 8.5), which agrees with the fact that all participants were able to complete the task successfully.

Table 12: SUS Score for each participant, including average and standard deviation

Participant No	SUS Score (0-100)	Participant No	SUS Score (0-100)
1	47.5	9	52.5
2	80	10	52.5
3	75	11	87.5
4	52.5	12	65
5	62.5	13	55
6	72.5	14	75
7	97.5	15	75
8	65	16	Decline to reply
Average	67.67	Standard Deviation	14.14

To evaluate further the developed system, a custom questionnaire was designed called Human-Robot Co-manipulation Questionnaire, which was inspired by [54–56] and can be found in Table 8. The Human-Robot Co-manipulation Questionnaire included 15 sentences that the participants had to score between 0 (which means strongly disagree) and 10 (which means strongly agree), answer 3 open-ended questions, and lastly provide any additional comments/feedback for the robotic system. The 15 sentences were grouped under the following five categories; perceived usefulness, perceived safety and trust, perceived ease of use, perceived interaction and ethical considerations.

It is worth noting that the average scores for perceived usefulness, safety and trust, and ease to use were above 7 as shown in Table 8. Additionally, the participants felt that they did not have to learn more about robots in order to be able to interact with the system (average score of 2.12), which demonstrated that the participants were comfortable interacting with the robot. One important point is that in the sentence *"I am concerned about my privacy when using the robot"* and even if the system included cameras, the average score was 1 and the standard deviation was 1.41. This is an interesting finding that will be discussed further in Section 4.4. From the open-ended questions, interesting ideas were provided, such as using a body-mounted camera instead of a head-mounted camera, enabling the user to select which gesture corresponds to which robot command, having additional speed modes, and adding additional grippers to the robot for different tasks.

From the presented results of the user study, it can be observed that the participants found the system potentially useful and easy to use and they perceived it as safe and comfortable. In the next section, the objective metrics of evaluating the developed system are discussed.

4.3 System Evaluation

To evaluate the developed system, it is important to compare it with a system that is considered State-of-the-Art (SOTA). The selected SOTA system is the OptiTrack motion capture system [57], which is available at the SCU's Robotic Systems Lab. The OptiTrack system consists of twelve Flex V100 cameras (see Table 13 for the specifications of Flex V100) and it tracks retroreflective markers to provide the 3D pose of the objects or humans. In our scenario, the cameras are mounted on the ceiling (see Figure 17) and the markers are attached to the robot's end-effector and the extended object (see Figure 5). For tracking the human hands, bands with markers are used. Figure 18 presents the view of the OptiTrack system on its own software called Motive.

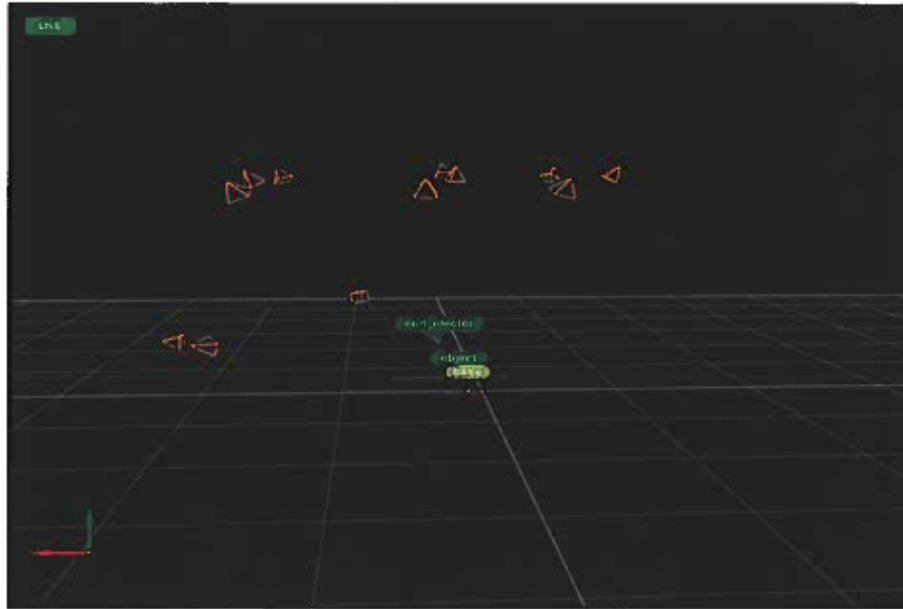


Figure 18: OptiTrack View in Motive Software(OptiTrack cameras in orange and robot end-effector in green)

Table 13: Specifications for Flex V100 camera used in OptiTrack

Frame Rate:	100hz
Default Shutter Time	1ms (1/1000th of a second)
Minimum Shutter Time	20us (1/50,000th of a second)
Resolution	640 x 480
Imaging modes	Pre-processed, Greyscale, Pre-processed objects
Interface	USB 2.0 Hi-speed, mini USB B-type
Mounting	Standard tripod -20 thread
Latency	10ms
Accuracy	2D sub millimeter, depending on marker size and distance to camera
Operating Range	15cm - 6 meters, depending on marker size
LED Ring	IR @ 850nm, 30 LEDs 45 degree FOV, adjustable brightness, removable
Status LEDs	two digit numeric LED and 2 status LEDs with PC control
Lens	45 degrees, IR 800nm bandpass coated
Lens mount	M12 Lens Holder
Power	5V @490ma, including IR LED Ring
Dimensions	1.78"(W) x 2.75"(H) x (0.81"(D)+ 0.60"(D LED))

Figure 19 shows the position of the robot's end-effector in the X-axis as measured by the Optitrack and calculated by the robot's forward kinematics (based on the position of the robot's joints) in comparison to the desired position generated by the OSAG control (Section 3.4). The figure shows the absolute error in the X-axis between the calculated robot's end-effector position and the measured position by the Optitrack (top), between the desired position of the robot's end-effector and the measured position by the Optitrack (center) and between the desired position of the robot's end-effector and the calculated robot's end-effector position (bottom). As can be observed in Figure 19, the minimum absolute error is 0 m and the maximum absolute error is 0.018 m between the calculated robot's end-effector position and the measured position by Optitrack, 0.027 m between the desired position of the robot's end-effector and the measured position by the Optitrack, and 0.024 m between the desired position of the robot's end-effector and the calculated robot's end-effector position. The average error is shown in Table 14.

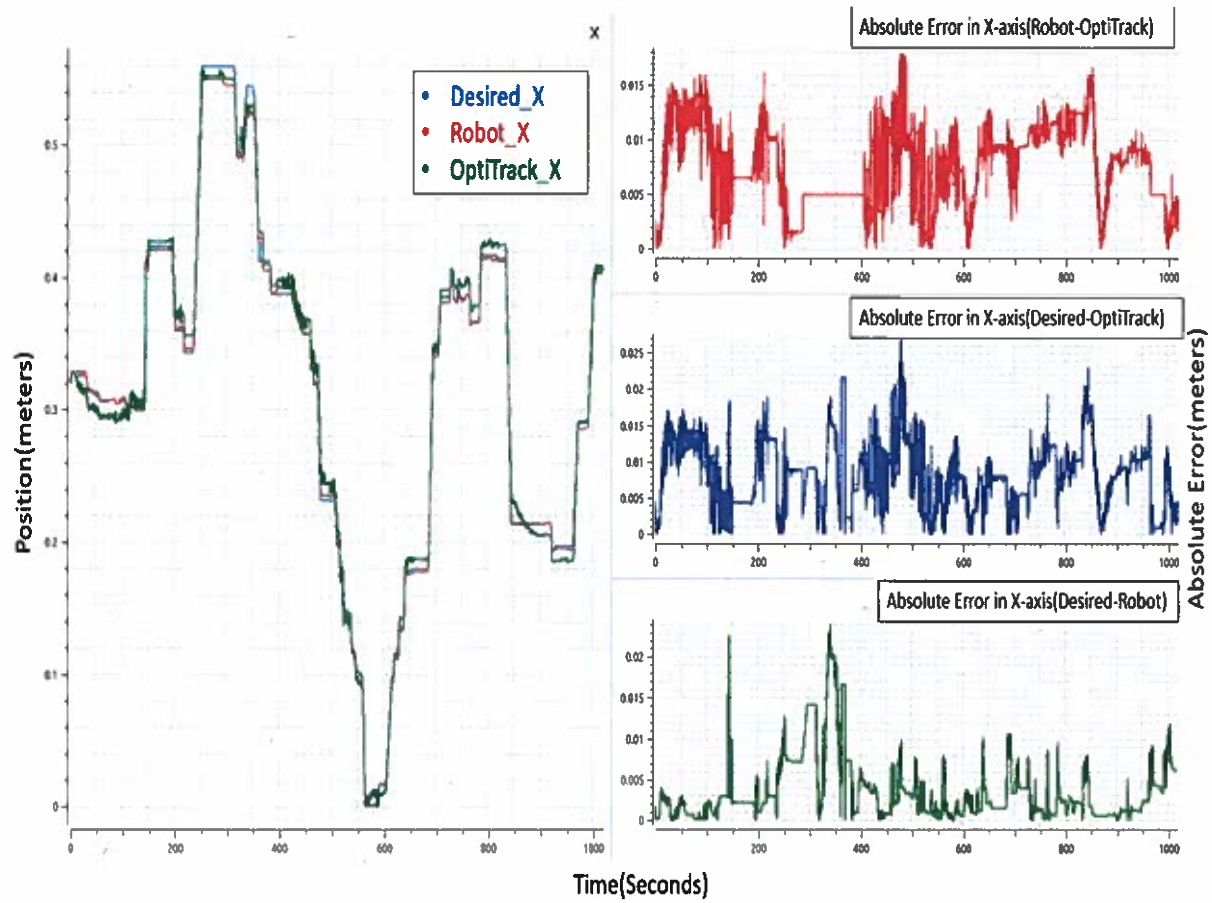


Figure 19: Robot's End-Effector Position in X-axis; Desired, measured by Optitrack and calculated by robot's forward kinematics (left). Absolute error in X-axis (right)

Table 14: Minimum, Maximum and Average Absolute Error for Robot-OptiTrack, Desired-OptiTrack and Desired-Robot in X-axis

Absolute Error in m (X-axis)	Min	Max	Average
Robot-OptiTrack	0.000001	0.017967	0.007603
Desired-OptiTrack	0.000000	0.026914	0.008146
Desired-Robot	0.000001	0.023986	0.003545

Figure 20 shows the position of the robot's end-effector in the Y-axis as measured by the Optitrack and calculated by the robot's forward kinematics (based on the position of the robot's joints) in comparison to the desired position generated by the OSAG control (Section 3.4). It also shows the absolute error in the Y-axis between the calculated robot's end-effector position and the measured position by the Optitrack (top), between the desired position of the robot's end-effector and the measured position by the Optitrack (center) and between the desired position of the robot's end-effector and the calculated robot's end-effector position (bottom). As can be observed in Figure 20, the minimum absolute error is 0 m and the maximum absolute error is 0.03 m between the calculated robot's end-effector position and the measured position by Optitrack, 0.036 m between the desired position of the robot's end-effector and the measured position by the Optitrack, and 0.024 m between the desired position of the robot's end-effector and the calculated robot's end-effector position. The average error is shown in Table 15.

Table 15: Minimum, Maximum and Average Absolute Error for Robot-OptiTrack, Desired-OptiTrack and Desired-Robot in Y-axis

Absolute Error in m (Y-axis)	Min	Max	Average
Robot-OptiTrack	0.000006	0.030553	0.013514
Desired-OptiTrack	0.000000	0.036480	0.013441
Desired-Robot	0.000000	0.023474	0.003275

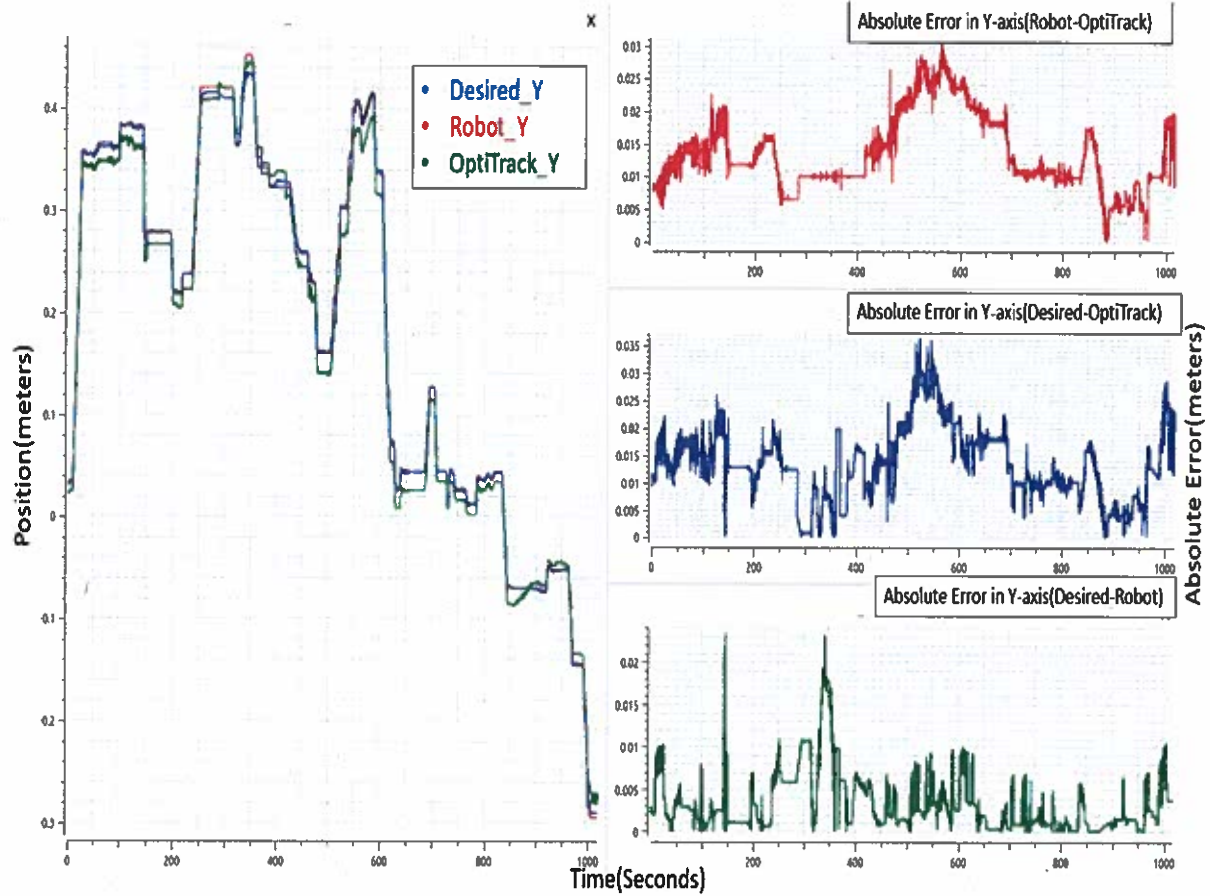


Figure 20: Robot's End-Effector Position in Y-axis; Desired, measured by Optitrack and calculated by robot's forward kinematics (left). Absolute error in Y-axis (right)

Figure 21 shows the position of the robot's end-effector in the Z-axis as measured by the Optitrack and calculated by the robot's forward kinematics (based on the position of the robot's joints) in comparison to the desired position generated by the OSAG control (Section 3.4). It also shows the absolute error in the Z-axis between the calculated robot's end-effector position and the measured position by the Optitrack (top), between the desired position of the robot's end-effector and the measured position by the Optitrack (center) and between the desired position of the robot's end-effector and the calculated robot's end-effector position (bottom). As can be observed in Figure 21,

the minimum absolute error is 0 m and the maximum absolute error is 0.014 m between the calculated robot's end-effector position and the measured position by Optitrack, 0.03 m between the desired position of the robot's end-effector and the measured position by the Optitrack, and 0.03 m between the desired position of the robot's end-effector and the calculated robot's end-effector position. The average error is shown in Table 16.

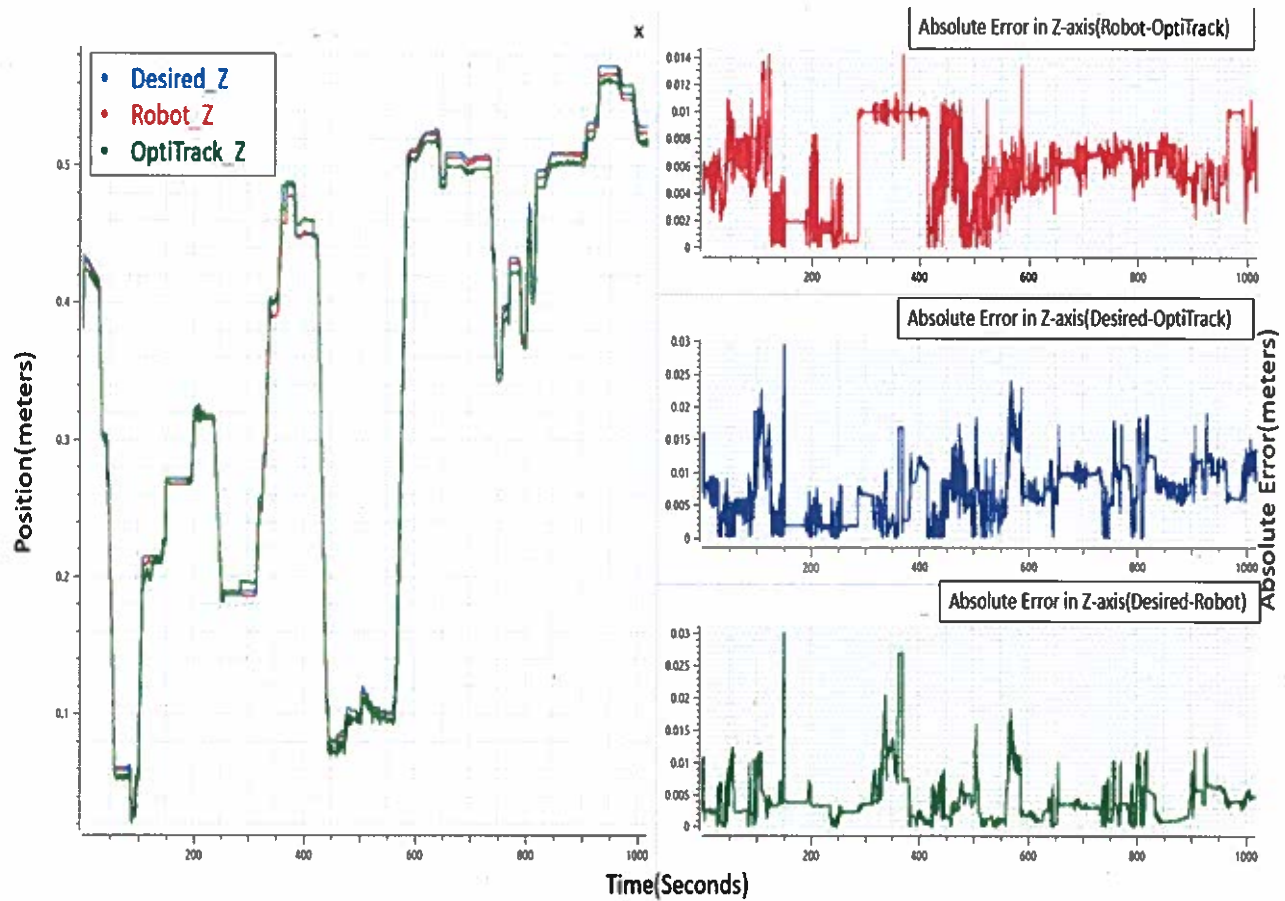


Figure 21: Robot's End-Effector Position in Z-axis; Desired, measured by Optitrack and calculated by robot's forward kinematics (left). Absolute error in Z-axis (right)

Table 16: Minimum, Maximum and Average Absolute Error for Robot-OptiTrack, Desired-OptiTrack and Desired-Robot in Z -axis

Absolute Error in m (Z-axis)	Min	Max	Average
Robot-OptiTrack	0.000000	0.014334	0.005740
Desired-OptiTrack	0.000001	0.029444	0.006958
Desired-Robot	0.000000	0.030315	0.003912

For our scenario, which involves the co-manipulation of an extended object and placing it into tubes, the mean errors in the X, Y, and Z axes are acceptable as the task was completed successfully. The developed system can be deployed in scenarios where a maximum error of 0.03 m is acceptable. It is worth noting that the Optitrack system costs a few thousand dollars while our vision-based system of two cameras is less than a thousand dollars. Therefore, our solution is affordable and low-cost. However, in scenarios that require very fine precision (e.g. assembly of electronics), our system will not be acceptable.

4.4 Discussion: Ethical Considerations and Privacy

Our proposed system received overall positive feedback from the users who evaluated it in the user study. The proposed system can support workers who are required to manipulate heavy and extended objects, e.g. in warehouses or construction, by taking care of the heavy lifting which the workers can provide guidance for the manipulation. The proposed system could improve the working conditions for workers who are required to do the heavy lifting by minimizing musculoskeletal disorders [58].

As our system consists of cameras, privacy concerns are important to be addressed [59, 60]. In our case, our system does not record any videos or images and it does the processing on the fly. Due to this reason, our participants in the user study were not concerned about the privacy

of their data (see Section 4.2). A recent study by Vervier et al. [61] with 125 participants for two working scenarios (a cobot and a chatbot) found that “*the more fun it is to use and the higher the expected performance, the higher the acceptance of technology using personal data*”. Therefore, this demonstrates that there is a connection between the performance of robots and the acceptance of robots having access to personal data.

Another important point that needs to be addressed is the fear that robots will replace human workers [62]. Our system is not designed to replace human workers but to enhance their abilities and improve their workplace. The human is an important component of our system and responsible for guiding the robot through the task. Our system has the potential to be extended to enable teleoperation for applications that are dangerous for humans, for example, the manipulation of hazardous materials. The human operator could be in a safe environment while using hand gestures to control the robot in a hazardous and polluted environment (e.g. manipulation of nuclear waste).

5 Conclusion and Future Work

This thesis presented a novel framework that enabled naive users to co-manipulate objects with a cobot, by using hand gestures. The users were able to control the robot in all dimensions providing a high level of flexibility and not being application specific. The framework took into account the user's position in the workspace in order to move the manipulated object in the desired motion, which enabled the user to freely move within the cobot's workplace. The proposed system was developed in ROS and was implemented into a 7-Degrees of Freedom Cobot equipped with a camera on its end-effector and a wearable camera that was worn by the user.

For the evaluation of the proposed framework, an experimental scenario was developed where the user was required to co-manipulate an extended object by guiding the robot to pick it up from the table and place it into four tubes with different orientations in different locations in the workspace. A study with 16 participants was conducted to evaluate the system's performance and usability. The developed system was compared to the state-of-the-art OptiTrack system and had an average error of 0.01 m which is acceptable for our application. Moreover, the system was evaluated positively by the participants in the study.

The contributions of this Thesis are summarized as follows:

- A novel control method for object co-manipulation, called OSAG, is developed that based on spatially aware information of the operator and their hand gestures can generate object-centric motion.
- The OSAG control method is generic and enables object co-manipulation in all dimensions (rotation and translation in all 3 axes).
- The OSAG control method does not depend on a specific application and it can be imple-

mented on any ROS-enable collaborative robotic arm.

- The developed OSAG control method was evaluated on an experimental scenario with 16 participants and all of them were able to successfully complete the required task without extensive training.

To further evaluate the feasibility and usability of the developed framework, it would be interesting to test it in a study with industrial workers and in different types of co-manipulation tasks (e.g. transportation of heavy objects, manipulation of waste or hazardous materials, etc.). This further testing would provide insights into how the system could be improved for the end-users (i.e. industrial workers).

It would also be interesting to expand the proposed system to a multi-arm robotic setup where a human could co-manipulate objects by controlling multiple robotic arms based on the required tasks. However, this will possibly require additional gestures and re-mapping of gestures and human position estimation to multi-robot motions. Moreover, with the recent advantages of ChatGPT [63], the proposed system could be extended into a multimodal interface that could utilize speech commands as an additional modality for object co-manipulation with multi-robot systems. While gestures could be used for continuous control of the cobots, speed changes or cobot selections could be done through speech commands. This approach may improve the perceived usefulness of the system.

References

- [1] D. Papanagiotou, G. Senter, and S. Manitsaris, “Egocentric gesture recognition using 3d convolutional neural networks for the spatiotemporal adaptation of collaborative robots,” *Frontiers in Neurorobotics*, vol. 15, p. 703545, 2021.
- [2] Logitech, “C615 webcam,” 2023, <https://www.logitech.com/en-us/products/webcams/c615-webcam.960-000733.html> [Accessed: November 17, 2023].
- [3] ZED, “Zed mini camera and sdk overview,” 2023, <https://cdn.stereolabs.com/assets/datasheets/zed-mini-camera-datasheet.pdf> [Accessed: November 17, 2023].
- [4] K. H. Tantawi, A. Sokolov, and O. Tantawi, “Advances in industrial robotics: From industry 3.0 automation to industry 4.0 collaboration,” in *2019 4th Technology Innovation Management and Engineering Science International Conference*. IEEE, 2019, pp. 1–4.
- [5] J. Fryman and B. Matthias, “Safety of industrial robots: From conventional to collaborative applications,” in *ROBOTIK 2012; 7th German Conference on Robotics*. VDE, 2012, pp. 1–5.
- [6] Z. Cséfalvay, “As “robots are moving out of the cages”—toward a geography of robotization,” *Eurasian Geography and Economics*, vol. 64, no. 1, pp. 89–119, 2023.
- [7] A. S. George and A. H. George, “The cobot chronicles: Evaluating the emergence, evolution, and impact of collaborative robots in next-generation manufacturing,” *Partners Universal International Research Journal*, vol. 2, no. 2, pp. 89–116, 2023.
- [8] C. Taesi, F. Aggogeri, and N. Pellegrini, “Cobot applications—recent advances and challenges,” *Robotics*, vol. 12, no. 3, p. 79, 2023.

- [9] A. Giammarino, J. M. Gandarias, P. Balatti, M. Leonori, M. Lorenzini, and A. Ajoudani, "Super-man: Supernumerary robotic bodies for physical assistance in human-robot conjoined actions," *arXiv preprint arXiv:2201.06365*, 2022.
- [10] L. Vianello, W. Gomes, P. Maurice, A. Aubry, and S. Ivaldi, "Cooperation or collaboration? On a human-inspired impedance strategy in a human-robot co-manipulation task," Feb. 2022, working paper or preprint. [Online]. Available: <https://hal.science/hal-03589692>
- [11] T. B. Ionescu and S. Schlund, "Programming cobots by voice: A human-centered, web-based approach," *Procedia CIRP*, vol. 97, pp. 123–129, 2021.
- [12] L. Guo, Z. Lu, and L. Yao, "Human-machine interaction sensing technology based on hand gesture recognition: A review," *IEEE Transactions on Human-Machine Systems*, vol. 51, no. 4, pp. 300–309, 2021.
- [13] S. Fabbriizzi, S. Papadopoulos, E. Ntoutsis, and I. Kompatsiaris, "A survey on bias in visual datasets," *Computer Vision and Image Understanding*, vol. 223, p. 103552, 2022.
- [14] F. Al Farid, N. Hashim, J. Abdullah, M. R. Bhuiyan, W. N. Shahida Mohd Isa, J. Uddin, M. A. Haque, and M. N. Husen, "A structured and methodological review on vision-based hand gesture recognition system," *Journal of Imaging*, vol. 8, no. 6, p. 153, 2022.
- [15] R. Jain, R. K. Karsh, and A. A. Barbhuiya, "Literature review of vision-based dynamic gesture recognition using deep learning techniques," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 22, p. e7159, 2022.
- [16] J. Qi, L. Ma, Z. Cui, and Y. Yu, "Computer vision-based hand gesture recognition for human-robot interaction: a review," *Complex & Intelligent Systems*, pp. 1–26, 2023.

- [17] Q. Gao, J. Liu, Z. Ju, and X. Zhang, "Dual-hand detection for human–robot interaction by a parallel network based on hand detection and body pose estimation," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 12, pp. 9663–9672, 2019.
- [18] T. Zhang, Z. Su, J. Cheng, F. Xue, and S. Liu, "Machine vision-based testing action recognition method for robotic testing of mobile application," *International Journal of Distributed Sensor Networks*, vol. 18, no. 8, p. 15501329221115375, 2022.
- [19] C. Nuzzi, S. Pasinetti, M. Lancini, F. Docchio, and G. Sansoni, "Deep learning-based hand gesture recognition for collaborative robots," *IEEE Instrumentation & Measurement Magazine*, vol. 22, no. 2, pp. 44–51, 2019.
- [20] O. Alonso, "Practical lessons for gathering quality labels at scale," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2015, pp. 1089–1092.
- [21] R. Chen, A. Shek, and C. Liu, "Robust and context-aware real-time collaborative robot handling via dynamic gesture commands," *IEEE Robotics and Automation Letters*, 2023.
- [22] S. Yuanyuan, L. Yunan, F. Xiaolong, M. Kaibin, and M. Qiguang, "Review of dynamic gesture recognition," *Virtual Reality & Intelligent Hardware*, vol. 3, no. 3, pp. 183–206, 2021.
- [23] M. Selvaggio, M. Cognetti, S. Nikolaidis, S. Ivaldi, and B. Siciliano, "Autonomy in physical human-robot interaction: A brief survey," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7989–7996, 2021.
- [24] R. L. Williams, F. W. Harrison, and D. L. Soloway, "Shared control of multiple-manipulator, sensor-based telerobotic systems," in *Proceedings of International Conference on Robotics and Automation*, vol. 2. IEEE, 1997, pp. 962–967.

- [25] M. Mujica, M. Crespo, M. Benoussaad, S. Junco, and J.-Y. Fourquet, "Robust variable admittance control for human-robot co-manipulation of objects with unknown load," *Robotics and Computer-Integrated Manufacturing*, vol. 79, p. 102408, 2023.
- [26] J. Chen and P. I. Ro, "Human intention-oriented variable admittance control with power envelope regulation in physical human-robot interaction," *Mechatronics*, vol. 84, p. 102802, 2022.
- [27] L. Peternel, N. Tsagarakis, D. Caldwell, and A. Ajoudani, "Robot adaptation to human physical fatigue in human-robot co-manipulation," *Autonomous Robots*, vol. 42, pp. 1011–1021, 2018.
- [28] L. van der Spaa, M. Gienger, T. Bates, and J. Kober, "Predicting and optimizing ergonomics in physical human-robot cooperation tasks," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1799–1805.
- [29] S. Hignett and L. McAtamney, "Rapid entire body assessment (reba)," *Applied ergonomics*, vol. 31, no. 2, pp. 201–205, 2000.
- [30] E. Mielke, E. Townsend, D. Wingate, and M. D. Killpack, "Human-robot co-manipulation of extended objects: Data-driven models and control from analysis of human-human dyads," *arXiv preprint arXiv:2001.00991*, 2020.
- [31] E. Mielke, E. Townsend, D. Wingate, J. L. Salmon, and M. D. Killpack, "Human-robot planar co-manipulation of extended objects: data-driven models and control from human-human dyads," *Frontiers in Neurorobotics*, vol. 18, 2024.

- [32] Z. Al-Saadi, Y. M. Hamad, Y. Aydin, A. Kucukyilmaz, and C. Basdogan, "Resolving conflicts during human-robot co-manipulation," in *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*, 2023, pp. 243–251.
- [33] J. E. Solanes, L. Gracia, P. Munoz-Benavent, J. V. Miro, M. G. Carmichael, and J. Tornero, "Human–robot collaboration for safe object transportation using force feedback," *Robotics and Autonomous Systems*, vol. 107, pp. 196–208, 2018.
- [34] M. Ambauen, "Robot-assisted manipulation: Utilizing communication of user intent through tactile sensing and spatial hand tracking," *Santa Clara University, Mechanical Engineering Master's Thesis*, 2011. [Online]. Available: https://scholarcommons.scu.edu/cgi/viewcontent.cgi?article=1034&context=mech_mstr
- [35] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.
- [36] J. P. C. de Souza, L. F. Rocha, V. M. Filipe, J. Boaventura-Cunha, and A. P. Moreira, "Low-cost and reduced-size 3d-cameras metrological evaluation applied to industrial robotic welding operations," in *2021 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2021, pp. 123–129.
- [37] K. Hutchinson, M. S. Yasar, H. Bhatia, and H. Alemzadeh, "A reactive autonomous camera system for the RAVEN II surgical robot," in *2020 International Symposium on Medical Robotics (ISMR)*. IEEE, 2020, pp. 195–201.

- [38] C. Wang, X. Chen, Z. Yu, Y. Dong, R. Zhang, and Q. Huang, "Intuitive and versatile full-body teleoperation of a humanoid robot," in *2021 IEEE International Conference on Advanced Robotics and Its Social Impacts (ARSO)*. IEEE, 2021, pp. 176–181.
- [39] FrankaEmika, "Panda datasheet," 2023, https://www.wiredworkers.io/wp-content/uploads/2019/12/Panda_FrankaEmika_ENG.pdf [Accessed: November 17, 2023].
- [40] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. Yong, J. Lee *et al.*, "MediaPipe: A framework for perceiving and processing reality," in *Third workshop on computer vision for AR/VR at IEEE computer vision and pattern recognition (CVPR)*, vol. 2019, 2019.
- [41] MediaPipe, "Hand landmarks detection guide," 2023, https://developers.google.com/mediapipe/solutions/vision/hand_landmarker [Accessed: November 18, 2023].
- [42] MediaPipe, "Model card - MediaPipe hands (lite/full)," 2023, [https://storage.googleapis.com/mediapipe-assets/Model%20Card%20Hand%20Tracking%20\(Lite_Full\)%20with%20Fairness%20Oct%202021.pdf](https://storage.googleapis.com/mediapipe-assets/Model%20Card%20Hand%20Tracking%20(Lite_Full)%20with%20Fairness%20Oct%202021.pdf) [Accessed: November 18, 2023].
- [43] StereoLabs, "3D object detection overview," 2023, <https://www.stereolabs.com/docs/object-detection/> [Accessed: November 20, 2023].
- [44] A. Addison, "How to convert a quaternion into euler angles in Python," 2023, <https://automaticaddison.com/how-to-convert-a-quaternion-into-euler-angles-in-python/> [Accessed: November 30, 2023].
- [45] M. Mayr and J. M. Salt-Ducaju, "A C++ implementation of a cartesian impedance controller for robotic manipulators," *arXiv preprint arXiv:2212.11215*, 2022.

- [46] O. Khatib and B. Siciliano, *Springer Handbook of Robotics*. Springer International Publishing, 2016.
- [47] N. Hogan, "Impedance control: An approach to manipulation," *J. Dynamic Systems, Measurement, and Control*, pp. 107–1, 1985.
- [48] C. Ott, *Cartesian impedance control of redundant and flexible-joint robots*. Springer, 2008.
- [49] O. Khatib, "Inertial properties in robotic manipulation: An object-level framework," *The International Journal of Robotics Research*, vol. 14, no. 1, pp. 19–36, 1995.
- [50] A. Ben-Israel and T. N. Greville, *Generalized inverses: theory and applications*. Springer Science & Business Media, 2003, vol. 15.
- [51] J. Brooke, "Sus: a 'quick and dirty' usability," *Usability evaluation in industry*, vol. 189, no. 3, pp. 189–194, 1996.
- [52] S. G. Hart, "NASA-task load index (NASA-TLX); 20 years later," in *Proceedings of the human factors and ergonomics society annual meeting*, vol. 50, no. 9. Sage publications Sage CA: Los Angeles, CA, 2006, pp. 904–908.
- [53] B. Klug, "An overview of the system usability scale in library website and system usability testing," *Weave: Journal of Library User Experience*, vol. 1, no. 6, 2017.
- [54] J. Schmidtler, K. Bengler, F. Dimeas, and A. Campeau-Lecours, "A questionnaire for the evaluation of physical assistive devices (quead): Testing usability and acceptance in physical human-robot interaction," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2017, pp. 876–881.

- [55] K. Kodur, M. Zand, M. Tognotti, C. Jauregui, and M. Kyrarini, "Structured and unstructured speech2action frameworks for human-robot collaboration: A user study," *Authorea Preprints*, 2023.
- [56] G. Charalambous, S. Fletcher, and P. Webb, "The development of a scale to evaluate trust in industrial human-robot collaboration," *International Journal of Social Robotics*, vol. 8, pp. 193–209, 2016.
- [57] NaturalPoint, "Optitrack - motion capture system," 2024, <https://optitrack.com/> [Accessed: January 3, 2024].
- [58] A. Mocan, A. Gaureanu, G. Szabó, and B. Mrugalska, "Arguments for emerging technologies applications to improve manufacturing warehouse ergonomics," *Sustainability and Innovation in Manufacturing Enterprises: Indicators, Models and Assessment for Industry 5.0*, pp. 115–164, 2022.
- [59] A. Chatzimichali, R. Harrison, and D. Chrysostomou, "Toward privacy-sensitive human–robot interaction: Privacy terms and human–data interaction in the personal robot era," *Paladyn, Journal of Behavioral Robotics*, vol. 12, no. 1, pp. 160–174, 2020.
- [60] K. A. Demir, G. Döven, and B. Sezen, "Industry 5.0 and human-robot co-working," *Procedia computer science*, vol. 158, pp. 688–695, 2019.
- [61] L. Vervier, P. Brauner, and M. Ziefle, "Perception of privacy and willingness to share personal data in the smart factory," in *International Conference on Human-Computer Interaction*. Springer, 2023, pp. 213–231.

- [62] P. K. McClure, ““You’re fired,” says the robot: The rise of automation in the workplace, technophobes, and fears of unemployment,” *Social Science Computer Review*, vol. 36, no. 2, pp. 139–156, 2018.
- [63] S. Vemprala, R. Bonatti, A. Bucker, and A. Kapoor, “ChatGPT for robotics: Design principles and model abilities,” *Microsoft Auton. Syst. Robot. Res*, vol. 2, p. 20, 2023.