ROBOT ARM MANIPULATION USING DEPTH-SENSING CAMERA AND INVERSE KINEMATICS

By

© Akhilesh Kumar Mishra

A Thesis submitted to the

School of Graduate Studies

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Memorial University of Newfoundland

January 2015

St. John's Newfoundland

ABSTRACT

Robotic arms can be controlled by human operators using different types of controllers or manipulators. For example, a Titan IV robot arm can be mounted on a ROUV (remotely operated underwater vehicle) for seafloor operation and can then be remotely controlled by a sophisticated manipulator, a "Master Controller". An operator needs a lot of training with this type of controller before they could apply their skills manipulating real robot arms in the field. There are, however, simulators like GRI Simulations Inc.'s manipulator trainer which help a user train virtually on a particular robotic arm using either a master controller or a joystick. Compared to joysticks, master controllers are much more sophisticated and expensive devices. On the other hand, joysticks are not as convenient as controller mechanisms, since they are more generic products and do not map the functionalities of master controllers as well as the custom master controllers.

This thesis presents a new technique to manipulate a robotic arm which uses an inexpensive depth-camera to capture the user input and inverse kinematics to define the motion of the robotic arm. Along with the easier manipulation of the robotic arm, the presented technique also adds some gesture commands to control the end-effector which makes the interaction more intuitive. To test the efficacy and efficiency of the proposed method, a user study was conducted in which 18 participants were asked to perform two placement tasks using a keyboard, a joystick and the depth-camera based interface. The presented technique is inexpensive and the results of the study suggest the technique is a good option for controlling robot arms with configurations similar to that of the Titan IV.

ACKNOWLEDGEMENTS

First off, I would like to acknowledge and thank my supervisor Dr. Oscar Meruvia-Pastor for his help and support during my masters here in Memorial University. The research would not have been possible without his insightful advice and continuous support. I would also like to thank Dr. Lourdes Pena-Castillo for her help with statistical analysis of the data gathered from the user study. I sincerely thank her for her time and support.

I would also like to thank GRI simulations Inc. for providing their simulator to us for using in this research. I thank them for letting us visit their office in St. John's NL and for their thorough help in developing an interface for their manipulator trainer which we could use with our depth-camera.

Finally, I would like to thank my family and friends for their unconditional support and encouragement throughout my masters.

Parts of this work were presented as a research article at the OCEANS 2014 conference held at St. Johns, NL under the category "Remotely operated vehicles-II" [112]. Posters based on this research were presented at the AI/GI/CRV conference held in University of Montreal in 2014, where the poster was awarded as the best research poster in HCI category [114] and at the Nova Scotia Energy R&D Conference 2014 under the category "Seabed Engineering" [113].

Table of Contents

List of Tables

List of Figures

List of Appendices

Chapter 1

Introduction

In recent years there have been significant developments in the field of HCI, where the focus of the research has been on providing a natural, easy and intuitive means for humans to manipulate or control robots. There has been a significant increase in the usage of robotic arms that require human manipulation, in industrial, medical and offshore applications. One of the earliest manipulation problems studied in the field of robotics was the insertion of a peg into a hole using a robotic arm while preventing the wedging or jamming of the peg in 1982 [1]. Since then there has been a significant advancements in manipulation techniques and robot control. Robotic arms can be controlled by human operators using different types of controllers or manipulators. For example, a Titan IV robot (Fig. 1a) arm, which can be mounted on a ROUV (remotely operated underwater vehicle) for seafloor operation, can be remotely controlled by a sophisticated manipulator, a "Master Controller" (Fig. 1b). An operator needs a lot of training with this type of controller before they could apply their skills manipulating real robot arms in the field. There are, however, simulators like GRI Simulations Inc.'s[2] manipulator trainer which help a user train virtually on a particular robotic arm using either a master controller or a joystick. Compared to joysticks, master controllers are much more sophisticated and expensive devices. On the other hand, joysticks are not as convenient as controller mechanisms, since they are more generic products designed for gaming, and do not map the functionalities of master controllers as well as the custom master controllers.

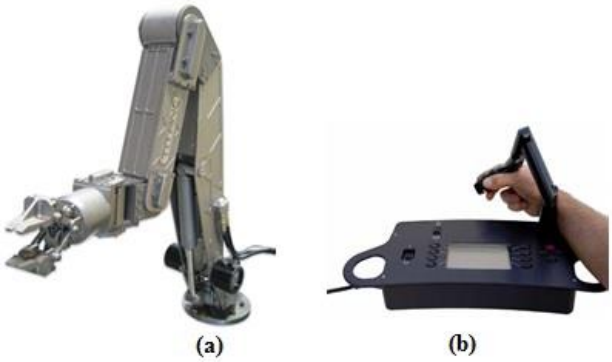

Figure 1(a) Titan IV robotic arm by FMC technologies (b) Master controller for Titan IV robotic arms. Image courtesy: fmctechnologies.com

1.1 Problem Statement

Over the last two decades there has been a significant development in robot control strategies. Most of the robot control mechanisms use a master controller (such as Titan IV's master controller in Fig. 1b) to control the robot which might be hard to learn for the operators and could also be expensive (thousands of dollars). The existing control mechanisms which use the master controller are expensive and require a lot of training. To counter this, an easy to learn and a cost effective control mechanism needs to be developed.

1.2 Motivation

As discussed in section 1.1, the aim of this research is to make an easy to learn and cost effective control method for robot arms. Since the advent of depth-cameras like Microsoft's Kinect [3] or Intel's Creative™[4] gesture camera, several applications have been developed where users interact with applications using gestures and speech

commands. However, most of these applications are related to gaming or simulations which are used for training purposes. To solve the problem mentioned in section 1.1, we decided to use the recently developed depth-sensing technology to make an easy to use interface which will help operators control the robot with much less training as compared to the conventional approaches.

1.3 Approach

Depth-cameras provide some essential information about the user such as the hand position, which can be used to control the robot. This thesis presents a new technique to manipulate a robotic arm in which an operator uses an inexpensive depth-camera to capture the user input and inverse kinematics to define the motion of the robotic arm. To test this technique, an articulated arm (Fig. 6) type robot simulator was developed in OpenGL. This simulator can be controlled by a keyboard, a joystick, and a depth-camera. For the keyboard and the depth-camera based input, the user was expected to control a target ball to point to an object which needs to be picked up. The inverse kinematics algorithm takes care of the rotation of the joints of the arm. However, for the joystick interface the user was expected to control the angular rotation of each joint manually using forward kinematics. Along with the easier manipulation of the robotic arm, the presented technique also adds some gesture commands to control the end-effector which makes the interaction more intuitive.

1.4 Research Questions

The objective of this research was to develop an easy and intuitive interface to control a robotic arm and compare it to the existing controllers such as keyboard control or joystick control, to answer the following research questions.

1. **How will the presented approach of using a depth–camera to control a robotic arm perform compared to a current approach?**

   As the depth-camera based input uses inverse kinematics to calculate the rotation angles and the joystick based control uses the forward kinematics in which the users control the rotation angles on its own, these two approaches of interaction, one is automatic (uses inverse kinematics) and second is manual (forward kinematics), are different and therefore, these two approaches need to be compared.

2. **Which approach would be harder to learn for the users?**

   The depth-camera based input method is new to the users as compared to the keyboard and the joystick based input methods. We need to find out whether the users will take a longer time to learn a new device as compared to the older, known devices.

3. **What will be the user preference for an input method?**

   As each input method has its pros and cons, we wanted to find out which input device would be preferred by the users. As the depth-camera was the newest

device for users, it was necessary to find out whether the users will prefer a new method and device or a more commonly used input method like joystick or a keyboard.

4. **Would there be a preferred input method for a particular task?**

   Two tasks were created which each user was supposed to perform using each input method. The objective of the first task was to reach to the randomly generated object and pick it. The objective of the second task was to pick the randomly generated object and drop it into the bin. Since the two tasks are different and require a different level of user control, we wanted to know if the users will prefer a particular input method for Task 1 and a different input method for Task 2.

1.5 User Study

A user study was necessary to test the efficacy and efficiency of the presented approach. Also, to answer the research questions mentioned above, a user study was important. We recruited 18 participants for the user study and each user was asked to perform two placement tasks using each input interface (keyboard, joystick and depth-camera). Each task was repeated 10 times. Statistical analysis was performed on the results which will be discussed in later part of the thesis.

1.6 Organization of the Thesis

The remainder of this thesis is organized as follows: An overview of the related work in the field of robot manipulation is provided in the next chapter, Chapter 2. Chapter 3 is the system overview and it explains the implemented system in detail. It also talks about the alternate approaches we tried to control the robot, using the depth camera, which didn't work, and Chapter 4 presents the details of the user study and the results from the user study. It also talks about the users' overview of the system as a whole. Chapter 5 talks about the conclusions obtained from this research and the user study. It also talks about future work related to this research as well.

Chapter 2

Related Work

This section briefly discusses similar work in robot manipulation and control techniques. One of the earliest manipulation problems studied in the field of robotics in 1982 was the insertion of a peg[1] into a hole using a robotic arm while preventing the wedging or jamming of the peg. Since then, the complexities of manipulation tasks have increased and contemporary robots can perform complex manipulations tasks. To aid these manipulation tasks several control techniques for manipulating a robot arm have been proposed, involving variety of position-controlled manipulators [5, 6, 7] closed loop manipulator control [8], joystick-based controllers [9, 5], speech [10, 11] and gesture based controller [12, 11, 13, 14] and sensor based interfaces[15].

2.1 Position-Controlled Manipulators

Position controlled manipulation (PCM) is probably the most common control method in robotics in which the robot is either controlled in a joint space or in Cartesian space [16, 17, 18, 6]. Since the inception of robotics, and its usage in industrial applications such as assembly, packaging and loading, many position controlled manipulators have been proposed. A.M. Sharaf presented a fuzzy logic based position controller for a single link manipulator[19]. Fuzzy logic was introduced in 1965 and is a form of a many valued logic which means that there are more than two truth values. The most famous form of fuzzy logic is the three valued logic, in which the truth table contains “true”, “false” and

“unknown” values [20]. The fuzzy logic based controller[21] is motivated by the system uncertainties and variable load excursions. Sharaf’s experimental results indicated that the fuzzy logic position controller is an effective and robust controller. However its performance needs to be evaluated for a multi jointed, articulated robotic arm [21]. There are position and force controllers such as [22, 23, 24] in which a dynamic system is implemented to control both the position and force . In some position based control methods, a PID (proportional-integral-derivative) control [25] is used to control the position of a robot [26, 27, 28], J.Jafar presented a PID based position control for a 2-DOF (degrees of freedom) robotic finger [29] which had two joints and two links. The PID control loop was used to control the position. The PID control parameters were used to tune the performance of transient response, overshoot, among others. PID parameters helped reduce the noise and vibrations from the mechanical part of robot. The study concluded that the PID control method improves the performance of the robotic finger for object manipulation. However, PID control systems are expensive and require special training to perform object manipulation. Also, it was not designed to perform the complex manipulation tasks which an articulated arm can perform.

Position controlled methods are also being used in controlling more than one manipulator. H. Carroll provided an adaptive position controlled method of dual-arm manipulators [17] in which two robot arms cooperate with each other in performing a task. This is a unique method as two robots communicate with each other to transfer a load from a point to a target position. Although controlling more than one manipulator is a relatively new topic, significant research has been done in that area such as the works of

Tarn T.J and X Yun in 1986[30] and C.Alford [31] in 1984. This could be an important addition to the presented research, whereby the operators of the robotic arm can manipulate two or more robotic arms together, to perform complex manipulation tasks. One of the industrial position-controlled manipulators such as Titan IV (Fig. 1a) are operated by a master controller (Fig. 1b), are used for training of remotely operated underwater vehicles (ROUVs) and are quite expensive [33, 2]. Also, for people who suffer from motor impairment, joystick or position based controller may not be a suitable method as it requires quite precise movements of the hand. However, the technique presented in this thesis provides an easy interaction method in which the operator just needs to point towards the target position and the robot rotates on its own to reach the target.

2.2 Voice Based Manipulation

There are other methods to control a robotic arm in which the users can interact with the robotic arm using speech, such as the *VoiceBot* presented by Brandi. H and Jonathan. M [11]. It uses the non-verbal voice such “ck” and “ch” sounds to control the gripper and it helps people with motor impairment. Similar to the *VoiceBot* there is another system which is known as the “*The Vocal Joystick*” presented by J. Blimes [33, 34] and is designed to help people with motor impairment to make use of voice commands to control objects generated on the computer screen and to control a robotic arm. Other non-verbal voice, such as humming-based control [35] and whistling based [36] control could be useful for people with some motor impairment, however these voice based solutions

are not best suited for longer durations, as after a while the system could become inconvenient to use.

2.3 Sensor Based Manipulation

In recent years, there has been a lot of development in sensor based robot control techniques [37, 38, 39, 40]. Sensor based control methods employ sensors to gather the state of the robot. Without the sensors, the robot end-effector would have to go through a path without any feedback, and that may cause limitations to the kind of tasks a robot can perform. Using sensors can give adaptability to the robots.[41, 38]. R. Das and A. Pandey presented a dual sensor based robot control system[37] in which a MEMS (Microelectromechanical systems) sensor [42] and an ultrasonic sensor were used to detect hand motion of the user and the gestures from the user. The use of these sensors provides a low-cost gesture recognition based robot as compared to the systems which use stereo cameras for feature extraction and gesture recognition[43, 44], however these sensors have a limited range and are not robust for commercial applications and they can only perform in a controlled environment. In another sensor based approach, Y. Song and C. Wusheng presented a sensor based control of a telerobotic system [40] in which multisensors are used to obtain the robot's position, velocity and force. This sensory information is then sent over LAN/internet for the human operator. The robot was also equipped with an intelligent control algorithm which based on the robot state, selects whether to operate by human control or local autonomous control. This method improves the accuracy of teleoperation and it can also be applied to some intelligent robot system, however it might not be suitable for under water applications as some of the sensors, such

as the infra-red sensors and the force/torque sensors, might behave differently under water (because of the buoyancy of water the force/torque sensors would have to be calibrated), therefore the performance of this system would have to be evaluated for underwater applications. S. Ma and X. Wu presented a sensor driven neural controller for a snake like robot which was self-adaptive and collision free [41]. Three IR (Infra-red) sensors were deployed to obtain the information about the obstacles in front of the snake locomotive. The neural network was designed after analyzing the motion of the snake locomotive which drives the CPG (central pattern generator) oscillators [41]. This approach however, is more suited for locomotive robots and its suitability for manipulating a robotic arm like a Titan IV arm will not be part of this research.

2.4 Vision & Stereo Camera-Based Manipulation

In vision based robot manipulation the users generally deploy a camera based sensor like Microsoft's Kinect[14] or other stereo or RGB camera to obtain the information about the environment. Since the inception of depth cameras many vision based control approaches have been researched [43-49]. One such method called *perception by proxy* [43], uses stereo cameras to obtain the information about the surroundings, which allows the operators of the robots to perform their tasks faster. It uses stereo cameras instead of depth cameras to obtain the accurate depth information about the obstacles in the environment. There are existing methods which use stereo cameras[49, 50, 51] or depth cameras like *Microsoft Kinect* to control a robot, for example H.B Suay's approach to control a humanoid using depth cameras [49]. This method uses depth images and skeletal tracking software to track the user movements and controls the humanoid robot

based on user input. This method is mostly suited for humanoids where the hand and leg gestures are mapped to the appropriate part of the humanoid; however for a robotic arm we need a different type of controller and an intuitive interface. There are other vision based approaches in which multiple cameras [52] or stereo cameras [53] are used to obtain information such as depth and position, of the objects used for pose correction and estimation, but these methods are effective only when the target object is small in size and the workspace of the end effector is large such as in assembly related industrial tasks [52, 54]. Y. M Zhao proposed a solution which utilizes multiple cameras and multiple target points to overcome this problem[54], the multiple cameras and the LEDs used help in improving the performance of the robot. Shirwalkar's approach in [110] provides a means for tele-manipulation of a robotic arm using a Kinect camera and hand gestures. The gestures were developed to open and close the grip of the end-effector and to map the hand movements to move the robot arm. The user controls the arm manually using a mapping method which maps the user's hand velocity to the robot movement. This approach is more suitable for techniques where haptic feedback from the arm is available; also this approach mainly focuses on controlling the gripper or the end-effector of the robot. A similar approach was also proposed by Raheja in [111] in which gestures were used to control a robotic arm. This approach uses a simple RGB camera to obtain the gesture information and matches it against the gestures stored in a database. This method provides a limited number of movements for the robotic arm which can be controlled by the gestures, which limits the workspace of the robot. In this research the depth camera can control the robotic arm in 3D as long as the constraints of the arm are met. The

above mentioned approaches use the stereo camera, or multiple cameras for feature extraction of the robot and obstacles. Based on the data obtained from the stereo or multiple cameras, a control strategy is devised to manipulate the robot. In the approach presented in this thesis the depth-camera acts as an interface between the robot arm and the operator and users interact with the robot arm using the depth-camera. The user specifies the target position using the depth-camera interface and the motion of the robot is defined by inverse-kinematics.

2.5 Robot arm configurations and Kinematics

The goal of this section is to provide the reader an insight into basic robot configurations and robot kinematics. This section provides an overview of the background needed to understand some aspects of the thesis, such as simulator design and control.

2.5.1 Robot arm configurations

One of the most common ways to classify robotic arms is by the arm configuration. There are five important arm configurations as explained below.

Cartesian coordinate robot: As the name suggests, Cartesian coordinate robots are rectilinear robots which move in the Cartesian coordinate frame along the X, Y, Z axis[55]. Figure 2 shows a basic structure of a Cartesian robot. Cartesian robots are commonly used for positioning applications such as cutting, dispensing and routing. The payload capability is dependent on the axis length and support structure [56].

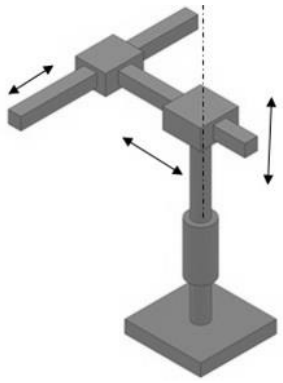

Figure 2: Basic Configuration of Cartesian coordinates Robot, image courtesy: nptel.ac.in [109]

Cylindrical Coordinate Robots: Unlike the Cartesian coordinate robots these robots have a rotational axis as well as two translational joints. Figure 3 shows the basic configuration of the cylindrical robot. As shown in the figure, the arm can move vertically, closer and farther and it can also rotate around its base which makes a cylindrical workspace for the robot [57, 58]. The vertical axis of the robot can rotate about 270 degree; because of its rigid structure the payload capability of cylindrical coordinate robots is higher than Cartesian coordinate robots[59].

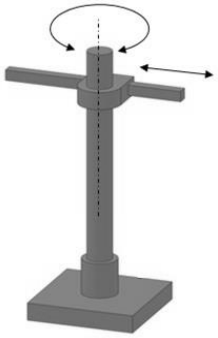

Figure 3: Basic configuration of cylindrical coordinate robot. Image courtesy: nptel.ac.in [109]

Spherical Coordinate robots: Spherical coordinate robots have two rotational joints and one linear joint. As shown in Figure 4 the robot can rotate around its base and around he Z axis, it can also move linearly. The robot's arm pivots can provide short rotary vertical strokes of about 60 degrees. The robot's base arm can swing horizontally around its base by about 210 degrees. Spherical robots are used in industrial tasks like material handling and welding .[60]

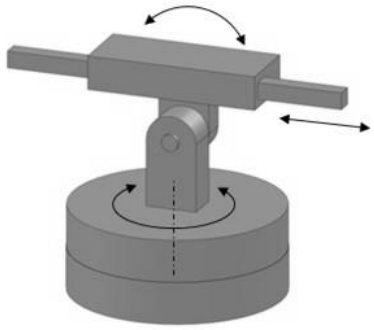

Figure 4: Basic configuration of spherical coordinate robot.

image courtesy: nptel.ac.in [109]

SCARA Robots: SCARA stands for Selective Compliance Assembly Robot Arm, SCARA robots have become very popular in the last decade in industrial applications such as painting, welding and packaging. [61, 62]. They are a combination of the cylindrical and the articulated robotic arm (discussed below) configurations. As shown in Figure 5, the arm has several revolute joints. SCARA robots have high speed and flexibility because of their precise angular controls.

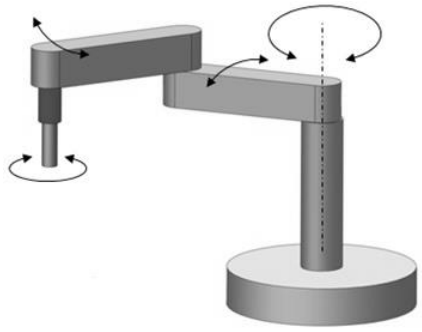

Figure 5: Basic configuration of a SCARA robot

image courtesy: nptel.ac.in [109]

Articulated or Jointed arm: The articulated or the jointed arm is the most commonly used arm configuration. These robots resemble human articulations design-wise, and have similar joint structures. As shown in Figure 6, the arm has number of revolute joints. θ_1 and θ_2 are connected with a rigid segment. Similar to the human arm $\theta_1$and θ_2 also have angular constraints. For this research we are developing a simulation of an articulated jointed arm. The main advantage with articulated robotic arms is that they minimize the floor space requirements. These arms are widely used for offshore applications such as sea floor excavation and sea bed engineering. As this research tries to provide a simpler control technique of Titan IV and similar types of articulated arms, we will focus on articulated arms for this research.[61]

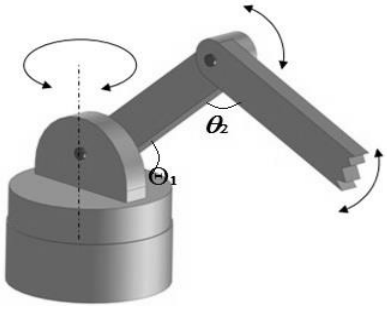

Figure 6: Basic configuration of articulated robot

image courtesy: nptel.ac.in [109]

2.5.2 Robot Kinematics

Kinematics is the study of motion of bodies or system of bodies without considering the cause of motion[63]. Robot kinematics is defined as the analytical study of the motion of the robot arm manipulator; it studies the relationship between the joints and the position of the kinematic chain of the robot. Robot kinematics can be divided into forward kinematics and inverse kinematics.

2.5.3 Forward Kinematics

In simple terms the forward kinematics problem can be stated as "given the joint angles, what will be the position of the end-effector (the gripper or the grabber mounted at the end of the robot)". Figure 7 shows a 2 link manipulator, where the forward kinematics problem is: given the link lengths l_1 and l_2 and the joint angles $\theta_1$andθ_2, what will be the position of the end-effector E (Fig. 7)?

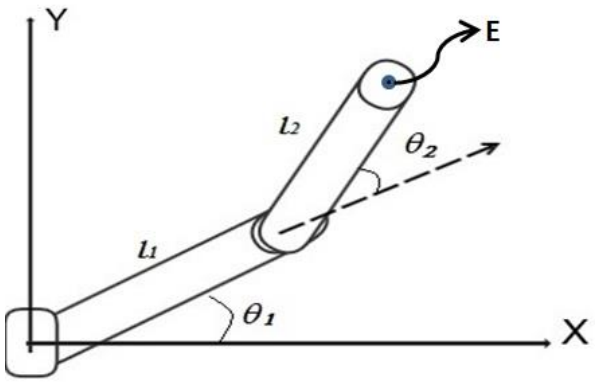

Figure 7: A simple two link manipulator.

This problem can be solved by two methods. The first method is the geometric approach which uses simple trigonometry to calculate the end-effector position. For the manipulator shown in Figure 7, let's assume that the end-effector is located at (X_{end}, Y_{end}) then X_{end}, and Y_{end} can be expressed in terms of l_1 and l_2 and the joint angles $\theta_1$and $\theta_2$as shown in equations 1 and 2.

$$Xend = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) \quad (1)$$

$$Yend = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) \quad (2)$$

The geometric approach works fine for 2 or 3 link manipulators, however, for a higher number of links the approach becomes tedious to formulate. There are however some algebraic methods to evaluate the forward kinematics which gives better and efficient solutions [64]. To understand these methods it is essential that we understand the simple matrix transformations associated with rigid body transformations such as rotation and translation.

2.5.4 Forward Kinematics Using Matrix transformations

Forward kinematics problem can also be solved using simple matrix operations. Figure 8 shows a 3 link chain with link lengths as l_1, l_2 and l_3. As shown in the Figure 8, each link has its own coordinate frame. E.g. link 1 is in the coordinate frame X_1 Y_1 similarly link 2, 3 also have their own coordinate system. As shown in Figure 8, the point P which is in the coordinate frame X_4 Y_4, is the end-effector of the whole arm and O is the origin at the base in X_0 Y_0 coordinate frame.

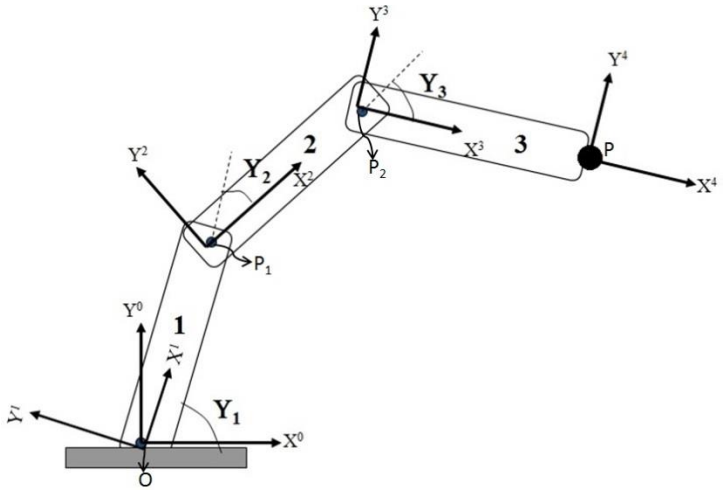

Figure 8: A simple 3 link manipulator

Using the matrix transformations we can express the position of the end-effector P based on the coordinate frame X_0 Y_0. For example, the end-effector of link 1, P_1, can be obtained from a rotation in frame X_0 Y_0. Similarly, the position of P_2 can also be obtained. The following equation gives us the relationship of the end-effector of the entire arm, P with respect to the base frame, X_0 Y_0.

$$Result = R_z(Y_1) \times T_{X_1}(l_1) \times R_z(Y_2) \times T_{X_2}(l_2) \times R_z(Y_3) \times T_{X_3}(l_3) \quad (3)$$

Where R_z means rotation along Z axis and T_x means translation along X axis. Rotation and the translation matrix are mentioned below in equation number (4) and (5) [65]. Equation 3 takes us to the coordinate frame X_4 Y_4, which is the end-effector P's local coordinate frame, to obtain the coordinates of P w.r.t to the base frame we need to multiply the *Result* matrix with P's coordinates as showed in equation number (6).

$$R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4)$$

$$T_x = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \quad (5)$$

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = [Result] \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (6)$$

2.5.5 Denvit and Hartenberg (DH) Parameters

The robot manipulators consist of several joints, these joints could be Cartesian controls (also called prismatic) (Figure2), revolute (Figure 5 and Figure 6) or spherical (Figure 4). To obtain the position of the end-effector based on these joints, it is important to know how each joint is connected to a previous joint and to the next joint.

There is a manual way to keep track of the coordinate transformation from joint to joint, however that is a recursive process and is mathematically tedious. Denvit and Hartenberg (DH) presented the DH parameters in 1955 [66, 67], which provides an easy presentation of joint connection and makes it easier to understand the coordinate transformation

moving from one joint to other. Normally, to represent a link correctly, 6 parameters are needed out of which 3 are for position and 3 are for rotation. Using the DH table these 6 variables can be converted into 4 linked parameters[67]. However, there are certain rules one must follow to represent DH coordinate frames. These rules are mentioned below.

2.5.6 Rules for Assigning Frames.

In the following discussion we will use the canonical axes X, Y and Z, as a set of three perpendicular vectors in 3D space.

Rule 1: Z_{i-1} is the axis of actuation (where the actuator is) of joint number *i*. In case of a revolute joint the axis Z_{i-1} becomes the axis of rotation, and if it is a prismatic joint, the Z_{i-1} is the axis for translation.

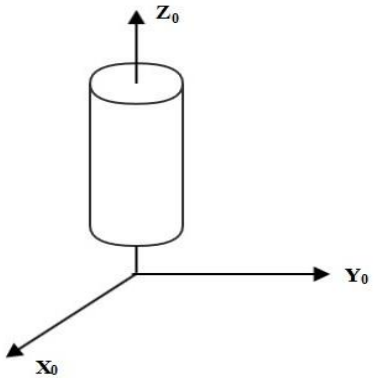

Figure 9: Axis of actuation is assigned to Z_0

Figure 9 shows a simple robot which rotates around its base, as the base is the axis of actuation, we assign Z_0 to the base. X_0 and Y_0 axis can be assigned as per user's choice. Consider a right hand rule where the thumb is Z_0 and X_0 and Y_0 could be assigned any of the index or the middle finger [66].

Rule 2: To apply the rule 2 to assign the coordinate frames, the following 3 cases need to be considered.

I. The axes Z_{i-1} and Z_i are not coplanar: If the axes Z_{i-1} and Z_i are not coplanar, then there exists only one line possible for X_i which will be perpendicular to both Z_{i-1} and Z_i. As shown in the Figure 10, the point of intersection where the line X_i intersects with Z_i is defined as the origin O_i. The axis Y_i is then chosen based on the right hand rule.[66, 59, 68]

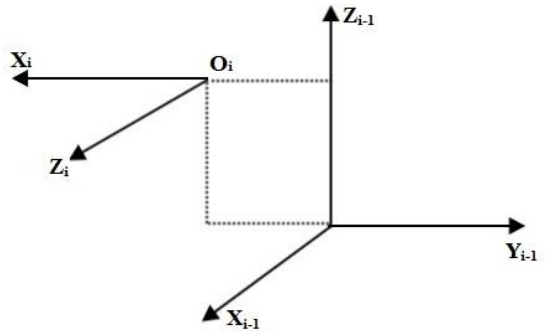

Figure 10: Axis X_i is assigned to the line perpendicular to Z_{i-1}

II. The axes Z_{i-1} and Z_i are parallel: Figure 11 shows a case where the axes Z_{i-1} and Z_i are parallel to each other there could be infinite number of possibilities for assigning X_i from Z_{i-1} and Z_i. In this case usually it is the best practice to choose X_i such that it will pass through O_{i-1}. The origin O_i is located at the intersection of X_i and Z_i.[69, 59]

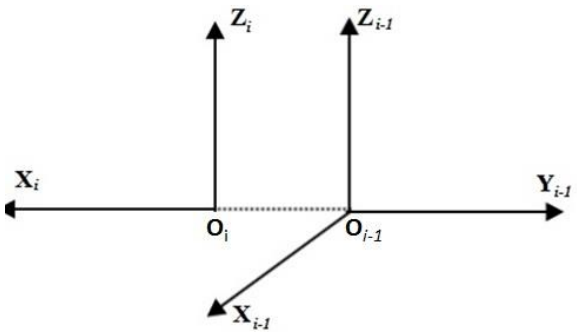

Figure 11: Axis X_i is assigned to the line which crosses O_{i-1} (or where O_i is parallel to O_{i-1})

III. The axes Z_{i-1} and Z_i intersect: If the axes Z_{i-1} and Z_i are intersecting with each other, then the X_i is assigned to a normal to the plane of Z_{i-1} and Z_i. The origin O_i can be anywhere on the axis Z_i. As shown in Figure 12, origin O_i is located at the intersection of X_i and Z_i [70, 66, 59].

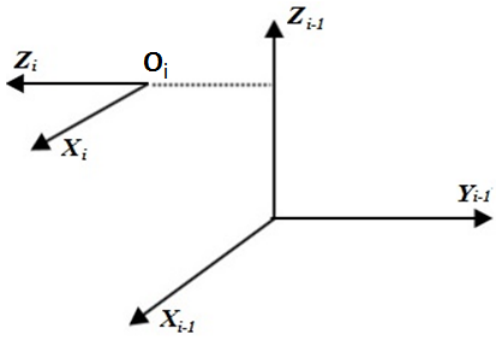

Figure 12: Axis X_i is assigned to the line which is normal to Z_{i-1} and Z_i

2.5.7 Finding the DH parameters

The four DH parameters are a_i, d_i, α_i and θ_i. Figure 13 shows the DH parameters.

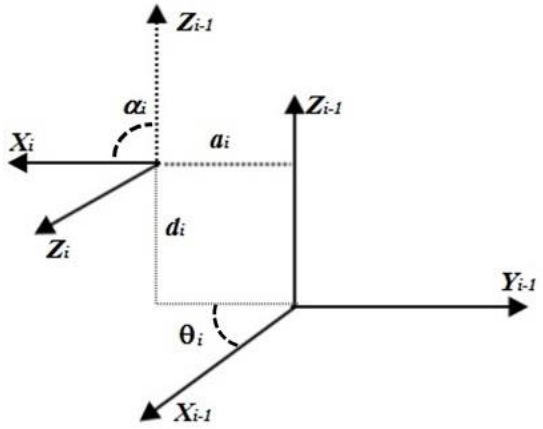

Figure 13: Example of DH parameters.

The parameter a_i is the distance between Z_{i-1} and Z_i measured along X_i. α_i is the angle between Z_{i-1} and Z_i measured about X_i. d_i is the distance between X_{i-1} and X_i measured along Z_{i-1}. θ_i is the angle between X_{i-1} and X_i measured about Z_{i-1}. After obtaining the DH parameters we can obtain the forward kinematics model using the homogenous transformations as shown in equation (7) [68], which can be used to obtain the position of the end-effector.

$$\boldsymbol{A_i = R_{z,\theta} \times T_{z,d} \times T_{x,a} \times R_{x,\alpha}} \quad \textbf{(7)}$$

2.5.8 Inverse Kinematics

Inverse kinematics problem can be stated as "given the target end-effector position what will be the joint parameters ?" [69]. Figure 14 shows a simple 2 link manipulator. To

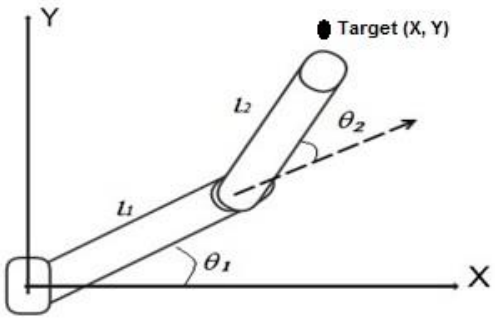

Figure 14: Simple 2 link manipulator

solve the inverse kinematics problem for the manipulator, shown in Figure 14, when the target position coordinates for the end-effector "Target(X, Y)" are given, $\theta_1$and θ_2 need to be calculated. Simplest way to obtain the joint angles $\theta_1$and θ_2 is to use trigonometry. Using the equations 1 and 2 we can see the relationship between the joint angles and the target positions. Using equations 1 and 2 we can write equations for target position (Tx, Ty)

$$\boldsymbol{Tx = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2)} \quad (8)$$

$$\boldsymbol{Ty = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2)} \quad (9)$$

After solving the simultaneous equations 8 and 9 for$\theta_1$and θ_2, we can write $\theta_1$and θ_2 in terms of target coordinates and the link lengths as given below.

$$\theta_2 = cos^{-1}\frac{({T_x}^2+{T_y}^2-{l_1}^2-{l_2}^2)}{(2l_1l_2)} \quad (10)$$

$$\theta_1 = \frac{(-T_x\, l_2\, \sin\theta_2 + T_y(l_1 + l_2\cos(\theta_2)))}{(-T_y\, l_2\, \sin\theta_2 + T_x(l_1 + l_2\cos(\theta_2)))} \quad (11)$$

This trigonometric solution is simple to understand and implement. However, for a higher number of links the formulation becomes tedious and more computationally expensive as

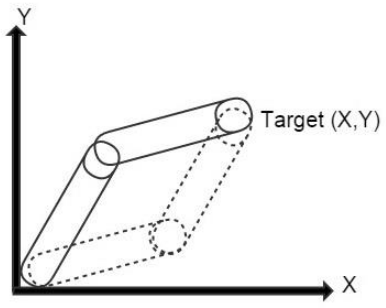

Figure 15: Multiple solutions for the same target position

it involves a lot of floating point calculations [71]. Along with the computational complexity there is another ambiguity in the trigonometric approach. For same target positions there could be multiple solutions which can lead to some erroneous state. For instance, Figure 15 shows an ambiguous state of the solution where there are two possible positions of the arm which lead to same target point, which might introduce computational errors (as the joint angles for both positions are different) and lead to vibration of the robotic arm around both solutions. Although the geometric solutions for forward and inverse kinematics are easy to understand and implement, they can get quite tedious and computationally expensive if there are more than 3 links. In addition, the geometric method would not work for a structure where the joints can move in different planes. For example, the joints in the arm in Fig. 15 rotate along Z axis but if the joints could also rotate along Y or X axis, the equations 10 and 11 would not be valid [72, 64] .

2.6 Inverse Kinematics Based Manipulation

Inverse kinematics is extensively used in manipulation techniques for humanoid [73–77] robots or animated robots [78]. T. Uzunovic presented a neural network based inverse kinematics method to control a Delta robot [51] in which the robot's inverse kinematics model was developed using the neural networks. It was stated that their method provides a significant improvement in mapping the task space coordinates to joint space coordinates, which improved the accuracy of the manipulation tasks. Similar neural network based inverse kinematics has also been proposed in [79,26, 80]. The advantage with neural network based inverse kinematics module is that it reduces the ambiguities (section 2.5.8) in the Inverse Kinematics (IK) solution and returns a solution which

doesn’t have the problems discussed in section 2.5.8 (Fig. 15). However, for this research CCD (Cyclic coordinate descent) algorithm [81, 82] was used since it is efficient to implement and is computationally inexpensive [71, 83]. Fedor in his comparative analysis in [84] reported that the CCD algorithm was a good compromise between speed and accuracy. Also, the simulator developed for this research has physical constraints, which shortens the solution space. The CCD algorithm returns a direction to rotate as well, which avoids the ambiguous solution problem mentioned in section 2.5.8 [71, 84, 85]. Y Kung presented a FPGA [86] implementation of Inverse kinematics and servo controller for a robot manipulator. Because of the FPGA implementation the computation time of inverse kinematics was really efficient. Also, digital control logic was implemented within the FPGA logic. Similar work has also been done in [87, 88, 89]. The FPGA implementation makes the computation faster, however the solution would be specific to a particular robot arm, as the arm configuration is hard coded and cannot be changed after deployment, so the control method might not be portable.

2.7 Brief Overview of Inverse Kinematics Algorithms

There are several algorithms for solving IK, coming originally from robotics applications. The most popular ones include CCD (Cyclic Coordinate Descent methods) [82, 90] pseudo-inverse methods [91], Jacobian transpose methods[91, 69, 92], damped-least squared methods [91, 93] and Triangulation methods [83]. The pseudo-inverse methods when used for robots with large number of links can be computationally demanding [84] as they produce large matrices to obtain the kinematic model of the system, and matrix operations consume a significant amount of processing power. Jacobian methods which

are more common than the pseudo-inverse methods, also require tedious matrix operations and hence are inefficient as compared to CCD [83, 90]. Jacobian transpose methods [91], overcome the drawbacks of Jacobian method by simply transposing the Jacobian matrices instead of finding its inverse or pseudo-inverse, however, Jacobian transpose methods lack in performance [94] . The triangulation method is as efficient as the CCD algorithm, it utilizes the properties of the triangles to accurately rotate the kinematic chain to make the end-effector closer to the target, and it makes a triangle between the end-effector and the target position using the length of the joint as the third point. Using the law of cosine it obtains the angle of rotation. The Triangulation method doesn't guarantee optimal rotations and tedious formulations might be needed for robots with a large number of links [83]. For this research we chose the CCD algorithm as it is easy to implement and is computationally inexpensive, as it does not involve calculating DH matrices, matrix inverse, transpose or partial derivatives which would involve a lot of floating point calculations.

2.8 Summary

This chapter discussed the various manipulation techniques, such as position control manipulation, sensor based manipulation, voice based manipulation and vision & stereo camera based manipulation. The position control techniques are based on user input, which means that a user controls the position of the robot. However, other discussed techniques employ sensor, voice and stereo images in conjunction with user input to control the robot. The presented research uses the depth-sensing camera as a user input device as compared to other vision based approaches discussed in section 2.4.

The chapter also provides and insight into robot kinematics in which forward and inverse kinematics was discussed. DH parameters and its rules for assigning coordinate axes were discussed. A brief overview of inverse kinematics algorithms was presented, out of which the CCD algorithm was chosen because of its simplicity and ease of implementation. CCD is efficient when compared to other algorithms, since it does not involve complex matrix and vector calculations.

Chapter 3

System Overview

The block diagram in Figure 16 shows the overall system. The system consists of four essential components. The user specifies the target position by moving his/her hands in front of a depth sensing camera. The user input processing module accepts the user input from the depth-camera, processes it, and passes it to the hand driven command module, where the processed user input is converted into 3D coordinates which can be passed to the inverse kinematics module as input. The inverse kinematics module calculates the joint angles for the robot arm simulator.

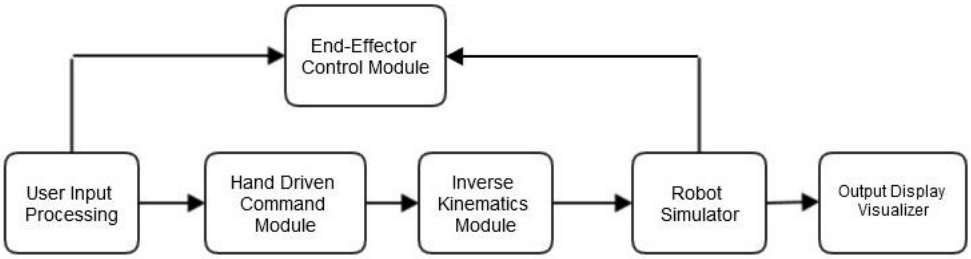

Figure 16: System Overview

After the joint angles have been calculated the robot arm simulator module applies the calculated rotations and the end-effector reaches the target. Once the target position has been reached the user can issue a gesture command to interact with the end-effector control module to open and close the grabber at the end-effector. The output display visualizer updates and displays the robot simulator along with other information such as

the depth-camera status, user's hand position, task iteration ID and task number. Each module has been discussed in detail below.

3.1 User Input Processing Module

The user input is captured using the depth sensing camera. For this research Intel's Creative depth-sensing camera was used to capture the user's hand motion. Intel's camera runs on Perceptual computing SDK (PCSDK) [4] which returns the user information such as hand information and the distance from the depth- camera. It returns the hand coordinates in Cartesian coordinates. Using the PCSDK, the right (or left) hand coordinates can be obtained. PCSDK also returns the coordinates of each finger's top, which is used to draw a skeletal view of the user's hand (as shown in Figure 19).

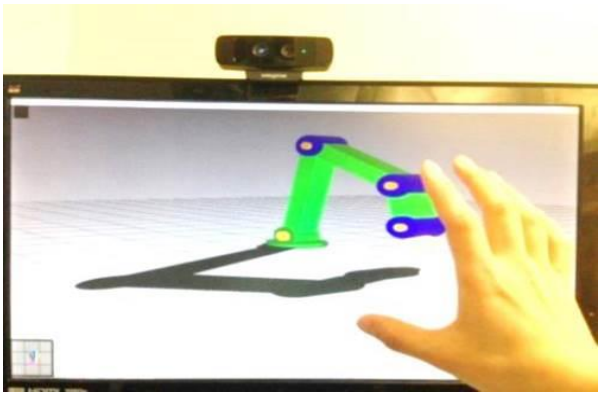

Figure 17: User interacting with the depth camera interface.

The creative depth sensing camera is designed for short ranges unlike Microsoft's Kinect which performs better for medium to long ranges [95]. For this research, a short range camera is more suitable, which is why Intel's creative depth camera was chosen. Figure 17 shows a user interacting with the depth camera. The camera is mounted on top of a desktop monitor and the user interacts with the robot arm simulator by moving his/her

hand in front of the camera. Users' hand motions are then processed and sent to the next module which is the hand driven command module.

3.2 Hand Driven Command Module.

Intel's Creative depth camera was used to get the data about the user's hand position. The depth-camera SDK [4] provides the hand position and its distance from the camera,

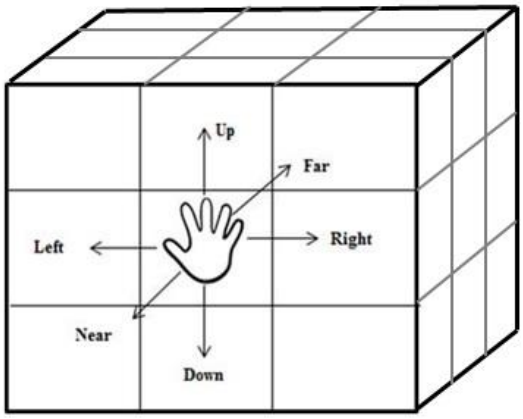

Figure 18: The 3D input space for user

which is used to control a target ball which can move in a 3D input space; this target ball specifies the target position for the end-effector of the robotic arm to go to. The camera field of view is mapped to the user's hand as shown in Figure 18 where the user controls the ball in 3 dimensions using his/ her hand. Placing the hand at the center of the grid stops the motion of the target ball. The grid cells Up, Down, Left, Right, Near and Far show the direction of motion. For example, if the hand is in the Up cell then the ball's Y coordinate will be incremented by 1 as long as the hand is in the Up grid cell. Similarly, if the hand is in Down grid cell then the ball's Y coordinate will be decremented by 1 for as long as the hand is in Down grid cell. The user interacts with X and Z coordinates in a

similar way. Figure 19 shows the various positions of the target ball with respect to the user's hand. The grid shown in Fig. 19 shows the hand skeleton of the user and the grid is the implementation of what is shown in Fig. 18.

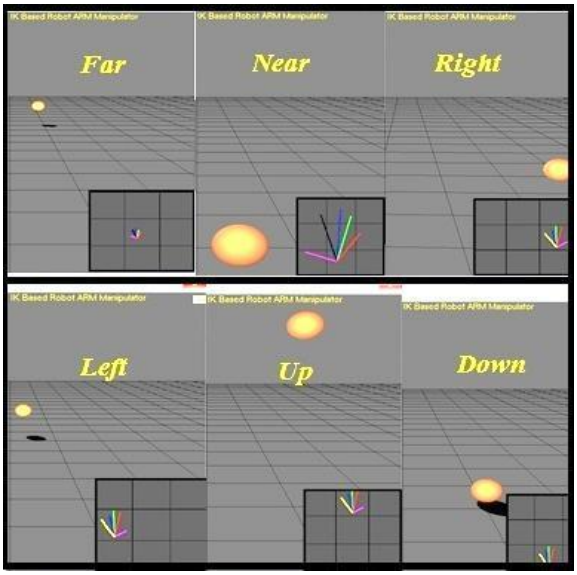

Figure 19: Ball position with respect to the user's hand.

3.2.1 Coordinate Processing

The position of the users' hand can be obtained from the depth-camera using the following code. The code shown below initializes the camera, makes an instance of the UtilPipeline class (defined in the C++ header file "util_pipeline.h" in the PCSDK), which gives us access to the gestures and alert functions defined in the class. The member functions OnGesture() and OnAlert() functions are built-in and return important information about the user. The OnGesture() function can loop the camera frames for built-in gestures. We use this function later to issue commands to the robot simulator to

pick and drop an object. The function OnAlert() provides information about an active node (Nodes are user's hand).

```
/* Step by step walk through of setting up the depth camera*/
Step 1: Include cameras headers.
Step 2: Declare variables to collect user's hand position
Step 3: Make a class which implements the UtilsPipeline class from SDK.
Step 4: Set up the OnGesture() and OnAlert() functions.
Step 5: Assign the gestures to the camera control, "Peace" for pick and "Thumbs_UP" for dropping an object.
Step 6: Declare global variables for camera tracking and active node tracking.
The following code snippet obtains the user's active hand information.
void handController()
{
pipeline.Init(); // Initialize the Camera
// acquire a frame to check for data
pipeline.AcquireFrame(true);
// Query the users hand position.
pipeline.QueryGesture()->QueryNodeData(Hand_data);
xCoordinate=Hand_data.positionImage.x;
yCoordinate=Hand_data.positionImage.y;
```

```
zCoordinate=Hand_data.positionWorld.y;}
```

As the depth-camera X, Y, Z coordinates at a rate of 35 samples per second, even a slight change in the hand position leads to unwanted movement of the target ball. To avoid the unnecessary flickering and displacement of the target ball, we decided to average out the last *n* coordinates, where n is a variable and can be kept between 5-10 to get smoother hand movement, so at any given instant the X, Y and Z coordinates would be the average of the last *n* coordinates.

The depth-sensing camera has a resolution of 320x240 which makes the returned xCoordinate anywhere from 0 to 320. Similarly, the returned yCoordinate is from 0 to 240. We inserted a buffer zone as shown in Figure 20, so the user can complete the interaction with the simulator without the risk of going out of the field of view of the camera. After adding a buffer zone in the field of view the *X* coordinate ranges from 10 to 290 and the *Y* coordinate ranges from 10 to 210. The *Z* value, zCoordinate, ranges from 0 to 1. As shown in Figure 18, the camera field of view is divided in a 3x3x3 cube or 27 cells. The central grid is for stopping any movements.

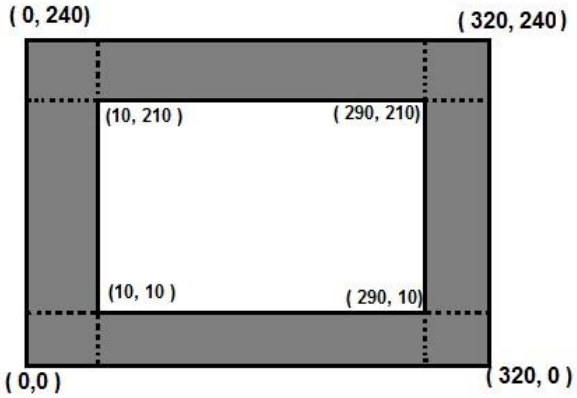

Figure 20: Camera field of view with the buffer zone.

The pseudo-code below shows the logic for the control function of the hand driven command module by which the user interacts with the simulator. The following pseudo-code explains how the 27 cells are formed and the target ball is controlled with each grid cell.

```
//Obtain the hand coordinate from handController() function.
xCoordinate=Hand_data.positionImage.x;
yCoordinate=Hand_data.positionImage.y;
zCoordinate=Hand_data.positionWorld.y;
Vect3D targetBall; // where targetBall is a 3D vector.
targetBall(0)// X Coordinate of the target ball
targetBall(1)// Y Coordinate of the target ball
targetBall(2)// Z Coordinate of the target ball
Function gridSetter():
// only one control is active at one time, hence, separate if and
// elif  statements are used
if (xCoordinate is between 0 to 90):
```

```
      targetBall(0)--;   // Move Left
elif (xCoordinate is between 200 to 290):
      targetBall(0)++;   // Move Right
elif (yCoordinate is between 0 to 70):
      targetBall(1)--;   // Move Up
elif (yCoordinate is between 140 to 210):
      targetBall(1)++;   // Move Down
elif (zCoordinate is between 0.1 to 0.3):
      targetBall(2)--;   // Move Far
elif (zCoordinate is between 0.6 to 1):
      targetBall(2)++;   // Move Near
elif (Camera tracking is On && hand node is detected):
      stopMovingBall= true;
```

The coordinates obtained from the hand controller module are sent to the inverse kinematics module where the joint angles for the robotic arm are calculated using the inverse kinematics algorithm.

3.3 Inverse Kinematics Module

This module is the core of the system and is responsible for applying the rotations to the joints of the robotic arm. As mentioned in section 2.5, the inverse kinematics problem can be stated as "given the target location, determine the joint angles"[59]. Figure 21 shows the robot arm simulator developed for this research. The figure also illustrates the target ball (orange ball) which specifies the target position, and joints 0, 1, 2 and 3. Joint 0 is the base of the robot and it can rotate 360^{o} around Y axis, which enables the robotic arm to

grab any object within its workspace. For this research, the CCD algorithm was used to solve the inverse kinematics problem. The CCD algorithm was chosen because of its computational efficiency[83, 96]. Also, the CCD algorithm is easier to implement as compared to Jacobian and pseudo-inverse methods, as it doesn't involve tedious matrix operations and floating point calculations [91, 94].

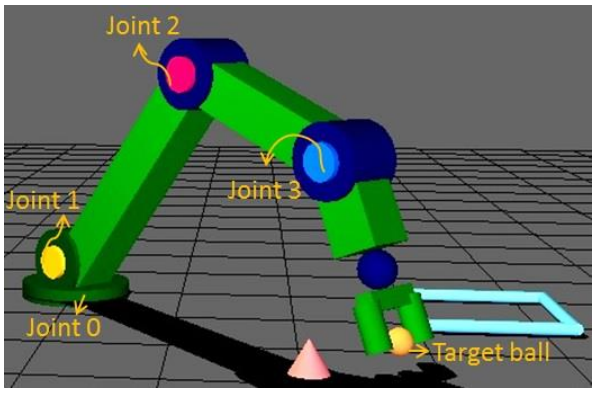

Figure 21: The simulator developed for this research. The base can rotate 360 degrees around Y axis.

3.3.1 Cyclic Coordinate descent (CCD)

CCD is the recursive algorithm we used. It solves the inverse kinematics problem through mathematical optimization. It loops from the last joint to root joint (the base joint in a kinematic chain). Each joint gets adjusted in a way that will bring end-effector as close to the target position as possible. As shown in Figure 22, the algorithm starts by measuring the difference between the two vectors formed between the effector position, E to C and from C to target position T. It then calculates the rotation and direction to reduce the angular distance between two vectors to zero (see Figure 22). It does this for each joint, iterating from the end-effector to the root joint of the kinematic chain. The rotation is

calculated by the dot product of two vectors and the direction is calculated by the cross product of two vectors defined in pseudo-code 1 for the CCD algorithm. To reach the target the equations 12 and 13, shown below, are solved for each joint until the distance between the end-effector and target is zero or the number of iterations has reached its limit.

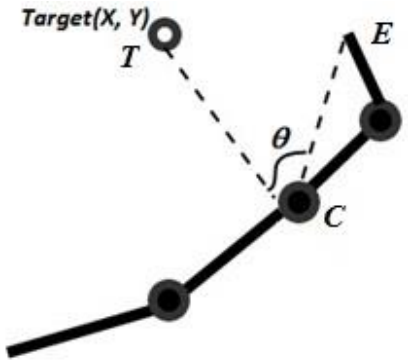

Figure 22: CCD algorithm, the vector formed by C, E rotates to make θ zero

$$\cos\theta = \frac{(E-C)}{(\|E-C\|)} \cdot \frac{(T-C)}{(\|T-C\|)} \tag{12}$$

$$\overrightarrow{\text{dir}} = \frac{(E-C)}{(\|E-C\|)} \times \frac{(T-C)}{(\|T-C\|)} \tag{13}$$

The basic pseudo-code of the 2D CCD algorithm is shown below:

Pseudo-code 1: CCD_2D_InverseKinematics

(Tx ,Ty) : Target coordinates

(Ex, Ey) : End Effector coordinates

(Cx_i, Cy_i) :Position of i-th link

Ti: Target vector for i-th link = ($Tx-Cx_i$, $Ty- Cy_i$)

Vi: End Effector vector for i-th link = ($Ex-Cx_i$, $Ey-Cy_i$)

θi: Angle between Ti and Vi

FOR i = n to 1:

// Where n is the number of joints.

Ti = ($Tx-Cx_i$, $Ty-Cy_i$)

Vi = ($Ex-Cx_i$, $Ey-Cy_i$)

θi = (Ti).(Vi)

dir = (Ti)x(Vi)

//Rotate ith link by *θi* in direction *dir* such that Vi aligns with Ti

END FOR

3.3.2 Obtaining a 3D solution using a 2D solution and a rotation

The algorithm mentioned above can solve the inverse kinematics problem for 2D. To obtain the 3D solution for the robotic arm, firstly, a 2D solution is obtained using the (X, Y) coordinates of the target ball, along the plane where the arm's joints are found then the baseAngle is calculated to find the rotation of the arm on the X-Z plane. The user specifies the target position *T'* on the plane#1 as shown in Figure 23. Plane #1 is the vertical plane that intersects the controller and passes through the axis of rotation of the arm. This target position is then transformed to the plane #2 (the arm's plane) to *T*. Using the Z coordinate of the target ball the *baseAngle* between plane #2 and plane #1 is

calculated. The solution is obtained on plane #2. The pseudo-code of the algorithm is mentioned below.

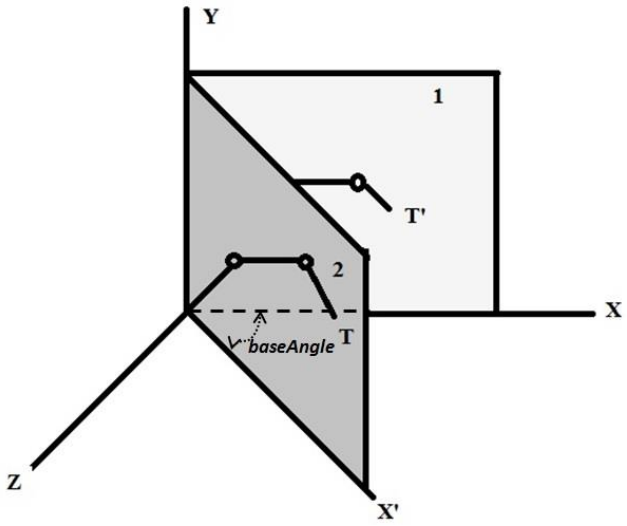

Figure 23: Obtaining a 3D solution using the 2D IK algorithm and a rotation

Pseudo-code for obtaining a 3D solution using the 2D IK solution:

```
Pseudo-code 2: 2Dto3D_IK algorithm.
Tx, Ty, Tz: Target ball's coordinates.
Ex, Ey, Ez: end-effector coordinates.
jointAngle[3]: Array containing joint angles corresponding to
each joint of the robot.
Tx = targetBall (0)
Ty = targetBall (1)
Tz = targetBall (2)
```

```
baseAngle = atan2(Tz/ Tx) // Angle between two planes.
// Obtain the target point on plane #2
T= Ry_baseAngle(T')  // Transform the target point.
                //Where Ry is rotation by baseAngle along Y axis.
jointAngle[3]= CCD_2D_InverseKinematics(T, E)
// calculate the joint angles using the pseudo-code 1
Rotate all the joint angles.
```

After the *jointAngles* are calculated using the algorithm mentioned in pseudo-code 1, the robot simulator applies these rotations and the end-effector reaches the target. The user can then issue a command to open or close the grabber.

3.4 End-Effector control module

The task that we are implementing with this IK system is a picking and dropping task, where objects are picked from the vicinity of the robot arm, and the user can drop them at different locations. This module uses the Intel SDK's image processing APIs [4] to let the user interact with the end-effector using gesture commands like "Thumbs up" or "Thumbs down" for pick and drop commands. The user is recommended to issue these gestures from the center part of the grid (Fig. 18), where no displacement is specified. Once the joint angles have been applied the user can interact with the End-effector and grab/ release the objects by hand gestures, keyboard input or joystick command. This module also checks for any possible collisions of end-effector and obstacles. Following is a basic outline of the collision detection algorithm.

Algorithm for collision detection:

Pseudo-code 3: Collision detection algorithm.

(Ex, Ey): Position of End-effector

(Ox,Oy) : Position of Obstacle

Re: Radius of Sphere surrounding End-effector

Ro: Radius of sphere surrounding obstacle

d: Distance between obstacle and end-effector

Calculate d;

d = $sqrt\left((E_x - O_x)^2 + \left(E_y - O_y\right)^2\right)$

IF d <= (Re + Ro) **THEN:**

Collision with Object

ELSE

No Collision

END IF

The above mentioned collision detection function is called along with CCD_2D_InverseKinematics() function to check if the end-effector made a contact with an object. There is just one object in the scene which is randomly generated with each iteration.

3.5 Interfaces

The user is provided with 3 types of input methods to control the robotic arm. The first is the option to control the target ball using a standard keyboard and camera movement using mice. The second option is to control the robotic arm with an Extreme 3D Pro(tm) joystick in which the user selects a joint and rotates it (this approach is based on forward

kinematics). The third option is where the user controls the target ball with his/her hand as described in section 3.2.

Keyboard interface + Inverse Kinematics: This enables the user to use the standard keyboard as an interacting device to perform manipulation of the target ball, while the IK module takes care of moving the arm's end effector towards the target ball. The user controls the target ball in 3D using the keys W, S, A, D, Z, X following the standard convention for keyboard interfaces for games (where the keys W and S control forward and backward movement and keys A and D control left and right movement of the player. We added keys Z and X control near and far movements). As shown in Figure 24A, the user controls the UP and Down movement of the target ball with the keys W and S, Right and Left movements with keys A and D, near and far movements with keys Z and X. Figure 24A shows the direction control for the target ball, whereas Figure 24B, shows the camera controls used to move the camera around the robot, KeyUP and KeyDown move the camera in a vertical direction, LeftKey and RightKey move the camera in a horizontal direction. The user can also move the camera across Z axis by using the PageUP and PageDown keys. The keyboard interface makes use of the Inverse Kinematics module to obtain the joint angles automatically based on the target ball movements.

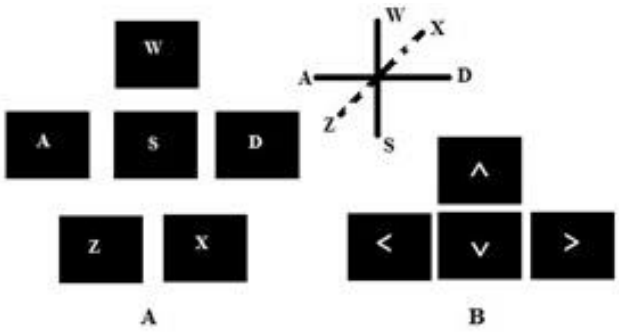

Figure 24 : (A) the directional keys (B) The camera control keys

The Joystick Interface + Forward Kinematics: Unlike the keyboard or the depth camera interface, the joystick module (Figure 25) uses forward kinematics, in which the user needs to control the robot by selecting a joint and then applying rotation. We chose forward kinematics for the joystick interface, because it is common practice when controlling most mechanical devices and simulators such as GRI Simlations Inc's manipulator trainer also use forward kinematics when using a joystick controller and we wanted to compare the inverse kinematics based approach with it, having it represent the baseline or control for the experiment. The direction X controls the rotation of the robot around its base around Y axis. The direction Y controls the rotation of the joints 1, 2 and 3. When the joystick is moved forward in Y direction (as shown in Figure 25) the rotations are closing the robot arm and when the joystick is moved towards the back the rotations are opening the robotic arm. The joystick interface was designed to follow the human finger analogy, which can be described as, when the root joint (joint 1 of the robot, see Figure 21) has reached its angular limit the next joint starts to rotate automatically. For instance, if a user selected joint number 1 and rotated it up to its angular limit, then the control will be transferred to joint number 2 and similarly when

joint number 2 has reached its limit, joint 3 gets selected automatically. This goes until all the limbs are stretched and the robot arm looks like a one straight inclined limb. If a user wants to control individual joints apart from the finger analogy, then the user needs to select a joint using the numbered keys 1 to 3 as shown in Figure 25, and then move the joystick across the Y direction (Figure 25) to obtain the desired rotation. The user can also pick and drop objects using the joystick button (4) as a toggle button.

Figure 25: Extreme 3D Joystick by Logitech™.

The Depth camera interface + Inverse Kinematics: As discussed in section 3.2, hand driven commands are used to move the target ball in 3D as shown in Figure 26. Figure 26 shows the hand skeleton in the right cell of the input space. The user moves the hand in up, down, right, left, near, far directions to control the target ball. The Inverse kinematics module calculates the angles based on the target position and applies it to the robot arm simulator, as previously explained in Section 3.3.

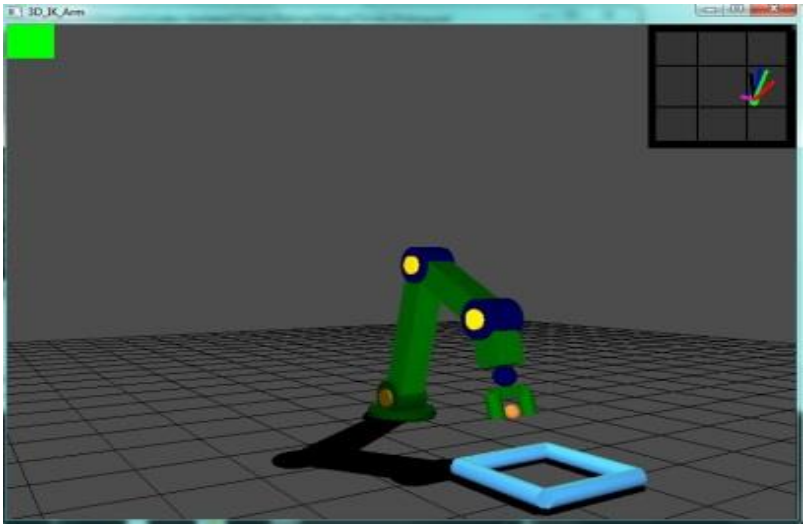

Figure 26: The user's hand skeleton view and the input space

3.6 Other Control Approaches.

Before finalizing the depth-camera interface mentioned above we tried the following methods to control the robot simulator. Initially, we wanted to design the interface in such a way that the target ball would follow the user's hand directions continuously instead of the grid based approach mentioned in section 3.2. To implement this method we need to solve two important problems. Firstly, as the depth-camera returns 35 frames per second, even a minute change in the hand position will displace the target ball, because of that the target ball would flicker, which causes vibrations in the robot arm. To avoid this vibration of the robotic arm we needed to discard the small changes in the hand position to get a smooth movement of the target ball. Secondly, to control the end-effector and to stop and start the motion of the target ball gesture commands were needed.

To solve the first problem, a method was implemented in which the hand data was filtered to remove any noise. Also a sliding window buffer was implemented and at any given instant the coordinates would be the average of the previous *n* coordinates. The size

of n could range from 1 to 10. As shown in Figure 27, the buffer stores *n* coordinates and keeps on popping the oldest coordinate out as soon as we get a new coordinate.

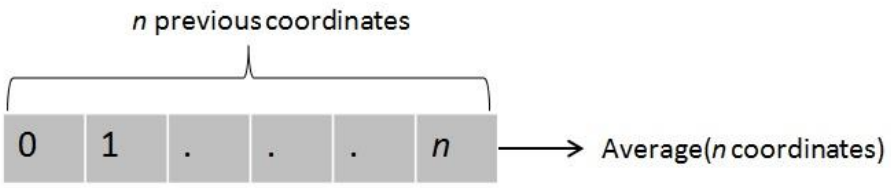

Figure 27 : Buffer implementation to average the past *n* coordinates

The buffer method improved the motion of the target ball, which resolved the vibration problem of the simulator. The next step was to design a gestural interface which can control the robot. The gestures for "Stop", "Start", "Left", "Right", "Pick" and "Drop" were needed to be implemented. To implement these gestures hidden Markov models were used[97][98] as described below.

3.6.1 Hidden Markov Models

A *Hidden Markov model* (HMM) is a statistical Markov model, and are the simplest case of a dynamic Bayesian network (DBN) [99]. HMM's are widely used for gesture recognition[100] and speech recognition[99]. A hidden Markov Model could be described as a collection of states that are connected by transitions from one state to another state with a unique transition probability. Each state in a HMM has two sets of probabilities associated with it: a transition probability and an emission or output probability.

A simple HMM can be described as shown in equation 14:

$$\lambda = (A, B, \pi) \qquad (14)$$

If N is the number of states and M is the number of distinct observation symbols per state, A is the NxN state transition probability distribution, represented by the matrix $A=\{a_{ij}\}$, B is the NxM observation symbol probability distribution and can be represented by the matrix $B=\{b_j(k)\}$ and π is the initial probability distribution.

There are three problems associated with HMMs:

a) Evaluation: The Evaluation problem is**,** given a HMM and a sequence of observations, evaluate the probability of the given output sequence. The *Forward* algorithm was used to solve this problem. The *Forward* algorithm [99] looks at the sequence of observations and calculates the probability that a sequence of states might have caused the observations.

b) Decoding: The second problem is, given an output sequence, calculate the probability of the most likely sequence of states. This problem can be easily solved by the famous Viterbi algorithm [101]. This is how we identify what gesture the user must have had made.

c) Learning:

HMM is used for many things, among them as a training based model, which means it needs training data to make predictions. To calculate the HMM parameters (A,B,π), the HMM needs to be trained using training data. For this research, 15 instances of a particular gesture were recorded and added to the gesture database. This process is scalable, which means if a new gesture needs to be added then, just the training data needs to be entered in gesture vocabulary. After obtaining the training data, Baum-Welch algorithm [102] was used to obtain the HMM parameters.

The HMM was trained on “Left”, “Right”, “Stop”, “Start” and “Circle” gesture commands. However, at the end HMM based gesture recognition was not used to control the robotic arm, different users draw gestures at a different speeds, which produces inaccurate results as the clustering algorithm (K-means) makes inconsistent clusters for the same gesture and because of that, the accuracy of gesture inference was low. Furthermore, controlling the entire robot arm with just gestures is a complex and unintuitive control interface for two reasons; first, it would need a lot of training for users to make the gestures correctly. Second, the users might not be able to make a gesture in time to control the robot. E.g. if the robot has been told to go left to reach to an object and once the end-effector has reached near the object the user fails to issue a “stop” gesture. This might lead to frustration and the user might not finish the task. Considering all the issues mentioned above, the control module discussed in sections 3.1 and 3.2 were implemented.

3.7 Summary

This chapter discussed the modules of the system presented in this research. The system consists of four important modules, which are the user input processing module, the hand driven command module, the inverse kinematics module and the robot simulator module. The user input processing module captures the user input from the depth-camera and passes it to the hand driven command module, where based on the user input the target ball coordinates are obtained. The target ball coordinates are passed to inverse kinematics module where the joint angles for the robot are calculated, which takes the robot to the target position. The robot simulator module applies the joint angles calculated through the

inverse kinematics module. The user can also control the end-effector of the robot to pick or drop an object.

The chapter also discusses the three interfaces which can control the robot simulator. The three devices are keyboard, joystick and the depth-camera. The keyboard and the depth-camera interface use inverse kinematics to control the robot, whereas the joystick interface uses forward kinematics. The approaches which were unsuccessful and he reasons behind it are also discussed.

Chapter 4

System Validation

4.1 Purpose

User studies are an important part of any user oriented research field such as human computer interaction, healthcare, medicine, or information systems. The researchers can verify their hypotheses and make comparisons between different systems by conducting a user study [103]. In the last few decades, several methods have been proposed for robot manipulation that make the human robot interaction easy and intuitive [6, 16, 17, 18, 20]; therefore it is an essential and standard practice to compare the proposed manipulation method in this research with existing manipulation techniques.

The design and execution of the user-centered evaluation of the presented manipulation technique is a difficult process because of the potential complexity of the tasks. Also, the users need to understand how to use or control the robot arm simulator in detail before they can proceed with the tasks. To compare the proposed manipulation technique with other commonly used manipulation techniques such as keyboard manipulation and joystick based manipulation, we designed two tasks. The complexity of the tasks was carefully chosen as the participants would have to perform the task in limited time in a controlled environment and without prior training.

4.2 Hypotheses

Based on the literature survey and understanding of the existing robot manipulation techniques the following hypotheses were proposed.

H 1: The keyboard based input method is the easiest to learn.

The keyboard based input uses a key mapping convention that is very similar to a gaming environment. We used the keys W, A, S, D, Z and X keys for 3D motion control of the target ball as these keys are widely used in gaming. We used the arrow keys for moving the camera around the scene. Considering many of the users may have had some experience using keyboard to either play a game or to perform computer-based tasks, we formulated the hypothesis that it will be the easiest input method to learn followed by the joystick and the depth-camera based input method.

H2: The keyboard and the depth-camera based input methods will be easier to use when compared to the joystick based input.

As the keyboard and the depth-camera based input methods are powered by inverse kinematics, we hypothesized that the keyboard and the depth-camera based input method will be easy to use as the user just has to control the target ball in a Cartesian coordinate based frame. On the other hand, the joystick based input method is driven by forward kinematics so the user has to perform the manipulation by rotating the joints of the robot.

H3: Participants will find that depth-camera based input method is not as ergonomic as the other two interfaces.

As the depth camera is placed in a third-person perspective, i.e. the camera is mounted on top of the desktop monitor, the user has to lift his/her hand in front of the camera when interacting with the robot arm simulator and that might cause discomfort to the user when the depth camera interface is used for longer duration.

H4: Participants will find that the depth-camera based input and the keyboard based input methods perform better as compared to the joystick for completing a task.

As the user has to control the joints of the robotic arm manually, the joystick will take more time to complete the task. Also, learning the joystick may take longer than the keyboard based input and the depth-camera based input, which adds to the task completion time as well. As mentioned in H2 the depth-camera and the keyboard based input are automated, meaning the user is not concerned about controlling the robot, the user just controls the target ball in 3D and the inverse kinematics algorithm takes care of applying the rotations to the robot arm simulator.

H5 Participants are more likely to prefer the keyboard or the joystick based input method as their overall preference. Based on the fact that the depth-camera is less ergonomic when compared to the keyboard and the joystick, we can expect that the participants will not prefer the depth-camera based input method for longer periods of time. In addition,

the depth camera is a relatively new interaction device for most people, so people might prefer to use devices they already use in everyday life.

H6: The users' will perform better with Keyboard as compared to the depth-camera and joystick both.

Since most of people are familiar with keyboards and perform well using keyboards in their daily tasks, it is expected that they would perform better with keyboard than with the other devices.

H7: Keyboard will be the most preferred input method.

Due to keyboards easy operability and the user friendliness, the participants will prefer the keyboard as the overall preferred input method.

H8: Participants will find that they perform better with the depth-camera based input after training and practice.

The depth-camera is the newest and least common way of interaction with a computer. It is very likely that for most of the participants the depth-camera will be a new input device. Therefore, we expect that there would be a learning effect with depth-camera and participants will get better with depth-camera interface after completing some iterations of the task.

H9: Participants will find the joystick most useful for completing a task.

Although we hypothesized that keyboard would be the most preferred input method because it is easy to learn and use. Participants might find a joystick more useful for the manipulation tasks. The reason for that could be that the joystick is a more common way of robot manipulation as compared to the keyboard and depth-camera.

H 10: Participants will find out that their performance improves as they complete more iterations of the same task across the input devices.

The participants will be given different order of inputs to complete the task. For instance, some users might begin with depth-camera then move to keyboard and end the experiment with joystick. We expect that the device which was used at the 3rd place will perform better as the user will be familiar with the task by then.

H 11: Participants will prefer the depth-camera interface over the joystick interface.

As the depth-camera interface works on inverse kinematics and is hypothesized to be easier to learn as compared to the joystick interface which works on forward kinematics, we expect that the users will prefer the depth-camera interface over the joystick interface.

4.3 Methodology

The user study was designed as a within subject [104] study to compare the three input methods (i.e. keyboard, joystick, and depth-camera). To keep the experiment unbiased and unadulterated for every participant, we asked each participant to start with a

randomly picked input method, with respect to input order, to perform a task, e.g. user A starts with keyboard then proceeds to depth-camera and ends with joystick whereas the user B starts with joystick then proceeds to depth-camera and ends with keyboard interface.

4.4 User Selection

Posters were put around the university to advertise the need for participants for this experiment. We received interested participants from engineering, science, medicine and bio chemistry fields. 18 users were recruited to perform this experiment. The users were divided in six groups, and users were assigned evenly to each of these groups. Each group follows a different order of input methods which the user uses to complete the task. Table 1 shows the 6 groups and the order of input methods. Each input method was assigned a number as follows.

Keyboard=1, Depth-Camera =2, Joystick= 3

Group	Input method order
1	1,2,3
2	1,3,2
3	2,1,3
4	2,3,1
5	3,1,2
6	3,2,1

Table 1: Group numbers and the input order

As a result, 3 participants were recruited for each group. The users were randomly put in a group. The table below shows the 18 users with randomized group number and order number.

No	Random_Group	Order_Input
1	1	1,2,3
2	6	3,2,1
3	5	3,1,2
4	3	2,1,3
5	4	2,3,1
6	2	1,3,2
7	5	3,1,2
8	3	2,1,3
9	6	3,2,1
10	4	2,3,1
11	1	1,2,3
12	2	1,3,2
13	3	2,1,3
14	2	1,3,2
15	4	2,3,1
16	6	3,2,1
17	5	3,1,2
18	1	1,2,3

Table 2: User ID, Group numbers and the input order.

The users IDs were assigned to the users as arrived and when they were recruited. Each user was asked to sign a consent form before proceeding to the experiment. The consent form had the basic introduction of the research topic and can be found in Appendix A.

4.5 Experimental Tasks

To validate the proposed robotic arm manipulation solution, two tasks were developed in which the user was required to interact with the robot arm simulator using each input method. The tasks required the user to have an understanding of how to control the simulator. Training and a live demonstration was provided for each input method prior to starting a task.

Task 1: This task was a selection/picking task, in this task the users were just expected to reach the object and grab it. Depending on which input method the user was using to interact with the simulator, there were different commands to pick/grab an object, which

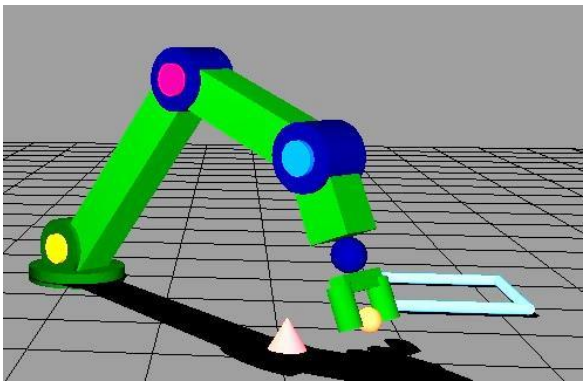

Figure 28: Task 1 for user study.

are explained in Chapter 3. As soon as the user starts the experiment and selects the task 1, an object (cone, sphere or cube) is randomly generated and placed on the floor and the user is expected to control the robot so as to grab the object. The system records the time from the instant the object was generated to the instant it was grabbed and writes this information in a text file. As soon as the user grabs the object, the iteration is completed and the robot is set to an initial configuration. There were 10 iterations in this task. At the start of each iteration an object is randomly generated within the reach of the robot.

Figure 28 shows the setup for Task 1. The user just needs to reach the cone and pick it and the task is completed.

Task 2: In this task the user is expected to grab the objects and drop them in a bin. As shown in Figure 29a the user grabs the object (green colored cube as shown in Figure

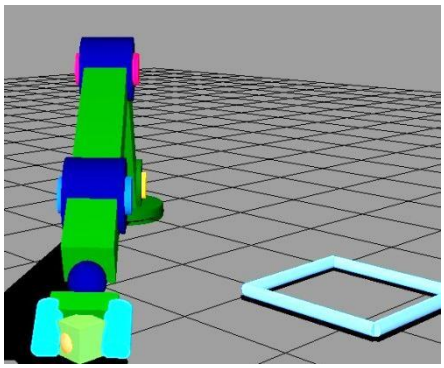

Figure 29a: Object is grabbed when end-effector is blue.

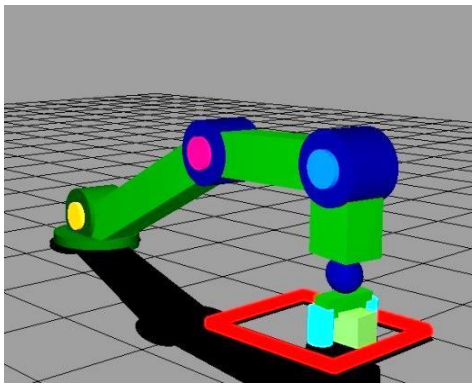

Figure 29b: Object can be dropped when the bin turns red.

29a) and places it in the bin. As soon as the user grabs the cube, the end-effector turns blue indicating an object has been grabbed, and then the user can move the arm towards the bin. If an object can be dropped into the bin, the color of the bin changes red as showed in Figure 29b. As soon as an object is dropped in the bin the robot is configured at its original resting position and a random object is generated again. There will be 10 repetitions of this task. The system notes the time since an object was generated, until it was dropped in the bin.

4.6 Results from the User's Feedback

After the experiment we gathered the data from all 18 users and performed some statistical analysis. After finishing the experiment each user was asked to answer a

questionnaire which had 17 questions in total with 1 open ended question. Appendix B presents the questionnaire in detail.

The following section presents the findings from the user's answers to the questionnaire. We will also discuss the answers to each question from the questionnaire in brief.

4.6.1 User Familiarity with Input Devices:

Questions 1 to 3 were designed to ask about the familiarity of the users with each input device. Figure 30 shows the familiarity of each user with the use of each input device to perform a manipulation task.

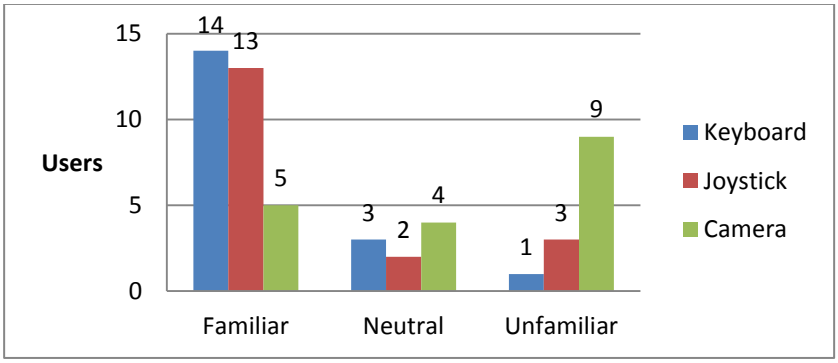

Figure 30: Familiarity of users with each input device.

Figure 30 shows that the users were most unfamiliar with the depth-camera and most familiar with joystick and keyboard. As shown in Figure 30, 14 (77.8%) participants said that they were familiar with keyboard and 13 (72.22%) participants said that they were familiar with joystick. The familiarity for keyboard and the joystick is almost equal, however, only 5 out of 18 (27.78%) participants said that they were familiar with the depth-camera and 9 participants (50%) said that they were unfamiliar with depth-camera, which indicates that participants were most unfamiliar with depth-camera.

4.6.2 Ease of Learning

Question 4 (Q4) in the questionnaire was "*Rate each input method for ease of learning. Was the operation of the input method easy to learn?"*

Figure 31 shows the user perception about ease of learning of each input method. 14 (77.78%) participants strongly agreed that the keyboard method was easy to learn, 12 (66.66%) participants strongly agreed that the joystick method was easy to learn and 15 (83.33%) participants strongly agreed that the depth-camera method was easy to learn.

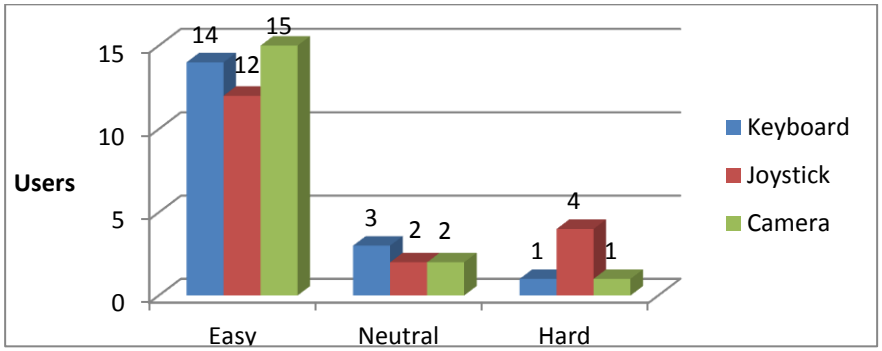

Figure 31: User responses to ease of learning.

From Figure 31 we can see that almost same number of users strongly agreed for each input method being easy to learn. However, 4 (22%) participants said that the joystick was hard to learn, whereas just 1 (5%) of the participants said that keyboard or depth-camera was hard to learn. It is interesting to note that users found the depth-sensing camera easy to learn, as it was least familiar device to users. The result presented above suggests that hypothesis H 1 which states that the keyboard is the easiest method to learn is not true. However, as shown in Figure 31, in general, users found all the input devices easy to learn.

4.6.3 User Perception about Performance.

Question 5 (Q5) in the questionnaire was "*Rate each input based on performance. Did the input perform appropriately?"*

The Figure 32 shown below shows the user perception on performance of each input device. We expected that the users will report that the keyboard performs *appropriately* the most followed by the depth-camera and then the joystick. However, as the Figure 32 shows 16 (88.89%) participants strongly agreed that the keyboard performs *appropriately* and 14 (77.78%) participants strongly agreed that joystick performs *appropriately*, whereas 17 (94.45%) out of 18 participants strongly agreed depth-camera performs *appropriately*.

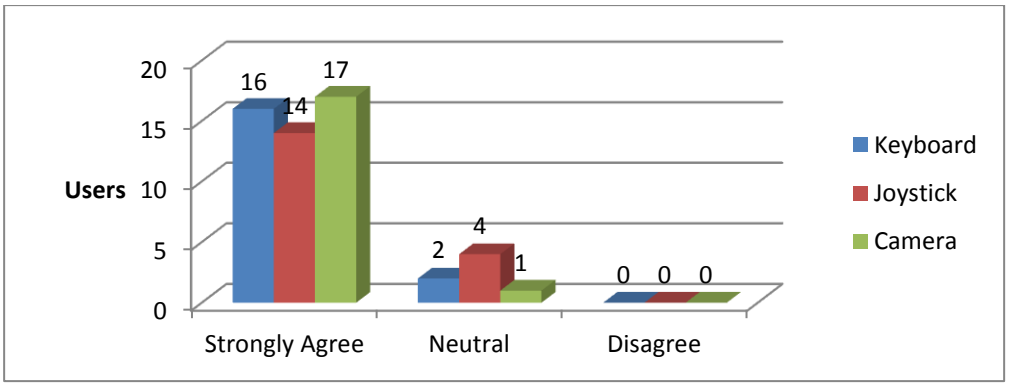

Figure 32: User perception of performance for each input device.

The overall impression of the users is that all methods performed appropriately.

4.6.4 Ease of Use

Question 6 (Q6) in the questionnaire was "*Rate each input based on ease of use. Was the input easy to use?"*

Figure 33 shows the users' response to Q6. We expected that the user will report the keyboard is the easiest to use and the depth camera will be hardest to use. From the results shown in Figure 33, we gathered that 4 (22.22%) participants perceived that joystick is slightly harder to use, whereas none of the participants said that the depth-camera was hard to use and just one user said that keyboard is hard to use. On the other hand 14 (77.78%) participants said that the keyboard is easy to use, 12 (66.66%) participants said joystick is easy to use and 17 (94.4%) participants said that they found depth-camera easy to use.

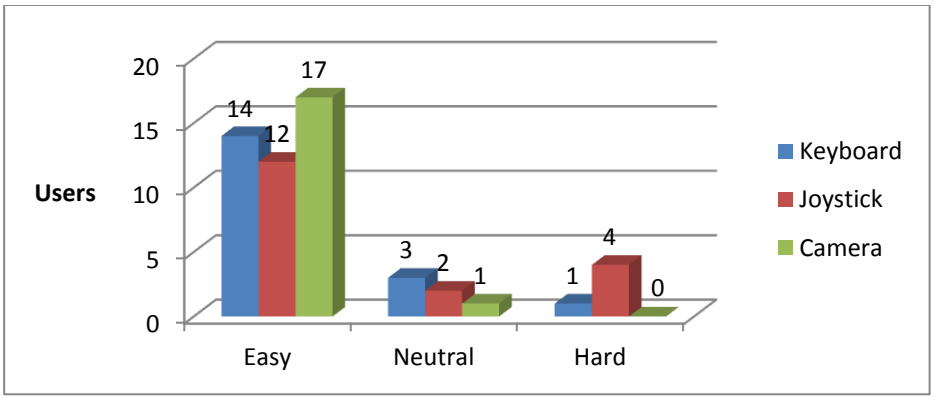

Figure 33: Ease of use for each input device.

This result tends to support our hypothesis H2, which stated that the users will find that depth-camera based input method and the keyboard based input method will be comparatively easier to use than the joystick, but again, this is only a trend that is not statistically significant.

4.6.5 Ergonomics

Question 7 (Q7) in the questionnaire was *"Rate each input for ergonomics. Was the input comfortable to operate?"*

We expected that the users will rate keyboard and the joystick higher than the depth-camera in ergonomics. The Figure 34 shows the user perception of ergonomics for each input. The keyboard and joystick was rated equal in ergonomics, 13 (72.22%) users strongly agreed that the keyboard and joystick are ergonomic interfaces.

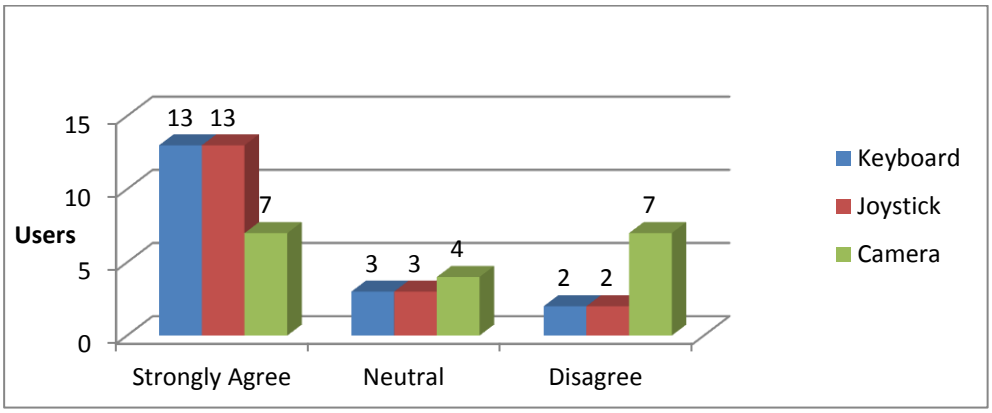

Figure 34: Ergonomics rating of each input device

The camera was rated lowest in ergonomics, and only 7 (38.89%) participants strongly agreed that the camera was ergonomic to use, and 7 (38.89%) participants disagreed that the camera was an ergonomic interface and 4 (22.22%) participants were neutral about this question. We believe that the reason for that was the camera position. The camera was mounted on top of the desktop monitor and the users had to lift their hand to control the target ball and to issue gesture commands to pick and drop and object. After using the depth-camera interface for a couple of iterations the hand starts to hurt. Many participants reported this. This result seems to confirm our hypothesis *H3* which stated that users will report that the depth-camera interface is not as ergonomic as the keyboard and the joystick interface.

4.6.6 Usefulness of an Input Method

Question 8 (Q8) in the questionnaire was *"Rate each input for its usefulness. Is the input useful for the given tasks?"*

This question asks the users about their perception of how useful each input device was for performing the given tasks. Figure 35 shows the user responses for Q8. 11 (61.11%) participants said that they found keyboard and joystick equally useful for completing both tasks. Whereas slightly more number of participants, 16 (88.88%), said that they found the depth-camera to be the most useful for completing both tasks. It is interesting to see that for depth-camera, slightly less number of participants (just 2 participants, as compared to 6 participants for keyboard or 5 participants for joystick), were either neutral or disagreed that it was useful for the given tasks. This result tends to reject our hypothesis H9 which states that users will find joystick to be the most useful for performing the tasks.

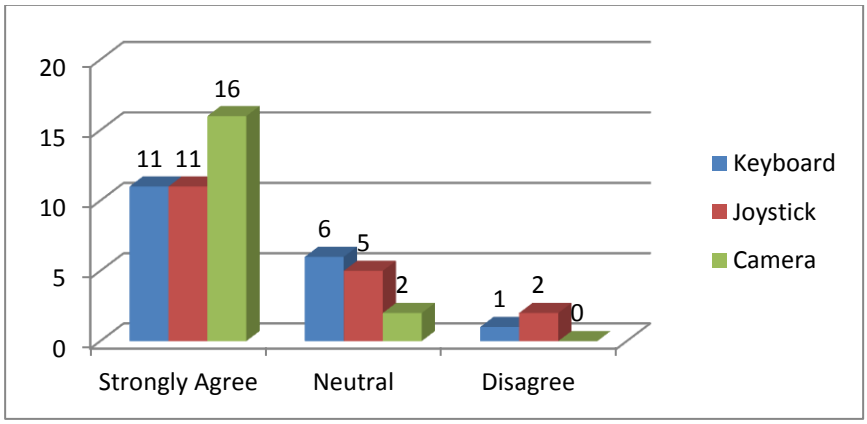

Figure 35: Usefulness rating of each input method

4.6.7 Input Preference for a Task

Questions 9 and 10 (Q9 and Q10) in the questionnaire were *"Which input type is best suited for Task 1/ Task2?*

Figure 36 shows the user responses for both Task 1 and Task 2. According to the user perception, the keyboard and the depth-camera was the most preferred input devices for Task 1 and Task 2 both. For Task 1, 7 (39%) participants strongly agreed that the keyboard is best suited input method for it, 8 (44%) participants strongly agreed that the depth-camera is the best suited whereas just 3 (17%) participants said that the joystick is the best suited input method. Similarly for Task 2, 7 (39%) participants said that the keyboard is the best suited input method and 9 (50%) participants reported that depth-camera is the best method for it, whereas just 2 (11%) participants were in favor of joystick.

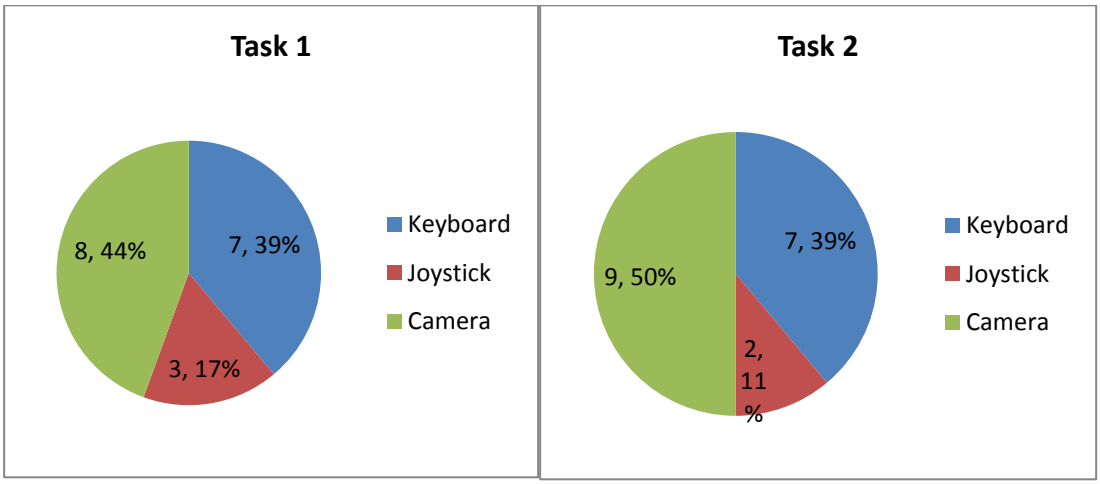

Figure 36: Input preference for each task.

From the results mentioned above, we can see that the user response for both Task 1 and Task 2 is very similar from which we can conclude that there is no preferred input device

for a particular task. Additionally, more participants perceived keyboard and the depth-camera as the preferred input method for both tasks as compared to joystick which supports the hypothesis H4.

4.6.8 Training Time for Input Devices

Question 11 (Q11) in the questionnaire was *"Which input method requires least amount of training?"*

The user responses are shown below in Figure 37. 9 participants (50%) believed that keyboard takes the least amount of time; the depth-camera was rated second in training time, 6 (33%) participants said that the depth-camera requires the least amount of training. The joystick was rated lowest as only 3 (17%) participants said that joystick

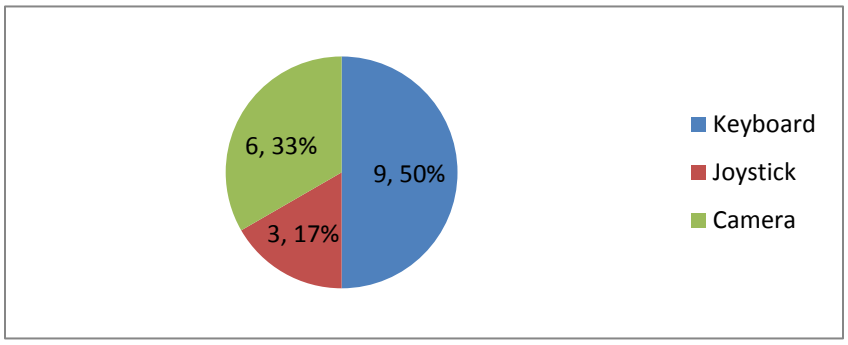

Figure 37: User responses for least amount of training time.

requires the least amount of training. The reason for that could be the manual control of the joystick, because the joystick was controlled using forward kinematics, and the user is expected to control the joint angles to reach the target position. From this result we can conclude that according to the participants the keyboard takes less time to train as compared to the joystick and the depth camera which supports the hypothesis H 1.

4.6.9 Perception of Performance

Question 12 (Q12) in the questionnaire was *"Which input method performs best after training & practice?"*

The user responses to Q12 are shown in Figure 38. 11 (61%) participants reported that the depth-camera performs best, 5 (28%) participants reported that keyboard performs best and only 2 (11%) participants chose the joystick as best performer after training and practice. This result tends to support our hypothesis H8, which states that the users will report that the depth camera performs better.

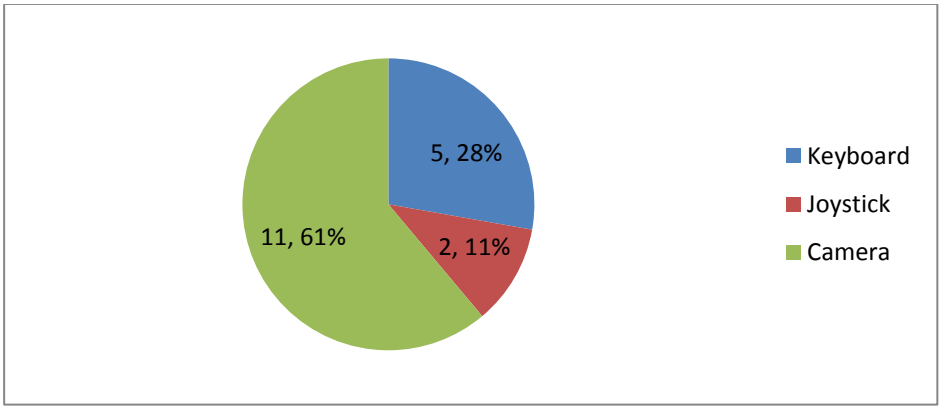

Figure 38: Users' perception of performance .

4.6.10 Input Preference

Question 13 (Q13) in the questionnaire was *"Would you prefer to use the depth camera interface over joystick? "*

Figure 39 shows the users responses to Q13. 12 (67%) participants strongly agreed that they would prefer the depth-camera interface over the joystick interface, only 2 (11%) participants strongly disagreed, whereas 4 (22%) were neutral about it.

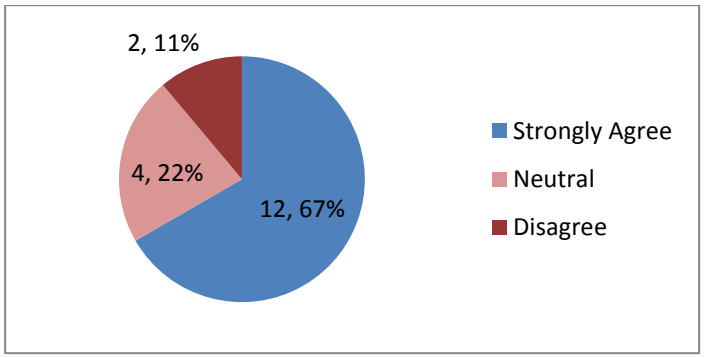

Figure 39: Users' response for selecting depth-camera over joystick

From the results we can conclude that most users preferred depth-camera interface over joystick which supports hypothesis H11.

Question 14 (Q14) in the questionnaire was *"Would you prefer to use the depth camera interface over keyboard? "*

Figure 40 shows the response to Q14. 11 (61%) participants strongly agreed that they would prefer the depth-camera interface over the keyboard interface; only 4 (22%) participants disagreed with it, whereas 3 (17) % participants had a neutral opinion about it. This result rejects our hypothesis H 7, which says that the keyboard will be the most preferred input method amongst all three. However as the sample size is small this result is not statistically significant.

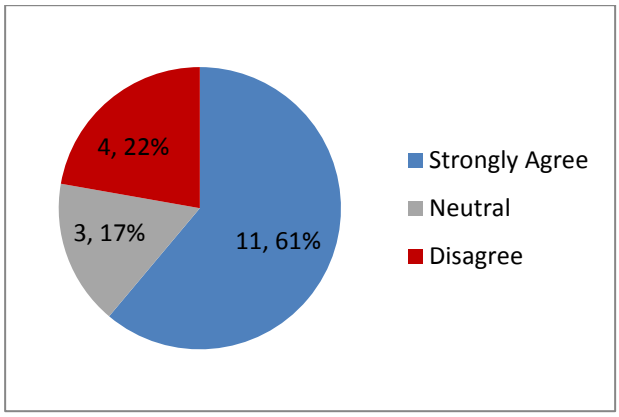

Figure 40: Users' response for selecting depth-camera over keyboard.

Question 15 (Q15) in the questionnaire was "*Would you prefer to use the joystick interface over keyboard?"*

Figure 41 shows the users response to Q15. 7 (39%) participants strongly agreed that they would prefer the joystick over the keyboard, 6 (33%) participants disagree with it and 5 (28%) participants had a neutral opinion about it. This result is inconclusive as almost same numbers of participants agreed, disagreed and were neutral about it, which rejects our hypothesis H 7. However as the sample size is small, this result is not statistically significant.

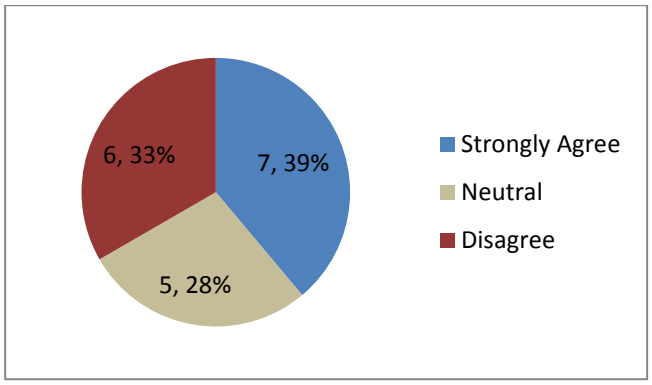

Figure 41: Users' response for selecting joystick over keyboard

4.6.11 Overall Input Preference

Question 16 (Q16) in the questionnaire was "*Which input type would you prefer overall?*" Figure 42 shows the users response to Q16. 9 (50%) participants responded that they would prefer the depth-camera based input method over keyboard and the joystick based input methods. The second most preferred option was the keyboard based input method, where 7 (39%) participants said that they preferred the keyboard above all.

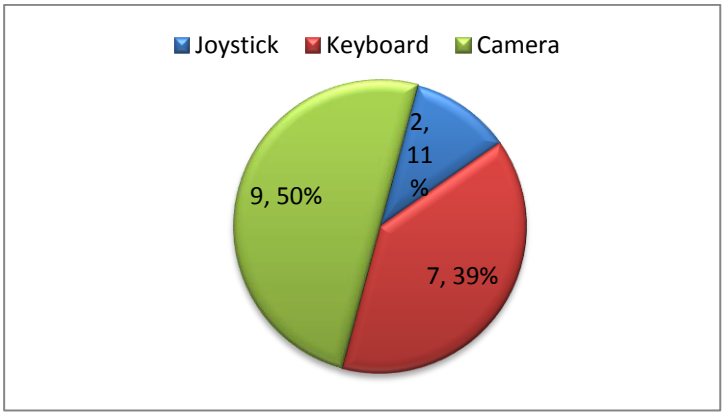

Figure 42: Overall input preference.

The least preferred option is the joystick based input method, where just 2 (11%) participants reported that they prefer the joystick based input method. The result tends to

reject the hypothesis H7, in which we stated that keyboard would be the most preferred input method. In addition, it also rejects the hypothesis H5 which stated that the users will prefer either the keyboard or the joystick based input method as the depth-camera is not ergonomic to use.

4.6.12 Unstructured User Feedback

The question number 16 in Appendix B asked the users to write their comments about the system as a whole. The questions were designed to gather suggestions and critical remarks. This section talks about the most common remarks made by the users.

1. *Depth camera position:* Many users reported that the camera position plays an important role in overall performance of the depth-camera based interface. The depth-camera was mounted on top of the desktop monitor and for many users that was not a convenient position. Participants reported that instead of the mounting the depth-camera on top of the monitor the depth-camera should be kept beside the keyboard on the same plane as keyboard facing upwards so that the user does not have to lift their hand too high to interact with camera. Also, some users said that there should be a support for the elbow if the camera was to be mounted on top of the desktop monitor.

2. *Keyboard control keys:* Some users said that the keyboard control keys should be assigned in such a way that only one hand is enough to interact with the simulator. Also, instead of using the keys “Z” and “X” for controlling the depth of the ball, a user suggested that the keys “W” and “S” make a better match for it. Some participants said that keyboard controls are confusing when the user viewpoint or

perspective changes i.e. when the camera viewing parameters (direction and orientation) are changed.

3. *Joystick joints:* Some of the users reported that the joystick interface is confusing to use as compared to the depth-camera and the keyboard because sometimes it is not clear which joint is selected and working in the simulator. The simulator should highlight the joint which is selected when the user is manipulating the arm using the joystick.

4.7 Researcher Findings

. We will briefly discuss some of the findings from the experiments in this section. For the joystick interface, we noticed that some users completed both tasks just by using the finger analogy feature (mentioned in section 3.5) of the joystick interface, which is interesting to see because even though the finger analogy method is simpler (user just needs to select and control joint 1 instead of control all 3 joints) it takes longer to complete an iteration of a task. However, some users preferred the slow and simpler method over the faster but slightly more complex method (select the joint depending on the location of the target object).

The user can rotate the OpenGL camera around the scene by dragging the mouse left or right. We expected that this feature will be used extensively, as it provides a better understanding of the target ball's position. However, many users reported that it is confusing to change the camera position in the scene because the user perspective changes completely and it is harder for them to control the target ball.

4.8 Statistical Analysis

A balanced study was conducted as mentioned in section 4.4. A total of 18 participants were recruited, and each user performed the experiment with all 3 devices with 10 trials per task and one of 6 different input orders was given to a user. Execution time in seconds was measured for each iteration. The following table shows the number of observations per input, order, trial and input-order combination.

Total Number of observations	Per Input	Per Order	Per Trial	Per Input-Order combination
540	180	180	54	60

Table 3: Number of observations

4.8.1 ANOVA tests

Two-way ANOVA tests were conducted on both Task 1 and Task 2. There was a significant difference in execution times and in variances of Task 1 and Task 2 (obtained from Bartlett's and Shapiro-Wilk tests). Some non-parametric tests were also applied along with two-way ANOVA tests. The details of the statistical analysis can be found in Appendix C.

For the following results, the following key was used in order to establish whether a result is statistically significant or not.

p-value	Inference
Less than 0.001	Highly significant
Less than 0.01	Significant
Less than 0.5	Borderline significant

Table 4: Key for statistical significance

4.8.2 Mean execution time per input

Table 5 shows the mean execution time per iteration with each input device. To confirm whether the difference between the execution times is statistically significant, an ANOVA test was conducted and we obtained a p-value of `2e-16` which is much less than `0.001`, which indicated that the difference in execution times of the input devices is statistically significant. Similarly, for Task 2 we obtained a p-value of `6.78e-07` which means that the difference in the mean execution time as a function of each input device is statistically significant.

Input device	Keyboard	Camera	Joystick
Task 1	17.333	18.083	28.733
Task 2	29.59	30.75	37.98

Table 5: Mean execution time per input

Figure 43 shows the mean execution times per trial for each input device for task 1 and Figure 44 shows the mean execution times per trial for each device for task 2. As shown in Fig. 43 and Fig. 44 the difference between the execution times of depth-camera and joystick and keyboard and joystick is significant. However, the depth-camera and the keyboard execution times are almost similar. To understand whether the difference between the execution times of depth-camera and the keyboard is statistically significant we performed the Tukey multiple comparison [105] test on the data.

The following table shows the p-values obtained by the comparison test for both Task 1 and Task 2.

Comparison pair	Task 1 p-value	Task 2 p-value
Depth-camera - keyboard	0.846458	0.7714063
Joystick - Keyboard	0.000000	0.0000025
Joystick – Depth-camera	0.000000	0.0000610

Table 6: Difference in execution times of input devices.

As indicated in Table 6, the p-value obtained from the comparison of the depth-camera and keyboard is much greater than 0.001, therefore there is no statistically significant difference between the execution times of the depth-camera and the keyboard interface. However, the comparison between the joystick and the keyboard resulted a p-value of 0. for Task 1 and 0.0000025 for Task 2, which means that there is a statistically significant difference in the execution times of the joystick and the keyboard. Similarly, for the joystick and the depth-camera interface, a p value much less than 0.001 was obtained which confirms that there is a statistically significant difference in the execution times of the joystick interface and the depth-camera interface. The above mentioned result confirms the hypothesis H4 as the depth-camera and the keyboard both performed better than the joystick. However, the results rejected the hypothesis H6 as there is no significant difference in the execution times of the depth-camera and the keyboard providing only statistical support to part of H6, namely that the keyboard would perform better than the joystick.

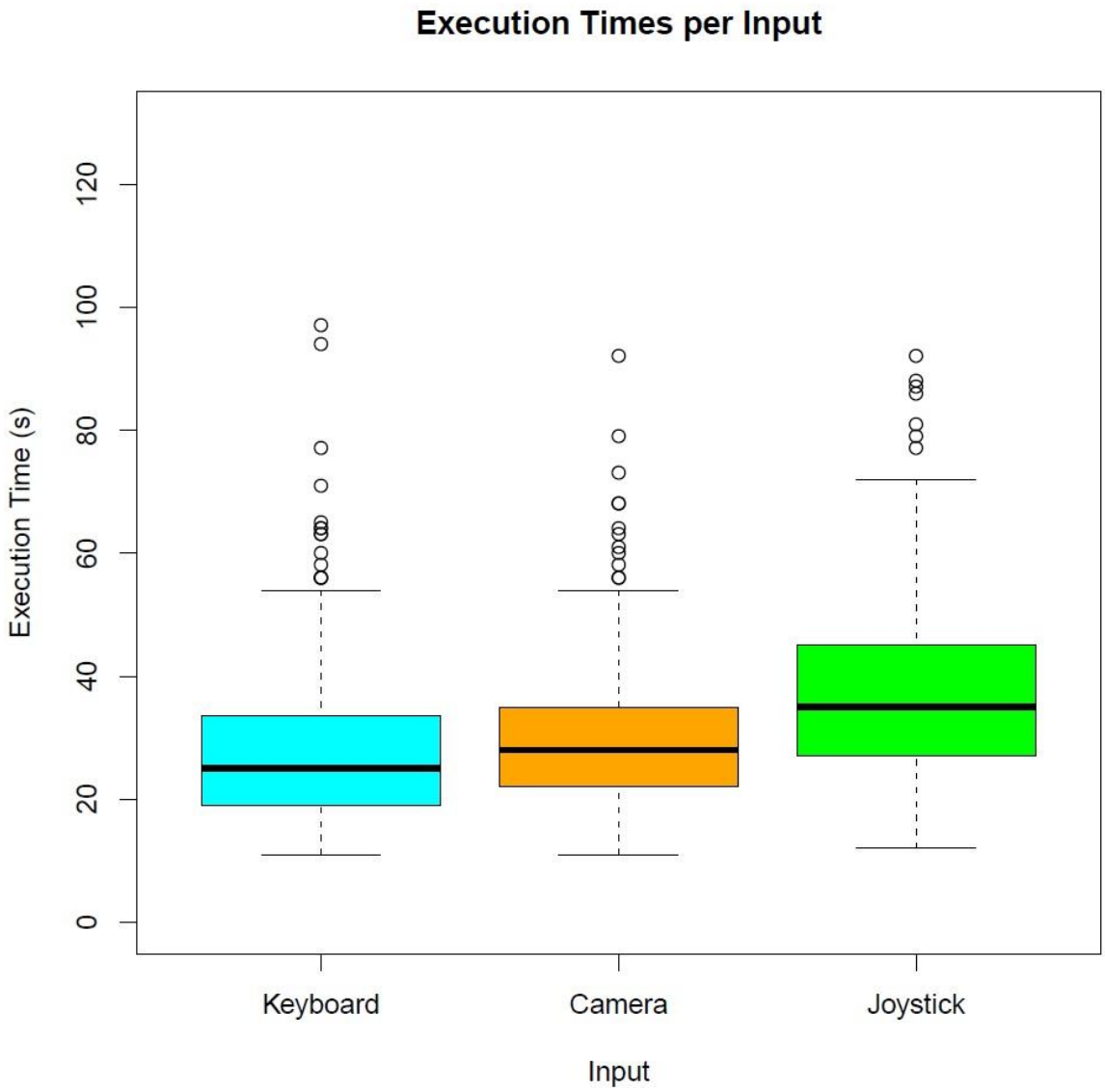

Figure 43: Execution time per input for Task 1

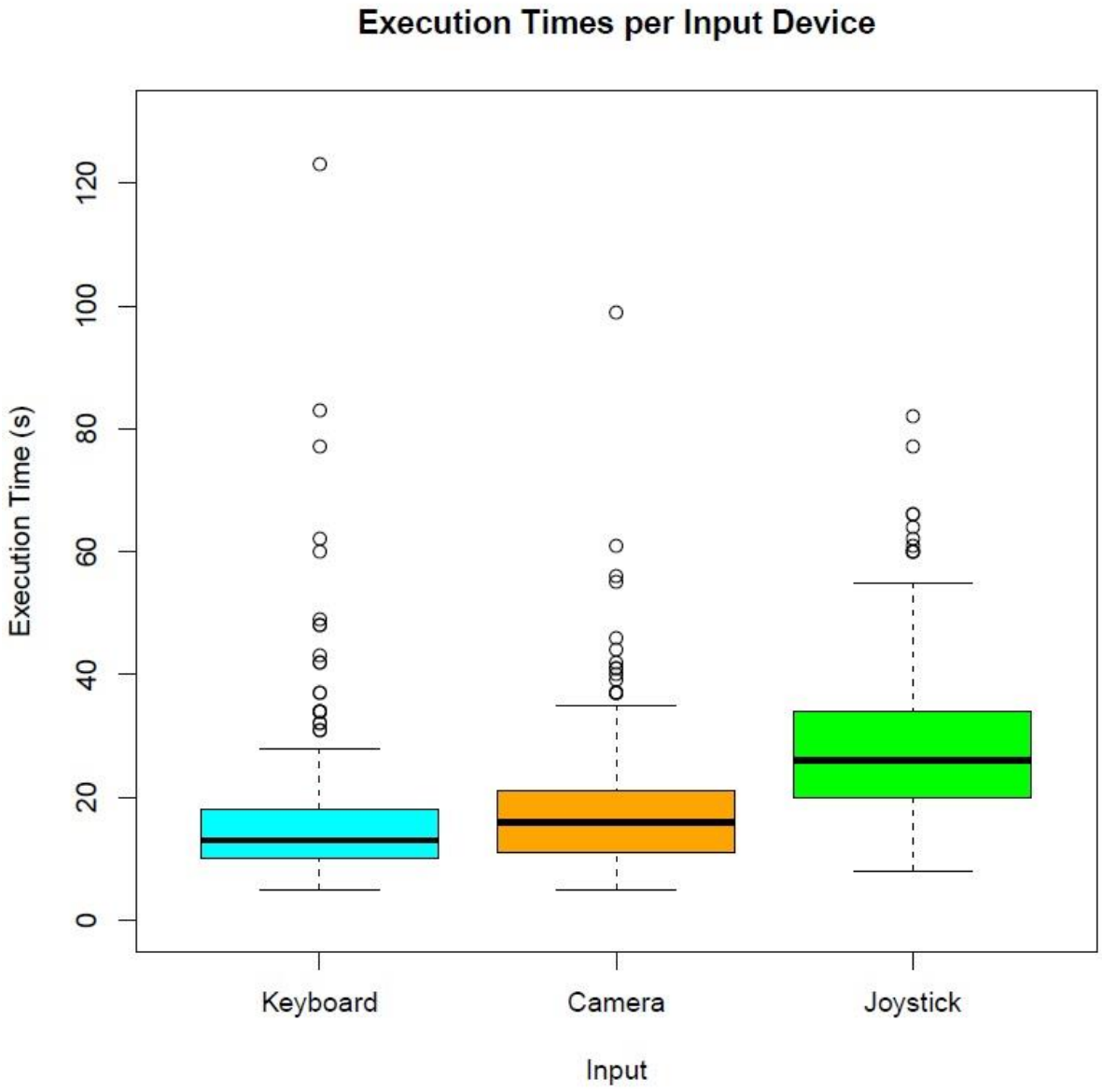

Figure 44: Execution time per input for Task 2

4.8.3 Means of Execution Times by Order.

The Table 7 shows the mean execution time and the standard deviation of each device in different order for Task 1, and Table 8 shows the mean execution time and the standard deviation of each device in different order for Task 2. From the tables 7 and 8, it can be seen that the keyboard's mean execution time for 3rd order decreases significantly as compared to 1st and the 2nd order. However, for the depth-camera the execution times seem to increase with the order; while the joystick's mean execution times do not change significantly by the order. To confirm this behavior Wilcoxon rank sum test [106] was conducted and for the keyboard for both tasks a p-value < 0.0200 was obtained, which confirms that the keyboard's execution times for the 3rd order is significantly lower than those for the 1st and 2nd order. For depth-camera, however, the execution times for Task 1 for the 3rd order were significantly larger than for the 1st order (p-value 0.03) and for Task 2 the execution times for the 3rd order were significantly larger than for the 2nd order (p-value =0.0088).

Input device	1st	2nd	3rd
Keyboard	19.4830 ± 11.670	20.350 ± 18.590	12.167 ± 10.540
Camera	15.933 ± 8.830	17.050 ± 9.860	21.267 ± 15.360
Joystick	29.867 ± 14.970	27.650 ± 12.370	28.683 ± 11.190

Table 7: Mean execution time and standard deviation of each input device at different order for Task 1

Input device	1st	2nd	3rd
Keyboard	32.23 ± 21.76	30.70 ± 15.79	25.85 ± 13.53
Camera	29.82 ± 9.89	26.98 ± 10.33	35.45 ± 17.46
Joystick	39.60 ± 16.51	37.47 ± 21.61	36.88 ± 14.17

Table 8: Mean execution time and standard deviation of each input device at different order for Task 2

Figure 45 and Figure 46 show the graphical representation of the execution times at different order per input device for Task 1 and Task 2 respectively. In case of the keyboard the execution time is significantly less when it was used at 3rd place. This is same for both Task 1 and Task 2. However, for the joystick there is not a significant difference in execution time by the order, for both Task 1 and Task 2. But the depth camera behaves differently than both keyboard and the joystick. The depth-camera execution times increases by the order, i.e. depth camera was slowest when it was used at the third place. The Figure 45 and Figure 46 both confirm the same behavior. The results discussed above reject hypothesis H 10 as the users' performance did not improve with the order across input devices; however, the order did affect the execution times of two of the input devices (keyboard's execution times improve and those of the camera became slightly worse).

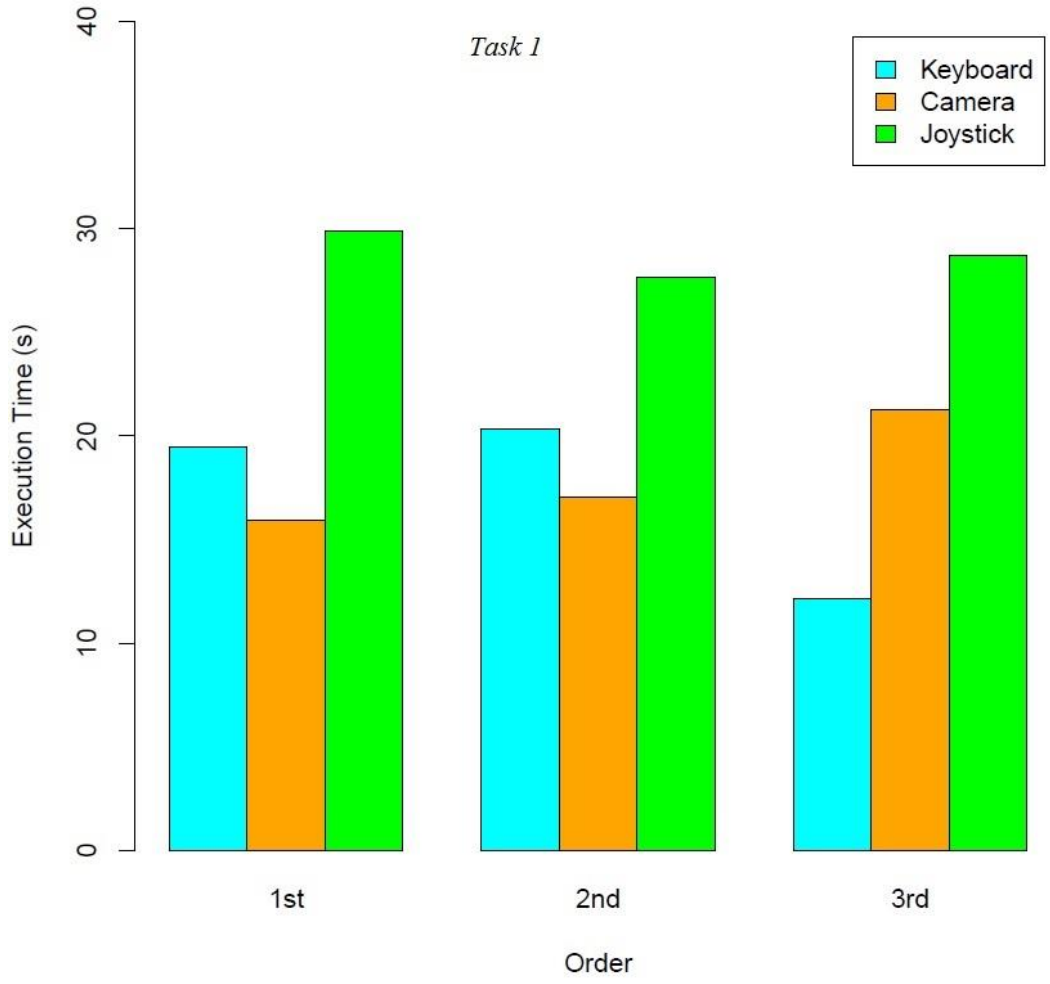

Figure 45: Mean execution time of each input device at different order for Task 1

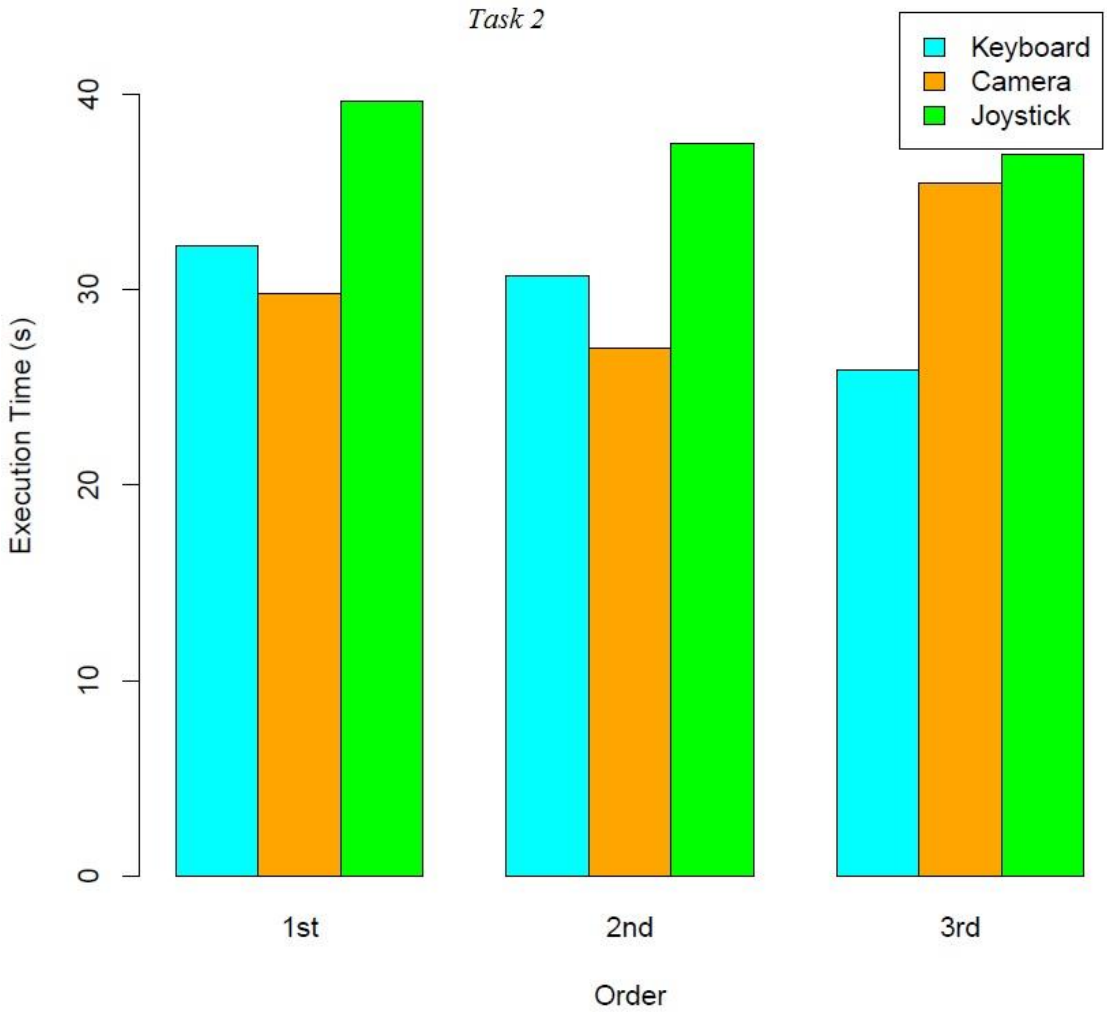

Figure 46: Mean execution time of each input device at different order for Task 2

Figures 47 and 48 show the execution times sorted by input order and input device. In Figures 47 and 48, the first digit showing 1 is the keyboard (shown in blue), 2 is the depth-camera (yellow) and 3 is for the joystick (green). For example, 1.1 denotes input 1(which is keyboard) and order 1. Similarly, the orders for all inputs are shown in both figures. From both figures it is clearly evident that the keyboard execution times a

significantly lower for 3rd order as compared to 1st and 2nd order. But for depth-camera the execution times are higher for 3rd order as compared to 1st and 2nd order. There was no significant change in the execution times for the joystick as the p-value obtained was higher than 0.05.

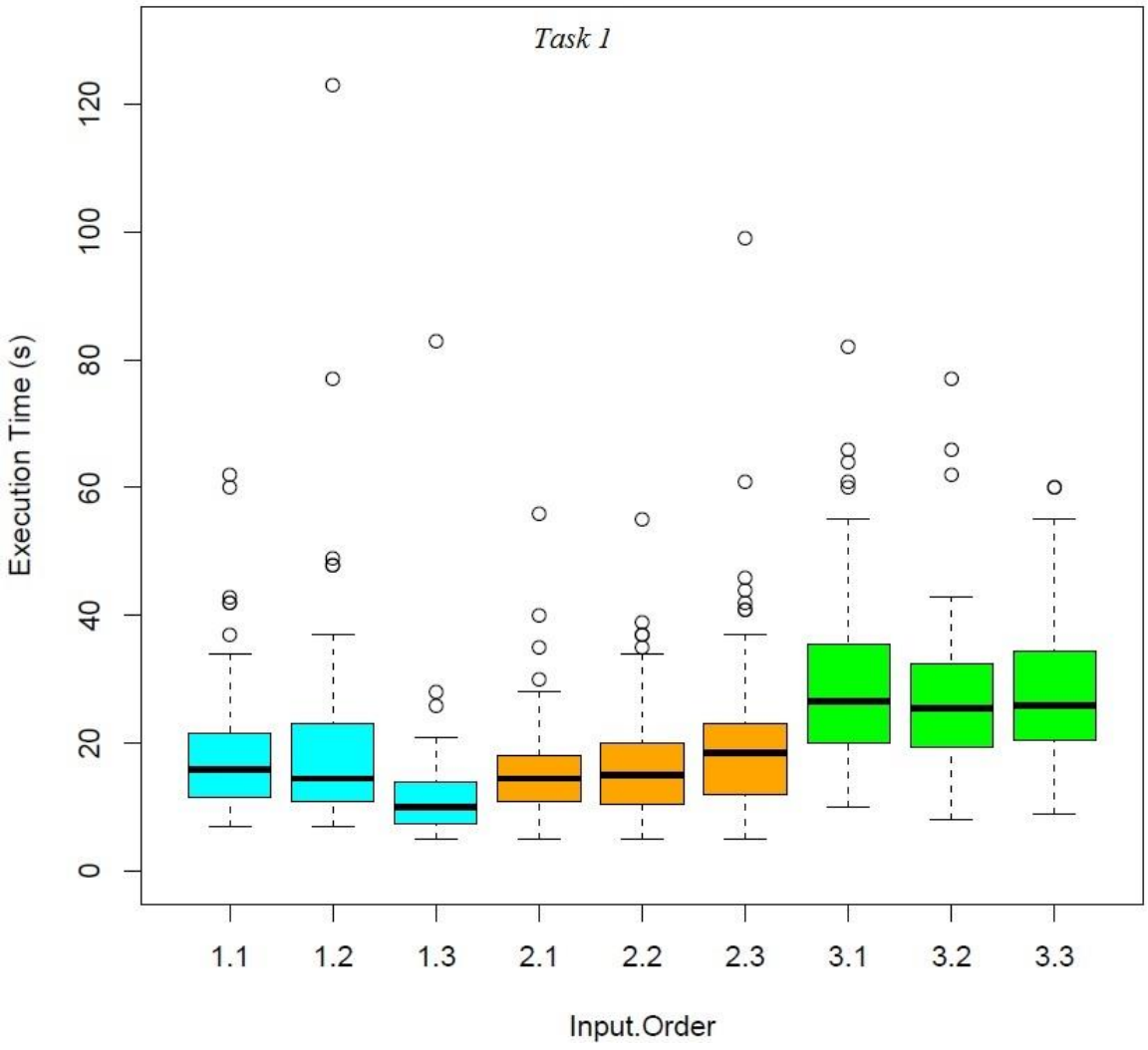

Figure 47: Mean execution times of each input sorted by order for Task 1

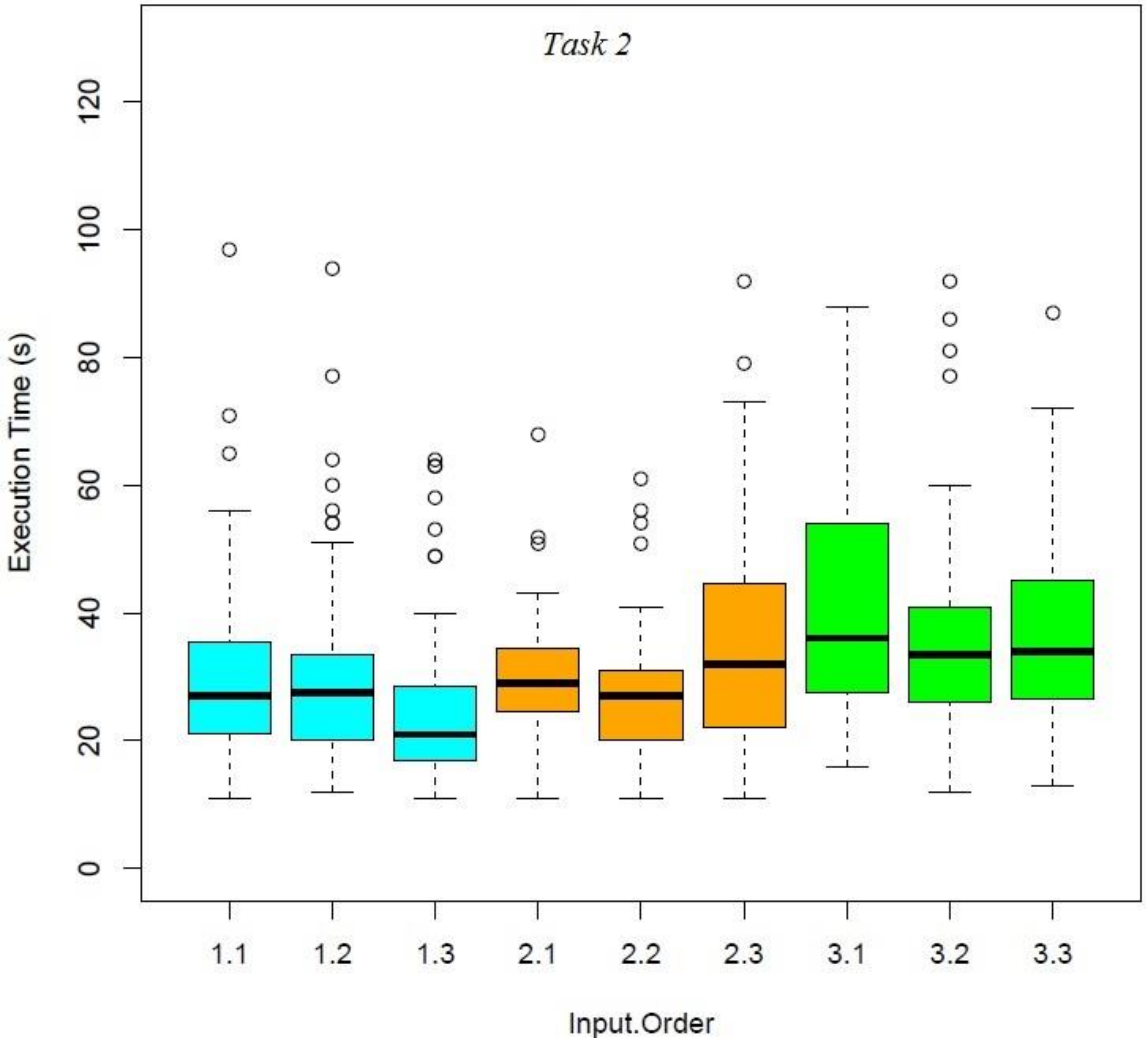

Figure 48: Mean execution times of each input sorted by order for Task 2

4.8.4 Learning effect

The following plots show the learning effect with each device over the 10 trials. The execution time for each trial for each input was analyzed and we found that the depth-camera has the clearest learning effect amongst all the input devices. For Task 1 we obtained a p-value of 9e-04 (Fig. 49) using paired Wilcoxon rank sum test which confirms that the learning effect is highly significant in case of depth-camera.

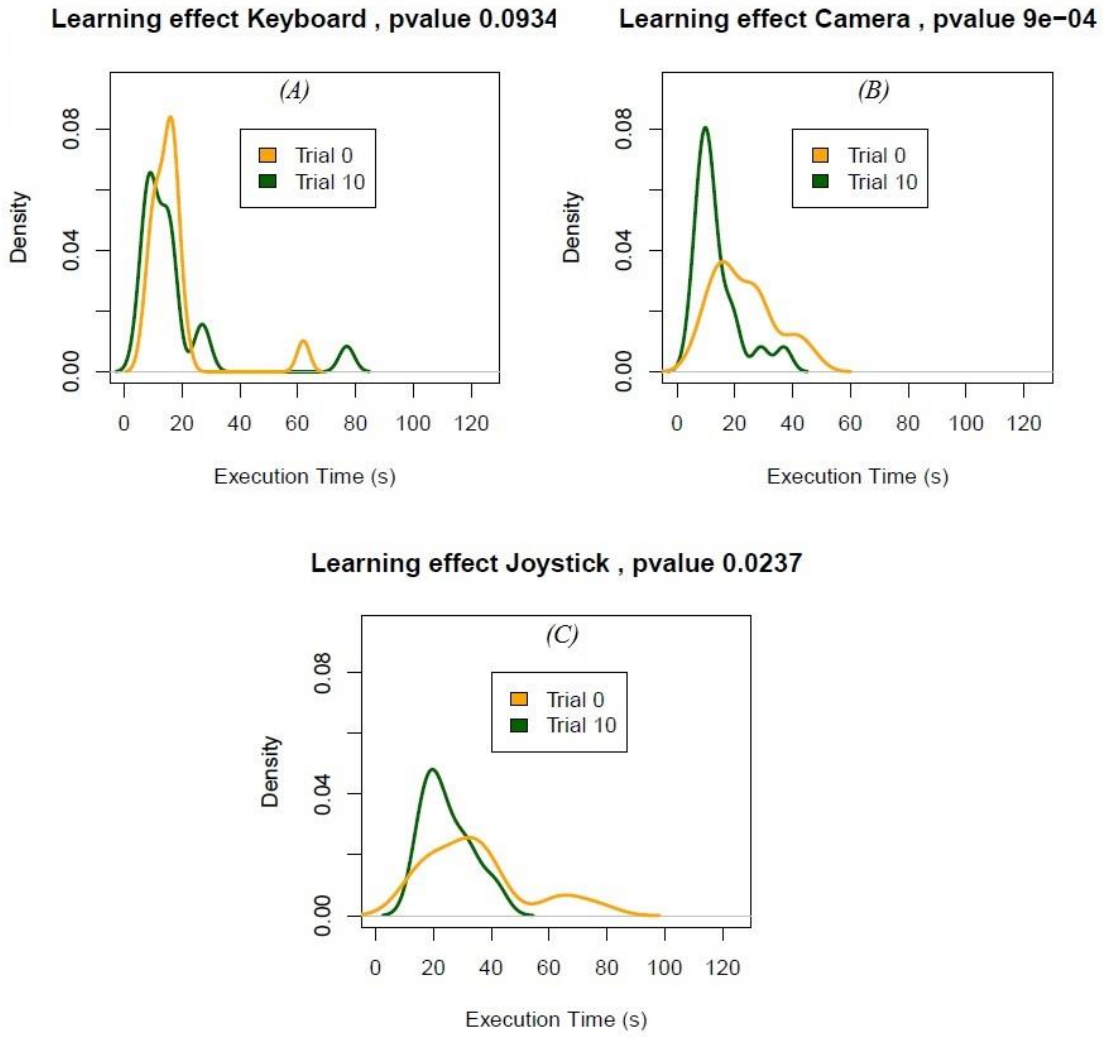

Figure 49: Learning effect for each input device for Task 1

The joystick also shows a significant (p-value < 0.05) learning effect, however not as clear as the depth-camera's. The keyboard on the other hand doesn't have a significant (p-value > 0.05) learning effect. From Task 2 results (Fig. 50) also it is confirmed that the depth-camera has a significant learning effect (p-value < 0.05) over the trials. However, the keyboard and the joystick interface doesn't show a significant learning (p –value > 0.05) curve for Task 2. The above mentioned result is in accordance with user perceptions as predicted in hypothesis H8, which states that the participants will find that the depth-camera based input will perform better.

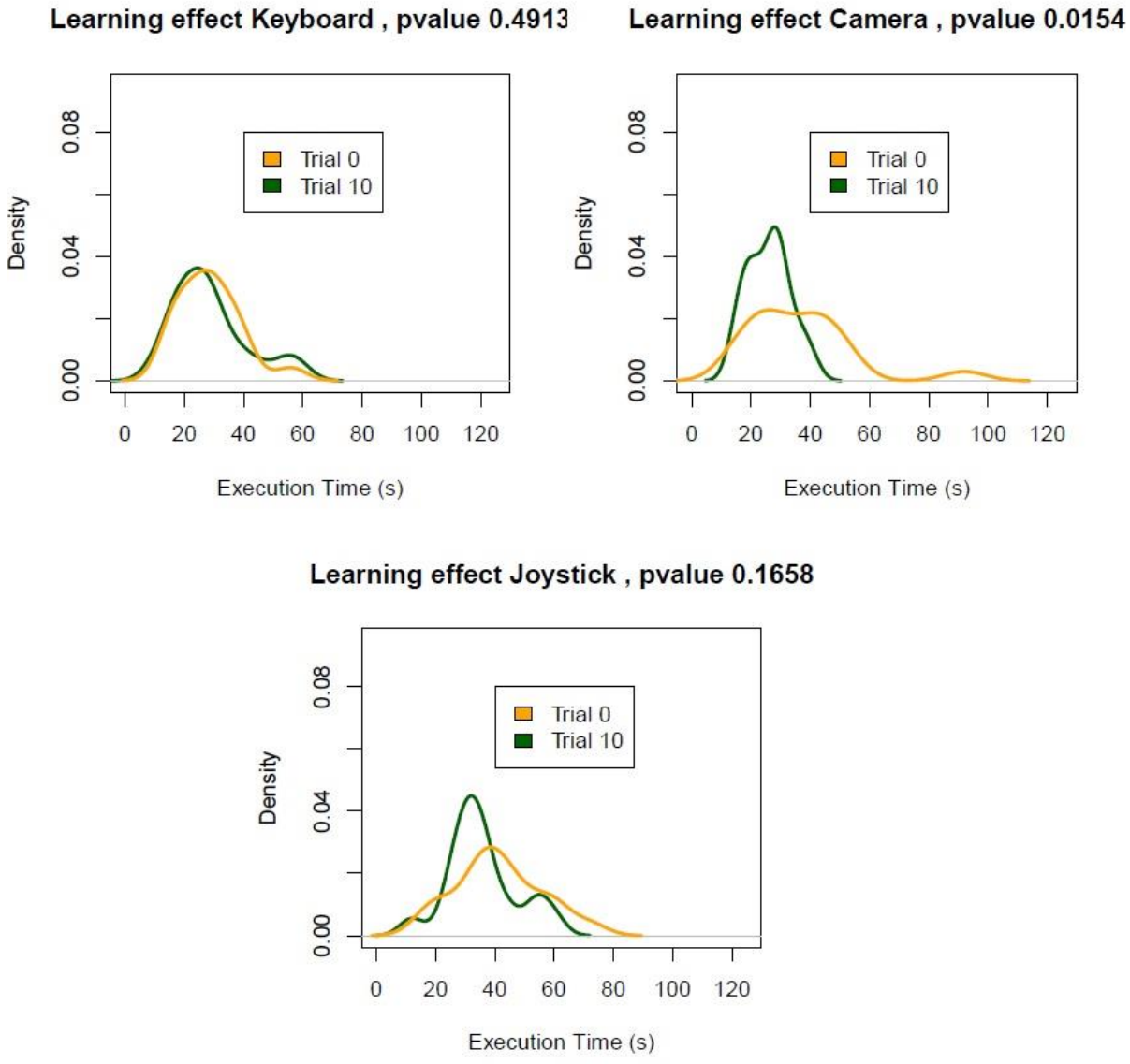

Figure 50: Learning effect for each input device for Task 2

4.9 Summary of User Feedback and Statistical Analysis

In this section we summarize the user feedback and the statistical analysis, and how that relates to our hypotheses.

- The hypothesis H1, which stated that the keyboard based input would be the easiest to learn, was rejected by the user feedback as all three devices were rated almost equally. However, as the results showed that each input was rated comparably in ease of learning by the users, we can say in general all the input methods were considered to be easy to learn.
- The hypothesis H2, which stated that the keyboard and the depth-camera based input methods will be easier to use as compared to joystick, was not clearly supported by the users' feedback, as the users found all the input devices were easy to use, but the depth-camera and the keyboard were rated slightly higher than the joystick.
- The hypothesis H3, which stated that the depth-camera based input method is not as ergonomic as the other two methods, was supported by the results, many participants said that the depth-camera was inconvenient to use because of its position. The keyboard and the joystick were rated equal in ergonomics ratings.
- The hypothesis H4, which stated that the users will perceive that the depth-camera based input and the keyboard based input method perform better than the joystick based method, was supported by the users' responses, and the statistical analysis mentioned in section 4.8 shows that the keyboard and the depth-camera based input methods performed significantly better than the joystick.

- The hypothesis H5, which stated that the participants are more likely to prefer the keyboard or the joystick based input method as their overall preference for an input method, was rejected by users' feedback as most of the users preferred the depth-camera based input over both the keyboard and the joystick both.
- The hypothesis H6, which stated that users of the keyboard will perform better than the users of the depth-camera and the joystick both, was rejected by the statistical analysis, as we found out that the keyboard and the depth-sensing camera performed equally. Only part of H6 was confirmed. This is the part that suggested that the keyboard would perform better than the joystick..
- The hypothesis H7, which stated that keyboard will be the most preferred input method, was rejected by the user feedback, as according to the users the depth-camera was the most preferred input method. Keyboard was the second most preferred input method.
- The hypothesis H8, which stated that the depth-camera based input will perform better, was supported by the user feedback as many participants said that they found the depth camera to be performing better. Also, from statistical analysis, we found out that the camera has a steep learning effect out of all the input methods, which confirms hypothesis H8.
- The hypothesis H9, which stated the participants will find the joystick to be the most useful input method, was rejected by the user feedback as the users found the depth-camera to be the most useful input method to complete the task.

- The hypothesis H10, which stated that the users will improve their performance as they complete more iterations of the same task across the input devices, was rejected by statistical analysis, which indicated that the order of input had a significant effect on the performance of the keyboard and the depth-camera based input methods. The keyboard execution times were significantly shorter when it was used at 3rd place, whereas the depth-camera execution times were higher for 3rd order and lower when the depth-camera was used at the 1st place, the reason for that could be the fact that the users get tired by the 3rd order and depth-camera causes discomfort to users especially when they were already tired.
- The hypothesis H11, which stated that the participants will prefer the depth-camera interface over the joystick interface, was supported by the user feedback, as most of the users said that they would prefer the depth-camera over joystick interface which was interesting to see, because as shown in section 4.6.1, the depth-camera was the least familiar method to the users.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this thesis a new approach to manipulate a robotic arm is presented. The presented approach makes use of an inexpensive depth-camera to capture user input and inverse kinematics to define the motion of the robotic arm. An OpenGL based robot arm simulator was developed which could be controlled by a standard keyboard, a gaming joystick and a depth-camera. A user study was conducted to test the presented approach. 18 participants were recruited to participate in the study. The participants were divided in 6 groups and each group followed a different order of input conditions. The users were randomly assigned to a group. Each user was asked to complete two tasks, one a picking task and the other a picking and dropping task. Task completion time was recorded at every iteration for both tasks. A statistical analysis was performed on the data obtained from the experiment. From the experimental results of Chapter 4 the following conclusions can be drawn about the presented approach.

The execution times for depth-camera and the keyboard were significantly lower than the joystick interface. On an average keyboard execution times are 11.40 seconds shorter and camera execution times are 10.65 seconds shorter than the joystick. However, the standard deviations for keyboard, depth-camera and joystick are 14.46, 11.87, and 12.9 for Task 1 (17.79, 13.43, and 17.68 for Task 2), respectively. This shows that there was a lot of variation in execution times of each device, since sometimes the random placement

of the object were close to the target and sometimes far. Since contemporary manipulators use a joystick-like interface to control robots we can say that the two other interfaces might offer a faster approach for some manipulation tasks.

We concluded that the input order plays an important role in the performance, or execution times, of the keyboard and the depth-camera interface. For the keyboard interface in 3rd order (i.e. when the keyboard was used after the user had used the depth-camera and the joystick interface), the execution times was significantly lower as compared to the 1st and the 2nd order. This could be explained by the fact that by the third time the user is aware of how the system works and hence the keyboard execution times are short. However, for the depth-camera the execution times are significantly higher for 3rd order as compared to the 1st order, which implies that the depth-camera execution times are longer over the time which might be due to the fact that by the 3rd order the users are tired. From the user feedback which was gathered after the experiment (Appendix B), the users reported that the depth-camera was inconvenient to use for longer durations because of its position. Also, it was rated low in the ergonomics ratings (38% users strongly disagreed that the camera was an ergonomic design, whereas 22% responded neutrally) which also explains the longer execution times when users perform the tasks using the depth-camera after having used the keyboard and the joystick. Therefore, it is important to keep the depth-camera at a convenient position so it doesn't cause discomfort to the user over longer durations.

The results in Chapter 4 also showed that there is a significant learning effect in the case of the depth-camera as compared to the keyboard and the joystick based interfaces.

The average execution time improved from trial number 1 to 10. Also, in the exit questionnaire, 61% users reported that the depth-camera performed best whereas 28% users reported that the keyboard performed best and just 11% users said that the joystick was the one that performed best. The user feedback suggests that the camera has a clear learning effect and becomes efficient after training and practice. Hence, we can conclude that the depth-camera requires a bit of getting used to and training but after that, the users preferred the depth-camera over joystick and keyboard both.

50% of the users reported that they would prefer the depth-camera interface over the joystick and the keyboard interface, 39% users reported that they would prefer the keyboard interface over the depth-camera and the joystick interface, whereas only 11% users reported that they would prefer the joystick interface over the depth-camera and the keyboard interface. The reason why the joystick was least preferred could be related to the fact that the execution times of the joystick are significantly longer as compared to the depth-camera and the keyboard. Also, joystick is the only interface which is operated by forward kinematics (user controls the joint and the rotation on its own) so we can conclude that the inverse kinematics based approaches were faster as compared to the forward kinematics.

We can also conclude that there is no preferred input method for a particular task as the results from Task 1 and Task 2 are similar for both tasks. The users also reported the same in the exit questionnaire. 38.8% of the users said that the keyboard was best suited for both the Task 1 and Task 2, 44% users said that the depth-camera was best suited for Task 1 and 50% users said that the depth-camera was best suited for Task 2. Whereas

just 16% users responded that the joystick was best suited for task 1, and 11% responded that the joystick was best suited for task 2. This result also shows that the users indicate that the depth-camera and the keyboard were the preferred input methods for both tasks.

5.2 Future Work

The user comments in the questionnaire provided important feedback about the presented approach. A common feedback from the users was about the position of the depth-camera; the position of the depth-camera plays an important role in overall comfort of the users of the depth-camera interface. We would like to explore the possibility placing the depth-camera at a more convenient position so the user doesn't need to lift his/her hand too much to interact with the depth-camera interface. In the future, the depth-camera could be placed on the same plane as of the keyboard facing upwards which could make the depth-camera interface more comfortable to use. Another option is to provide a support for the user's elbow so that it doesn't cause pain over the long term.

The user experience with the joystick interface can be improved by implementing colored feedback in the simulator, e.g. the selected joint should change the color after joint selection. Similarly, in case of the depth-camera the grid mentioned in Fig. 18 could be implemented with a colored feedback, so when a user is in a specific cell, the gird shows that appropriately.

Currently the implementation is limited to finding the solution in one plane and all the joint rotations occur in one plane. In future, the presented method would be extended to support not only cylindrical joints but spherical and prismatic joints as well so that a solution can be obtained in multiple planes using the inverse kinematics method.

This research mainly focused on the control of the robot simulator to reach a target position using the depth-camera and inverse kinematics. However, it does not focus on the end-effector module in detail and the randomly generated objects which were used for tasks were relatively uniform in shape. However, in the real world the object which may need to be manipulated might not be uniform in shape or size. In the future, we would like to add an intelligent algorithm to the end-effector control module which will find a way to grab the object based on its shape [107], [108]. Also, the possibility of the remote operation of the robotic arm can be explored where the operators are manipulating the robotic arm using a wireless network or through LAN.

The presented approach could also be extended to control more than one manipulator robots where the robots can communicate with each other and transfer a load from a point to a target position [30, 31]. This could enable the manipulator to perform complex manipulation tasks.

Finally, the presented approach would be tested in a real world environment, with an actual robot arm.

5.3 Publications from this Research

Parts of this work were presented as a research article at the OCEANS 2014 conference held at St. Johns, NL under the category “Remotely operated vehicles-II” [112]. Posters based on this research were presented at the AI/GI/CRV conference held in University of Montreal in 2014, where the poster was awarded as the best research poster in HCI category [114] and at the Nova Scotia Energy R&D Conference 2014 under the category “Seabed Engineering” [113].

Bibliography

[1] D. E. Whitney, "Quasi-static assembly of compliantly supported rigid parts," *J. Dyn. Syst. Meas. Control*, vol. 104, no. 1, pp. 65–77, 1982.

[2] "GRI Simulations Inc. In Manipulator Trainer (2014)." [Online]. Available: http://www.grisim.com/products.html#ManipTrainer. [Accessed: 10-Jun-2014].

[3] Microsoft, "Microsoft Kinect for Windows," 2014. [Online]. Available: http://www.microsoft.com/en-us/kinectforwindows/. [Accessed: 15-Dec-2014].

[4] Intel, "Perceptual Computing SDK," 2013. [Online]. Available: https://software.intel.com/en-us/perceptual-computing-sdk. [Accessed: 11-Nov-2013].

[5] N. Sian, K. Yokoi, and S. Kajita, "Whole body teleoperation of a humanoid robot-development of a simple master device using joysticks," *Intell. Robot.*, no. October, 2002.

[6] R. Manikandan and R. Arulmozhiyal, "Position control of DC servo drive using fuzzy logic controller," *2014 Int. Conf. Adv. Electr. Eng.*, pp. 1–5, Jan. 2014.

[7] Jan, V., Marek, S., Pavol, M., Vladimir, V., Stephen, D. J., & Roy, P. Near-time-optimal position control of an actuator with PMSM. In *Power Electronics and Applications, 2005 European Conference on* (pp. 10-pp). IEEE.

[8] J. Yuan, "Closed-loop manipulator control using quaternion feedback," *Robot. Autom. IEEE J.*, vol. 4, no. 4, pp. 434–440, 1988.

[9] M. Keerio, "A friendly and human-based teleoperation system for humanoid robot using joystick," *2008 7th World Congr. Intell. Control Autom.*, pp. 2283–2288, 2008.

[10] V. Ng-Thow-Hing and S. Okita, "Synchronized gesture and speech production for humanoid robots," *2010 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pp. 4617–4624, Oct. 2010.

[11] B. House, J. Malkin, and J. Bilmes, "The VoiceBot: a voice controlled robot arm," *Proc. SIGCHI Conf.*, pp. 183–192, 2009.

[12] Corradini, A., & Gross, H. M.. Camera-based gesture recognition for robot control. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on* (Vol. 4, pp. 133-138). IEEE, 2000.

[13] S. Oniga, A. Tisan, D. Mic, A. Buchman, and A. Vida-Ratiu, *Hand Postures Recognition System Using Artificial Neural Networks Implemented in FPGA*. 2007.

[14] "Real-time 3D pointing gesture with Kinect for object-based navigation by the visually impaired," in *2013 ISSNIP Biosignals and Biorobotics Conference: Biosignals and Robotics for Better and Safer Living (BRC)*, 2013, pp. 1–6.

[15] C. Chen, M. M. Trivedi, and C. R. Bidlack, "Simulation and animation of sensor-driven robots.," *IEEE Trans. Rob. Autom.*, vol. 10, no. 5, pp. 684–704, Oct. 1994.

[16] R. Kikuuwe, "A Sliding-Mode-Like Position Controller for Admittance Control With Bounded Actuator Force," *IEEE Journals Mag.*, vol. 19, no. 5, pp. 1489–1500, 2014.

[17] H. B. R. L. Carroll, H. Bolandi, R. L. Carroll, and Y. Chen, "Adaptive Model-Reference Position Control of Dual-Arm Manipulators," in *American Control Conference, 1991*, 1991, no. 18, pp. 3068–3069.

[18] W. Feng and I. Postlethwaite, "A Simple Robust Control Scheme for Robot Manipulators with Only Joint Positon Measurements," in *Intelligent Control and Instrumentation, 1992. SICICI '92. Proceedings., Singapore International Conference on*, 1992, vol. 1, pp. 335–340.

[19] A. M. Sharaf, E. E. Depamnent, A. Mohammed, and S. El-kom, "Laboratory Implementation of Fuzzy Logic Position Controller for a Single-Link Robotic Manipulator," *Control Appl. 1994., Proc. Third IEEE Conf. Date 24-26 Aug. 1994*, no. 4, pp. 547–552, 1994.

[20] P. Hajek, "Fuzzy Logic," *http://plato.stanford.edu/entries/logic-fuzzy/*, 2002. [Online]. Available: http://plato.stanford.edu/entries/logic-fuzzy/. [Accessed: 12-Dec-2014].

[21] A. Rubaai, J. Jerry, and S. Smith, "Performance evaluation of fuzzy switching position controller for automation and process industry control," *Ind. Appl. IEEE* , vol. 47, no. 5, pp. 2274–2282, 2011.

[22] S. Diego, "Non-Linear Controller for Position and Force Control," *Decis. Control. 1997., Proc. 36th IEEE Conf. (Volume2)*, no. December, pp. 1353–1354, 1997.

[23] S. Han and J. Lee, "Tele-operation of a mobile robot using a force reflection joystick with a single hall sensor," *Commun. 2007. RO-MAN 2007. …*, pp. 206–211, 2007.

[24] J. Mills, “Robotic manipulator control of generalized contact force and position,” *Syst. Man Cybern. IEEE Trans.* ,vol. 24, no. 3, 1994.

[25] Control-Solutions, “Basics of PID Control,”. [Online]. Available: http://www.csimn.com/CSI_pages/PIDforDummies.html. [Accessed: 13-Dec-2014].

[26] Y. Li, Z. Wang, and L. Zhu, “Adaptive neural network PID sliding mode dynamic control of nonholonomic mobile robot,” in *Information and Automation (ICIA), 2010 IEEE International Conference on*, 2010, pp. 753–757.

[27] B. Wang, Y. Sun, J. Cao, and G. Zhang, “Control and simulation of an underwater robot using single neuron PID control based on immune feedback mechanism,” in *Intelligent Control and Automation (WCICA), 2010 8th World Congress on*, 2010, pp. 2507–2511.

[28] W. Yu and J. Rosen, “Neural PID Control of Robot Manipulators With Application to an Upper Limb Exoskeleton,” *Cybern. IEEE Trans.*, vol. 43, no. 2, pp. 673–684, Apr. 2013.

[29] J. Jaafar and R. L. A. Shauri, “PID position control for 2-DOF robotic finger,” *2013 IEEE 4th Control Syst. Grad. Res. Colloq.*, pp. 152–157, Aug. 2013.

[30] T. J. Tarn, A. K. Bejczy, and X. Yun, “Coordinated control of two robot arms,” in *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, 1986, vol. 3, pp. 1193–1202.

[31] C. O. Alford and S. M. Belyeu, “Coordinated control of two robot arms,” in *Robotics and Automation. Proceedings. 1984 IEEE International Conference on*, 1984, vol. 1, pp. 468–473.

[32] S. Hirose and R. Chu, “Development of a light weight torque limiting M-Drive actuator for hyper-redundant manipulator Float Arm,” in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, 1999, vol. 4, pp. 2831–2836 vol.4.

[33] J. A. Bilmes, J. Malkin, X. Li, S. Harada, K. Kilanski, K. Kirchhoff, R. Wright, A. Subramanya, J. A. Landay, P. Dowden, and H. Chizeck, “The Vocal Joystick,” in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, 2006, vol. 1, pp. I–I.

[34] Malkin, J., House, B., & Bilmes, J. Control of simulated arm with the vocal joystick. In *CHI 2007 Workshop on Striking a C [h] ord: Vocal Interaction in Assistive Technologies, Games, and More*, April, 2007.

[35] S. Y. Won, D.-I. Lee, and J. Smith, “Humming control interface for hand-held devices,” in *Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility*, 2007, pp. 259–260.

[36] A. J. Sporka and P. Slavík, “Vocal control of a radio-controlled car,” *ACM SIGACCESS Access. Comput.*, no. 91, pp. 3–8, 2008.

[37] D. Rao and D. Bhattacharya, “Dual Sensor Based Gesture Robot Control Using Minimal Hardware System,” *Int. J.*, vol. 3, no. 5, pp. 3–5, 2013.

[38] D. B. Stewart, D. E. Schmitz, and P. K. Khosla, “CHIMERA II: a real-time multiprocessing environment for sensor-based robot control,” *Proceedings. IEEE Int. Symp. Intell. Control 1989*, pp. 265–271.

[39] Weiss, L. E., Sanderson, A. C., & Neuman, C. P. (1987). Dynamic sensor-based control of robots with visual feedback. *Robotics and Automation, IEEE Journal of*, *3*(5), 404-417.

[40] C. Wusheng and W. Tianmiao, “Sensor-based autonomous control for telerobotic system,” *Proc. 4th World Congr. Intell. Control Autom. (Cat. No.02EX527)*, pp. 2430–2434, 2002.

[41] X. Wu and S. Ma, “Sensor-driven neural controller for self-adaptive collision-free behavior of a snake-like robot,” *2011 IEEE Int. Conf. Robot. Autom.*, pp. 191–196, May 2011.

[42] Freescale-dot-com, “MEMS-Based Sensor Technology.” [Online]. Available: http://www.freescale.com/webapp/sps/site/overview.jsp?code=SNSMEMSOVERVIEW. [Accessed: 14-Dec-2014].

[43] Atherton, J. A., & Goodrich, M. A. (2011, March). Perception by proxy: humans helping robots to see in a manipulation task. In *Human-Robot Interaction (HRI), 2011 6th ACM/IEEE International Conference on* (pp. 109-110). IEEE.

[44] Utsumi, A., & Ohya, J. (1998, June). Direct manipulation interface using multiple cameras for hand gesture recognition. In *Multimedia Computing and Systems, 1998. Proceedings. IEEE International Conference on* (pp. 264-267). IEEE.

[45] Cai, C., Dean-Leon, E., Mendoza, D., Somani, N., & Knoll, A. (2013, November). Uncalibrated 3D stereo image-based dynamic visual servoing for robot manipulators. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on* (pp. 63-70). IEEE.

[46] T.-I. Kim, W. Bahn, C. Lee, T. Lee, M. M. Shaikh, and K. Kim, “Vision system for mobile robots for tracking moving targets, based on robot motion and stereo vision information,” *2011 IEEE/SICE Int. Symp. Syst. Integr.*, pp. 634–639, Dec. 2011.

[47] H. Nagayama and T. Murakami, “An approach to high operational control of a slave mobile robot by master touch manipulation,” *Adv. Motion Control (AMC)*, pp. 557–562, 2014.

[48] X. Song, “A robust downward-looking camera based velocity estimation with height compensation for mobile robots,” *Autom. Robot.* no. December, pp. 7–10, 2010.

[49] H. B. Suay and S. Chernova, “Humanoid robot control using depth camera,” *Proc. 6th Int. Conf. Human-robot Interact. - HRI ’11*, p. 401, 2011.

[50] C. Parga, X. Li, and W. Yu, “Tele-manipulation of robot arm with smartphone,” *2013 6th Int. Symp. Resilient Control Syst.*, pp. 60–65, Aug. 2013.

[51] T. Uzunovic and E. Golubovic, “Configuration space control of a parallel Delta robot with a neural network based inverse kinematics,” *(ELECO), 2013 8th* , no. October, pp. 1914–1919, 2013.

[52] J. Stavnitzky and D. Capson, “Multiple camera model-based 3-D visual servo,” *Robot. Autom. IEEE Trans.*, vol. 16, no. 6, pp. 732–739, Dec. 2000.

[53] I. Kim, D. Kim, and Y. Cha, “An embodiment of stereo vision system for mobile robot for real-time measuring distance and object tracking,” *Control. Autom.* pp. 1029–1033, 2007.

[54] Y. M. Zhao, W. F. Xie, and X. W. Tu, “Multiple cameras-multiple target points visual servoing in large scale 3D manufacturing systems,” *2011 6th IEEE Conf. Ind. Electron. Appl.*, pp. 2107–2113, Jun. 2011.

[55] Fu, K. S., Gonzalez, R. C., & Lee, C. G. (1987). *Robotics* (pp. 163-189). McGraw-Hill, New York.

[56] EngineersHandbook.com, “Cartesian Coordinate Robots.” [Online]. Available: http://www.engineershandbook.com/Components/robclasscartesian.htm. [Accessed: 10-Sep-2014].

[57] EngineersHandbook.com, “Cylindrical Coordinate Robots.” [Online]. Available: http://www.engineershandbook.com/Components/robclasscylindrical.htm. [Accessed: 09-Oct-2014].

[58] Brighthubengineering.com, “Cylindrical Coordinate Robots.” [Online]. Available: http://www.brighthubengineering.com/robotics/29395-base-bodies-of-robots-cylindrical-base-robot/. [Accessed: 10-Oct-2014].

[59] Shankar, K. S., & Harmon, T. L.. Introduction to Robotics. *IEEE Expert*, *1*(2), 108-108, 1986.

[60] EngineersHandbook.com, “Spherical Robot.” [Online]. Available: http://www.engineershandbook.com/Components/robclassspherical.htm. [Accessed: 10-Sep-2014].

[61] F. Lewis, D. Dawson, and C. Abdallah, *Robot manipulator control: theory and practice*, Second Edi. 2003.

[62] EngineersHandbook.com, “SCARA Robot.” [Online]. Available: http://www.engineershandbook.com/Components/robclassscara.htm. [Accessed: 10-Sep-2014].

[63] Beggs, J. S. (1983). *Kinematics*. CRC Press., 1983.

[64] Lewis, F. L., Abdallah, C. T., & Dawson, D. M.. *Control of robot manipulators* (Vol. 236). New York: Macmillan, (1993).

[65] R. S. Wright and B. Lipchak, *OpenGL SuperBible (3rd Edition)*. Indianapolis, IN, USA: Sams, 2004.

[66] Denavit, J., & Hartenberg, R. S.. Kinematic synthesis of linkages. *Department of Mechanical Engineering and Astronomical Sciences. Northwestern University*, *7(1964)*, (1964).

[67] G. Legnani, F. Casolo, P. Righettini, and B. Zappa, “A homogeneous matrix approach to 3D kinematics and dynamics—I. Theory,” *Mech. Mach. Theory*, vol. 31, no. 5, pp. 573–587, 1996.

[68] P. Corke, *Robotics, vision and control: fundamental algorithms in MATLAB*, vol. 73. Springer, 2011.

[69] Merat, F.. Introduction to robotics: Mechanics and control. *Robotics and Automation, IEEE Journal of*, *vol 3*(2), 166-166, (1987).

[70] Craig, J. J.. *Introduction to robotics: mechanics and control* (pp. 144-146). Upper Saddle River, NJ, USA:: Pearson/Prentice Hall, (2005).

[71] Aristidou, A., & Lasenby, J.. *Inverse kinematics: a review of existing techniques and introduction of a new fast iterative solver. Cambridge University Department of Engineering Technical Report*. CUED/F-INFENG/TR-632. 2009.

[72] Kucuk, S., & Bingul, Z.. Robot kinematics: forward and inverse kinematics. *Industrial Robotics Theory Modelling & Control. ARS/pIV, Germany*, pp 117-148, (2006).

[73] A.-G. Ebrahim, "Robotics system optimal task control (neuro-inverse kinematics approach)," *2006 IEEE GCC Conf.*, pp. 1–5, Mar. 2006.

[74] C. Kuo, Y. Lai, K. Chiu, and S. Lee, "Motion planning and control of interactive humanoid robotic arms," *2008 IEEE Work. Adv. Robot. Its Soc. Impacts*, pp. 1–6, Aug. 2008.

[75] M. Shimizu, H. Kakuya, W.-K. Yoon, K. Kitagaki, and K. Kosuge, "Analytical Inverse Kinematic Computation for 7-DOF Redundant Manipulators With Joint Limits and Its Application to Redundancy Resolution," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1131–1142, Oct. 2008.

[76] N. Vahrenkamp and D. Berenson, "Humanoid motion planning for dual-arm manipulation and re-grasping tasks,"*Robot. Syst.*, no. Section IV, pp. 2464–2470, 2009.

[77] Zhao, J., & Badler, N. I.. *Real time inverse kinematics with joint limits and spatial constraints* (No. MS-CIS-89-09). *Moore school of electrical engineering philadelphia pa graphics lab*, (1989).

[78] F. Burget, A. Hornung, and M. Bennewitz, "Whole-body motion planning for manipulation of articulated objects," *2013 IEEE Int. Conf. Robot. Autom.*, pp. 1656–1662, May 2013.

[79] S. F. M. Assal, S. Member, K. Watanabe, and K. Izumi, "Neural Network-Based Kinematic Inversion of Industrial Redundant Robots Using Cooperative Fuzzy Hint for the Joint Limits Avoidance," vol. 11, no. 5, pp. 593–603, 2006.

[80] Z. M. Li, C. G. Li, and S. J. Lv, "A method for solving inverse kinematics of PUMA560 manipulator based on PSO-RBF network," *2012 8th Int. Conf. Nat. Comput.*, no. Icnc, pp. 298–301, May 2012.

[81] L. Wang, W. Rong, L. Qi, and Z. Qin, "Optimal design of a 6-DOF parallel robot for precise manipulation," *Mechatronics Autom.* , pp. 3933–3937, 2009.

[82] Kenwright, B.. Inverse Kinematics–Cyclic Coordinate Descent (CCD). *Journal of Graphics Tools*, *16*(4), 177-217., 2012.

[83] Muller-Cajar, R., & Mukundan, R.. Triangualation-a new algorithm for inverse kinematics.

[84] M. Fêdor, "Application of inverse kinematics for skeleton manipulation in real-time," in *Proceedings of the 19th spring conference on Computer graphics*, 2003, pp. 203–212, (2007).

[85] T. Uzunovic, E. Golubovic, E. a. Baran, and A. Sabanovic, "Configuration space control of a parallel Delta robot with a neural network based inverse kinematics," *2013 8th Int. Conf. Electr. Electron. Eng.*, pp. 497–501, Nov. 2013.

[86] Y. Kung, K. Tseng, and C. Chen, "FPGA-implementation of inverse kinematics and servo controller for robot manipulator,",2006. ROBIO'06., pp. 1163–1168, 2006.

[87] K. Gac, G. Karpiel, and M. Petko, "FPGA based hardware accelerator for calculations of the parallel robot inverse kinematics," *Proc. 2012 IEEE 17th Int. Conf. Emerg. Technol. Fact. Autom. (ETFA 2012)*, pp. 1–4, Sep. 2012.

[88] C. C. Wong and C. C. Liu, "FPGA realisation of inverse kinematics for biped robot based on CORDIC," *Electron. Lett.*, vol. 49, no. 5, pp. 332–334, Feb. 2013.

[89] Z. Yili, S. Hanxu, J. Qingxuan, and S. Guozhen, "Kinematics control for a 6-DOF space manipulator based on ARM processor and FPGA Co-processor," *2008 6th IEEE Int. Conf. Ind. Informatics*, pp. 129–134, Jul. 2008.

[90] L.-C. Wang and C.-C. Chen, "A combined optimization method for solving the inverse kinematics problems of mechanical manipulators," *Robot. Autom. IEEE Trans.*, vol. 7, no. 4, pp. 489–499, 1991.

[91] Buss, S. R.. Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods. *IEEE Journal of Robotics and Automation*, *17*, 1-19, (2004).

[92] Meredith, M., & Maddock, S.. *Real-time inverse kinematics: The return of the Jacobian*. Technical Report No. CS-04-06, Department of Computer Science, University of Sheffield, (2004).

[93] Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," *J. Dyn. Syst. Meas. Control*, vol. 108, no. 3, pp. 163–171, 1986.

[94] W. Wang, Y. Suga, H. Iwata, and S. Sugano, “Solve inverse kinematics through a new quadratic minimization technique,” *2012 IEEE/ASME Int. Conf. Adv. Intell. Mechatronics*, pp. 306–313, Jul. 2012.

[95] Stimulant.com, “Depth Sensor Shootout: Kinect, Leap, Intel and Duo.” [Online]. Available: http://stimulant.com/depth-sensor-shootout/. [Accessed: 17-Nov-2014].

[96] R. Mukundan and S. Member, “A Fast Inverse Kinematics Solution for an n -link Joint Chain,” no. Icita, pp. 349–354, 2008.

[97] Tanguay, D. O.. *Hidden Markov models for gesture recognition* (Doctoral dissertation, Massachusetts Institute of Technology), (1995).

[98] Shastry, K. R., Ravindran, M., Srikanth, M. V. V. N. S., & Lakshmikhanth, N.. Survey on various gesture recognition techniques for interfacing machines based on ambient intelligence. arXiv preprint arXiv:1012.0084, (2010)

[99] Rabiner, Lawrence. "A tutorial on hidden Markov models and selected applications in speech recognition." *Proceedings of the IEEE* 77.2 (1989): 257-286.

[100] Yang, Jie, and Yangsheng Xu. *Hidden markov model for gesture recognition*. No. CMU-RI-TR-94-10. Carnegie-Mellon Univ Pittsburgh Pa Robotics Inst, 1994.

[101] Forney Jr, G. D.. The viterbi algorithm. *Proceedings of the IEEE*, *61*(3), 268-278, (1973).

[102] L. R. Welch, “Hidden Markov models and the Baum-Welch algorithm,” *IEEE Inf. Theory Soc. Newsl.*, vol. 53, no. 4, pp. 10–13, 2003.

[103] M. Q. Patton, “Qualitative Research,” in *Encyclopedia of Statistics in Behavioral Science*, John Wiley & Sons, Ltd, 2005.

[104] M. Shuttleworth, “Within Subject Design.” [Online]. Available: https://explorable.com/within-subject-design. [Accessed: 11-Jun-2014].

[105] Minitab.com, “Tukeys Comparision Test.” [Online]. Available: http://support.minitab.com/en-us/minitab/17/topic-library/modeling-statistics/anova/multiple-comparisons/what-is-tukey-s-method/.

[106] Mathworks.com, “Wilcoxon rank sum test.” [Online]. Available: http://www.mathworks.com/help/stats/ranksum.html. [Accessed: 01-Dec-2015].

[107] Taylor, M., Blake, A., & Cox, A. . Visually guided grasping in 3D. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on* (pp. 761-766). IEEE, 1994.

[108] N. Vahrenkamp, S. Wieland, P. Azad, D. Gonzalez, T. Asfour, and R. Dillmann, "Visual servoing for humanoid grasping and manipulation tasks," in *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, 2008, pp. 406–412.

[109] NPTEL.ac.in, "Common Robot Configurations." [Online]. Available: http://nptel.ac.in/courses/112103174/module7/lec5/3.html. [Accessed: 13-Nov-2014].

[110] Shirwalkar, S.; Singh, A.; Sharma, K.; Singh, N., "Telemanipulation of an industrial robotic arm using gesture recognition with Kinect," *Control, Automation, Robotics and Embedded Systems (CARE), 2013 International Conference on* , vol., no., pp.1,6, 16-18 Dec. 2013

[111] J. L. Raheja, R. Shyam, U. Kumar, and P. B. Prasad, "Real-Time Robotic Hand Control Using Hand Gestures," *2010 Second Int. Conf. Mach. Learn. Comput.*, pp. 12–16, 2010.

[112] Mishra, A.K.; Meruvia-Pastor, O., "Robot arm manipulation using depth-sensing cameras and inverse kinematics," *Oceans - St. John's, 2014* , vol., no., pp.1,6, 14-19 Sept. 2014. doi: 10.1109/OCEANS.2014.7003029

[113] Mishra, A.K.; Meruvia-Pastor, O, "A New Approach for Robot Arm Manipulation Using Depth Cameras and Inverse Kinematics," *Poster in Nov. Scotia Energy R&D Conf.*, Halifax, CA, April, 2014.

[114] Mishra, A.K.; Meruvia-Pastor, O., "Robot arm manipulation using depth-sensing camera and inverse kinematics," *Poster in AI/GI/CRV, 2014*, Montreal, Canada, 4-9 May. 2014

Appendix A

User Study Documentation

This appendix includes the formal approval received from the Interdisciplinary Committee on Ethics in Human Research (ICEHR) for the user evaluation and the consent form.

Informed Consent Form

Title: *Assessment of gesture recognition performance for the operation of a simulated robotic arm*

Researcher(s): *Akhilesh Kumar Mishra*
Masters Computer Science
Department of Computer Science.
Akm565@mun.ca

Supervisor: Dr. Oscar Meruvia-Pastor
Department of Computer Science.
oscar@mun.ca

You are invited to take part in a research project entitled *"Assessment of gesture recognition performance for the operation of a simulated robotic arm".*

This form is part of the process of informed consent. It should give you the basic idea of what the research is about and what your participation will involve. It also describes your right to withdraw from the study at any time. In order to decide whether you wish to participate in this research study, you should understand enough about its risks and benefits to be able to make an informed decision. This is the informed consent process. Take time to read this carefully and to understand the information given to you. Please contact the researcher, *Akhilesh Kumar Mishra,*

if you have any questions about the study or for more information not included here before you consent.

It is entirely up to you to decide whether to take part in this research. If you choose not to take part in this research or if you decide to withdraw from the research once it has started, there will be no negative consequences for you, now or in the future.

Introduction

As part of my Masters/Honours thesis, I am conducting research under the supervision of Dr. Oscar Meruvia-Pastor. The research is funded by RDC (Research & Development Corporation of Newfoundland and Labrador)

The idea behind this research is to develop an input interface which is based on depth sensing cameras, to control a simulated robotic arm. Current simulators make use of a variety of interfaces to control a robotic arm, such as a commercial controller designed to control the robotic arm or gaming joysticks. In this research we are adding another interface style to control a robotic arm: gesture-base manipulation using depth-sensing cameras, such as Microsoft's Kinect and Intel's Perceptual Computing camera.

Purpose of study:

The purpose of this study is to design a gesture based input module to control a robotic arm simulator. Existing manipulators for robotic arms such as Titan IV are operated via cabled controllers which are relatively expensive. Operators of these controllers require a lot of training and experience before they apply their skills in the field. As the operated arm is deployed under the ocean, training a user to control the arm is expensive and risky. There are, however, computer simulators which are used to train users on particular manipulator arms. However, commercial simulators work using forward kinematics and require a high degree of skill from the operator requiring extensive training. The proposed method in this research enables the user to operate a robotic arm with ease, which requires less training. Also, as the proposed method uses a depth sensing camera to capture the user input, the cost of this method is less.

What you will do in this study:

The users are expected to complete a placement task using a robotic arm simulator connected to one of the three possible operating interfaces: a standard simulation joystick, a standard keyboard control method and a gesture-based control method which used the depth sensing camera.

The Task:

The users will be asked to complete several iterations of a placement task by controlling a simulated robotic arm made of 4 joints and a grabber handle at the very end. In the simulator screen the users will see a robotic arm and several objects strewn on the floor in the vicinity of the robotic arm. The users will be asked to grab an object and place it into a bin using a variety of interfaces.

Upon completion of the tasks, the participants will fill an anonymous feedback form.

To obtain a measure of performance we will gather the task completion times and perform statistical analysis of the results. The task completion times will be recorded by the system automatically: before each trial starts, the controller of the experiment will set an internal timer that will be running until the user completes the placement task.

The participants will be offered $10 for participation in the study as compensation.

Length of time:

The total amount of time that we expect that each user spends using the simulator is about 50 minutes, so 15 minutes per condition, plus 5 minutes to complete a task satisfaction survey at the end of the session.

Withdrawal from the study:

Participation is entirely voluntary and participants can withdraw anytime. As for the compensation, if the participant withdraws in the stage of the first training session, i.e. before starting to try to accomplish the actual tasks from which experimental data will be obtained, no compensation will be provided, as this amounts to a conscious decision of not taking part in the

experiment. If for any reason, a participant who has started doing a task does not complete it because the participant decides to withdraw at that point, full compensation will be provided and the data for that particular task will be purged. However, if any of the above mentioned tasks is completed by the participant, the data gathered up to that point will remain as part of the study and it will be aggregated along with the other data from the other participants and hence it cannot be removed from the study.

Possible benefits:

Students participating in the study will benefit through exposure to the type of experimentation needed for validation of scientific studies, so if they will eventually perform their own user studies, they will have the experience of having participated as subjects in a prior study.

This research will provide information to the scientific community and the public in general about the potential advantages and disadvantages of the use of gesture-based controllers using depth cameras for manipulation tasks which are relevant to a wide variety of applications. In particular, the study will be relevant to the remote operation of a robotic arm, which is an important task where human dexterity and control are necessary to complete a certain task.

Possible risks:

There is a certain risk that some participants might feel upset or frustrated if they are unable to complete the tasks. To reduce the anxiety participants will be assured that they do not need to feel an obligation to complete the experiment and that they will be able to move on to the next trial or next stage of the experiment if they have failed all the trials under a particular condition and wish to continue.

Confidentiality and Storage of Data:

During this study no information about the identity of participants will be used during the conduct of the research or the release of the findings. The only place where the participants' names or identifying information will be recorded is in the informed consent form and in the videos. The participants will be assigned sequential ID numbers in the internal computer systems, statistics analysis and in the release of the findings. In the case of the video recordings there is likelihood

that the face of the participants is captured. When distributing the results of the research we may show snapshots of the experimental setup, but we will not show the face of any participant and will blur any identifying feature in the snapshot. In general, we will not be publishing any sequence of video as part of this research.

The participants will fill an anonymous feedback form and the information on the feedback is the only thing we need for research. Upon completion of the study the informed consent forms and completed surveys will be archived in the office of the Principal Supervisor. These forms will be kept for a minimum of five years and may be destroyed after that. Video recordings will be kept in a secured computer with password protection in the office of the Principal Supervisor and in a secured server of the Department of Computer Science at MUN as a backup. The data will only be accessible to the principal investigator and the supervisor. All data will be retained for a minimum of five years, as required by Memorial University policy on Integrity in Scholarly Research.

Anonymity:

Participant's identity will be kept anonymous. Also, the feedback form which the participants will complete after the experiment will be anonymous. Participants will not be mentioned in the thesis or publication without their explicit permission.

Recording of Data:

To improve our understanding of the limitations of our implementation and inform our conclusions about the research we will record the session using a video-camera that will be placed such that it should capture both the screen that the participant is looking at and the gestures of the participant.

Reporting of Results:

The results of the user study will be used in the thesis and since the data collected from the participants is an anonymous feedback form, there will not be any mention of the personal identity of the participants. We will gather the data from the feedback forms and perform statistical analysis on it to understand the performance of the gesture-based method as compared to a Joystick based manipulation method.

Sharing of Results with Participants:

Upon completion of experiments and thesis submission, the thesis will be available for public viewing. The participants will be informed of the availability of the thesis report by email.

Questions:

You are welcome to ask questions at any time during your participation in this research. If you would like more information about this study,

Please contact: *Akhilesh Kumar Mishra, Email: akm565@mun.ca*

Oscar Meruvia-Pastor, Email : Oscar@mun.ca

The proposal for this research has been reviewed by the Interdisciplinary Committee on Ethics in Human Research and found to be in compliance with Memorial University's ethics policy. If you have ethical concerns about the research (such as the way you have been treated or your rights as a participant), you may contact the Chairperson of the ICEHR at icehr@mun.ca or by telephone at 709-864-2861.

Consent:

Your signature on this form means that:

- You have read the information about the research.
- You have been able to ask questions about this study.
- You are satisfied with the answers to all your questions.
- You understand what the study is about and what you will be doing.
- You understand that you are free to withdraw from the study at any time, without having to give a reason, and that doing so will not affect you now or in the future.
- You understand that if the data gathered upon your withdrawal is not complete, i.e. if you leave without completing any task, the data gathered up to that point will be purged. However, if you complete any one of the tasks, the data will be kept and will not be removed from the study.

If you sign this form, you do not give up your legal rights and do not release the researchers from their professional responsibilities.

Your signature:

I have read what this study is about and understood the risks and benefits. I have had adequate time to think about this and had the opportunity to ask questions and my questions have been answered.

☐ I agree to participate in the research project understanding the risks and contributions of my participation, that my participation is voluntary, and that I may end my participation at any time.

☐ I agree to be video-recorded during the experiment.

☐ I agree to the use of quotations but do not want my name to be identified in any publications resulting from this study.

A copy of this Informed Consent Form has been given to me for my records.

_____ _____

Signature of participant Date

Researcher's Signature:

I have explained this study to the best of my ability. I invited questions and gave answers. I believe that the participant fully understands what is involved in being in the study, any potential risks of the study and that he or she has freely chosen to be in the study.

_____ _____

Signature of Principal Investigator Date

MEMORIAL UNIVERSITY

Interdisciplinary Committee on Ethics in Human Research (ICEHR)

Office of Research Services
St. John's, NL Canada A1C 5S7
Tel: 709 864 2561 Fax: 709 864 4612
www.mun.ca/research

ICEHR Number:	20140556-SC
Approval Period:	October 22, 2013 – October 31, 2015
Funding Agency:	Supervisor's RDC grant [Title: *3D Telepresence for Education, Training and Control*]
Responsible Faculty:	Dr. Oscar Meruvia-Pastor Department of Computer Science, Faculty of Science
Title of Project:	*Assessment of Gesture Recognition Performance for the Operation of a Simulated Robotic Arm*
Amendment #:	*01*

October 16, 2014

Mr. Akhilesh Mishra
Department of Computer Science, Faculty of Science
Memorial University of Newfoundland

Dear Mr. Mishra:

The Interdisciplinary Committee on Ethics in Human Research (ICEHR) has reviewed the proposed revisions for the above referenced project, as outlined in your correspondence dated October 10, 2014, and is pleased to give approval to remove the comparison between first- and third- person control, and to add the additional input interface (i.e. the standard keyboard) as requested, provided all previously approved protocols are followed.

If you need to make any other changes during the conduct of the research that may affect ethical relations with human participants, please forward an amendment request form with a description of these changes to icehr@mun.ca for further review by the Committee.

Your ethics clearance for this project expires October 31, 2015, before which time you must submit an annual update form to ICEHR. If you plan to continue the project, you need to request renewal of your ethics clearance, and include a brief summary on the progress of your research. When the project no longer requires contact with human participants, is completed and/or terminated, you need to provide the annual update form with a final brief summary, and your file will be closed. The annual update form is on the ICEHR website at http://www.mun.ca/research/ethics/humans/icehr/applications/.

The Committee would like to thank you for the update on your proposal and we wish you well with your research.

Yours sincerely,

Gail Wideman

Gail Wideman, Ph.D.
Vice-Chair, Interdisciplinary Committee on Ethics in Human Research

GW/hw

copy: Supervisor – Dr. Oscar Meruvia-Pastor, Department of Computer Science, Faculty of Science

Appendix B

Exit Questionnaire

1) Rate your familiarity with use of keyboard to manipulate a robotic arm.

1	2	3	4	5
(Very familiar)		*(Neutral)*		*(Very unfamiliar)*

2) Rate your familiarity with use of joystick to manipulate a robotic arm.

1	2	3	4	5
(Very familiar)		*(Neutral)*		*(Very unfamiliar)*

3) Rate your familiarity with use of depth camera to manipulate a robotic arm.

1	2	3	4	5
(Very familiar)		*(Neutral)*		*(Very unfamiliar)*

4) Rate each input method for ease of learning. Was the operation of the input method easy to learn?

a) Joystick

1	2	3	4	5
(Strongly Disagree)		*(Neutral)*		*(Strongly Agree)*

b) Keyboard

1	2	3	4	5
(Strongly Disagree)		*(Neutral)*		*(Strongly Agree)*

c) Depth Camera

1	2	3	4	5
(Strongly Disagree)		*(Neutral)*		*(Strongly Agree)*

5) Rate each input based on performance. Did the input perform appropriately?

a) Joystick

1	2	3	4	5
(Strongly Disagree)		*(Neutral)*		*(Strongly Agree)*

b) Keyboard

1	2	3	4	5
(Strongly Disagree)		*(Neutral)*		*(Strongly Agree)*

c) Depth Camera

1	2	3	4	5
(Strongly Disagree)		*(Neutral)*		*(Strongly Agree)*

6) Rate each input based on ease of use. Was the input easy to use?

a) Joystick

1	2	3	4	5
(Strongly Disagree)		*(Neutral)*		*(Strongly Agree)*

b) Keyboard

1	2	3	4	5
(Strongly Disagree)		*(Neutral)*		*(Strongly Agree)*

c) Depth Camera

1	2	3	4	5
(Strongly Disagree)		*(Neutral)*		*(Strongly Agree)*

7) Rate each input for ergonomics. Was the input comfortable to operate?

a) Joystick

1	2	3	4	5
(Strongly Disagree)		*(Neutral)*		*(Strongly Agree)*

b) Keyboard

1	2	3	4	5
(Strongly Disagree)		*(Neutral)*		*(Strongly Agree)*

c) Depth Camera

1	2	3	4	5
(Strongly Disagree)		*(Neutral)*		*(Strongly Agree)*

8) Rate each input for its usefulness. Is the input useful for the given tasks?

a) Joystick

1	2	3	4	5
(Strongly Disagree)		*(Neutral)*		*(Strongly Agree)*

b) Keyboard

1	2	3	4	5
(Strongly Disagree)		*(Neutral)*		*(Strongly Agree)*

c) Depth Camera

1	2	3	4	5
(Strongly Disagree)		*(Neutral)*		*(Strongly Agree)*

9) Which input type is best suited for Task 1?

a) Joystick

b) Keyboard

c) Depth Camera

10) Which input type is best suited for Task 2?

a) Joystick

b) Keyboard

c) Depth Camera

11) Which input method requires least amount of training?

a) Joystick

b) Keyboard

c) Depth Camera

12) Which input method performs best after training & practice?

a) Joystick

b) Keyboard

c) Depth Camera

13) Would you prefer to use the depth camera interface over joystick?

1	2	3	4	5
(Strongly Disagree)		*(Neutral)*		*(Strongly Agree)*

14) Would you prefer to use the depth camera interface over keyboard?

1	2	3	4	5
(Strongly Disagree)		*(Neutral)*		*(Strongly Agree)*

15) Would you prefer to use the joystick interface over keyboard?

1	2	3	4	5
(Strongly Disagree)		*(Neutral)*		*(Strongly Agree)*

16) Which input type would you prefer overall?

a) Joystick

b) Keyboard

c) Depth Camera

17) Is there anything else you would like to comment on?

……

……

……

……

……

……

……

……

……

……

Appendix C

ANOVA Tests and Results

The following statistical analysis is the work of Dr. Lourdes Pena-Castillo. We would like to thank her for her contribution to data analysis.

We have a balanced study with 18 users, 3 input devices, 10 trials per user and 6 alternative orderings of the input methods. Execution time in seconds was measured for every user per trial.

Number of observations	Per Input	Per Order	Per Trial	Per Input-Order combination
540	180	180	54	60

Data for each of the two tasks was analyzed independently. Statistical threshold for significance was set to 0.01. Data was tested for homogeneity of variance using the Bartlett's test and for normality using the Shapiro-Wilk test. There was a significant difference in the variances, and the distribution of the execution times significantly deviated from normality for both tasks. Therefore non-parametric tests were applied in addition to two-way ANOVA.

Results task 1

Anova model: Time ~ Input * Order
Table of means of execution times

```
# Grand mean
# 21.38333

# Input
# Keyboard   Camera Joystick
  # 17.333   18.083   28.733

# Order
   # 1st    2nd    3rd
# 21.761 21.683 20.706

# Input:Order
          # Order
# Input      1st    2nd    3rd
  # Keyboard 19.483 20.350 12.167
  # Camera   15.933 17.050 21.267
  # Joystick 29.867 27.650 28.683
```

```
# Standard errors for differences of means
        # Input Order Input:Order
        # 1.363 1.363       2.361
# replic.   180   180          60
```

ANOVA results

```
             # Df Sum Sq Mean Sq F value   Pr(>F)
# Input         2  14637    7318  43.750  < 2e-16 ***
# Order         2    125      62   0.372 0.689281
# Input:Order   4   3398     849   5.078 0.000506 ***
# Residuals   531  88825     167
# ---
# Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Conclusion: The input device has a very significant effect in the execution time and there is a significant interaction between Input and Order.

```
# Tukey multiple comparisons of means
# 95% family-wise confidence level
# Fit: aov(formula = Time ~ Input * Order, data = t1_r4A)
# $Input
                    # diff      lwr      upr    p adj
# Camera-Keyboard    0.75 -2.45422  3.95422 0.846458
# Joystick-Keyboard 11.40  8.19578 14.60422 0.000000 ***
# Joystick-Camera   10.65  7.44578 13.85422 0.000000 ***
# $Order
                # diff       lwr      upr     p adj
# 2nd-1st -0.07777778 -3.281998 3.126443 0.9982072
# 3rd-1st -1.05555556 -4.259776 2.148665 0.7190090
# 3rd-2nd -0.97777778 -4.181998 2.226443 0.7534191
# $`Input:Order`
                                    # diff          lwr          upr     p
adj
# Camera:1st-Keyboard:1st    -3.5500000 -10.90472047   3.80472047 0.8538528
# Joystick:1st-Keyboard:1st  10.3833333   3.02861286  17.73805381 0.0004487 ***
# Keyboard:2nd-Keyboard:1st   0.8666667  -6.48805381   8.22138714 0.9999907
# Camera:2nd-Keyboard:1st    -2.4333333  -9.78805381   4.92138714 0.9828192
# Joystick:2nd-Keyboard:1st   8.1666667   0.81194619  15.52138714 0.0169584
# Keyboard:3rd-Keyboard:1st  -7.3166667 -14.67138714   0.03805381 0.0524087
# Camera:3rd-Keyboard:1st     1.7833333  -5.57138714   9.13805381 0.9979009
# Joystick:3rd-Keyboard:1st   9.2000000   1.84527953  16.55472047 0.0035058 ***
# Joystick:1st-Camera:1st    13.9333333   6.57861286  21.28805381 0.0000002 ***
# Keyboard:2nd-Camera:1st     4.4166667  -2.93805381  11.77138714 0.6345377
# Camera:2nd-Camera:1st       1.1166667  -6.23805381   8.47138714 0.9999344
# Joystick:2nd-Camera:1st    11.7166667   4.36194619  19.07138714 0.0000330 ***
# Keyboard:3rd-Camera:1st    -3.7666667 -11.12138714   3.58805381 0.8073634
# Camera:3rd-Camera:1st       5.3333333  -2.02138714  12.68805381 0.3694984
# Joystick:3rd-Camera:1st    12.7500000   5.39527953  20.10472047 0.0000036 ***
# Keyboard:2nd-Joystick:1st  -9.5166667 -16.87138714  -2.16194619 0.0020740 ***
# Camera:2nd-Joystick:1st   -12.8166667 -20.17138714  -5.46194619 0.0000031 ***
```

```
# Joystick:2nd-Joystick:1st  -2.2166667  -9.57138714   5.13805381 0.9906320
# Keyboard:3rd-Joystick:1st -17.7000000 -25.05472047 -10.34527953 0.0000000 ***
# Camera:3rd-Joystick:1st    -8.6000000 -15.95472047  -1.24527953 0.0089880 ***
# Joystick:3rd-Joystick:1st  -1.1833333  -8.53805381   6.17138714 0.9998980
# Camera:2nd-Keyboard:2nd    -3.3000000 -10.65472047   4.05472047 0.8986274
# Joystick:2nd-Keyboard:2nd   7.3000000  -0.05472047  14.65472047 0.0534943
# Keyboard:3rd-Keyboard:2nd  -8.1833333 -15.53805381  -0.82861286 0.0165612
# Camera:3rd-Keyboard:2nd     0.9166667  -6.43805381   8.27138714 0.9999856
# Joystick:3rd-Keyboard:2nd   8.3333333   0.97861286  15.68805381 0.0133455
# Joystick:2nd-Camera:2nd    10.6000000   3.24527953  17.95472047 0.0002997 ***
# Keyboard:3rd-Camera:2nd    -4.8833333 -12.23805381   2.47138714 0.4964849
# Camera:3rd-Camera:2nd       4.2166667  -3.13805381  11.57138714 0.6917429
# Joystick:3rd-Camera:2nd    11.6333333   4.27861286  18.98805381 0.0000392 ***
# Keyboard:3rd-Joystick:2nd -15.4833333 -22.83805381  -8.12861286 0.0000000 ***
# Camera:3rd-Joystick:2nd    -6.3833333 -13.73805381   0.97138714 0.1490789
# Joystick:3rd-Joystick:2nd   1.0333333  -6.32138714   8.38805381 0.9999638
# Camera:3rd-Keyboard:3rd     9.1000000   1.74527953  16.45472047 0.0041215 ***
# Joystick:3rd-Keyboard:3rd  16.5166667   9.16194619  23.87138714 0.0000000 ***
# Joystick:3rd-Camera:3rd     7.4166667   0.06194619  14.77138714 0.0462804
```

Conclusion: Both camera and keyboard have significantly shorter execution times than joystick. On average keyboard execution times are 11.40 seconds shorter and camera execution times are 10.65 seconds shorter than those of joystick. There is no significant difference between the execution times of camera and keyboard.

Results non-parametric tests

As the data deviated from normality and violated the assumption of homogeneity of variances, a Kruskal-Wallis rank sum test (which is a non-parametric test) of the null that the location parameters of the distribution of the execution times are the same for each input device was performed. The conclusion is that the execution times per input device significantly differ. Execution times are not significantly different when grouped based on order or trial.

```
# Kruskal-Wallis rank sum test
# data:  Time by Input
# Kruskal-Wallis chi-squared = 135.9753, df = 2, p-value < 2.2e-16


# #   Kruskal-Wallis rank sum test
# data:  Time by Order
# Kruskal-Wallis chi-squared = 1.6793, df = 2, p-value = 0.4319

# #   Kruskal-Wallis rank sum test
# data:  Time by Trial
# Kruskal-Wallis chi-squared = 12.4988, df = 9, p-value = 0.1866
```

To account for the effect of order and trial in the execution times, a Friedman rank sum test with unreplicated blocked data was performed. This test can be used instead of two-way ANOVA when the normality

assumption may be violated. The null hypothesis is that apart from the effect of order and trial, the location parameter of the distribution of the execution times per input device is the same. Conclusion: Execution times per input device significantly differ.

```
 # Friedman rank sum test

# data:  tmp$x, tmp$Group.1 and tmp$block
# Friedman chi-squared = 39.4667, df = 2, p-value = 2.691e-09
```

Differences in the execution times per input were also tested using a Wilcoxon signed rank test of the null that the mean of the differences of ranks of the execution times between two samples is different from zero. Same conclusion as above; namely, both camera and keyboard have highly significant shorter execution times than joystick.

```
# Pairwise comparisons using paired Wilcoxon rank sum test
# data:  Time and Input
         # Keyboard Camera
# Camera   0.047    -
# Joystick <2e-16   <2e-16
# P value adjustment method: BH
```

Differences in the execution times per order for input device were also tested using a Wilcoxon signed rank test. Conclusion: Execution times for keyboard – 3rd order are significantly shorter than those for keyboard in the 1st or 2nd order. Execution times for camera – 3rd order are in the threshold of being significantly longer than those for camera in the 1st or 2nd order.

```
## keyboard
# #   Pairwise comparisons using Wilcoxon rank sum test
# data:  Time and Order
    # 1st     2nd
# 2nd 0.16    -
# 3rd 8.4e-08 8.2e-06
# P value adjustment method: BH

#Camera
# #   Pairwise comparisons using Wilcoxon rank sum test
# data:  Time and Order
    # 1st   2nd
# 2nd 0.299 -
# 3rd 0.030 0.064
# P value adjustment method: BH
```

Results task 2

Trial was included as a factor for the analysis of task 2 based on the interaction plots.

Anova model: Time ~ Input * Order + Trial
Table of means of execution times

```
# Grand mean

# 32.77593

# Input
# Keyboard   Camera Joystick
# 29.59    30.75    37.98
# Order
  # 1st   2nd   3rd
# 33.88 31.72 32.73

# Trial
    # 0     1     2     3     4     5     6     7     8     9
# 35.46 38.04 30.69 38.04 32.59 31.85 33.24 27.83 29.93 30.09

# Input:Order
          # Order
# Input      1st   2nd   3rd
  # Keyboard 32.23 30.70 25.85
  # Camera   29.82 26.98 35.45
  # Joystick 39.60 37.47 36.88

# Standard errors for differences of means
        # Input Order Trial Input:Order
        # 1.683 1.683 3.073       2.915
# replic.   180   180    54          60
```

ANOVA results

```
             # Df Sum Sq Mean Sq F value   Pr(>F)
# Input         2   7442    3721  14.597 6.78e-07 ***
# Order         2    423     212   0.830  0.43664
# Trial         9   5821     647   2.538  0.00742 **
# Input:Order   4   3384     846   3.319  0.01064 *
# Residuals   522 133062     255
# ---
# Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Conclusion: The input device has a very significant effect in the execution time, trial also has a significant effect in the execution time, and there is a slightly significant interaction between Input and Order (p-value < 0.05).

```
  # Tukey multiple comparisons of means
    # 95% family-wise confidence level

# Fit: aov(formula = Time ~ Input * Order + Trial, data = t2_r4A)

# $Input
                      # diff       lwr       upr     p adj
# Camera-Keyboard   1.155556 -2.800068  5.111179 0.7714063
# Joystick-Keyboard 8.388889  4.433265 12.344512 0.0000025 ***
# Joystick-Camera   7.233333  3.277710 11.188957 0.0000610 ***

# $Order
             # diff       lwr      upr     p adj
# 2nd-1st -2.166667 -6.122290 1.788957 0.4028919
# 3rd-1st -1.155556 -5.111179 2.800068 0.7714063
# 3rd-2nd  1.011111 -2.944512 4.966735 0.8197306

# $Trial
              # diff        lwr        upr     p adj
# 1-0  2.574074e+00  -7.189140 12.3372886 0.9979510
# 2-0 -4.777778e+00 -14.540992  4.9854368 0.8688672
# 3-0  2.574074e+00  -7.189140 12.3372886 0.9979510
# 4-0 -2.870370e+00 -12.633585  6.8928442 0.9952892
# 5-0 -3.611111e+00 -13.374326  6.1521034 0.9758109
# 6-0 -2.222222e+00 -11.985437  7.5409923 0.9993615
# 7-0 -7.629630e+00 -17.392844  2.1335849 0.2800093
# 8-0 -5.537037e+00 -15.300252  4.2261775 0.7338139
# 9-0 -5.370370e+00 -15.133585  4.3928442 0.7673615
# 2-1 -7.351852e+00 -17.115066  2.4113627 0.3322877
# 3-1  2.131628e-14  -9.763215  9.7632145 1.0000000
# 4-1 -5.444444e+00 -15.207659  4.3187701 0.7526823
# 5-1 -6.185185e+00 -15.948400  3.5780293 0.5902117
# 6-1 -4.796296e+00 -14.559511  4.9669182 0.8661692
# 7-1 -1.020370e+01 -19.966918 -0.4404892 0.0322523
# 8-1 -8.111111e+00 -17.874326  1.6521034 0.2019091
# 9-1 -7.944444e+00 -17.707659  1.8187701 0.2270790
# 3-2  7.351852e+00  -2.411363 17.1150664 0.3322877
# 4-2  1.907407e+00  -7.855807 11.6706219 0.9998176
# 5-2  1.166667e+00  -8.596548 10.9298812 0.9999973
# 6-2  2.555556e+00  -7.207659 12.3187701 0.9980630
# 7-2 -2.851852e+00 -12.615066  6.9113627 0.9955126
# 8-2 -7.592593e-01 -10.522474  9.0039553 0.9999999
# 9-2 -5.925926e-01 -10.355807  9.1706219 1.0000000
# 4-3 -5.444444e+00 -15.207659  4.3187701 0.7526823
# 5-3 -6.185185e+00 -15.948400  3.5780293 0.5902117
# 6-3 -4.796296e+00 -14.559511  4.9669182 0.8661692
# 7-3 -1.020370e+01 -19.966918 -0.4404892 0.0322523
# 8-3 -8.111111e+00 -17.874326  1.6521034 0.2019091
# 9-3 -7.944444e+00 -17.707659  1.8187701 0.2270790
# 5-4 -7.407407e-01 -10.503955  9.0224738 1.0000000
```

```
# 6-4  6.481481e-01  -9.115066 10.4113627 1.0000000
# 7-4 -4.759259e+00 -14.522474  5.0039553 0.8715329
# 8-4 -2.666667e+00 -12.429881  7.0965479 0.9973070
# 9-4 -2.500000e+00 -12.263215  7.2632145 0.9983688
# 6-5  1.388889e+00  -8.374326 11.1521034 0.9999877
# 7-5 -4.018519e+00 -13.781733  5.7446960 0.9516837
# 8-5 -1.925926e+00 -11.689140  7.8372886 0.9998023
# 9-5 -1.759259e+00 -11.522474  8.0039553 0.9999072
# 7-6 -5.407407e+00 -15.170622  4.3558071 0.7600699
# 8-6 -3.314815e+00 -13.078029  6.4483997 0.9865724
# 9-6 -3.148148e+00 -12.911363  6.6150664 0.9907033
# 8-7  2.092593e+00  -7.670622 11.8558071 0.9996083
# 9-7  2.259259e+00  -7.503955 12.0224738 0.9992705
# 9-8  1.666667e-01  -9.596548  9.9298812 1.0000000

# $`Input:Order`
                               # diff         lwr        upr     p adj
# Camera:1st-Keyboard:1st    -2.4166667 -11.4963090  6.6629757 0.9959633
# Joystick:1st-Keyboard:1st   7.3666667  -1.7129757 16.4463090 0.2213211
# Keyboard:2nd-Keyboard:1st  -1.5333333 -10.6129757  7.5463090 0.9998528
# Camera:2nd-Keyboard:1st    -5.2500000 -14.3296424  3.8296424 0.6815630
# Joystick:2nd-Keyboard:1st   5.2333333  -3.8463090 14.3129757 0.6853643
# Keyboard:3rd-Keyboard:1st  -6.3833333 -15.4629757  2.6963090 0.4138087
# Camera:3rd-Keyboard:1st     3.2166667  -5.8629757 12.2963090 0.9736162
# Joystick:3rd-Keyboard:1st   4.6500000  -4.4296424 13.7296424 0.8073063
# Joystick:1st-Camera:1st     9.7833333   0.7036910 18.8629757 0.0237655
# Keyboard:2nd-Camera:1st     0.8833333  -8.1963090  9.9629757 0.9999979
# Camera:2nd-Camera:1st      -2.8333333 -11.9129757  6.2463090 0.9882149
# Joystick:2nd-Camera:1st     7.6500000  -1.4296424 16.7296424 0.1789728
# Keyboard:3rd-Camera:1st    -3.9666667 -13.0463090  5.1129757 0.9118480
# Camera:3rd-Camera:1st       5.6333333  -3.4463090 14.7129757 0.5913537
# Joystick:3rd-Camera:1st     7.0666667  -2.0129757 16.1463090 0.2728980
# Keyboard:2nd-Joystick:1st  -8.9000000 -17.9796424  0.1796424 0.0597453
# Camera:2nd-Joystick:1st   -12.6166667 -21.6963090 -3.5370243 0.0006061 ***
# Joystick:2nd-Joystick:1st  -2.1333333 -11.2129757  6.9463090 0.9983212
# Keyboard:3rd-Joystick:1st -13.7500000 -22.8296424 -4.6703576 0.0001066 ***
# Camera:3rd-Joystick:1st    -4.1500000 -13.2296424  4.9296424 0.8884558
# Joystick:3rd-Joystick:1st  -2.7166667 -11.7963090  6.3629757 0.9910684
# Camera:2nd-Keyboard:2nd    -3.7166667 -12.7963090  5.3629757 0.9382268
# Joystick:2nd-Keyboard:2nd   6.7666667  -2.3129757 15.8463090 0.3310959
# Keyboard:3rd-Keyboard:2nd  -4.8500000 -13.9296424  4.2296424 0.7683111
# Camera:3rd-Keyboard:2nd     4.7500000  -4.3296424 13.8296424 0.7882297
# Joystick:3rd-Keyboard:2nd   6.1833333  -2.8963090 15.2629757 0.4598541
# Joystick:2nd-Camera:2nd    10.4833333   1.4036910 19.5629757 0.0105752
# Keyboard:3rd-Camera:2nd    -1.1333333 -10.2129757  7.9463090 0.9999855
# Camera:3rd-Camera:2nd       8.4666667  -0.6129757 17.5463090 0.0899322
# Joystick:3rd-Camera:2nd     9.9000000   0.8203576 18.9796424 0.0208636
# Keyboard:3rd-Joystick:2nd -11.6166667 -20.6963090 -2.5370243 0.0024836 ***
# Camera:3rd-Joystick:2nd    -2.0166667 -11.0963090  7.0629757 0.9988797
# Joystick:3rd-Joystick:2nd  -0.5833333  -9.6629757  8.4963090 0.9999999
# Camera:3rd-Keyboard:3rd     9.6000000   0.5203576 18.6796424 0.0290486
# Joystick:3rd-Keyboard:3rd  11.0333333   1.9536910 20.1129757 0.0053442 ***
# Joystick:3rd-Camera:3rd     1.4333333  -7.6463090 10.5129757 0.9999117
```

Conclusion: Both camera and keyboard have significantly shorter execution times than joystick. On average keyboard execution times are 8.40 seconds shorter and camera execution times are 7.2 seconds shorter than those of joystick. There is no significant difference between the execution times of camera and keyboard. For task 2, order is not such a significant effect as for task 1.

Results non-parametric tests

Kruskal-Wallis test's conclusion is that the execution times per input device significantly differ. Execution times are not significantly different when grouped based on order, but they are in the threshold of significance when grouped by trial.

```
#     Kruskal-Wallis rank sum test
# data:  Time by Input
# Kruskal-Wallis chi-squared = 43.3175, df = 2, p-value = 3.924e-10

# Kruskal-Wallis rank sum test
# data:  Time by Order
# Kruskal-Wallis chi-squared = 2.5309, df = 2, p-value = 0.2821

#     Kruskal-Wallis rank sum test
# data:  Time by Trial
# Kruskal-Wallis chi-squared = 21.0169, df = 9, p-value = 0.01258
```

Based on the Friedman test, execution times per input device significantly differ even when the effect of order and trial is accounted for.

```
# #   Friedman rank sum test
# data:  tmp$x, tmp$Group.1 and tmp$block
# Friedman chi-squared = 20.0667, df = 2, p-value = 4.391e-05
```

Based on paired Wilcoxon test, both camera and keyboard have highly significant shorter execution times than joystick. Camera execution times are in the threshold of significance for being larger than those of keyboard.

```
# Pairwise comparisons using Wilcoxon signed rank test
# data:  Time and Input
         # Keyboard Camera
# Camera   0.018    -
# Joystick 1.5e-07  3.1e-06
# P value adjustment method: BH
```

Differences in the execution times per order for input device were also tested using a Wilcoxon signed rank test. Conclusion: Execution times

for keyboard – 3rd order are in the threshold of being significantly shorter than those for keyboard in the 1st or 2nd order. Execution times for camera – 3rd order are significantly longer than those for camera in the 2nd order.

Keyboard
Pairwise comparisons using Wilcoxon rank sum test
data: Time and Order
    # 1st   2nd
2nd 0.498 -
3rd 0.013 0.013
P value adjustment method: BH

Camera
Pairwise comparisons using Wilcoxon rank sum test
data: Time and Order
    # 1st   2nd
2nd 0.9734 -
3rd 0.1571 0.0088
P value adjustment method: BH

Appendix D

Instructions

You are expected to perform two tasks with each of the input method which is, a keyboard, joystick, depth camera.

Task 1: You are just expected to reach the object and grab it by the commands below. The commands are different for each input method.

- **Trial runs:** You will be given a trial time. You can do 5 iterations of task 1 in trial.
- **Experiment runs:** There will be 10 iterations of each task.
  So you will have to reach the target using one of the input devices and grab it. This repeats 10 times.

Task 2: You are expected to reach the object, grab it and drop it into the bin.

- Trial runs: You will be given a trial time. You can do 5 iterations of task 2 in trial.
- Experiment runs: There will be 10 iterations for this task as well.
  So you will have to reach the target using one of the input devices and grab it, then navigate to the bin and drop the object. This repeats 10 times.

Commands to pick (task 1) and drop (task 2) an object.

1. *Keyboard*
   For picking an object, press the Key "P".

   For dropping an object, press the key "O"

2. Joystick

   You can pick and drop the objects using the same key in joystick. The key is marked on the joystick.

3. Depth camera.
   For picking something when using the depth camera, perform a peace gesture.

Figure 2: Peace gesture to pick an object

Figure 2: Thumbs UP gesture to drop an object

Toggle Control: Press the key "T" when you want to start/stop the camera control.

4. **End-Effector Colors**

   The end-effector changes the color to red when an object can be picked and changes to blue when an object has been picked.

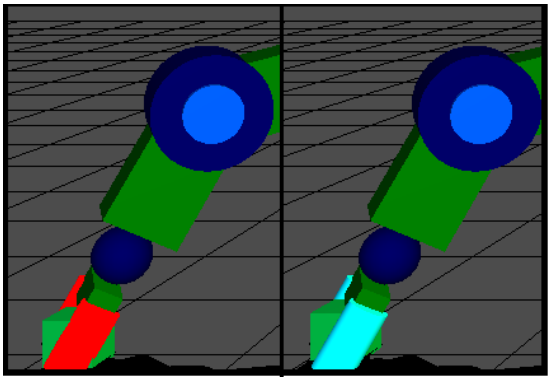

Figure 3: Red: Object can be picked. Blue: Object has been picked.