INTERACTIVE REMOTE ROBOTIC ARM CONTROL WITH HAND MOTIONS

By

JORDAN RALPH FOGG

Bachelor of Science in Electrical Engineering Technology
Oklahoma State University
Stillwater, Oklahoma
2021

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 2023

INTERACTIVE REMOTE ROBOTIC ARM
CONTROL WITH HAND
MOTIONS

Thesis Approved:

Dr. Huaxia Wang

Thesis Advisor

Dr. Chulho Yang

Dr. Amanda De Oliveira Barros

ACKNOWLEDGMENTS

I would first like to thank my advisor Dr. Huaxia Wang for giving me the opportunity to learn under him and for believing in me as a student. Allowing me to grow as an engineer and researcher over the last few years.

A thank you goes out to my coauthors of this research: Dr. Huaxia Wang, Dr. Chen Wang, Zeyu Deng, Long Huang, and especially Emory Meursing it is because of you guys that this research is where it is today.

I would also like to thank my fiancee Nicole for being my second half through college and especially through graduate school. Thank you for staying with me through the long hours and late nights, your love and support meant the world to me. I love you and I look forward to saying "I do" in July.

Finally, Mom and Dad, there is nothing I can say here that will show my gratitude I have for the love and support you have given me as I pursued my dreams through college. Allowing me to stay focused on my studies without ever worrying about finances. I love you guys and thank you for everything.

Acknowledgments reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

Name: JORDAN RALPH FOGG

Date of Degree: MAY 2023

Title of Study: INTERACTIVE REMOTE ROBOTIC ARM CONTROL WITH HAND MOTIONS

Major Field: ENGINEERING TECHNOLOGY

Abstract: Geographically-separated people are now connected by smart devices and networks to enjoy remote human interactions. However, current online interactions are still confined to a virtual space. Extending pure virtual interactions to the physical world requires multidisciplinary research efforts, including sensing, robot control, networking, and kinematics mapping. This paper introduces a remote motion-controlled robotic arm framework by integrating these techniques, which allows a user to control a far-end robotic arm simply by hand motions. In the meanwhile, the robotic arm follows the user's hand to perform tasks and sends back its live states to the user to form a control loop. Furthermore, we explore using cheap robotic arms and off-the-shelf motion capture devices to facilitate the widespread use of the platform in people's daily life. we implement a test bed that connects two US states for the remote control study. We investigate the different latency components that affect the user's remote control experience, conduct a comparative study between the remote control and local control, and evaluate the platform with both free-form in-air hand gestures and hand movements following reference curves. We also are investigating the possibility of using VR (Virtual Reality) headsets to enhance first-person vision presence and control allowing for smoother robot teleoperation. Finally, a user study is conducted to find out user satisfaction with different setups while completing a set of tasks to achieve an intuitive and easy-to-use platform.

TABLE OF CONTENTS

Chapter Page

I. INTRODUCTION . . . 1

1.1 System Overview . . . 1

1.2 Contributions . . . 2

1.3 Outline . . . 2

II. BACKGROUND . . . 4

2.1 Robotic Design . . . 4

2.2 Robotic Operating System . . . 7

2.3 Leap Motion Controller . . . 9

2.4 Meta Quest 2 VR headset . . . 10

2.5 Ricoh THETA 360 degree camera . . . 10

III. RELATED WORKS . . . 12

3.0.1 Teleoperation . . . 13

3.0.2 User Experience In Human-Robot Interaction . . . 14

IV. FIRST STUDY . . . 15

4.0.1 Introduction . . . 15

4.0.2 Hardware Integration . . . 15

4.0.3 Software Components . . . 16

4.1 Approach Design . . . 17

4.1.1 Gathering and Converting Motion Data . . . 17

4.1.2 Secure Wide Area Network Connection . . . 18

Chapter Page

4.1.3 Receiving Data and Interfacing with Robot 19

4.2 User Study with Low-cost Robotic Arm Platform 20

4.2.1 Control Delay Study 20

4.2.2 Local Control vs. Robot Replay 22

4.2.3 Remote vs. Local Control 23

4.2.4 Free-form vs. Reference Control 24

V. STUDY TWO 28

5.0.1 Introduction 28

5.0.2 Platform Overview 28

5.0.3 Hardware Integration 30

5.0.4 Software Components 30

5.0.5 Gathering and Converting Motion Data 32

5.0.6 Secure Local Area Network Connection 33

5.0.7 Receiving Data and Interfacing with Robot 33

5.1 Study Design 35

5.1.1 Control Latency 36

5.1.2 Motion Accuracy 36

5.1.3 Task Achievement 37

5.1.4 User Satisfaction 38

5.2 Performance Evaluation 39

5.2.1 Latency 39

5.2.2 Line Tracing Time and Similarity 40

5.2.3 Object Collection and Placement 41

5.2.4 User's Rating 42

VI. CURRENT WORK 47

Chapter Page

6.0.1 Live Video Feed Using VR Headset 47

6.0.2 Camera Movement Using VR Headset 47

6.0.3 Scaling Up the Robot 48

VII. CONCLUSIONS 49

7.0.1 Conclusion 49

REFERENCES 50

LIST OF TABLES

Table Page

1. Average transmission latency of two video streaming setups. 40

LIST OF FIGURES

Figure Page

1. The remote-controlled robotic arm platform 16
2. Rotation from LEAP Motion frame to PX-150 frame 19
3. Average time to execute one iteration of the program. 21
4. Average time taken to transfer data over the internet. 22
5. Execution time of different connection configurations. 23
6. Trajectories of drawing a "S" in different control scenarios. 24
7. DTW distances of the trajectories in different control scenarios. 25
8. Latency on the x-axis between the robot and human in remote control. . . . 26
9. Latency on y-axis between the robot and human in remote control. 26
10. Latency on z-axis between the robot and human in remote control. 27
11. Comparing the similarity between different control scenarios and within the same scenario. 27
12. Illustration of VR-enabled robotic arm control platform. 29
13. Camera viewing angles. 32
14. Example of curve line tracing. 34
15. Example zigzag line tracing. 35
16. Object collection and placement setup. 36
17. Average latency of each control loop. 37
18. Speed of task completion when the camera angle is 45°. 38
19. Speed of task completion when the camera angle is 90°. 39
20. Speed of task completion when the camera angle is 135°. 40
21. Control accuracy when the camera angle is 45°. 41
22. Control accuracy when the camera angle is 90°. 42
23. Control accuracy when the camera angle is 135°. 43
24. object collection time. 44
25. object collection. 44
26. Satisfaction of Latency. 45
27. Similarity of Trajectory. 45
28. Satisfaction of Task Achievement. 46

CHAPTER I

INTRODUCTION

1.1 System Overview

When this research started, the world was recovering from a pandemic that saw millions of people stuck in their homes and separated from loved ones. There was a huge shift in how people lived their daily life including how they interacted with other people around them. Companies like Zoom and Microsoft Teams saw users skyrocket as companies found ways to communicate with employees virtually. While there are lots of ways to virtually interact with people around the world there are very few that allow users to physically interact with that environment while being virtually connected.

Whether you are trying to interact with a family member during COVID-19 or trying to move hazardous material while keeping a safe distance, the need for telepresence is growing and this research hopes to fill a small portion of that growing field by designing and testing a low-cost and low-latency teleoperated robotic platform. This platform uses a hand tracking controller to capture the hand movements of a user and send it over a wide area network (WAN) where it is received and converted into data used to update the position of a robotic arm. While this occurring the user is receiving live images of the robot's environment through a video stream. This system allows for real-time robotic arm control using an intuitive and easy-to-use controller over long distances.

1.2 Contributions

Our contributions are summarized as the following:

• We developed a robotic arm framework using a low-cost robotic arm and the off-the-shelf motion capture device to enable remote real-time control over a Wide Area Network (WAN). It shows the potential to provide physically-augmented services to users with low cost and nonprofessional knowledge.

• A test-bed bridging two US states is built for estimating the availability of the remote motion-controlled robot.

• We examine different types of latency that impact the user experience and task performance. Moreover, we conduct two comparison studies, local vs. remote and reference-guided motions vs. free-form movements.

• Experiments with five participants show that the cheap robotic arm and vision-based motion sensor are available to provide physically-augmented services to users.

1.3 Outline

This is the layout of the rest of this paper:

Chapter 2: Is a thorough look into the background of the components that make up the system as well as components of current and future research.

Chapter 3: Is a review of the current literature and how their contribution is used to forward this area of research.

Chapter 4: This chapter talks about the first platform we tested using Leap Motion and Zoom to control the robotic arm from different distances from local control to over 600 miles away.

Chapter 5: In this chapter we look into using a VR headset paired with a 360-degree camera and the leap motion to test if having a larger field of view (FOV) can increase users' accuracy and happiness with the system.

Chapter 6: The focus of this chapter is to look at the current and future possibilities of the research and the direction for future researchers to continue advancing this area.

Conclusion: This chapter gives the conclusion of the research as well as a summary of what was covered in the paper

CHAPTER II

BACKGROUND

2.1 Robotic Design

There are two main robotic arms we used when completing this research which are the PincherX 150 Robot Arm (PX-150) and the Sawyer robotic arm. The first one is the PincherX 150 Robot Arm from Interbotix and sold by Trossen Robotics. It is a low-cost robot roughly $1,000.00 compared to larger industrial robots that can cost north of $20,000.00 and is easy to set up. The PX-150 is a five degree of freedom robot arm that has a reach of 450 mm and a span of 900 mm.

The arm is made up of eight DYNAMIXEL XL430-W250 servos that allow for a payload of 50g and movement accuracy of eight millimeters. There are five joints that make up the reach of the robotic arm starting with the waist that is moved by a single servo with a range of -180 to 180 degrees. The next is the shoulder joint which uses two XL430-W250 servos in tandem and both have a range of -113 to 111 degrees. The elbow joint is similar to the shoulder with a difference in range. The range of the elbow is -120 to 95 degrees. The wrist is separated into two different joints the first being the wrist angle and the second being the wrist rotation both controlled by separate servos. The wrist can pitch between -100 degrees and 123 degrees. While the rotation portion can rotate -180 to 180 degrees. The final joint of the robot is the gripper with a range of 30 mm to 74 mm being the opening of the gripper from the center of each carriage. The links and frame of the robot are constructed out of extruded aluminum to give it structure and help it withstand wear and tear. Both the base and the gripper of the robot are 3d printed allowing for custom-printed parts to be used based on application and need. In this research, the robot was left in the same configuration as it arrived at the factory aside from repairs and preventive maintenance. The recommended workspace of the PX-150 is 70% of the span of the robot as mentioned above the span is 900 mm so the recommended workspace for the robot is 630 mm.

The DYNAMIXEL XL430-W250 servos used on the robot are smart servos with a resolution of 4096 pulse/rev allowing very precise movement. The servos have a 258.5:1 gear ratio and weigh roughly 57.2g. They are powered through a power hub board and have a range of 6.5V to 12V and are daisy-chained together back to the U2D2 controller that converts USB to TTL allowing the robot to be controlled from a computer. Each servo provides a range of feedback information including position, velocity, load, input voltage, and current.

The kinematics of this robot follow the product of exponentials (POE) method which requires a reference frame as well as the position of the base frame and end effector at its zero point. That's why this position is often called the zero configuration of the robot. It is common and standard to get a frame for each n joints that a robot has, so in the case of the PX-150 $n = 5$. A screw axis for each joint based on its zero position must be found as well as the current joint variable angle from n_1 to n. this is known as the M matrix and S matrix of the robot. The M and S matrix for the PX-150 is

$$M = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.358575 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.25457 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{2.1.1}$$

$$S = \begin{bmatrix} 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & -0.10457 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & -0.25457 & 0.0 & 0.05 \\ 0.0 & 1.0 & 0.0 & -0.25457 & 0.0 & 0.2 \\ 1.0 & 0.0 & 0.0 & 0.0 & 0.25457 & 0.0 \end{bmatrix}^T \tag{2.1.2}$$

These matrices represent the forward kinematics of the robot in its home position so that when the robot's end effector is changed by some angle θ it can be represented as

$$T(\theta) = e^{[S_1]\theta_1}....e^{[S_n]\theta_n}M \tag{2.1.3}$$

Given 2.1.3 the inverse kinematics of the robot can be found by finding the angles of each joint that satisfies the equation $x = f(\theta)$ when x_d is the updated end effector position being moved too. The PX-150 uses Newton-Raphson iterative algorithm to find θ_d given the equation

$$g(\theta_d) = x_d - f(\theta_d) = 0 \tag{2.1.4}$$

allowing some error e in the solution allows for multiple approximated solutions to be found as $g(\theta_d) \to 0$ with the system choosing the closest solution to the root. If no solution can be found within the allowed error then that means that the convergence failed and the robot will not move. It should be noted that the PX-150 uses X, Y, and Z and the right-hand convention to represent its position relative to its base frame in millimeters. The orientation of the robot is represented in terms of Euler angles i.e roll, pitch, and yaw.

The PX-150 can be controlled from a computer by using its ROS API in Python. Interbotix built what is called IRROS structure to allow researchers to interact with the robot at a research-level while letting IRROS handle the lower-level drivers and configurations. IRROS is made up of 5 layers that slowly descend from the research layer down to the hardware layer. The first layer is the research layer, this layer allows users to move the robot by simply sliding a bar on a graphical user interface without needing to know anything else about how a robot moves. The second layer is the application layer made up of the inverse kinematics solver, ROS control packages, and the Move-it interface packages. The application layer is the layer used most commonly in this research allowing the integration of the PX-150 into the system without needing to reinvent or rebuild the robot's control packages but allowing for custom commands and packages that are needed to complete the research. The middle layer is called the Control layer and houses the configuration packages for each servo as well as parameters needed by ROS to simulate and control the physical robot. This layer is also home to the PX-150's unified robotics description format (URDF) file allowing ROS to show the model of the robot in applications such as Gazebo and RVIZ. The fourth layer of IRROS is the driver layer and is made up of the computer drivers needed to convert the incoming data from the USB cable into usable data for ROS. Finally, the hardware layer is made up of the physical hardware such as the U2D2, the servos, and any other sensors that might be connected to the robot.

2.2 Robotic Operating System

The Robotic Operating System (ROS) is a collection of libraries, applications, and tools that allow users to quickly develop robotic applications without having to start at the lowest levels of communications. ROS framework revolves around the ability to build pipelines between sensors, actuators, and control systems to efficiently and effectively move data and commands between hardware and software applications. This saves a lot of time by allowing the use of different products from different companies to interact without having to make sure they are compatible with each other. Sensors, robots, and cameras like the ones that will be covered later in this paper have a software interface that can now seamlessly be integrated together. ROS uses a system of nodes connected together and sharing messages through channels called topics. Nodes are any connection point that generates or receives data and would be how the sensors and hardware connect to the ROS system. Topics are the pipelines of ROS world as they transport data along them to different nodes connecting them in a web similar to a network of computers. The data that flows on the topics are called messages, these messages have unique structures based on the type of message being used.

There are a few ways a program can interact with the web of nodes through Python and that is by building one of the following: Publish and Subscriber, Services, or Action. The first one is Publish this is exactly as it sounds meaning it publishes messages about a topic. This is useful when you have a sensor like a thermometer that is reading temperature because it can publish the updated temperature to the topic periodically. The reverse of a publisher is a subscriber, this is how a program can pull the information from a topic to use. An example of this would be a mobile robot that has a distance sensor, a Python program could subscribe to the topic of the sensor, and when the distance falls below a set value then it would stop the robot. The next structure for moving data is a service and instead of publishing data to a topic continuously a service waits till it receives a call. Services have a client and a server relationship where the client request information and waits for a response from the server. Servers can have multiple clients connected at once but only one server can be connected at a time. The final one is an Action, in which is similar in service in that it waits for a request but is usually reserved for longer tasks. During the duration of the action, the client is receiving feedback from the server on the status of the action. Once the action is complete it will receive a response with the final data requested a the start.

ROS is not just for moving physical robots, it comes with an array of tools and applications that allow a user to test, train and gather data on simulated robots as well as digital twins of robots. One of these applications is called RVIZ which allows for a real-time simulated model of the actual robot including sensors and video feeds. This is a useful way of visualizing what the robot sees and is doing during testing. RVIZ subscribes to the topics of the robot to receive continuous updates about its position and current state and displays it in an easy-to-use interface. A similar but slightly different application that comes with ROS is Gazebo. Gazebo is a full-blown physics simulator that allows users to simulate what a virtual robot would do under different conditions without risking damage or destruction of the physical robot. Gazebo allows users to adjust world conditions like friction, wind, and gravity while also changing the scenery. If a user was testing a self-driving car algorithm then it would allow them to see how the car handles on ice or what the system does when faced with an obstacle in its path without endangering a physical car or environment.

2.3 Leap Motion Controller

The Leap Motion controller is a hand-tracking device that uses two optical sensors and infrared light to track the position of a user's hands in 3D space. The Leap Motion captures a frame for each sensor and triangulates the user's hand based on the reflection angle of the IR allowing the sensor to send back very accurate positional and rotation data about a user's hand. The coordinate system for the Leap Motion is a right-handed system with a positive Y-axis going in the up direction and a positive X-axis being toward the left side. Finally, the positive Z is in the forward position. The origin of the coordinate system is in the center of the device and positional data is measured in millimeters from the origin. The Leap Motion has a 150 degrees field of view and a sensing range from the origin to around 600 millimeters. The rotational data is measured in radians and uses Euler angles similar to the PX-150 robot. The Leap Motion has a built-in hand model that overlays the sensor data to send back data on specific parts of a hand. An example of this would be the distal phalanges, also known as the fingertips, of a human hand, which can be tracked for all five fingers of a user's hand sending back an X, Y, and Z and orientation for the fingertip. The area that is used for this research is the center of a user's hand also called the palm of the hand. The Leap Motion can run at 120 frames a sec but after triangulation outputs data at half that speed giving 60fps of usable hand tracking data. The sensor also has built-in gesture and tool detection such as circle, swipe, and screen tap as well as using a stylus or pen to point at a screen without touching the screen. These gesture and tool detection are useful in different applications but these will not be used for this research. The sensor interfaces with a computer through the API in Python allowing for easy integration with other platforms such as ROS.

2.4 Meta Quest 2 VR headset

The Meta Quest 2 headset is a state-of-the-art virtual reality (VR) headset developed by Meta that gives users 1832 X 1920 resolution in each eye and up to 90 Hz refresh rate for a clear and immersive feeling while they wear the headset. Powered by a Qualcomm Snapdragon XR2 CPU and 6 GB of RAM, the headset comes with 128 GB of storage allowing for a wide use of VR applications. The headset also has the option to track a user's hand position using 2 different methods. The first is using handheld controllers that use integrated sensors to feedback position and orientation of the headset. This is a commonly used method and is the way most VR headsets track hand position on the market today. The Quest 2 also supports controller-free hand tracking by using sensors similar to the Leap Motion but is integrated into the front of the VR headset. This allows users to ditch the controllers for a more natural feel when interacting with menus, games, and other applications on Quest 2. To program and interact with data from the Quest 2, the most common way is by Unity to update the display in the headset and pull data from onboard sensors. Unity uses a left-handed coordinate system to represent the incoming positional data as well as quaternions for orientation. Unity is programmed in C# and allows for a connection between Quest 2 and other scripts and programs.

2.5 Ricoh THETA 360 degree camera

The Ricoh Theta Z is a 360-degree camera that takes two fish-eyed 180-degree FOV cameras and stitches them together to form the full image. The camera has the capability to produce 4k 360-degree still photos as well as 2k video streams of 360-degree video. The camera weighs in at around 182g and comes with a four-channel microphone. It interfaces with the computer via a USB-C cable and allows video to be streamed directly from the camera to the computer. This gives us the ability to bring that video in and display it for users whether it is in a viewer or streamed to the user's VR headset. If it is to the headset then the user can turn their head and view the full 360-degree environment around the camera.

CHAPTER III

RELATED WORKS

Remote real-time motion tracking has found a growing interest in robotics for robot imitation control [35, 21, 31, 22, 2]. Motion re-targeting is the essential and challenging part of real-time robot control from human observations since humans and robots are dissimilar in size, degrees-of-freedom (DOF), and dynamics and mechanical limitations [19]. However, prior research on teleoperation systems used very expensive robotic arms (e.g., $\$20,000$), which implies overwhelming high costs of development and makes it unaffordable for home use. Inertial measurement units (IMUs) can be used for tracking human motions [23, 30, 25]. However, due to the estimate's drift, magnetic disturbances, and calibration issues, IMU-based human motion tracking always suffers from low tracking accuracy [9]. Moreover, these methods require multiple IMUs to be attached to different parts of the user's limbs (e.g., upper arm, forearm, etc.) to measure their altitudes and orientations, which is burdensome and obtrusive.

Owing to the wide deployment and ease of use, cameras have been well deployed for human motion tracking. Hwang *et al.* proposed MonoEye [14], a human motion tracking system using a single RGB camera mounted on the user's chest. A Microsoft Kinect depth camera can be used for device-free human motion tracking [32]. Bilesan *et al.* [4] developed a marker-based motion tracking system using the Kinect v2 sensor to capture joint angles of a human-like ankle flexion/extension motion. OpenPose was leveraged to implement a 3D markerless and device-free motion tracking technique [20, 28]. OptiTrack with multiple cameras was used to track human hand motions with low tracking errors (e.g., less than $0.20mm$) [12]. However, the above methods either suffer from low tracking accuracy or require the installation of expensive dedicated hardware. Instead, in this work, we propose to leverage the low-cost Leap Motion Controller for capturing a user's hand motion [3, 8], which can also achieve a relatively high tracking accuracy.

3.0.1 Teleoperation

Telerobotics has a wide range of applications such as being used to work in harsh environments (e.g., in space, underwater, etc.) where humans can hardly reach, used on the field for military purposes, used to conduct medical surgeries, and used for remote education. Telerobotics involves two major subfields, which are teleoperation and telepresence. Teleoperation refers to the operation of a device at a distance, while telepresence means the operator feels present in the remote environment via manipulating the remote robot while getting feedback from the immersive interface [34]. Most teleoperation studies consider haptic feedback and/or visual feedback. For example, haptic feedback has been used to facilitate teleoperation systems for micromanipulation [5] and control of unmanned aerial vehicles [16]. Cameras and monitors were leveraged to provide monovision feedback during teleoperation [10, 13]. However, the human perception of distance is still constrained due to the asymmetry between the mechanical structures of master and slave robots and the lack of stereoscopic vision [29].

To improve human awareness of real environments, virtual reality (VR) has been leveraged to facilitate teleoperation interfaces. Hetrick *et al.* [11] compared different VR robot teleoperation interfaces for performing dexterous manipulation tasks with a Baxter robot. Brizzi *et al.* [6] proposed an interface built on augmented reality (AR) to improve pick-and-place tasks' performance. Sun *et al.* [29] further developed a mixed reality (MR)-based teleoperation system, which combines real and virtual worlds. However, the above-mentioned methods all use 2D cameras as the source of the visual feedback, which still lacks depth information. Instead, this work proposes to use the 360° camera to better capture the remote environment at the robot end and stream it to the user end, which is then displayed in the

VR device.

3.0.2 User Experience In Human-Robot Interaction

User experience is a key factor of human-robot interaction and has been studied by many works. Lindblom *et al.* [17] investigated how to properly conduct user experience evaluation in human-robot interaction by answering several related methodological questions. Similarly, a user experience design cycle was proposed to include human factors in the design of the human-robot interface [24]. Alenljung *et al.* [1] explored the role and relevance of user experience related to socially interactive robots and identify the challenges to be addressed. Buchner *et al.* [7] further studied how user experience in human-robot interaction changes over time by asking the participants to fill out the questionnaire at three defined points in time. Different from most user experience studies, Jokinen *et al.* [15] explored using multimodal behavior such as gazing, facial expressions, and body posture to predict the user experience during human-robot interaction, rather than using questionnaires.

CHAPTER IV

FIRST STUDY

4.0.1 Introduction

The first study that was conducted was to verify that it was possible to have a platform that would allow for real-time control of a robotic manipulator using a hand tracking sensor when the controller and the robotic manipulator were separated by a long distance and on separate networks. To test this a system of sensors, computers, cameras, and a robotic manipulator was constructed and data was gathered across 4 different configurations to compare the latency between each system as well as the accuracy of the robotic movements vs the user's hand motion as distance between the controller and robotic manipulator grew.

4.0.2 Hardware Integration

This research uses three main pieces of hardware: a PC installed with Linux Ubuntu 18.04 [27], a PincherX 150 Robot Arm from Trossen Robotics, and a LEAP Motion Controller from Ultraleap. The LEAP Motion Controller is a motion tracking camera that uses three LEDs and two infrared cameras in order to triangulate an accurate value of a hand above the sensor at around 120 frames per second. The PincherX 150 Robot Arm, known further in this paper as the PX-150, is a robotic arm with 5 degrees of freedom (DOF) and a 900mm wingspan. When fully extended, it can reach up to 450mm in length and can pick up payloads of up to 50g. The PX-150's base and end effector are 3D printed and the servos used on the arm itself are DYNAMIXEL X-Series smart servos which are connected together through a DYNAMIXEL U2D2 controller. This controller connects to a computer through a USB

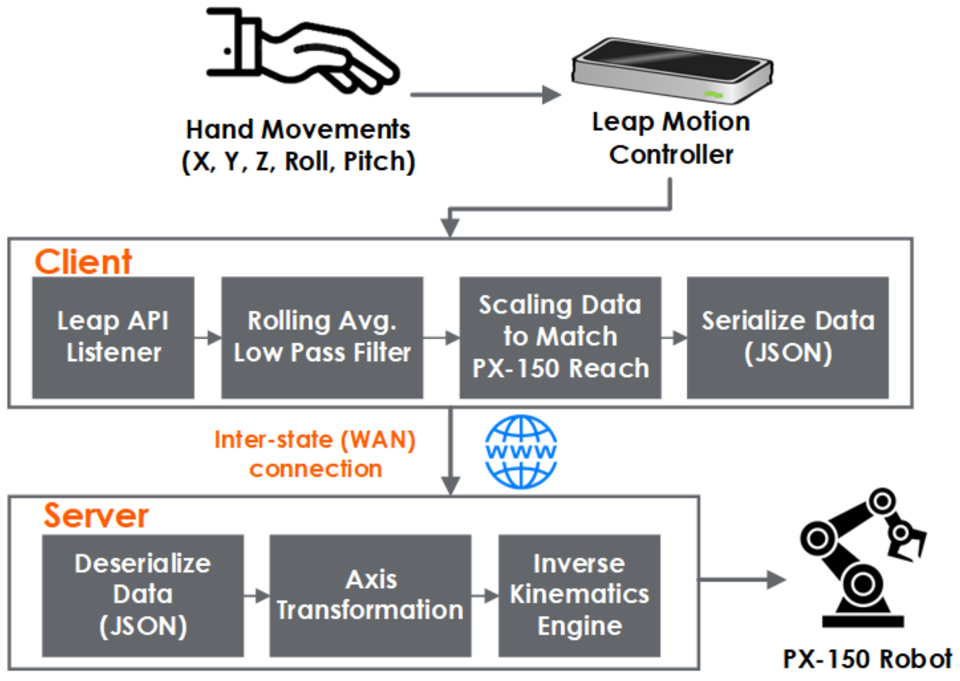
Hand Movements (X, Y, Z, Roll, Pitch)
Leap Motion Controller
Client
Leap API Listener
Rolling Avg. Low Pass Filter
Scaling Data to Match PX-150 Reach
Serialize Data (JSON)
Inter-state (WAN) connection
www
Server
Deserialize Data (JSON)
Axis Transformation
Inverse Kinematics Engine
PX-150 Robot

Figure 1: The remote-controlled robotic arm platform

serial connection and allows it to utilize the pre-built PX-150 ROS packages for Melodic. The kinematics engine used to move the robot is included in the software package of the PX-150. The PX-150 was selected for this research for its affordability (with price $945.95), repairability, and end-effector customization by 3D printing.

4.0.3 Software Components

Vision-based Hand Motion Tracker. The developing company for the LEAP Motion Controller, Ultraleap, provides an API that allows developers to utilize the functions of the LEAP Motion Controller to access data from the sensor. A few functions that are utilized in this research include the real-time position and orientation tracking of the palm of the hand as well as the accessing of previous frames captured by the sensor.

ROS-based Human-Robot Interface. The interface used to communicate with the PX-150 is a ROS wrapper that was built by Interbotix. Starting from the research layer (main code) there are three main layers that connect the application to the physical robot. These three layers are the application support layer, the control layer, and the driver layer. Starting with the application support layer, this layer houses the main ROS packages such as the inverse kinematics engine module, the ROS control package, and the MoveIt interface package. These packages are how the user's inputs are able to interface with the lower-level ROS packages. The control layer contains the necessary configuration files of the robot such as the robot's joint names, default joint values, as well as all the other robot parameters. The final layer between the application and the hardware layer is the driver layer, this layer holds the interbotix-ros-core package that directly communicates with the PX-150's onboard controller (U2D2). This completes the transfer of data from the application to the physical robot.

4.1 Approach Design

4.1.1 Gathering and Converting Motion Data

By accessing the LEAP Motion API, the code starts a listener that takes in tracking data from the sensor. After the listener has started, the program waits until a hand is in view above the controller to then start reading position values. When a hand is in view, the program then takes the current frame and the previous two frames, then divide them by the total number of frames in use, in this case, to create a rolling average of the three positional values of the palm center in order to get an accurate position value for the robotic arm to use. The code takes this rolling average in order to create a low-pass filter to reduce the extra noise caused by hands prone to twitching or jerky movements and increase the accuracy of the movements when sent to the PX-150. With this rolling average of position values, we then scale them into values that the robotic arm can recognize. With the PX-150, it has a workspace that is "a specification of the configurations that the end-effector of the robot can reach" [18], meaning that the robot has a space where the arm can create a valid configuration. In order for the PX-150 to configure into a correct position, we must scale the position values from the LEAP Motion Controller into positional values that can be accepted by the PX-150. To do this, we take the positions of the LEAP Motion Controller on the x, y, and z axes and convert them into a corresponding $\hat{x}$, $\hat{y}$, and $\hat{z}$ position that is valid for the PX-150. This scaling is necessary in order to make sure that when the hand is at the farthest sensor position above the LEAP Motion Controller, the PX-150 will be fully extended to a corresponding position. In addition to scaling the position data, the LEAP Motion Controller also looks for the distance between the forefinger and thumb to determine if the hand is in a fist position. We take this value and call it the grab strength, as it will be utilized by the PX-150 to open and close the end-effector. After the code has finished processing all of the necessary data, it then enters the position values, the roll value of the hand, and the grab strength value into a list which is then serialized into a JSON string to be sent over the web socket.

4.1.2 Secure Wide Area Network Connection

In order to get the values from the LEAP Motion Controller to the PX-150, our code establishes a secure client-server relationship between two computers using a web socket. When creating a web socket connection in different physical locations, a person must have two computers that have access to the same IP address, internet port, and header number to establish a proper connection. When the server is started, it will start listening on the pre-selected port number with the server computer's IP address. When the client is started, it will look on the same pre-selected port number as the server to find the server's IP address, then, after it successfully acknowledges the server's IP address, a connection is established and data can be sent until the connection is terminated. In the case of this research, the client computer connects to a VPN to increase the security of the communication and allow access to the same local network that houses the server computer. The server then actively listens to the port for messages from the client containing packets of positional values from

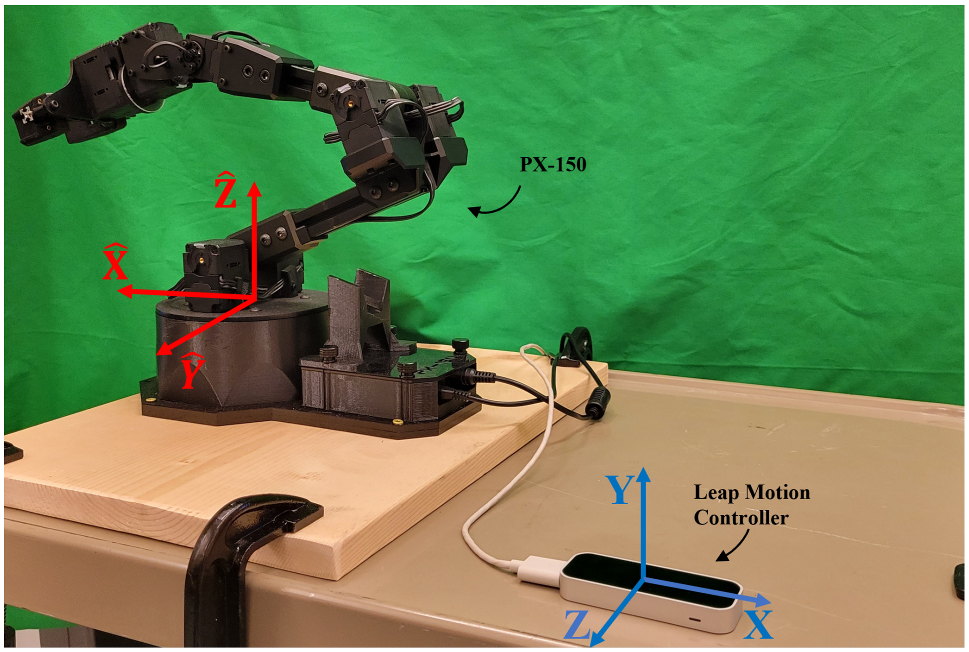

Figure 2: Rotation from LEAP Motion frame to PX-150 frame

the LEAP Motion Controller.

4.1.3 Receiving Data and Interfacing with Robot

Once the server has received the JSON string, the code then deserializes the string back into a list. As shown in Figure 2, before the received values can be used by the PX-150, they must be transformed into the correct orientation by rotating the values -90 degrees along the x-axis and then -90 degree along the z-axis.

After the position values have been orientated correctly, they can now be published to the kinematics engine using the *set-ee-pose-components* method. The *set-ee-pose-components* method is part of the arm library of the PX-150 that houses all of the kinematics calculation data. If a successful path was found, it will publish a list of angles for each of the servos to allow the arm to achieve the desired end-effector position. If a position is not found then it will return that the robot has failed to converge to the specified position and will stay in the last position it reached. Part of the functionality of the end-effector is the opening, closing, and rotation of the gripper. Before the code was sent, the grab strength of the hand was retrieved from the LEAP Motion API as a value between 0 and 1; if the user closes their hand, then the gripper on the end-effector will send the command to close. The LEAP Motion API also captures the rotation angle of the user's hand, which is also sent with the position data which is then passed to the robot as part of the end-effector's orientation. These movements will continue so long as there maintains an active connection between the server and the client computers.

4.2 User Study with Low-cost Robotic Arm Platform

4.2.1 Control Delay Study

One of our current experiments includes the transmission of data one string at a time over the internet. In this experiment, the goal was to find the total execution time of the code and confirm there is a delay in the transmission of data over a long-distance connection versus the transmission of data in a local environment. The data that was needed in this experiment was the length of travel time over the internet and the action time of the PX-150's motion. This experiment has been tested at four different connection configurations: a straight connection (a connection with no web socket), a LAN connection on the same computer, a LAN connection on a different computer, and a WAN connection across multiple states using a VPN. The results of the experiment were as follows: by comparing the total time for one iteration of the loop versus the times that were found throughout the experiment, it was shown that all of the delays throughout the code were accounted for, the movement of the PX-150 took up the majority of time spent per iteration, and that the time it took to send the data over each connection increased with the increased distance between each computer. After running multiple iterations of the experiment described above to collect data, the results have been gathered and shown in Figure 3 and Figure 4.

As we can observe from Figure 5, when the experiment is conducted through a straight

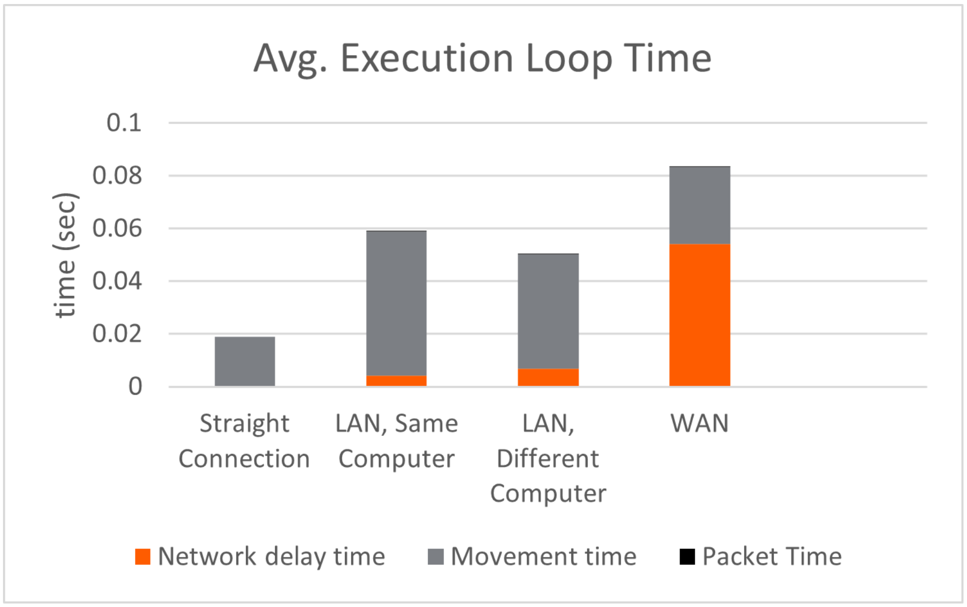

Figure 3: Average time to execute one iteration of the program.

connection, the execution time is mainly composed of the robot's movement time, which spans from $0.008s$ to $0.018s$. Similarly for the LAN connection on the same computer and the LAN connection on different computers, the robot's movement time still dominates the total execution time with a median of $0.021s$ and $0.032s$, respectively. The remaining part of the execution time for these two configurations mainly comes from the network delay, whose median is around $0.005s$ and $0.006s$, respectively. Different from the previous three configurations, the dominant part of the execution time for the WAN connection is the network delay, spanning from $0.042s$ to $0.091s$, while the robot's execution time spans from $0.009s$ to $0.047s$. For all these four configurations, the execution time contributed by the packet time is close to $0s$, which can thus be neglected. In regard to the total execution time, the straight connection achieves the shortest time around $0.02s$, the two LAN connections show a longer time between $0.05s$ and 0.06s, and the WAN connection turns out to have the longest time around $0.08s$.

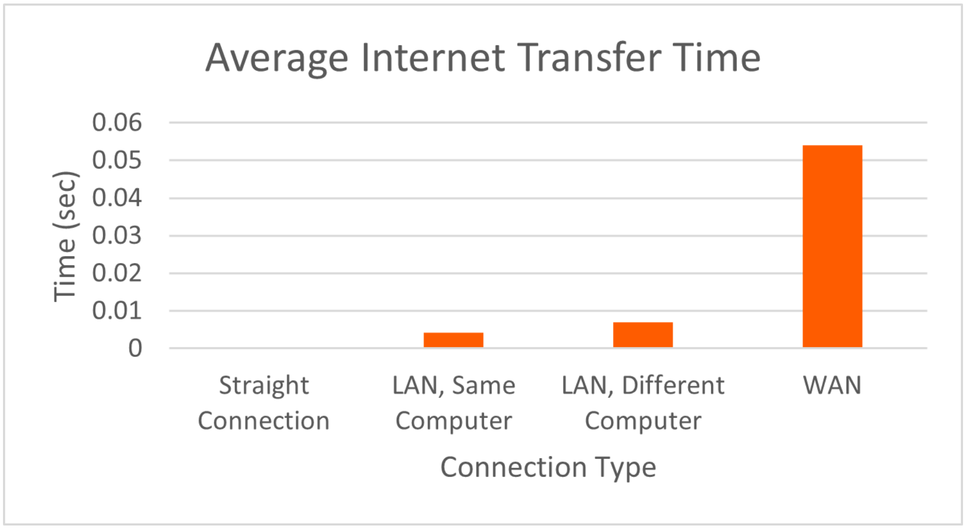

Figure 4: Average time taken to transfer data over the internet.

4.2.2 Local Control vs. Robot Replay

The robotic arm reads human motion as the end-effector trajectories frame by frame. Joint angle control commands are derived from the frame series leveraging inverse kinematics. As a result, it forms the robotic arm's end-effector trajectory. Moreover, the human hand motion is also sent to another robotic arm afterward, the robotic arm executes the commands to take a replay control rather than an interactive control.

We first compare the trajectories of human motion, controlled robot, and replayed robot to study the effects of transmission and robot execution on the motion trajectories. For example, the trajectories of drawing a "S" in the remote and local replay scenarios are shown in Figure 6. We can observe that the trajectories of the robot in both real-time control and replay control are close to the human hand trajectory, which indicates that the robot follows the human movement well and the latency during the remote control has a very slight impact on robot trajectories.

We further compare the trajectories by computing their Dynamic Time Warping (DTW) distance [26] between each other. The smaller DTW distance means a higher similarity between the trajectories. As illustrated in Figure 7, the DTW distances between the trajec-

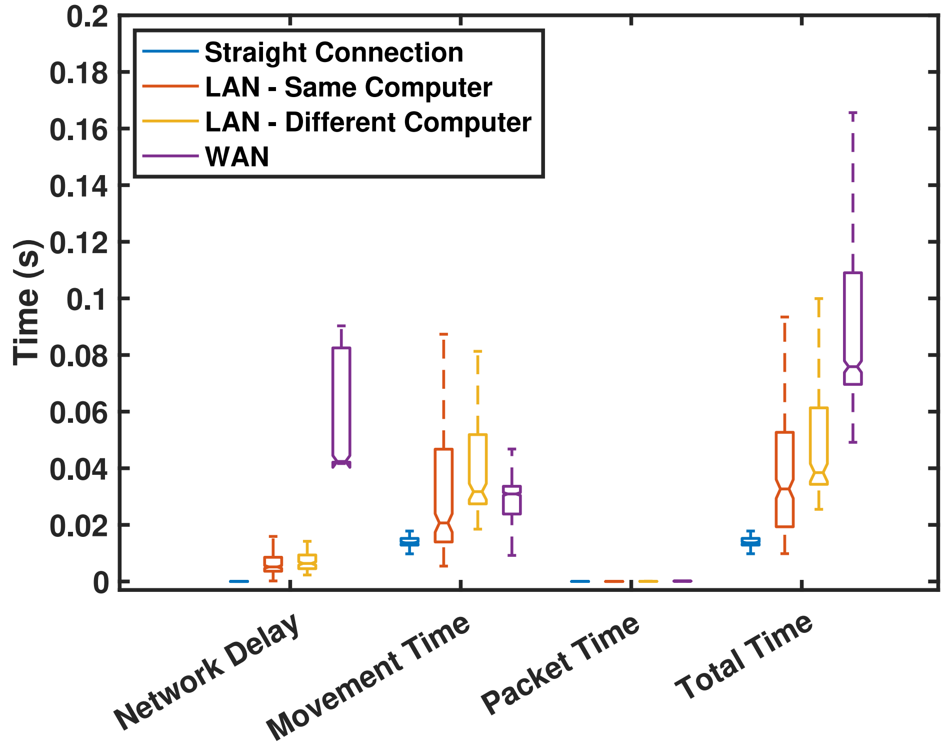

Figure 5: Execution time of different connection configurations.

tories of the robot in different control types and that of the human hand span in the same range, indicating these trajectories are close to each other and hard to be differentiated. Figure 8, Figure 9, and Figure 10 illustrate the time shift on the axes of the trajectories in 20 frames caused by the transmission and robot execution latency. Although the axes values are not differentiated, we can observe the robot's movement over time is not as smooth as human motion, and the time shift is not constant for every frame, which indicates the robot executing speed is swinging in a small range.

4.2.3 Remote vs. Local Control

We now present the experiments when the participants are controlling the robotic arm in remote control and local control scenarios. In these two scenarios, the participants repeat drawing letters (e.g., "S") in the air 20 times. The hand motions are captured by the Leap Motion sensor based on which we generate a time series of human hand 3D coordinates, The

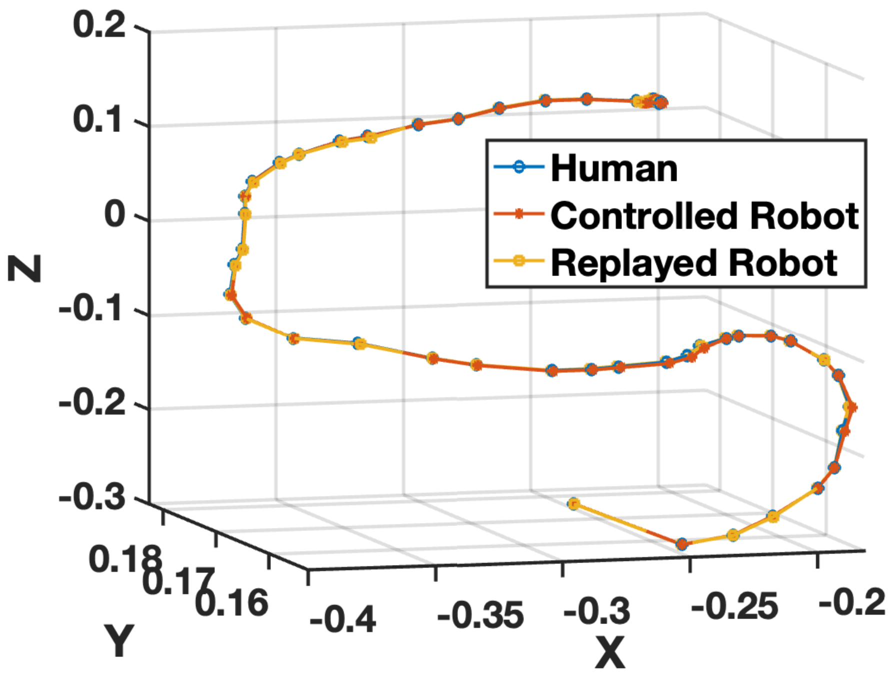

Figure 6: Trajectories of drawing a "S" in different control scenarios.

human motion is transmitted to the robotic arm concurrently through a WAN for remote control or through a LAN for local control. Both are in real-time. Figure 11 presents the DTW distances in the same control scenario as well as between the remote control and the local control scenarios. The results show that the user behaves differently in different scenarios while their motions are consistent in the same scenario.

4.2.4 Free-form vs. Reference Control

We next study the impact of a reference curve on the remote control performance, which guides the user to move the robotic arm by following a specific trajectory (e.g., letter) rather than an individually interpreted curve by heart. Specifically, we conducted experiments in two different settings. In the first set, the participants performed gestures in their own styles without a reference, which we call the free-form gesture control. While in the second set, the participants were required to control the robotic arm to follow the trajectory of a reference drawn on the paper. The comparison of the user's motion curve to the far-end robot with

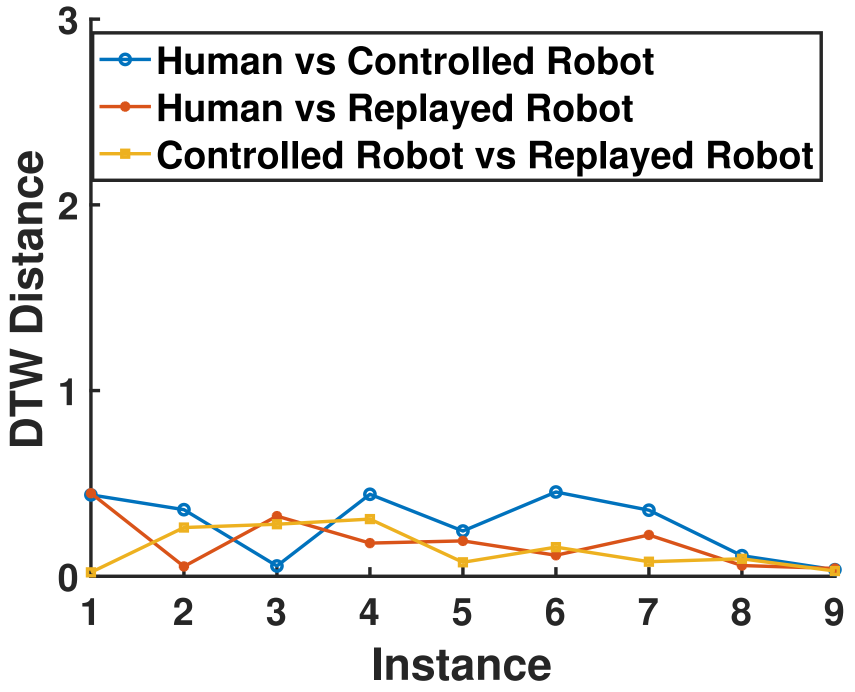

Figure 7: DTW distances of the trajectories in different control scenarios.

or without a reference is shown in Figure 11. We find that the DTW distances between the reference control and the free form control (i.e., red curve) are much higher than that calculated with either control scenario (i.e., orange curve). The results indicate that the user's behaviors in the two scenarios are different. Nonetheless, we find that the robot is able to replicate the user's motion with high similarity.

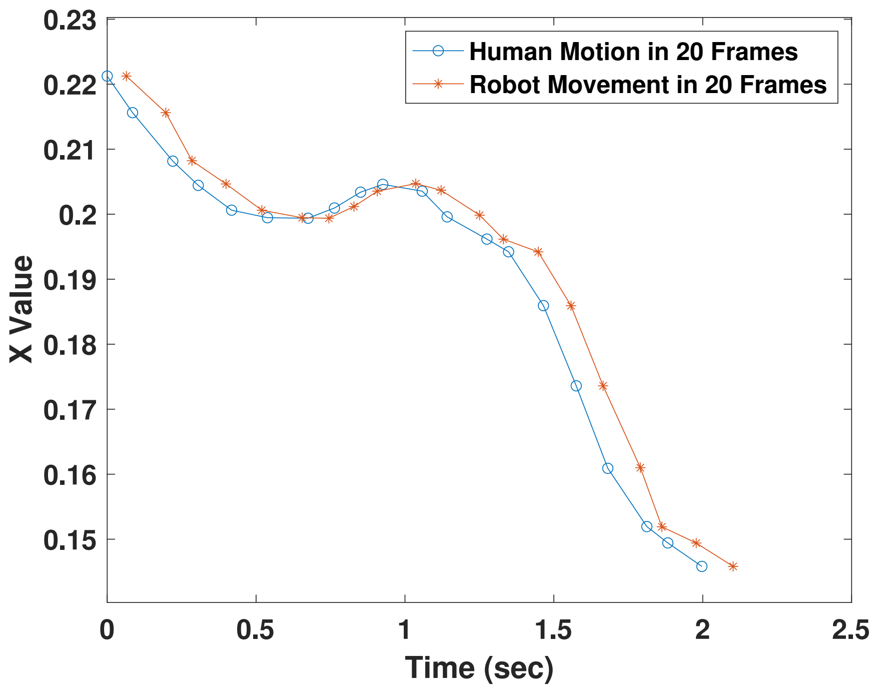

Figure 8: Latency on the x-axis between the robot and human in remote control.

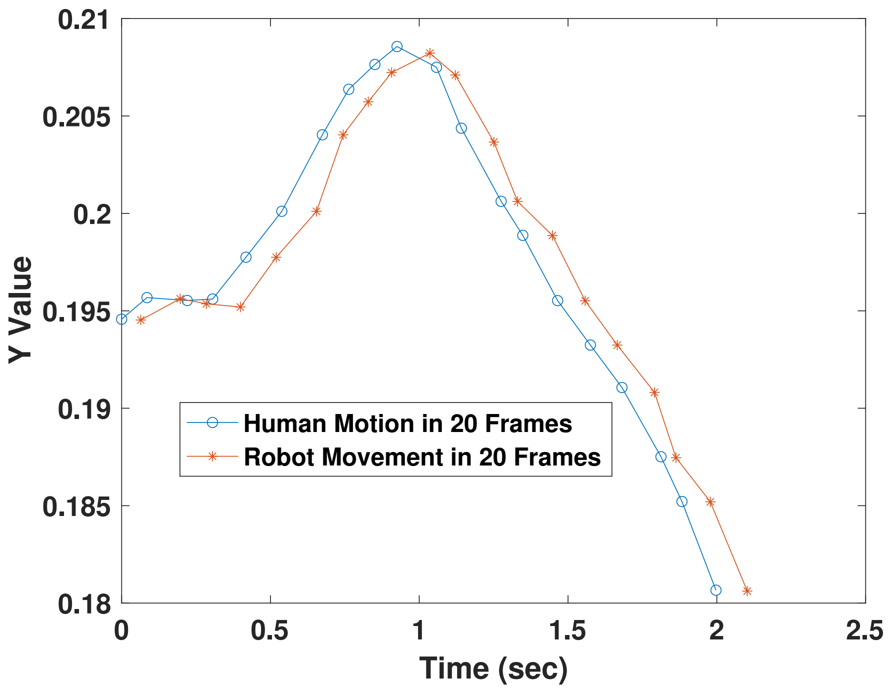

Figure 9: Latency on y-axis between the robot and human in remote control.

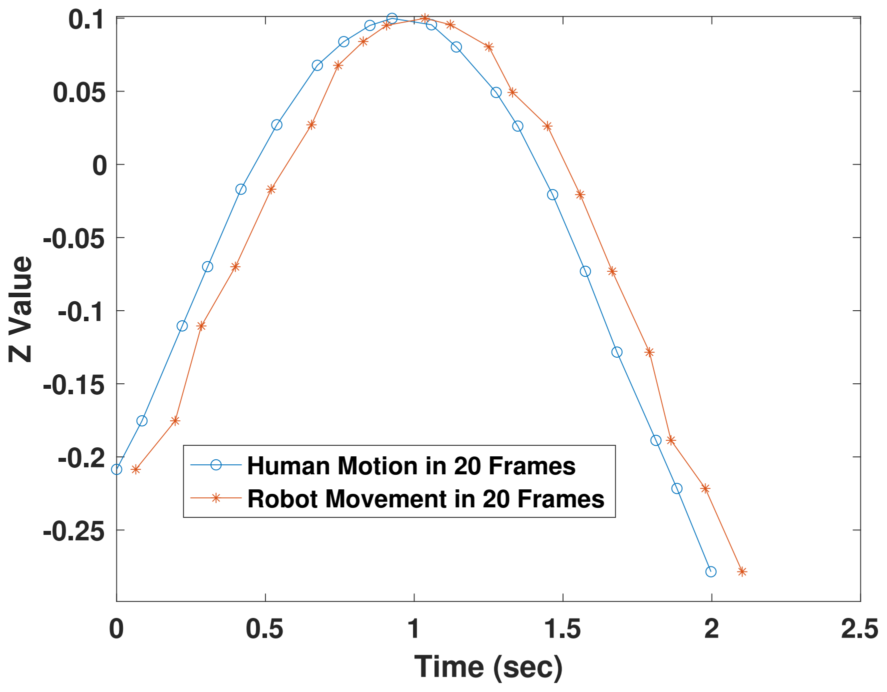

Figure 10: Latency on z-axis between the robot and human in remote control.

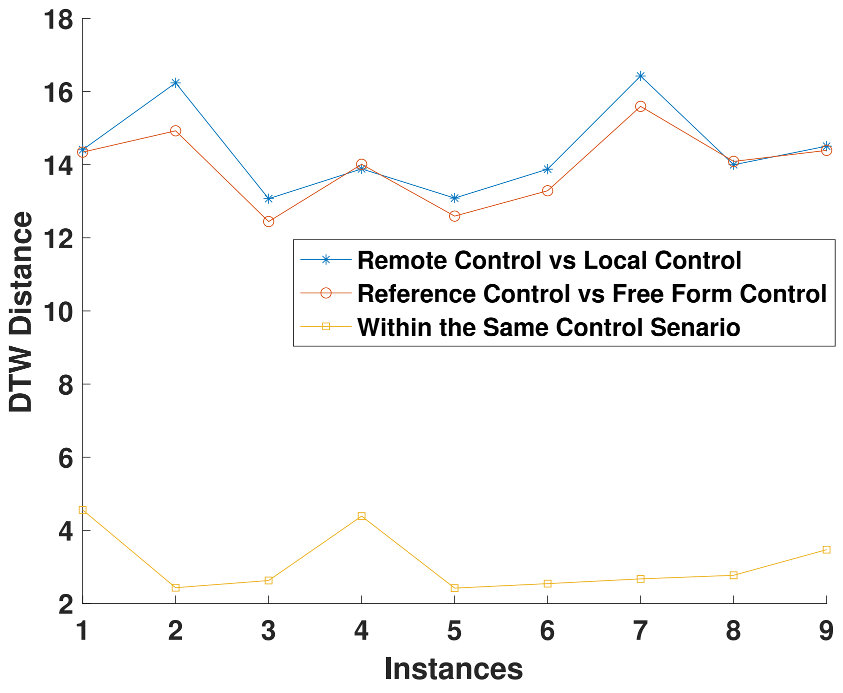

Figure 11: Comparing the similarity between different control scenarios and within the same scenario.

CHAPTER V

STUDY TWO

5.0.1 Introduction

The second study was conducted to compare different setups of the platform to determine which one allowed for a better user experience while allowing for fast and accurate completion of different tasks. The two tasks the participants were asked to complete were line tracing and object manipulation. Once the tasks were completed then the participants were asked to fill out a survey allowing for user satisfaction data to be gathered on each of the tasks to determine which setup was the most intuitive for the participants.

5.0.2 Platform Overview

The overview of our system and the setup environment is shown in Figure 12. To track the user's hand motion and control a robotic arm in real-time, a Leap Motion camera [33] with hand recognition API provided by Ultraleap is used to reconstruct palm center positions in a 3D coordinate, obtain hand orientation, and gripping behaviors. Then the hand trajectory is optimized for robot planning with scaling and offsets to overcome limitations from both the camera sensing range and the robotic arm work envelope. Moreover, we lower the motion control latency with Last-In-First-Out buffering which speeds up computation time and smooths the robotic arm execution with PID controllers. The optimized trajectory is sent to a PX-150 robotic arm through a web socket and then into the Interbotix run-time environment which allows the robot to move.

There are three different setups used to test the interactions between the user and a

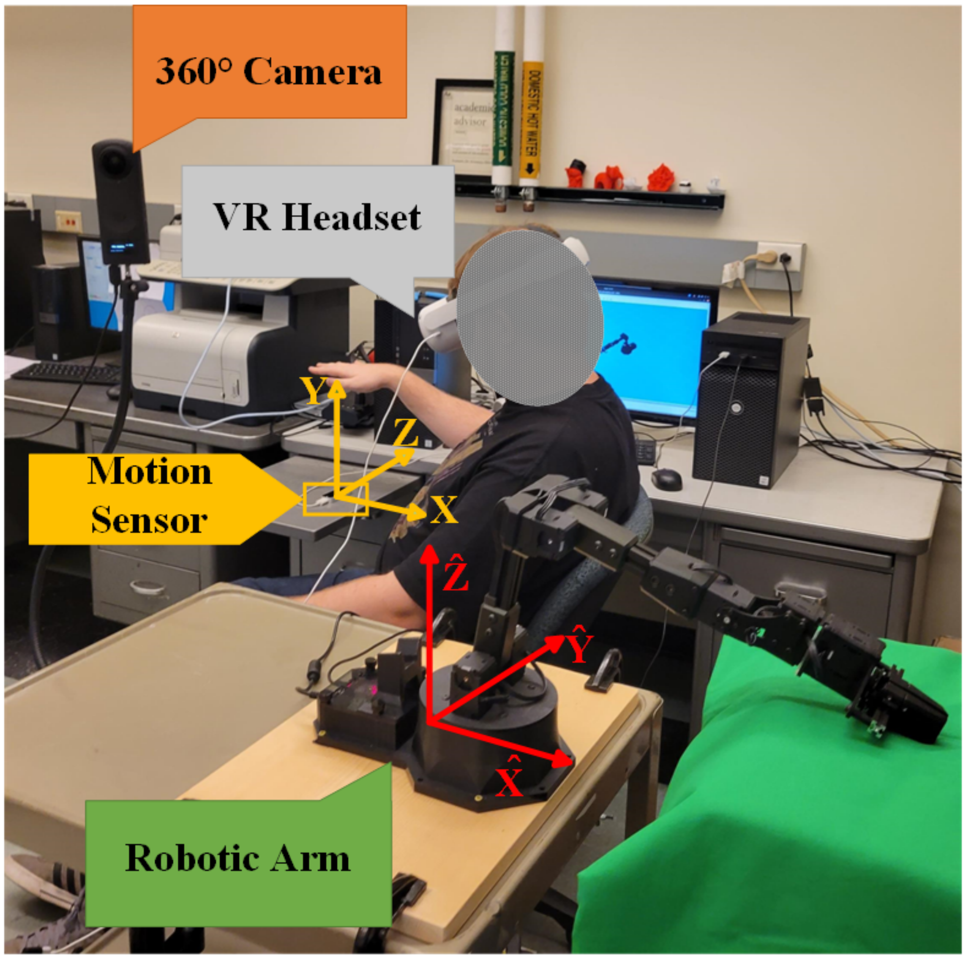

Figure 12: Illustration of VR-enabled robotic arm control platform.

robotic manipulator.

Line of sight The first of these setups is line of sight in which the user can see the robot and receives visual feedback from watching the robot in motion. This results in no feedback delay to the user.

2D Feedback The second setup consists of the user only receiving visual feedback from a web camera instead of a direct line of sight. This restricts the user to see the movements of the robot on a 2D computer monitor only.

VR Headset The final setup is viewing the movements of the robot in a VR headset. By live streaming 360° camera footage into a VR headset, the user is able to see the robot from the point of view of the camera. This is similar to the 2D setup as the user is restricted from having a line of sight of the robot. However, instead of a 2D monitor, the user wears a VR headset. This allows the user to look around the environment and get a full view of the robot and the area around it.

5.0.3 Hardware Integration

This research utilizes many different pieces of hardware: a PC installed with Linux Ubuntu 20.04, a PincherX 150 Robot Arm from Trossen Robotics, a LEAP Motion Controller from Ultraleap, an Oculus Quest 2, and a Ricoh THETA Z1 360° camera. The LEAP Motion Controller is a motion tracking camera that uses three LEDs and two infrared cameras in order to triangulate an accurate value of a hand above the sensor at around 120 frames per second. The PincherX 150 Robot Arm, known further in this paper as the PX-150, is a robotic arm with 5 degrees of freedom (DOF) and a 900mm wingspan. When fully extended, it can reach up to 450mm in length and can pick up payloads of up to 50g. The PX-150's base and end effector are 3D printed and the servos used on the arm itself are DYNAMIXEL X-Series smart servos which are connected together through a DYNAMIXEL U2D2 controller. This controller connects to a computer through a USB serial connection and allows it to utilize the pre-built PX-150 ROS packages for Noetic while the kinematics engine used to move the robot is included in the software package of the PX-150. The Meta Quest 2 is a factory-issued VR helmet from Meta and is able to work as a standalone helmet or connect to a computer to increase its capabilities and battery life. The Ricoh THETA Z1 360° camera was manufactured by the THETA 360 company and it is the latest in their 360° video and image-capturing hardware. The THETA Z1 camera is able to support up to 7K image quality for still images, and up to 4K video and live streaming.

5.0.4 Software Components

Vision-based Hand Motion Tracker The developing company for the LEAP Motion Controller, Ultraleap, provides an API that allows developers to utilize the functions of the

LEAP Motion Controller to access data from the sensor. A few functions that are utilized in this research include the real-time position and orientation tracking of the palm of the hand as well as the accessing of previous frames captured by the sensor.

ROS-based Human-Robot Interface The interface used to communicate with the PX-150 is a ROS wrapper that was built by Interbotix. Starting from the research layer (main code) there are three main layers that connect the application to the physical robot. These three layers are the application support layer, the control layer, and the driver layer. Starting with the application support layer, this layer houses the main ROS packages such as the inverse kinematics engine module, the ROS control package, and the MoveIt interface package. These packages are how the user's inputs are able to interface with the lower-level ROS packages. The control layer contains the necessary configuration files of the robot such as the robot's joint names, default joint values, as well as all the other robot parameters. The final layer between the application and the hardware layer is the driver layer, this layer holds the interbotix-ros-core package that directly communicates with the PX-150's onboard controller (U2D2). This completes the transfer of data from the application to the physical robot.

Unity and VR Implementation The Unity software suite is used to transfer the live streaming 360° camera feed from the computer to the VR headset. Unity uses a custom script on the computer side and interfaces with the VR headset using the built-in Quest link application. This allows for a direct pipeline from the 360° camera feed to the user in the VR headset. The user will see from the point of view of the camera and has the ability to look around seeing a 360° view of the surrounding area. To make it easier for the user to see the robot, the camera is centered so when the user is looking forward the robot is in front of them. The camera feed is also adjusted to align the robot with the user's hand direction.

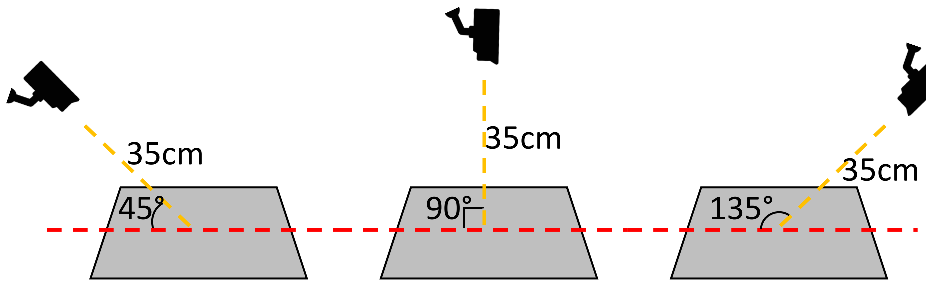

Figure 13: Camera viewing angles.

5.0.5 Gathering and Converting Motion Data

By accessing the LEAP Motion API, the code starts a listener that takes in tracking data from the sensor. After the listener has started, the program waits until a hand is in view above the controller to then start reading position values. When a hand is in view, the program then takes the current frame and the previous two frames, then divide them by the total number of frames in use, in this case, to create a rolling average of the three positional values of the palm center in order to get an accurate position value for the robotic arm to use. The code takes this rolling average in order to create a low-pass filter to reduce the extra noise caused by hands prone to twitching or jerky movements and increase the accuracy of the movements when sent to the PX-150. With this rolling average of position values, we then scale them into values that the robotic arm can recognize. With the PX-150, it has a workspace that is "a specification of the configurations that the end-effector of the robot can reach" [18], meaning that the robot has a space where the arm can create a valid configuration. In order for the PX-150 to configure into a correct position, we must scale the position values from the LEAP Motion Controller into positional values that can be accepted by the PX-150. To do this, we take the positions of the LEAP Motion Controller on the x, y, and z axes and convert them into a corresponding $\hat{x}$, $\hat{y}$, and $\hat{z}$ position that is valid for the PX-150. This scaling is necessary in order to make sure that when the hand is at the farthest sensor position above the LEAP Motion Controller, the PX-150 will be fully extended to a corresponding position. In addition to scaling the position data, the LEAP

Motion Controller also looks for the distance between the forefinger and thumb to determine if the hand is in a fist position. We take this value and call it the grab strength, as it will be utilized by the PX-150 to open and close the end-effector. After the code has finished processing all of the necessary data, it then enters the position values, the roll value of the hand, and the grab strength value into a list which is then serialized into a JSON string to be sent over the web socket.

5.0.6 Secure Local Area Network Connection

In order to get the values from the LEAP Motion Controller to the PX-150, our code establishes a secure client-server relationship between two computers using a web socket. When creating a web socket connection, a person must have access to the same IP address, internet port, and header number to establish a proper connection. When the server is started, it will start listening on the pre-selected port number with the server computer's IP address. When the client is started, it will look on the same pre-selected port number as the server to find the server's IP address, then, after it successfully acknowledges the server's IP address, a connection is established and data can be sent until the connection is terminated. In the case of this research, the client computer connects to the same local network that houses the server computer. The server then actively listens to the port for messages from the client containing packets of positional values from the LEAP Motion Controller.

5.0.7 Receiving Data and Interfacing with Robot

Once the server has received the JSON string, the code then deserializes the string back into a list. Before the received values can be used by the PX-150, they must be transformed into the correct orientation by rotating the values -90 degrees along the x-axis and then -90 degrees along the z-axis.

After the position values have been orientated correctly, they can now be published to the kinematics engine using the *set-ee-pose-components* method. The *set-ee-pose-components*

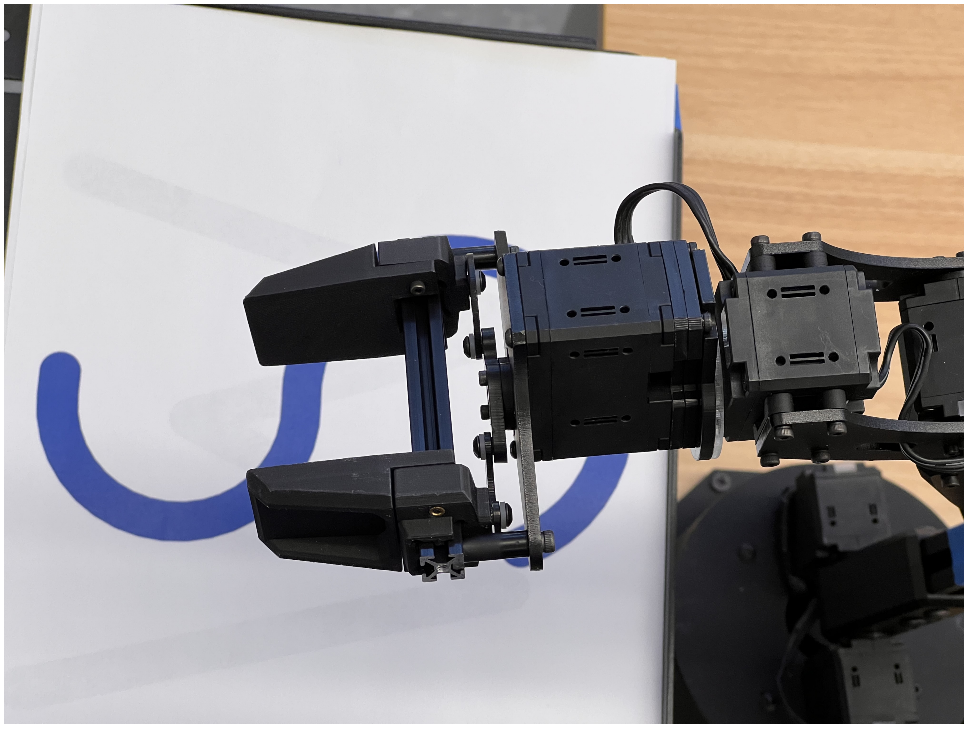

Figure 14: Example of curve line tracing.

method is part of the arm library of the PX-150 that houses all of the kinematics calculation data. If a successful path was found, it will publish a list of angles for each of the servos to allow the arm to achieve the desired end-effector position. If a position is not found then it will return that the robot has failed to converge to the specified position and will stay in the last position it reached. Part of the functionality of the end-effector is the opening, closing, and rotation of the gripper. Before the code was sent, the grab strength of the hand was retrieved from the LEAP Motion API as a value between 0 and 1; if the user closes their hand, then the gripper on the end-effector will send the command to close. The LEAP Motion API also captures the rotation angle of the user's hand, which is also sent with the position data which is then passed to the robot as part of the end-effector's orientation. These movements will continue so long as there maintains an active connection between the server and the client computers.

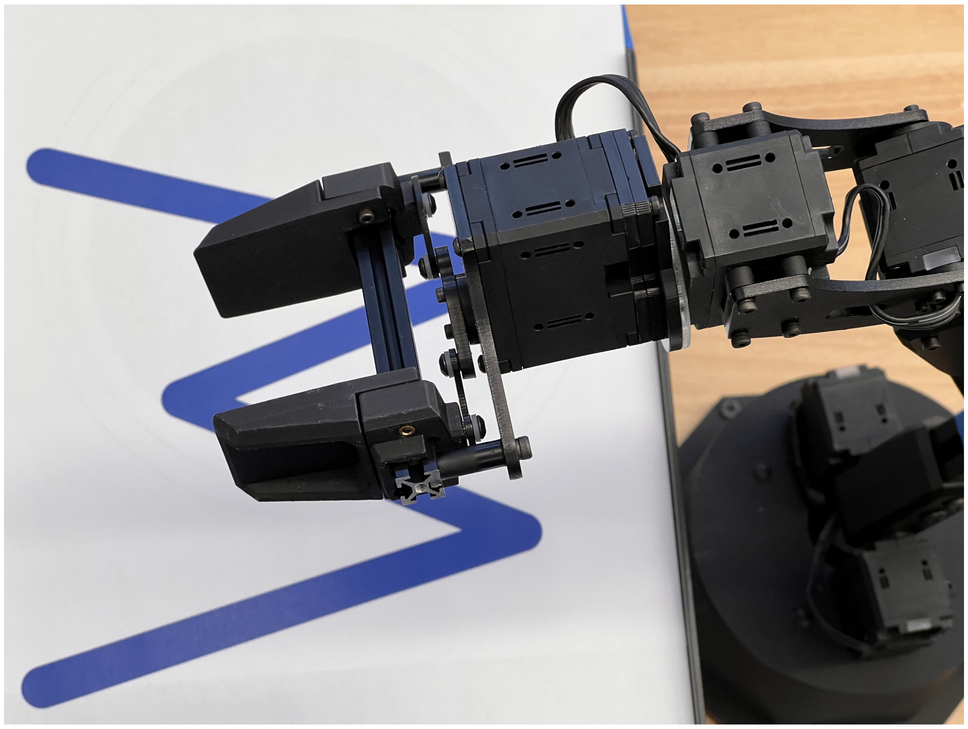

Figure 15: Example zigzag line tracing.

5.1 Study Design

To evaluate the smoothness, efficiency, and versatility of the VR robotic arm platform, we design three tasks to investigate control latency, motion accuracy, and overall task achievement. During the experiments, the robotic arm works as described in Section 5.0.2, and there is no line of sight between the robot and the user. The THETA Z1 360° camera, as well as an Elecom 4K webcam, are set to be 35cm away from the center of a work pad in 2D and VR experiments respectively. Each of the cameras has three viewing angle setups as illustrated in Figure 13.

In each experiment, seven participants conduct 10 trials in three tasks: in-air free-style writing, line tracing, and object collection and placement. Both robotic arm trajectory and time data were collected for evaluation.

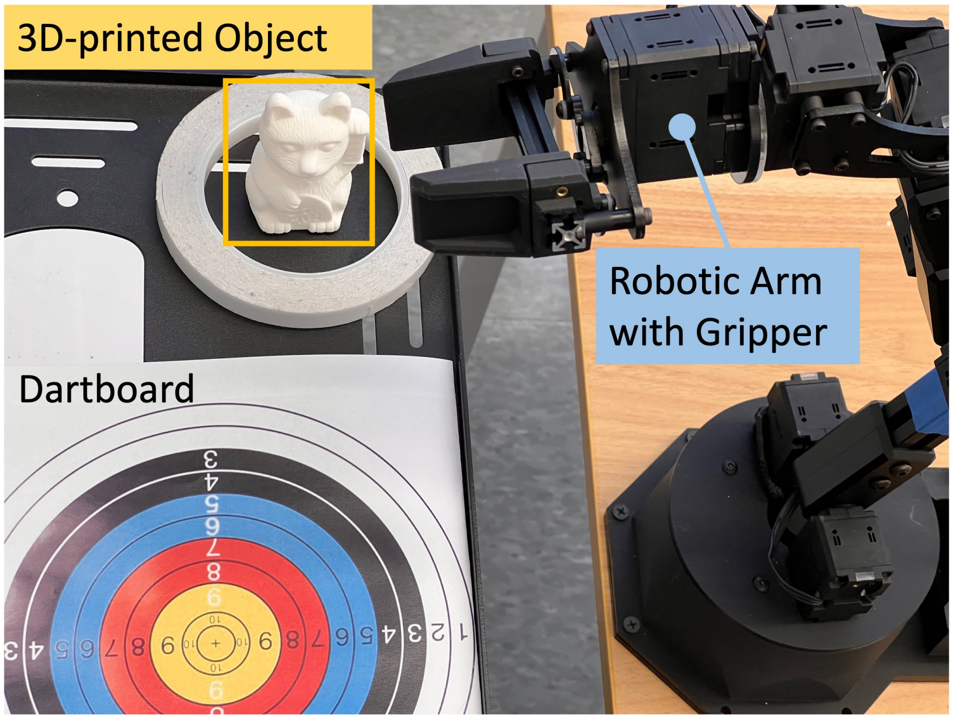

Figure 16: Object collection and placement setup.

5.1.1 Control Latency

In this task, participants wrote an “S” in the air without constraint to get familiarized with the platform. Meanwhile, the time of each component in the control loop is measured.

In a control loop, the motion captured by the sensor is packaged and sent to the robot server. When the packages are received, algorithms including Last-In-First-Out buffering are utilized to reduce the entire transmission time. Afterward, the robot server converted the motion data into robotic control commands with axes alignment and workspace mapping so the robotic arm can always reach a pose within its range as long as the hand motion is captured.

5.1.2 Motion Accuracy

This task asked the participants to control the robotic arm’s end-effector with their hand motion to track a line printed on the paper which is placed on the work pad. Specifically,

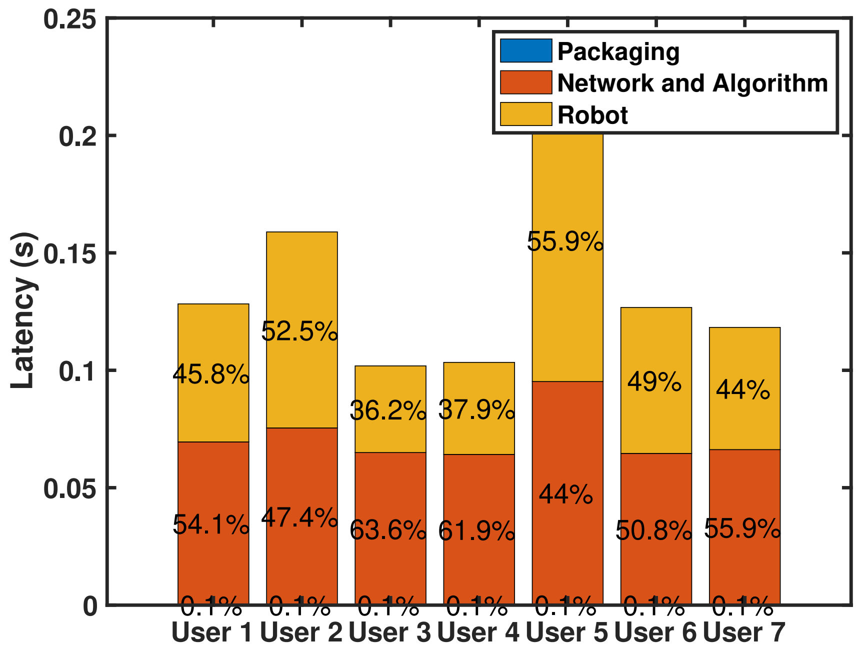

Figure 17: Average latency of each control loop.

there are two types of lines, including a curve line (e.g., an “S” as shown in Figure 14) and a zigzag line (e.g., a “W” as shown in Figure 15). All the lines were generated by Python programs therefore they can be utilized as reference lines with known 2D coordinates. Because there was no LOS between the participants and the robotic arm, they observed the work pad area with the visual feedback from the 2D video or the VR stream. In both setups, the participants repeated the experiments for the three camera viewing angles.

5.1.3 Task Achievement

This task is object-oriented and the participants are required to complete point-to-point moving, object collection, and placement.

Object Collection They first move the robotic arm’s end-effector from the start position to the object position, trying to grab the object that is placed in a circle. With gesture recognition enabled by motion tracking algorithms, the participants can open and close the gripper attached to the robotic arm. As illustrated in Figure 16, the object to be collected

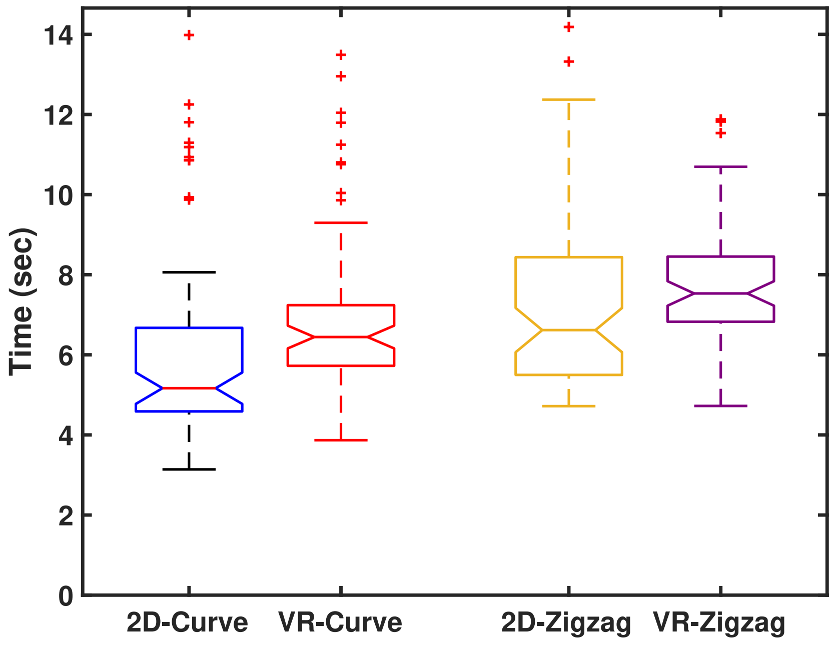

Figure 18: Speed of task completion when the camera angle is 45°.

is a 3D-printed cat, with a 4cm diameter base.

Object Placement Once grabbed, the participants move the object above a dartboard and then set the object on it. The circles on the dartboard help the participants to estimate the distance. Due to total control latency, human error, and lack of depth of view in the video, the object cannot always be placed in the center of the dartboard.

We count the time for the participants to successfully grab the object as well as the time to release the object, to indicate the effort it takes to complete the task. Meanwhile, the distance between where the object was placed and the center of the dartboard is measured to represent the achievement of the task.

5.1.4 User Satisfaction

After participants finished all the tasks, they were asked to answer a questionnaire as well as rate their experience of the platform and the interactive tasks. There are 12 questions related to control latency, motion accuracy, and task achievement which are scored between

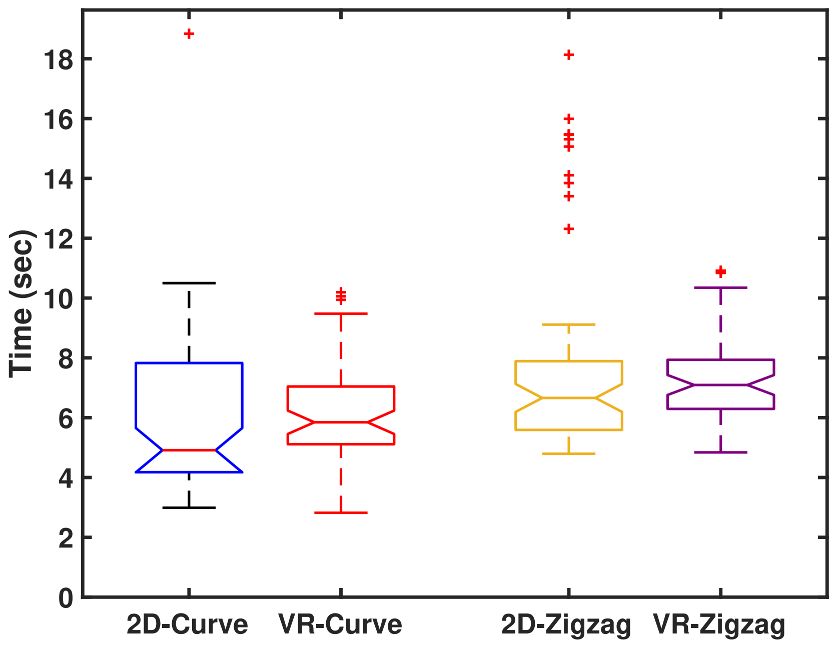

Figure 19: Speed of task completion when the camera angle is 90°.

1 to 10. They were also given the option to comment on the overall experience.

5.2 Performance Evaluation

5.2.1 Latency

The components of a control loop are categorized into three types, namely packaging, network and algorithm, and robot execution. As illustrated in Figure 17, the packaging time is negligible compared to the network transmission and robot execution time, and for most users' experiments, the robot execution time is less than the network time. The total time between motion packaging and the completion of the robot's execution is around 0.2s.

Moreover, we measured the transmission time for the 360° camera to stream to the VR headset and for the webcam to stream to a 2D external monitor, as shown in Table 1. The transmission latency was higher for the VR setup due to the 360° video being transmitted

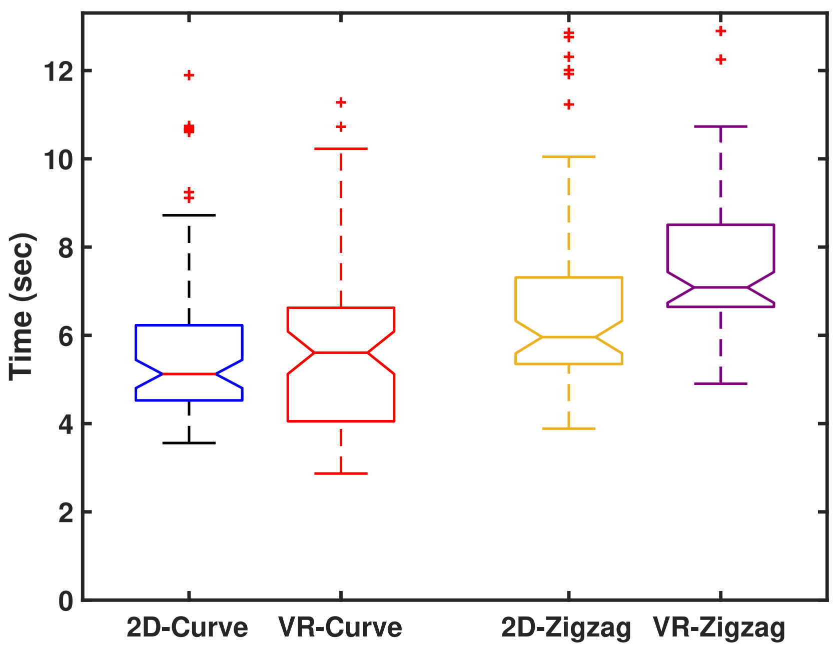

Figure 20: Speed of task completion when the camera angle is 135°.

through more software layers.

5.2.2 Line Tracing Time and Similarity

Figures 18,19,20 present the time it takes the participants to finish the line tracing task, and Figures 21,22,23 shows the similarity between the robot trajectory and the reference lines. Generally, it takes longer to complete line tracing in the VR setup due to a larger camera delay. However, the robot trajectories in the VR setup still have comparable performance to the 2D setup when following the reference line.

All the participants took a similar effort to complete the task and the trajectories had the same level of similarity when the camera angle changed from 45° to 90°, but when the

Table 1: Average transmission latency of two video streaming setups.

	VR 360° Streaming	2D Streaming
Latency (s)	0.27	0.13

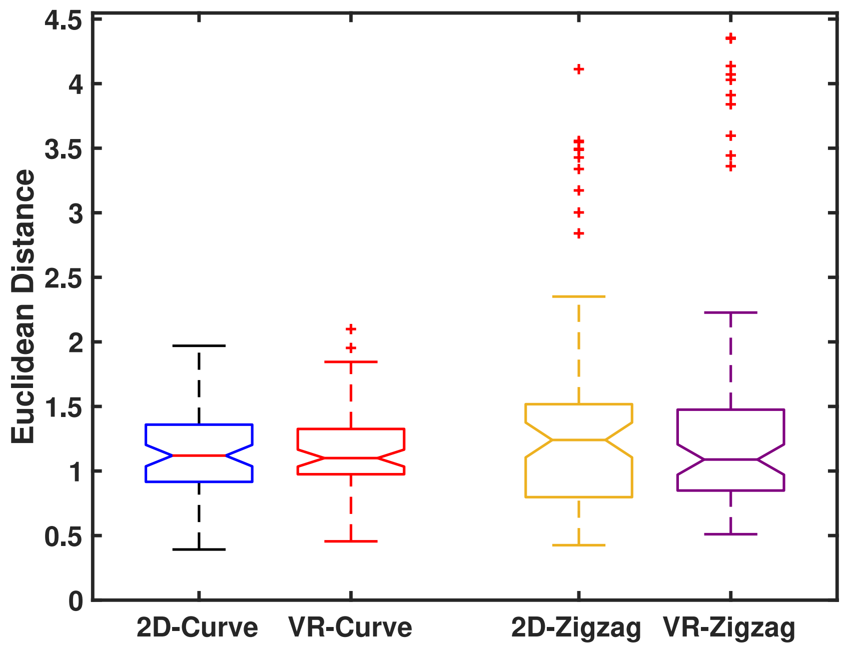

Figure 21: Control accuracy when the camera angle is 45°.

camera rotated beyond 90°, the viewing angle severely distorted the cognition of the axis vertical to the work pad, made the perception upside down. It is observed from Figure 23 that participants' robot trajectories following curve lines are affected both in the 2D and VR setups.

5.2.3 Object Collection and Placement

In this object-oriented task, the participants were not asked to follow a reference line, rather, they were required to successfully grab an object and place it in a desired position. As discussed in the previous sections, the participants experienced a larger control delay in the VR setup and took more effort to finish the tasks in general. However, as shown in Figure 24,25 it took the participants less time in the VR setup than in the 2D setup to successfully grab the object, and the final object placements were closer to the center of the dartboard despite the higher latency. The results indicate the VR setup augmented the user performance in this object-oriented task.

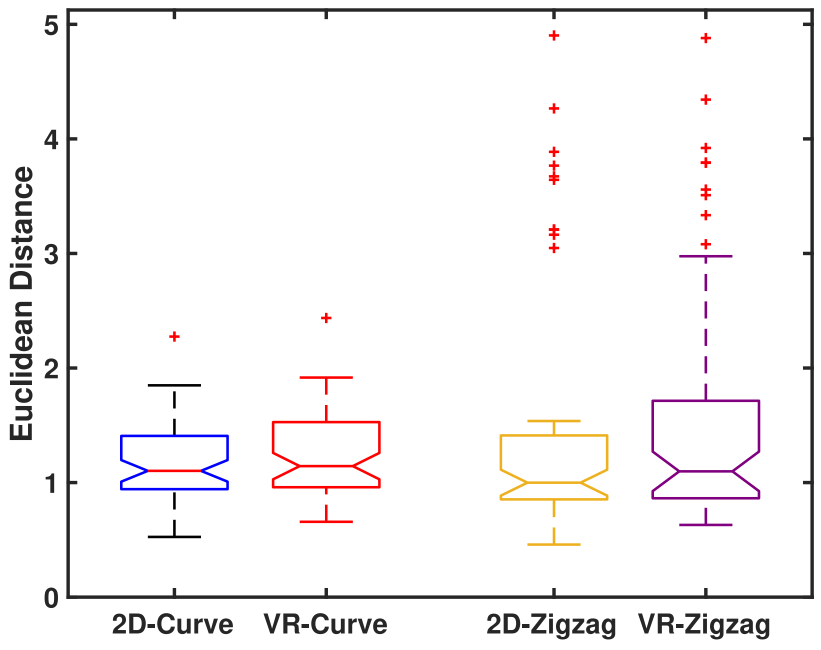

Figure 22: Control accuracy when the camera angle is 90°.

5.2.4 User's Rating

Figure 26 shows the participants' satisfaction with overall latency in LOS, 2D video, and VR streaming motion control experience. Since the 0.1s latency difference was noticeable for the participants, they rated LOS control with the highest score and followed by the 2D video-based streaming.

For the line tracing accuracy, the participants rated the similarity between their hand motion and the target line, between the robot motion and their hand motion, and between the robot motion and the target line. The result is shown in Figures 21,22,23, the participants considered the similarity between their hand motion and the target line was high, and thought the robot's following is acceptable.

The participants also rated the overall experience in the interactive tasks, particularly their impression of conducting line tracing, object grabbing, and placing both in 2D video and VR streaming. Participants agreed the experience of conducting experiments with the

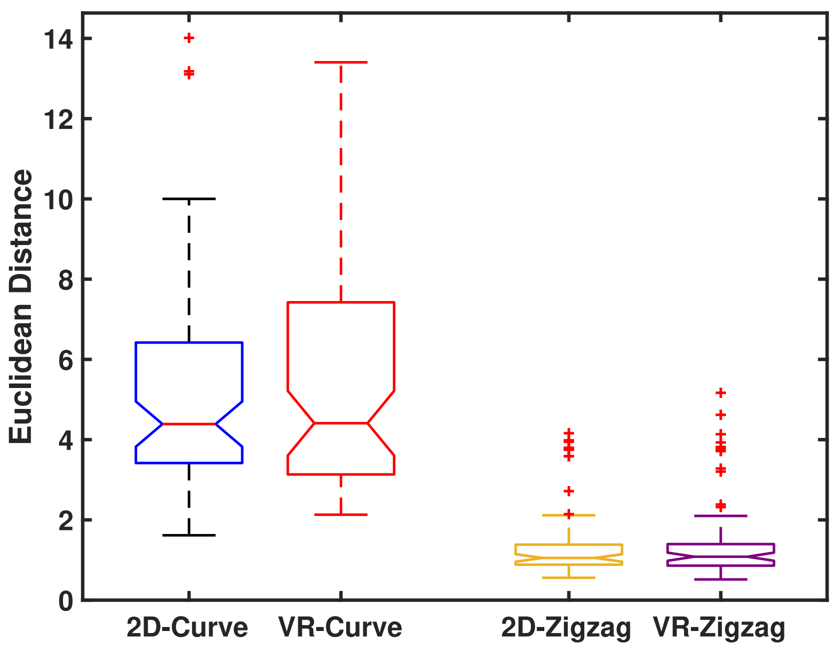

Figure 23: Control accuracy when the camera angle is 135°.

first-person robotic arm was excellent in general. Certain aspects, such as the latency, need time for them to adjust properly, especially when following the curve lines. However, these don't negatively impact the experience once they are used to them. Additionally, they found object placing to be much easier when done in VR than using the 2D screen. In VR they can tilt their head to adjust the angle of view to best suit their needs, and the most natural feeling camera angle in VR was when the camera was at 45°.

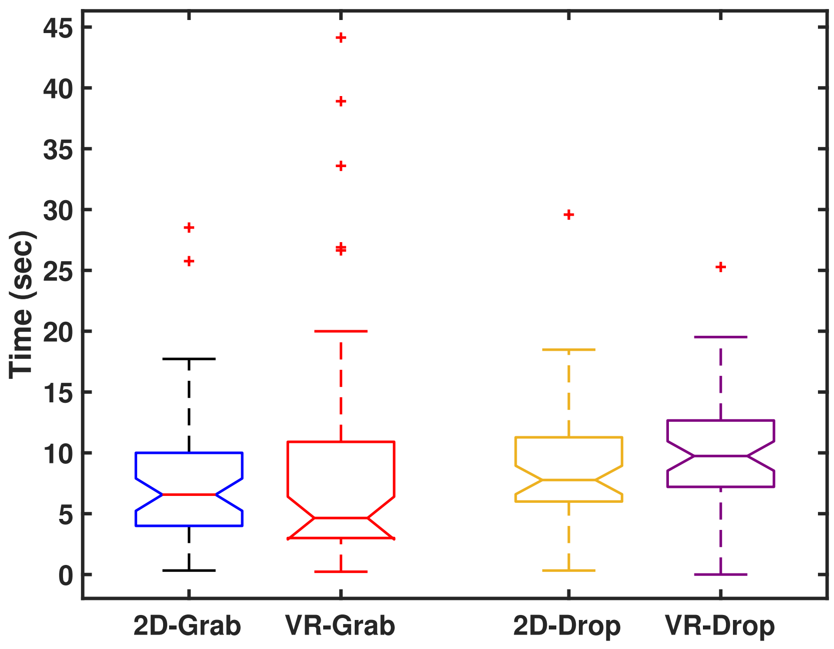

Figure 24: object collection time.

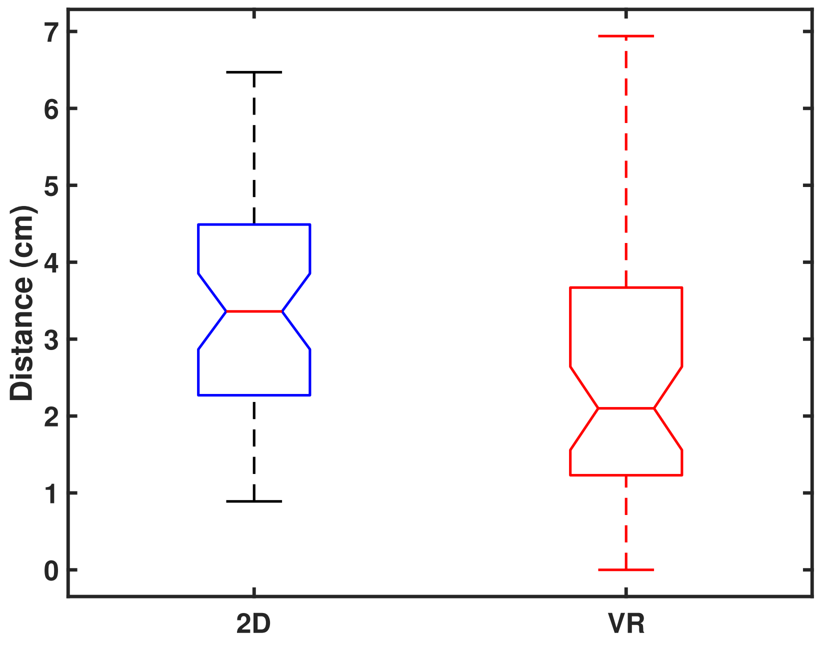

Figure 25: object collection.

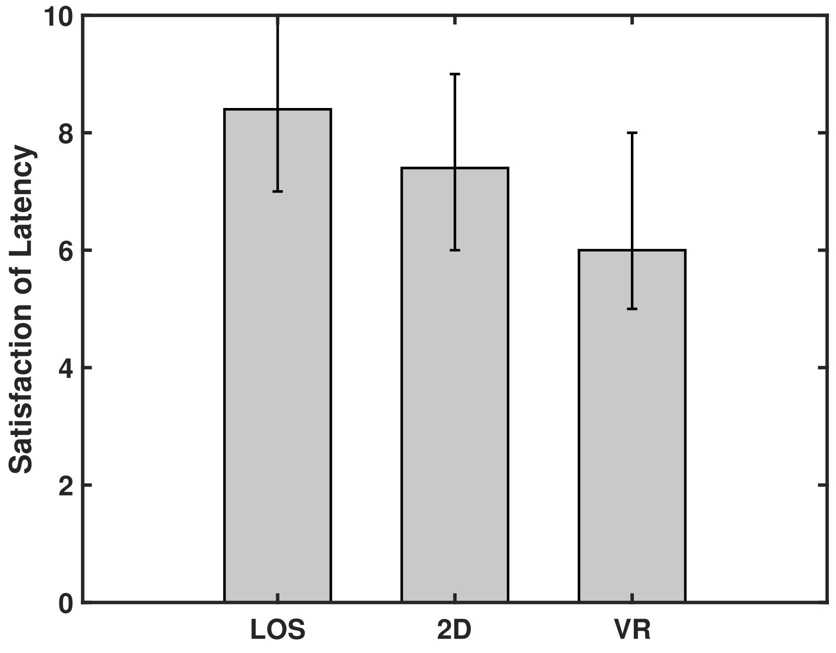

Figure 26: Satisfaction of Latency.

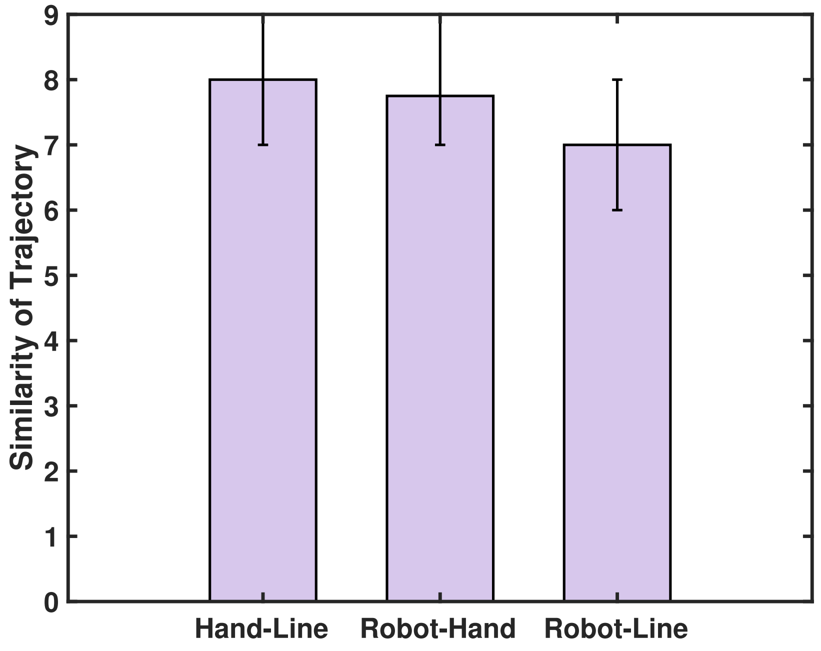

Figure 27: Similarity of Trajectory.

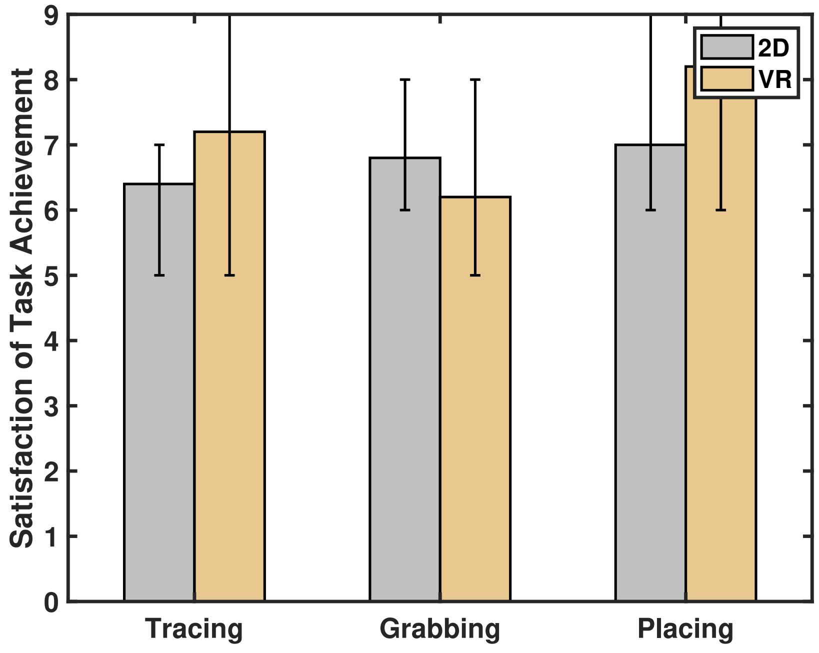

Figure 28: Satisfaction of Task Achievement.

CHAPTER VI

CURRENT WORK

6.0.1 Live Video Feed Using VR Headset

This research is ongoing and is continuing to evolve and improve every day. Currently, research is being done on the possibility of using the Quest 2 VR headset's built-in sensors to replace the leap motion for hand and head tracking capabilities. By using the sensors built into the Quest 2 that track your hand movements similar to the Leap Motion, it will be possible to remove the Leap Motion from the platform and directly use the hand's positional data to move the robotic platform. This will allow users to move freely without being constrained to one location and will eliminate the user from having to be aware of where the Leap Motion is in regard to their hand. The Quest 2 also has a built-in virtual model of a hand that the user will be able to see while wearing the VR helmet, giving them both the feeling of where their hand is as well as a visual aid of their hand location. Visual feedback of the current state of the robot has always been a problem with teleoperation, but by using the Quest 2's VR capabilities it will be possible to send a video stream over a web socket and display it to the user in the helmet giving them a low latency live look at the current position and state of the robotic manipulator.

6.0.2 Camera Movement Using VR Headset

The Quest 2 also tracks the angle at which the user is turning their head which allows that information to be passed to a pan and tilt camera mount on the robotic side of the platform to give the user the ability to look around the environment to check for hazards or other obstacles that might be a danger to the robotic manipulator. This also allows the user to adjust the camera angles to improve depth and distance perception of objects that the user might be trying to manipulate using the end effector of the robot.

6.0.3 Scaling Up the Robot

As mentioned above the current platform is using a low-cost 5 DOF robotic manipulator which does great for small tasks. If a larger task needs to be completed via teleoperation then the ability to scale this platform is critical. Current research is testing the feasibility of using this platform on an industrial 7 DOF robotic manipulator that is in the price range of $20,000 called Sawyer by Rethink Robotics. As it stands early testing has been promising, showing that it is possible by scaling the maximum sensing range of the hand tracking sensor to the maximum reach of the Sawyer robot allowing for a user to control the larger robotic platform by simply moving their hand in front of the leap motion or the Quest 2. More research is needed to get quantifiable data on the accuracy of the system as well as the ease of use felt by the user. The Sawyer robot also comes with a built-in motorized camera on its head which can be used in the testing of the previously mentioned future research of using the head positions from the Quest 2 to "look around" the environment.

CHAPTER VII

CONCLUSIONS

7.0.1 Conclusion

So, in conclusion, this research first showed through study number one that it is plausible to control a robotic manipulator over a WAN in real-time with high accuracy and low latency. This study was conducted using low-cost off the shelf components like the Leap Motion and the PX-150 robotic manipulator. Study one studied the control delay performance of our robotic arm platform under different LAN/WAN connection configurations, and observed that the dominant part of execution time for the WAN connection is the network delay instead of the robot's movement time. In addition, we comprehensively compared the remote control with the local control and conducted a contrasted study with both the reference-guided motions vs. free-form hand movements. The second study implements a first-person robot control platform with hand motions in virtual reality. We developed the platform by integrating a motion-tracking device, a robotic arm, a 360° camera, and a VR headset. In particular, we studied the latency, accuracy, and versatility of the VR robotic arm platform with free-style writing, line tracing, and object collection and placement tasks, and then compared it with the 2D video-streaming environment. Experiments show that our first-person robot control platform with a hand motion in virtual reality is capable of bringing physically-augmented remote human interactions. Moreover, participants' feedback suggests our platform offered a satisfactory and natural experience. Future work will continue to evolve this platform improving user satisfaction as well as decreasing latency and increasing the accuracy of the system.

REFERENCES

[1] Beatrice Alenljung, Jessica Lindblom, Rebecca Andreasson, and Tom Ziemke, *User experience in social human-robot interaction*, Rapid automation: concepts, methodologies, tools, and applications, IGI Global, 2019, pp. 1468–1490.

[2] Miguel Arduengo, Ana Arduengo, Adrià Colomé, Joan Lobo-Prat, and Carme Torras, *A robot teleoperation framework for human motion transfer*, arXiv preprint arXiv:1909.06278 (2019).

[3] D Bassily, C Georgoulas, J Guettler, Thomas Linner, and T Bock, *Intuitive and adaptive robotic arm manipulation using the leap motion controller*, ISR/Robotik 2014; 41st International Symposium on Robotics, VDE, 2014, pp. 1–7.

[4] Alireza Bilesan, Mohammadhasan Owlia, Saeed Behzadipour, Shuhei Ogawa, Teppei Tsujita, Shunsuke Komizunai, and Atsushi Konno, *Marker-based motion tracking using microsoft kinect*, IFAC-PapersOnLine **51** (2018), no. 22, 399–404.

[5] Aude Bolopion and Stéphane Régnier, *A review of haptic feedback teleoperation systems for micromanipulation and microassembly*, IEEE Transactions on automation science and engineering **10** (2013), no. 3, 496–502.

[6] Filippo Brizzi, Lorenzo Peppoloni, Alessandro Graziano, Erika Di Stefano, Carlo Alberto Avizzano, and Emanuele Ruffaldi, *Effects of augmented reality on the performance of teleoperated industrial assembly tasks in a robotic embodiment*, IEEE Transactions on Human-Machine Systems **48** (2017), no. 2, 197–206.

[7] Roland Buchner, Daniela Wurhofer, Astrid Weiss, and Manfred Tscheligi, *Robots in time: How user experience in human-robot interaction changes over time*, International Conference on Social Robotics, Springer, 2013, pp. 138–147.

[8] Guanglong Du, Ping Zhang, and Xin Liu, *Markerless human–manipulator interface using leap motion with interval kalman filter and improved particle filter*, IEEE Transactions on Industrial Informatics **12** (2016), no. 2, 694–704.

[9] Alessandro Filippeschi, Norbert Schmitz, Markus Miezal, Gabriele Bleser, Emanuele Ruffaldi, and Didier Stricker, *Survey of motion tracking methods based on inertial sensors: A focus on upper limb human motion*, Sensors **17** (2017), no. 6, 1257.

[10] K. Goldberg, M. Mascha, S. Gentner, N. Rothenberg, C. Sutter, and J. Wiegley, *Desktop teleoperation via the world wide web*, Proceedings of 1995 IEEE International Conference on Robotics and Automation, vol. 1, 1995, pp. 654–659 vol.1.

[11] Rebecca Hetrick, Nicholas Amerson, Boyoung Kim, Eric Rosen, Ewart J de Visser, and Elizabeth Phillips, *Comparing virtual reality interfaces for the teleoperation of robots*, 2020 Systems and Information Engineering Design Symposium (SIEDS), IEEE, 2020, pp. 1–7.

[12] Long Huang, Zhen Meng, Zeyu Deng, Chen Wang, Liying Li, and Guodong Zhao, *Towards verifying the user of motion-controlled robotic arm systems via the robot behavior*, IEEE Internet of Things Journal (2021).

[13] Panfeng Huang, Pei Dai, Zhenyu Lu, and Zhengxiong Liu, *Asymmetric wave variable compensation method in dual-master-dual-slave multilateral teleoperation system*, Mechatronics **49** (2018), 1–10.

[14] Dong-Hyun Hwang, Kohei Aso, Ye Yuan, Kris Kitani, and Hideki Koike, *Monoeye: Multimodal human motion capture system using a single ultra-wide fisheye camera*, Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology, 2020, pp. 98–111.

[15] Kristiina Jokinen and Graham Wilcock, *Modelling user experience in human-robot interactions*, International Workshop on Multimodal Analyses Enabling Artificial Agents in Human-Machine Interaction, Springer, 2014, pp. 45–56.

[16] Dongjun Lee, Antonio Franchi, Paolo Robuffo Giordano, Hyoung Il Son, and Heinrich H Bülthoff, *Haptic teleoperation of multiple unmanned aerial vehicles over the internet*, 2011 IEEE International Conference on Robotics and Automation, IEEE, 2011, pp. 1341–1347.

[17] Jessica Lindblom, Beatrice Alenljung, and Erik Billing, *Evaluating the user experience of human–robot interaction*, Human-Robot Interaction, Springer, 2020, pp. 231–256.

[18] Kevin M Lynch and Frank C Park, *Modern robotics*, Cambridge University Press, 2017.

[19] Guilherme Maeda, Marco Ewerton, Dorothea Koert, and Jan Peters, *Acquiring and generalizing the embodiment mapping from human observations to robot skills*, IEEE Robotics and Automation Letters **1** (2016), no. 2, 784–791.

[20] Nobuyasu Nakano, Tetsuro Sakura, Kazuhiro Ueda, Leon Omura, Arata Kimura, Yoichi Iino, Senshi Fukashiro, and Shinsuke Yoshioka, *Evaluation of 3d markerless motion capture accuracy using openpose with multiple video cameras*, Frontiers in sports and active living **2** (2020), 50.

[21] Christian Ott, Dongheui Lee, and Yoshihiko Nakamura, *Motion capture based human motion recognition and imitation by direct marker control*, Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots, IEEE, 2008, pp. 399–405.

[22] Yongsheng Ou, Jianbing Hu, Zhiyang Wang, Yiqun Fu, Xinyu Wu, and Xiaoyun Li, *A real-time human imitation system using kinect*, International Journal of Social Robotics **7** (2015), no. 5, 587–600.

[23] Lorenzo Peppoloni, Alessandro Filippeschi, Emanuele Ruffaldi, and Carlo Alberto Avizzano, *A novel 7 degrees of freedom model for upper limb kinematic reconstruction based on wearable sensors*, 2013 IEEE 11th international symposium on intelligent systems and informatics (SISY), IEEE, 2013, pp. 105–110.

[24] Elisa Prati, Margherita Peruzzini, Marcello Pellicciari, and Roberto Raffaeli, *How to include user experience in the design of human-robot interaction*, Robotics and Computer-Integrated Manufacturing **68** (2021), 102072.

[25] Sarvenaz Salehi, Gabriele Bleser, Norbert Schmitz, and Didier Stricker, *A low-cost and light-weight motion tracking suit*, 2013 ieee 10th international conference on ubiquitous intelligence and computing and 2013 ieee 10th international conference on autonomic and trusted computing, IEEE, 2013, pp. 474–479.

[26] Stan Salvador and Philip Chan, *Toward accurate dynamic time warping in linear time and space*, Intelligent Data Analysis **11** (2007), no. 5, 561–580.

[27] Mark G Sobell, *A practical guide to ubuntu linux*, Pearson Education, 2015.

[28] Uti Solichah, Mauridhi Hery Purnomo, and Eko Mulyanto Yuniarno, *Marker-less motion capture based on openpose model using triangulation*, 2020 International Seminar on Intelligent Technology and Its Applications (ISITIA), IEEE, 2020, pp. 217–222.

[29] Da Sun, Andrey Kiselev, Qianfang Liao, Todor Stoyanov, and Amy Loutfi, *A new mixed-reality-based teleoperation system for telepresence and maneuverability enhancement*, IEEE Transactions on Human-Machine Systems **50** (2020), no. 1, 55–67.

[30] Tasbolat Taunyazov, Bukeikhan Omarali, and Almas Shintemirov, *A novel low-cost 4-dof wireless human arm motion tracker*, 2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob), IEEE, 2016, pp. 157–162.

[31] Anand Thobbi and Weihua Sheng, *Imitation learning of arm gestures in presence of missing data for humanoid robots*, 2010 10th IEEE-RAS International Conference on Humanoid Robots, IEEE, 2010, pp. 92–97.

[32] Jing Tian, Chengzhang Qu, Wenyuan Xu, and Song Wang, *Kinwrite: Handwriting-based authentication using kinect.*, NDSS, vol. 93, Citeseer, 2013, p. 94.

[33] Ultrleap, *Leap motion controller*, 2022.

[34] Wikipedia, *Telerobotics*, 2022.

[35] Xiaojun Zhao, Qiang Huang, Zhaoqin Peng, and Kejie Li, *Kinematics mapping and similarity evaluation of humanoid motion based on human motion capture*, 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), vol. 1, IEEE, 2004, pp. 840–845.

VITA

JORDAN RALPH FOGG

Candidate for the Degree of

Master of Science

Thesis: INTERACTIVE REMOTE ROBOTIC ARM CONTROL WITH HAND MOTIONS

Major Field: Engineering Technology

Biographical:

Education:

Completed the requirements for the Master of Science in Engineering Technology at Oklahoma State University, Stillwater, Oklahoma in May, 2023.

Completed the requirements for the Bachelor of Science in Engineering Technology at Oklahoma State University, Stillwater, Oklahoma in 2021.

Publications:

Jordan Fogg, Zeyu Deng, Emory Meursing, Long Huang, Huaxia Wang, and Chen Wang. 2023. "Remote Robot Control with Low-cost Robotic Arms and Human Motions." *In Adjunct Proceedings of the 2022 ACM International Joint Conference on Pervasive and Ubiquitous Computing and the 2022 ACM International Symposium on Wearable Computers* (UbiComp/ISWC '22 Adjunct). Association for Computing Machinery, New York, NY, USA, 32–34. https://doi.org/10.1145/3544793.3560331

Emory Meursing, Zeyu Deng, Jordan Fogg, Long Huang, Huaxia Wang, and Chen Wang. 2023. "Demo: I Am a Robot: First-person Robotic Arm Control with Hand Motions in Virtual Reality." *In Adjunct Proceedings of the 2022 ACM International Joint Conference on Pervasive and Ubiquitous Computing and the 2022 ACM International Symposium on Wearable Computers* (UbiComp/ISWC '22 Adjunct). Association for Computing Machinery, New York, NY, USA, 85–87. https://doi.org/10.1145/3544793.3560328