RoSTAR: ROS-based Telerobotic Control via Augmented Reality

Chung Xue Er (Shamaine)

Supervised **by Dr Niall Murray** and **Dr Yuansong Qiao**

THESIS

Submitted to the Faculty of Engineering & Informatics, Athlone Institute of Technology, Athlone, Co. Westmeath, the Republic of Ireland in the partial fulfilment of the requirements for the degree of

Master of Science by Research

Athlone Institute of Technology

(September 2021)

Abstract

Real-world virtual world communication and interaction will be a cornerstone of future intelligent manufacturing ecosystems. Human robotic interaction is considered to be a fundamental element of factories of the future (FoF). Despite the advancement of different technologies such as wearables and new human-machine interfaces, human-robot interaction (HRI) is still extremely challenging. Whilst progress has been made in the development of different mechanisms to support HRI, and there are issues with cost, naturalistic and intuitive interaction, and communication across heterogeneous systems. To mitigate these limitations, RoSTAR is proposed.

RoSTAR is a novel open-source HRI system based on the Robot Operating System (ROS) and Augmented Reality (AR). RoSTAR enables the user to interact and communicate with an ABB robotic arm (both real and virtual) with a Microsoft HoloLens 2 headset. A computer vision-based framework called Vuforia SDK is used to track and register the virtual robotic arm at a fixed position via the HoloLens 2 front-facing camera's coordinate system. RoSTAR allows the user to teleoperate the ROS-enabled ABB robotic arm seamlessly from a distance through the method of augmented trajectories. Bezier Interpolation is used to generate a smooth and continuous curve for more complex paths following tasks such as robotic gluing, dispensing and arc welding as part of an interoperable, low cost, portable and naturalistically interactive experience.

This thesis also presents the results of various experimental studies performed on RoSTAR in terms of: accuracy; time; and frame rate to attest the efficiency and effectiveness of the proposed system. Overall, the results indicate that RoSTAR is a very promising solution for complex paths following tasks with high position accuracy. We also verify that the HoloLens 2 is capable of displaying significant 3D objects and performing the path planning task in real-time without any performance downscale. Finally, limitations in terms of teleoperating the ROS-enabled ABB robotic arm in terms of path accuracy are identified and discussed.

Acknowledgements

This research work was collaborative between Robotics and Drives Services (RDS) Ltd., the SFI Confirm Centre for Smart Manufacturing and the Athlone Institute of Technology. In particular, I would like to thank RDS along with CONFIRM Smart Manufacturing, Ireland for funding the research. I would like to acknowledge my supervisor, Dr Niall Murray, for providing me with the opportunity to pursue this work. Thank you for all the proofreading, editing, general support and keeping my spirit up by pushing, pulling, or prodding me when required. I would also like to show gratitude to Ken McNevin and John Henry from Robotics and Drives for providing me with the necessary resources to successfully finish my project in the lab. A special thanks to my co-supervisor, Dr Yuansong Qiao, for guiding me through this thesis. Without him, this thesis could not have been possible.

Finally, a big thank you goes to my friends and family for the support and encouragement along the way.

Affirmation

This master degree report, RoSTAR: ROS-based Telerobotic Control via Augmented Reality, was written as part of the master degree work needed to obtain a Master of Science by Research in Software Engineering program at Athlone Institute of Technology. All material in this report that is not my own is well identified and used appropriately and correctly. The central part of the work included in this degree project has not previously been published or used for obtaining another degree.

[illegible]

22/09/2021

Signature by the author

Date

Chung Xue Er (Shamaine)

Table of Contents

LISTS OF ABBREVIATIONS

- **AR** - Augmented Reality
- **AV -** Augmented Virtuality
- **CCD** - Couple-Charged Device
- **CMOS -** Complementary Metal- Oxide- Semiconductor
- **CPPS -** Cyber-Physical Production Systems
- **EMG** – Electromyography
- **FABRIK** - Forward and Backward Reaching Inverse Kinematics
- **FoF** - Factories of Future
- **FoV** - Field of View
- **HMD -** Head Mounted Display
- **HMI** - Human-Machine Interface
- **HPU -** Holographic Processing Unit
- **HRI -** Human-Robot Interaction
- **IMU -** Inertial Measurement Unit
- **IoT -** Internet of Things
- **IR -** Infra-Red
- **LMC -** Leap Motion Controller
- **MEMS -** Micro-Electro-Mechanical-System Sensors
- **MR -** Mixed Reality
- **MRTK** - Mixed Reality Toolkit
- **ROS** - Robot Operating System
- **RV -** Reality-Virtuality
- **ToF -** Time-of-Flight
- **URDF** - Universal Robotic Description Format
- **VE -** Virtual Environment
- **VR -** Virtual Reality
- **XML** - Extensible Markup Language

LISTS OF FIGURES

LISTS OF TABLES

1 INTRODUCTION

The fourth industrial revolution often referred to as Industry 4.0, is a general term for a new industrial model [1]. Industry 4.0 aims at building cyber-physical production systems (CPPS) (which comprise intelligent, real-time-capable, networked sensors and actuators) that unite both digital and physical worlds to make manufacturing increasingly smart by utilising the internet of things (IoT)[1][2]. Industry 4.0 [3] includes a range of digital technologies that includes but is not limited to Augmented Reality (AR), Virtual Reality (VR), Digital Twins (DT), predictive maintenance, cloud computing, Internet of Things (IoT), Artificial Intelligence (AI), and big data. This stack is essential in achieving smart factories of the future (FoF).

However, little research has been conducted to develop a user-centred HRI in the context of industry 4.0 environment requirements. HRI is the study of how humans communicate with robotic systems. It informs us on how to best design, evaluate, understand and implement robotic systems that are competent enough for carrying out collaborative tasks with the human stakeholder [4][5]. One of the major challenges encountered by HRI is that both humans and robots use extremely distinct methods of communication and data representation [6]. Current work in HRI seeks to discover a less training-intensive, more direct, safe, effective, natural and stress-free interaction [7].

Teleoperation or often defined as telerobotic [8] when controlling robotics systems, is commonly used in human-robot interaction (HRI) research. It is an approach that allows human operators to control machines or robotic systems from a distance. This approach has been intensively developed in the last decade. Teleoperated robots are mostly used in telesurgery [9], military search and rescue tasks [10], mining [11] as well as space exploration [12]. According to [13], teleoperated robots helps to reduce the risk and costs associated with human exposure to life-threatening conditions. The capabilities spectrum for telerobotic systems has been expanded from traditional 2D computer interfaces to immersive media-based [14] VR and AR controlled teleoperated robots and has attracted considerable attention in recent years [15][16]. AR provides more spatial awareness for the users if compared with the more immersive VR experience that may cause them to lose connection with the real-world surroundings.

The novel approach of using Augmented Reality (AR) creates a new design space and an opportunity for facilitating more naturalistic human-robotic interaction for applications such as teleoperation in real-time [17]. The key inspiration is that AR has the potential of displaying visual objects directly (from the field of view of the robotic system) into the field of view of the user without being disconnected from the physical world. A clear communication orientation and information visualisation are needed to mediate robot teleoperation [18][19]. On account of the ability of AR to augment the operator view and visualise the robotic corresponding task space, this allows the operator to retrieve the robot's sensor state information (temperature level, position, joint angle etc) via graphic overlays result in AR. This provides great potential as an interface for robotic teleoperation [20].

To the best of our knowledge, no studies have been performed to thoroughly analyse AR interfaces' effectiveness and accuracy in more complex path following tasks.

1.1 PROBLEM STATEMENT

In the context of the above-mentioned scope, the research question defined for this work is: *"Can augmented reality be used as an accurate and viable HRI tool in industry 4.0 scenarios?"*

This primary research question leads us to several sub-questions:

- What is an appropriate system architecture and what components are required to enable HRI with simulated and real robotic systems? (Appropriate System Architecture and its components can be found in Chapter Three)
- How can we embed safety within an AR-enabled HRI? (Safety feature is addressed in Chapter 3.1.3)
- How accurate is communication via AR with a real robotic system? (Robot accuracy results were mentioned in Chapter 4.2.1)
- Which is the most suitable 3D robotic simulator for this project? (Comparison descriptions can be found in Chapter 2.5.2)
- Which Inverse Kinematics solvers are the most efficient? (Results obtained in Chapter 4.2.2)

- What are the limitations with existing Human-Robot Interaction Interfaces? (Limitation of proposed HRI system can be found in Chapter 5.2)

1.2 RESEARCH AIM

The aim of this research is to design and develop an end-to-end system based on AR technology to facilitate an interactive human-robot interaction workflow for smart factories of the future (FoF). In this system, the human operator is able to plan robotic trajectories at a safe distance in hazardous environments through an AR interface. Once the planned robotic trajectory is verified, the trajectory messages are sent from Unity to ROS via a rosbridge server and executed by the simulated or real robotic system. We also aim to evaluate the AR interfaces' effectiveness and accuracy in more complex path following tasks.

The key research objectives for this work are:

1. Perform a state-of-the-art critique of
    I. Augmented Reality and HRI's.
    II. Teleoperation or Telerobotic Control.
    III. Robot Motion Planning Techniques.
2. Compare, analyse, and survey existing Human-Robot interaction methods (AR-related) with a focus on less training-intensive, more direct, safe, effective, and natural interaction.
3. Evaluate, analyse, and research a suitable 3D robotic simulator for the project.
4. Design and develop a system prototype to incorporate current technologies outlined in steps (1) & (2) that can facilitate communication from simulator to a real robot system.
5. Design and execute a testing strategy that evaluates the aforementioned system across a range of parameters such as:
    I. The trajectory accuracy between the HoloLens 2 and ABB robot arm.

II. The level of trajectory complexity is based on the number of waypoints/ shapes.

III. The HoloLens 2 stability (Frames per second) for dealing with the huge amount of real-time data.

IV. The robot arm trajectory completion time (start to goal) between the virtual (with FABRIK) and real ABB robot arm (with TRAC-IK).

6. Disseminate the project results effectively.

1.3 THESIS OUTLINE

The remainder of this thesis is structured in the following manner:

Chapter Two reviews the literature surrounding AR-based Human Machine Interfaces (HMI), the telerobotic Human-Robot Interaction (HRI)framework and covers the required background research on robot motion and path planning techniques.

Chapter Three presents a description and methodology in building the AR prototype system. This includes details relating to the system requirements and a walk-through of the high-level designed software system architecture as well as the detailed descriptions of software tools involved in this project.

Chapter Four describes the steps taken required to set up and test the physical ABB robot arm during the deployment phase as well as the experiment design. The experimental results are summarized and discussed respectively.

Chapter Five concludes our present work and outlines possible future work. In addition to the research presented, a link to a public GitHub repository is provided, containing the source code of the developed software prototype and related collateral resources.

2 LITERATURE REVIEW

For a project of this scope, a review of the state-of-the-art research covering four distinct areas was required. This is summarised as per Table 2-1. The four domains included: Human Machine Interfaces (HMI) Input devices, HMI Output devices focusing on immersive technologies, Human-Robot Interaction (HRI) which considers existing remote interaction frameworks, and finally, the robot motion and path planning techniques in the context of HRI. In brief, Human-Robot Interaction (HRI) refers to any interaction and communication that involves both user and a machine, HMI device is used to facilitate such interaction. To get a deeper insight into both Human Machine Interfaces (HMI) and Human-Robot Interaction (HRI), the following sections will describe the terminology in more detail.

Table 2-1. Overview of paper's principal aspects

Human Machine Interfaces (HMI) Input Devices • Keyboard and Mouse Control • Hand-held/ Joystick control • Hand gesture-based Control ➢ Non-Image Based ➢ Image-Based	Human Machine Interfaces (HMI) Output Devices • Virtual Reality, Mixed and Augmented Reality • The HoloLens 2 • AR Registration Techniques ➢ Manual registration ➢ Tracker-based registration ➢ Computer Vision-Based registration

Human-Robot Interaction (HRI)
 	
Remote Interaction Framework
 	
• Existing telerobotic Application
• 3D Robotics Simulators in terms of Robot Operating System Integration (ROS)
  ➢ ARGoS
  ➢ Gazebo
  ➢ V-Rep
	Human-Robot Interaction (HRI)
	
	Robot Motion and Path Planning
	
	• Parametric Interpolation curves
	  ➢ Cubic Hermite Interpolation
	  ➢ Bezier Interpolation.
	  ➢ B-Spline Interpolation
	• The Inverse Kinematic Solvers
	  ➢ Heuristic method
	  ➢ Gradient-Based method

2.1 HUMAN MACHINE INTERFACES (HMI)

Traditionally, the term Human Machine Interface (HMI) is comprised of an Input or Output (I/O) device that connects a human or operator to a machine, computer or device. It is now commonly known as user interface (UI) where both human and the machine will act as information-processing devices [21]. The human-machine interface comprises both hardware and software that is capable of translating human input into a form of instructions that could be understood by a machine (i.e., binary) and present the results in a human-readable format. (i.e., text, graphics, tactile, audio, or video) [21] [22].

2.2 HMI- INPUT DEVICES

An input device is defined as a physical piece of hardware that connects to a machine or computer with the intention of providing user input [23]. The evolution of the HMI input devices will be discussed in the following sub-sections.

2.2.1 Keyboard and Mouse Control

Traditional robot or industrial machinery is usually controlled by a set of keystrokes on a control keyboard. The keyboard design was originally inherited from a traditional typewriter according to [24], while the mouse claims to be more natural and faster in 1965 than a keyboard input before the introduction of keyboard hotkeys [25].

2.2.2 Hand-held/ Joystick control

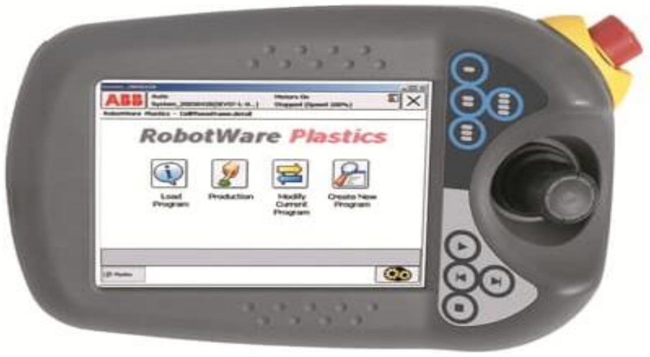


Figure 2-1. Conventional Human Machine Interface. The user can interact with the robotic arm through a traditional touchscreen ABB Flex Pendant.

A Joystick is a hand-held device that comprises a stick that is on a base and communicates its angle or direction to the machine it is manipulating. It was first created in 1972 by Ralph H. Bear for video game purposes. It has become very popular as a human-system interaction tool due to its reliability and robustness. For the operation of industrial robotics or machines, a high life cycle, strong, dust and waterproof joystick design are essential characteristics [26]. Unfortunately, low-level discrete actions such as a simple mouse click, straightforward keyboard input or joystick movements are not ideal for performing high-level or complex robotic tasks due to their physical limits (layout of keys and buttons). Modernized HMI allows operators to tap or touch the physical screen to control the robotic application as depicted in Fig. 2-1 besides buttons and switches. However, it is still considered as a hand-held device and there is still too much work involved and can be time-consuming, especially when programming the robot for complex robotic tasks.

2.2.3 Hand gesture-based Control

2.2.3.1 Non- Image-Based

The most common wearable human hand gestures on the market uses accelerometers and gyroscopes for motion detection. These sensors are considered non-image-based methods [27]. Micro-Electro-Mechanical-System Sensors (MEMS sensors) or Inertial Measurement Unit (IMU) are small, power-saving and weightless sensors consisting of Gyroscope, Accelerometers and Magneto sensors that bind together through sensor fusion technology to acquire human arm activity data. In recent times, we have started to see this technology applied in controlling a robotic arms context [28]. Commercial examples of such technology are the MYO Gesture Control Armband founded by a Canadian company called Thalmic Labs in 2014. The device detects gestures based on forearm muscle contractions using Electromyography (EMG) sensors together with 9-axes MEMS sensors for position, orientation and motion tracking. The challenges faced by MYO Armband are the restricted number of integrated categorisation actions and the random occurrence of misclassifications errors [29]. A corresponding technology called Finexus, a 3D input device that can track multiple fingertips in real-time using magnetic field sensing technique. It can robustly and accurately track a user's fingers motion using only 4 magnetometers by putting a one-dimensional electromagnet on the back of each fingernail. This system allows humans to perform effortless input tasks, for example, writing in the air or basic gesture-based gaming control. As an interaction technology for AR and VR enabled applications, the author believes that the Finexus system makes a better candidate due to its continuous, multipoint, real-time, precise, and occlusion-free finger tracking system [30].

2.2.3.2 Image-Based

Unlike non-Image-based sensors methods, image-based sensors require the use of the camera in order to recognise gestures. Depth sensing technology such as the Microsoft Kinect device was first introduced in [31]. It was capable of full-body tracking with high accuracy [25]. The Microsoft Kinect 2.0, introduced in 2013, was the enhanced version of Kinect 1.0 with a higher resolution or frame rate improved depth sensor, and the ability to track a considerable number of bodies and joints per body [27]. The Microsoft Kinect v1.0 had an Infra-Red (IR) laser projector to measure distance based on their distortion and the aid of a monochrome Complementary Metal-Oxide-Semiconductor (CMOS) sensor for reflected IR light detection, while the Kinect v2.0 used the time-of-flight (ToF) method to evaluate distance based on the speed of light [25] [31].

[32] describes depth sensor's accuracy as the differences between the ground truth distance and measured depth value while its precision depends on the frequency of consecutive depth calculation under fixed conditions. [31] summarised that Kinect v1 experiences a decrease in accuracy and precision in terms of depth detection when the distance between the user and Kinect v1 increases. Although the precision of the Kinect v2 decreases at different distances, its accuracy will remain the same. Nevertheless, Kinect v2 could experience flying pixels with less precision when the plane is not flat due to multipath interference. Overall, the Kinect v2 still providing better precision than Kinect v1 at the range of 1m but is highly dependent on energy intake as powerful LEDs are needed to broadcast the ToF signal across the scene [33].

The Leap Motion Controller (LMC) is considered one of the most advanced 3-Dimensional image-based sensors on the market. Unlike traditional image-based sensor which requires the generation of depth map to identify the matching points for geometry reconstruction and depth estimation, LMC applies advanced tracking algorithms to retrieve the raw sensor data without the need of generating depth map. It consists of two Couple-Charged Device (CCD) cameras and three Infrared LEDs [25]. It is primarily designed for hands and arms detection, as opposed to whole-body tracking applications. It has a maximum 150-degree horizontal Field of View (FoV) and 120 ° vertical FoV as shown in Table 2-2.

Table 2-2. Brief Technical Comparison of Depth Sensing Technologies

	Kinect V1	**Kinect V2**	**Leap Motion Controller (LMC)**	**Occipital Structure Core**	**Intel RealSense D435i**
Depth Camera Resolution	320×240 at 30 fps	512 X424 at 30 fps	640×240 at 60 fps	1280×960 at 54 fps 1280x 800 at 60fps	1280 x 720 up to 90fps
Depth Technology	Structured Light	ToF	Infrared with a mathematic algorithm	Structured Light with Global Shutter	Stereoscopic active with Global Shutter
Field of View (FoV)	57◦H, 43◦V	70◦H, 60◦V	150◦H,120◦V	160- D ultra-wide vision	87H x 58V x 90D
Suitability	Shorter Range, Lower accuracy	Longer Range, Higher accuracy	Shorter Range, Higher Accuracy	Shorter Range, Higher Resolution and Accuracy with Lower Latency	Longer Range, Lower resolution Higher accuracy at dynamic scene
Specified Measuring Distance	0.4 or 0.8 m–4 m	0.5–4.5 m	0.25 -60 cm	0.3 – 5m	0.16–10 m
Supported Libraries	The Kinect SDK V1.8	The Kinect SDK V2.0	Leap Motion SDK support Unity3D, Unreal Engine etc.	Structure SDK support Unity3D	Intel RealSense SDK 2.0 support Unity3D, Unreal Engine, OpenCV, ROS etc
IMU	Absence	Absence	Absence	Present	Present
Connectivity	USB 2.0 or 3.0	USB 3.0	USB 2.0 port	USB 2.0 or 3.0	USB-C 3.1 Gen 1

H—Horizontal, V – Vertical, D—Diagonal

In common with most image-based sensors, LMC suffers from occlusions. According to Chandel et al. [34], occlusion in stereo vision is defined as a problem that occurs when a part of an image or scene is visible on one image but goes out of sight on another image. This problem could limit the information within an image.

Although LMC consist of occlusion problems, it provides higher accuracy than both the Kinect version 1 and Kinect 2 [31]. In 2013, Weichert et al., [35] evaluated the

LMC accuracy in an attempt to increase its efficiency in the area of Human-Computer Interaction (HCI). An industrial robot (Kuka Robot KR 125/3) with a position error below 0.2mm was used to conduct the experiment. The experiment aimed to determine the accuracy and robustness of the Leap Motion controller in robot path drawing. According to the manufacturer, the theoretical accuracy of LMC in the fingertip position is approximately 0.01mm. However, the author reported accuracy of less than 2.5mm with an average of around 1.2mm and a standard deviation of 0.7mm per axis is possible when navigating.

The Occipital Structure Core [36] is an all-in-one wide vision depth camera sensor, with 160-degree Diagonal FoV as shown in Table 2-2. It uses structured light technology (IR laser projector to measure distance based on their distortion) similar to Kinect v1 and Orbbec Astra Embedded S. The main characteristic that differentiates Structure Core sensor from both Kinect and Orbbec is the presence of three cameras instead of two. The Structure Core sensor has a separate ultrawide monochrome camera with large FoV, and two stereo infrared cameras for depth-sensing). It also has an onboard IMU for 6 Degree of Freedom (DoF) inside-out simultaneous localisation mapping (SLAM) tracking, and an NU3000 processor that supports 3D image and Computer Vision processing with ultra-low-latency and jitter. According to [36], its depth precision is within the range of ± 0.29% (Plane-fit RMS at 1m.) However, there is a lack of research data related to the accuracy and precision of Occipital Structure Core [36]. While the Structure Core is a very new device, it has gained popularity in the last year after the release of an open-source AR headset- Project North Star [37].

A similar depth camera with a combined IMU sensor is the Intel RealSense D435i [38]. Unlike Structure Core Monochrome sensor, D435i has only 90-degree diagonal FoV, but it does have a quite advanced SDK, which is compatible with various open-source computer vision libraries. Both Structure Core and D435i support global shutter as shown in Table2-2. D435i is identical to the D435 in every aspect except the presence of an extra IMU sensor. This allows the robotic operating system to obtain precise positional data besides vision [39]. A company named Assemtica Robotics in India deploy a popular educational robot arm called Dobot together with the D435i depth functionality to localise and map the coordinates in the x, y & z-direction for robotic pick and place. This was accomplished with the use of python 3.6, OpenCV and custom Dobot APIs [40].

2.3 HMI- OUTPUT DEVICES

Devices that enable the user to visualise, hear, smell or touch everything that occurs in front of the user or the virtual environment are considered Human Machine Interfaces (HMI) output devices. In this section, a brief overview of the evolution of display and presentation devices, from standard computer monitor devices to current immersive head-mounted devices is presented [41]. Since the focus of this work is on Human-Robot Interaction with AR, the remaining extended reality (XR) technology review does not include fully immersive VR systems detail.

2.3.1 Virtual Reality, Mixed and Augmented Reality

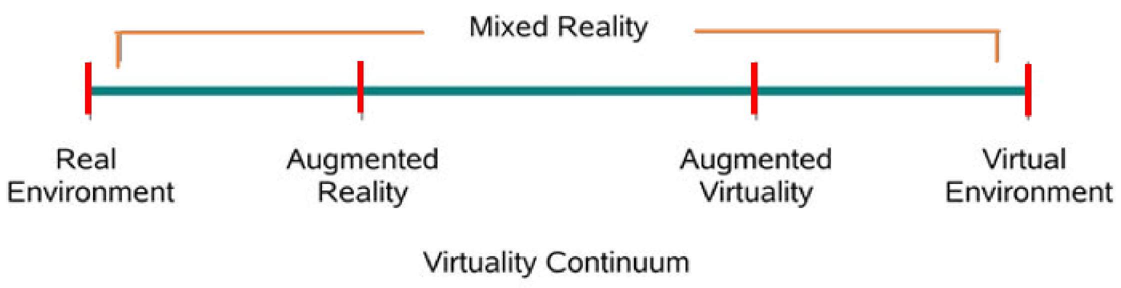


Figure 2-2. Reality-Virtuality Continuum defined by Paul Milgram and Fumio Kishino in 1994 adapter from [45]

In [41], Cipresso et al. stated that devices that enable the user to interact with the virtual environment are defined as the input device. Contrary to this, devices that enable the user to visualise, hear, smell or touch everything that occurs in front of the user, or the virtual environment is considered as output devices. In the following sections, an overview of the evolution from standard computer monitor output devices to current immersive head-mounted devices is presented. Since the focus of this work is on Human-Robot Interaction with AR or MR, the remaining XR technologies such as VR is not considered in detail.

However, to placed VR within context, the authors of [25][41] highlighted that the virtual reality (VR) concept is nothing new and it has existed since early 1992. The term “virtual reality” is described by [42] as “the use of the computer-generated 3D environment, that the user can navigate and interact with, resulting in real-time simulation of one or more of the user’s five senses.” The type of VR systems with different levels of immersion could indicate how closely the virtual environment (VE) matches the physical world. A range of popular and commercial VR headsets are currently available with specifications presented in Table 2-3. [43].

Table 2-3. Popular Commercial VR Headsets Comparison Chart

	Oculus Quest 2	HTC Vive Pro	Sony PlayStation VR	Valve Index VR Kit
Company	Facebook	HTC	Sony	Valve
Cost	€ 349	€679.00	€ 292	€ 1079
Platform	Oculus Home	SteamVR. VivePort	PlayStation 5, PlayStation VR	SteamVR

Mixed Reality (MR) or often referred to as hybrid reality according to [44], is identified as the blend of Augmented and Virtual Reality. Therefore, MR does not represent a segment along the reality-virtuality (RV) continuum family adapted from [45]. It characterises everything in between the real and virtual worlds. As a result, both AR and Augmented Virtuality (AV) are considered to be the subsets of MR according to the traditional RV continuum introduced by Milgram and Kishino [45]. The outdated framework appears to emphasise more on graphic display technologies than users' experience. The terms MR and AR are often used interchangeably by many, but some perceived MR as a stand-alone technology.

As mentioned by [46], AR has existed since 1968. In recent times, there has been a sudden surge in popularity with public AR frameworks such as Google ARCore and Apple ARkit (released in summer 2017 [47]) making it possible to develop AR applications for smartphone technologies. One such is that of Pokémon Go, which as claimed in [48] has helped bring AR technology to the masses. AR technology has the ability to superimpose digital information on top of real-world physical objects. The European Union has identified AR as part of the primary technologies that will boost the development of smart FoF [3][49]. A range of high-performing AR head-mounted displays on the market includes the latest Microsoft’s HoloLens 2, Magic Leap One and Nreal are actively used in the research and business industry.

A number of research works have critiqued the use of augmented reality-based trajectories on industrial robots [50]–[52]. Chacko et al. [50] presented an approach using a mobile smartphone with Google’s ARCore framework used to localise a 4 DOF manipulator robot. It allowed a user to successfully perform a simple pick-and-place task within the robot’s workspace with the aid of a 2D marker. However, the benefit of this system is constrained and tedious considering that the user needs to perform the gestures with one hand while holding the mobile phone in the other hand. This work also highlighted challenges in identifying complex shapes or picked object orientation due to the limitation of the mobile phone camera. In addition, the 3D object recognition feature was also not supported on the ARCore framework.

Similar to our approach, in [51] a user could execute a robot trajectory through free space or in contact with a surface using a Microsoft HoloLens. The user is able to move the 7DOF robot's end-effector along the path with an MYO [29] armband and speech recognition. Multiple waypoints can be set by the user by altering the orientation of their head in combination with speech commands. A user study highlighted that HoloLens required less physical effort to train (in comparison with traditional hand-held teaching pendant), the AR robotic interface required a higher mental effort. This was demonstrated when users require to remember several specific voice commands which must be performed in order. The authors also reported that the poor 3D environmental reconstruction from the HoloLens resulted in difficulty for accurate depth estimation, and higher registration error after constant user position displacements.

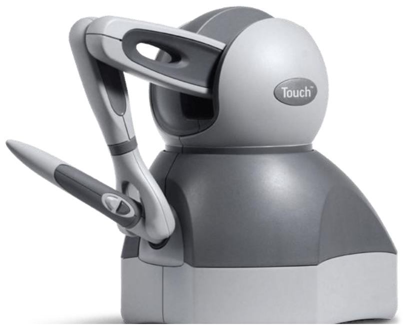


Figure 2-3. SensAble PHANToM OMNI haptic device adapted from [53]

Another augmented reality-based trajectory method is presented in [52] where a PC camera was used to compose the AR scene of the robot workspace, and a Kinect sensor was used to capture the 3D point cloud data of the workpiece. The PHANToM device [53] as shown in Fig. 2-3 which acts as a haptic interface is used to define the welding points and relocate the virtual robot along a surface of a workpiece. In order to overlay the virtual robot with respect to the coordinate system of the real robot, a fiducial marker is used to obtain the coordinate registration between these devices. The study concluded that the haptic and visual interface greatly improved the welding path accuracy in comparison with a visual-only interface. This approach, however, is not a flexible wearable type of AR interface but a PC AR display in a fixed remote position.

2.3.2 The HoloLens 2

The HoloLens 2 as shown in Fig 2-4. The main materials of the HoloLens 2 are carbon fibre (headband), memory foam (forehead), hard plastic, ANSI rated glass and some hardware components [54]. A USB-C charger, wall adapter, and microfiber cloth. The headband can be extended in length by tightening the adjustment wheel for better head alignment. It is much lighter than its predecessor (HoloLens 1) in terms of weight thanks to the carbon fibre build. It also has a wider diagonal field of view (FoV) of 52 degrees. It includes a new edition of 64-GB UFS 2.1 storage space, 4GB LPDDR4x system DRAM, the second generation of Holographic Processing Unit (HPU2.0) plus a more up-to-date Bluetooth connection (5.0)[55]. The overall hardware components of HoloLens 2 are shown in Fig2-5. The HPU 2.0 which is located on the front manages all the real-time computer vision algorithms on the device (spatial mapping as well as head, hand and eye-tracking etc.).

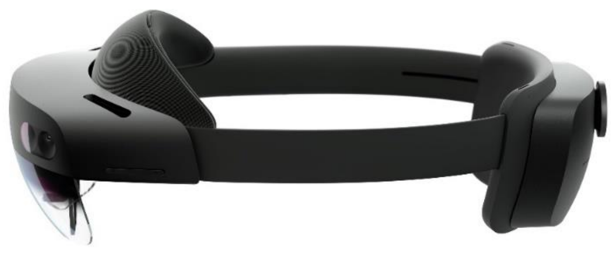


Figure 2-4. Physical appearance of HoloLens 2.

The HoloLens 2 also hosts the Deep Neural Network (DNN) learning core. In addition, it features a Qualcomm Snapdragon 850 with an 8-core 2.6 GHz CPU which is located on the back. The device is equipped with a Time-of-Flight (ToF) RGB depth camera similar to the depth camera of the Azure Kinect. Moreover, it consists of four grayscale visible light cameras, and an Inertial Measurement Unit (IMU) plus a built-in spatial sound speaker and 5-channel mic array [56].

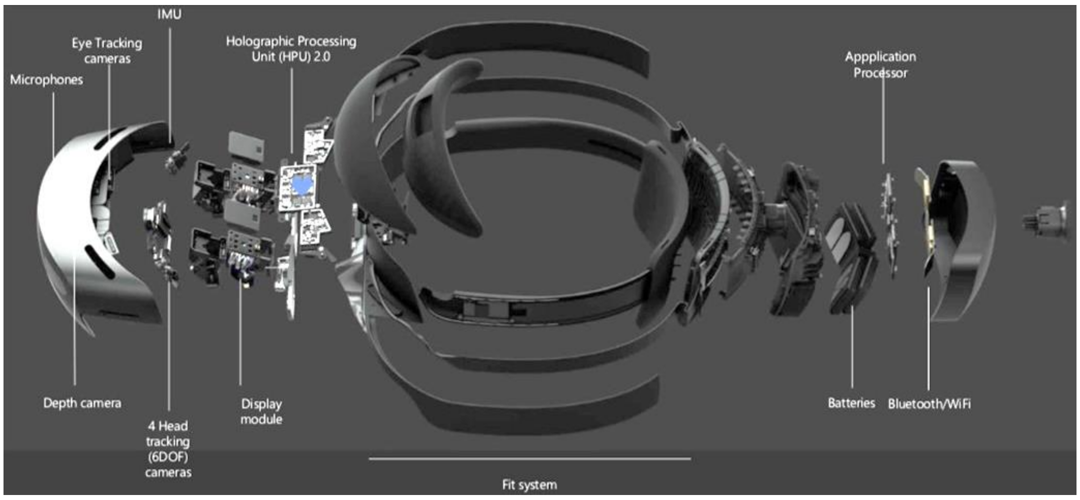


Figure 2-5. Hardware components of Hololens2, adapted from SphereGen. 2021 [54]

Finally, the HoloLens 2 provides research mode functionality and is specifically designed for academic and industrial researchers. It enables researchers to access all HoloLens 2 sensor streams and as such, collect useful data and discover new ideas in the areas of computer vision and robotics.

2.3.3 AR Registration Techniques

Yi-bo [57] defined AR registration as a process that combines computer-generated virtual objects with physical world images captured via a camera. Once the position between the end-user viewpoint and the virtual object is identified, the virtual objects are then displayed into the field of view of the user via projection transformation. The approaches for registering the holograms can be classified into three categories: manual registration, registration integrate external tracking sensors/ tracker-based registration, and computer vision-based registration [57] [58].

2.3.3.1 Manual Registration

Manual registration is considered the simplest AR registration method among the three. Here, the user will manually adjust the position of the hologram until the hologram is spatially aligned with the real-world object through basic hand gesture recognition software, for example, the Mixed Reality Toolkit library (MRTK) which supports HoloLens 2's articulated hand tracking. Although manual registration has proven effective, the registration process is often time-consuming and there is a high probability of chance for human error which will indirectly impact the registration accuracy. This problem can be seen in [51], where Camilo tried to manually register the hologram to the seven degrees-of-freedom (DoF) manipulator. The proposed framework can perform trajectory specification, robot motion preview, parameter visualisation, and robot task execution. This system, however, does not appear to be promising as when the hologram had an offset of about 3-5cm when the user views the robot from a different angle (e.g., by standing behind the robot).

2.3.3.2 Tracker-Based Registration

Tracker-based registration is an approach that requires not only a 3D-AR device but external tracking sensors. It can potentially improve the accuracy of spatial tracking and localization and is beneficial for clinical applications that involve multiple or moving object tracking. Rafael [59] presented an assembly training AR system that incorporates multiple Kinect cameras and a Microsoft HoloLens to track multiple parts of a piston motor in a room and render the virtual 3D part such as screw at the correct location for assembly guidance. The transformation from the HoloLens to the real asset of interest is required to augment a virtual model from the right angle. The point cloud obtained from each Kinect camera needs to be transformed into one global coordinate reference frame and fused into one point cloud data structure in order to detect every object of a specific size in a room-sized space. The author argued that the proposed systems yield a sufficient registration accuracy but the accuracy is unclear because there is no data available to support the claim. However, the registration errors and noticeable network latency due to the Wi-Fi connection and the Kalman filter are still present. In Christopher [58] analysis, the tracker-based approach consists of two limitations. The first will be the errors and drift in the spatial mapping system of the headset and the second is that this method requires larger tracking volumes.

2.3.3.3 Computer-Vision Based Registration

The front-facing camera on the HoloLens can also be utilized by computer-vision algorithms to register a virtual object at known locations by placing a 2D /3D computer vision target without the need for external sensors [58]. The HoloLens front-facing camera is at a predetermined position relative to its display results in its coordinates directly linked to the display coordinates. Developers or researchers could access the camera's spatial position through the HoloLens software development kit (SDK). The coordinates of 2D images or 3D Objects relative to the headset can be determined by utilizing the camera hardware and software tools in conjunction with computer vision algorithms. Popular open-source computer vision SDKs such as Vuforia Engine [60], ARCore [61] and OpenCV [62] are all capable of identifying and capturing planar images or simple 3D objects effortlessly through computer vision technology. By incorporating the ARCore library, a mobile pick-and-place robot arm AR interface application utilized a 2D fiducial marker in combination with markerless AR technology is used to localize the pose of the virtual robot arm while detecting a planar surface that contains the robot workspace [50]. This method is applicable for manipulating unidentified and arbitrarily positioned objects in the robot workstation. Another AR robot application in assembly tasks utilized two Vuforia Image Target markers to manipulate the real Kuka KR6 R900 robot arm using a HoloLens Head Mounted Device (HMD). A reference marker is needed to reference the robot's coordinate system concerning the HoloLens' coordinate system and must always be fixed within the workspace while the mounting aid marker can be positioned randomly. By linking the virtual and real Kuka robots, the real robot can perform predefined pick-and-place tasks based on start and end coordinates. The accuracy and precision of the marker are 1 mm-2 mm and 3mm-5mm respectively [63]. [58] conducted a literature review to compare the accuracy across these three registration methods and concluded that the combination of image targets and computer vision algorithms is considered the most accurate hologram registration technique in determining the position of the target relative to the headset. The author believes that this approach can resolve the headset's spatial mapping drift apart from performing precise registrations with external tracking sensors. The author evaluated that the required user accuracy when performing clinically relevant tasks of around 2 mm is sufficient.

2.4 HUMAN-ROBOT INTERACTION (HRI)

David et al., [4] defined Human-Robot Interaction (HRI) as a multidisciplinary field with contributions from Human-Computer interaction (HCI), natural language understanding, Robotics, Artificial Intelligence (AI), design, and psychology. HRI is the study of how human communicates with the robot, and how best to understand, design, evaluate and implement robot systems that are competent enough for carrying out collaborative chores in the human environment [5] [64]. As Michael and Alan mentioned in [5], Human-Robot Interaction can be categorised as remote interaction when both humans and robots are distant, isolated spatially or even for the time being while proximate interaction refers to humans and robots are nearly within reach of each other.

2.5 HRI - REMOTE INTERACTION

Teleoperation is part of remote interaction where a robot or machine is controlled by an operator in a separate environment. Remote interaction involves a physical manipulator (the master or local) with a human operator on one side and a different physical device (the slave) located at a distance on the other side is often considered as telemanipulation [64].

2.5.1 Existing telerobotic Application Framework

There are many diverse forms of remote Human-robot interaction (HRI); the most common and natural way of which is the use of hand gesture recognition. Spranger et al. proposed a free-hand gesture user interface that is capable of performing remote robot programming/ training without causing delays and inefficiency. The author utilises a Unity-based hand tracking software-Mano Motion SDK that builds on a Samsung Galaxy S7 mobile phone application to teleoperate a ROS enabled UR10 robotic arm. The Mano Motion SDK uses the phone's mono-camera to control the translation and orientation of the robot end-effector by capturing the cartesian position of the user's hand. The hand coordinates from the mobile device are sent out to the robotic arm through a Virtual Private Network (VPN) and communicate using TCP/IP [65].

A remote-control robot has been associated with ROS by Peppoloni et al. [16] for controlling and manipulating a KUKA Youbot arm with a Leap Motion controller and a Virtual Reality (VR) headset. The communication between the Leap Motion and ROS enabled robot arm at the remote site is realise via the ROS Leap interface. An external Kinect sensor is used to capture the hand position data from leap motion and saved waypoints from ROS at a remote site. The augmented 3D feedback is then streamed back to the user via an Oculus VR headset. Similar research also used the Leap Motion controller and ROS framework to teleoperate a mobile Turtlebot 2 with an HTC Vive VR headset. The author used a slightly different approach if compare with the Peppoloni approach by mounting the Unity enabled Leap Motion controller on the VR headset instead of connecting the Leap Motion controller directly to a ROS powered computer via a USB. The interaction between the Leap Motion and ROS enabled mobile robot is accomplish through the Rosbridge package that provides a WebSocket-based communication over the network [66].

The rosbridge has also been used by Krupke et al. [15] to connect a Microsoft HoloLens and ROS enabled Universal Robot UR5 industrial manipulator for performing remote pick-and-place tasks. The user interface on the HoloLens is built on a popular Unity game engine. QR code is used as part of the registration method to establish a common coordinate system between the HoloLens headset and the robotic manipulator. A comparison between an AR gaze-based interface and a gesture-based interface is conducted. The author claimed that AR gaze-based outperformed the hand gesture-based interaction. However, the usability test is done without controlling an actual robot. The trajectory accuracy between the real and virtual robots is unknown. In brief, the open-source Robot Operating System (ROS) has gained the favour of the robotic research community as their framework of choice when designing robot teleoperation systems.

2.5.2 3D Robotics Simulators

A suitable 3D robotic simulator will be chosen for the project during the initial code verification phase with the aim to reduce the cost of developing and performing real robot experiments. As a result, 3 popular simulators on the market namely, V-REP, GAZEBO and ARGoS were compared and reviewed accordingly with regards to their feature and performance.

Table 2-4. Feature and performance comparison of 3D robotics simulator

	V-REP	**GAZEBO**	**ARGoS**
ROS Compatible	YES	YES	YES
Ease of Use	Complex	Easy to integrate	Moderate
Licence	Free for Education only	Completely Open Source	Completely Open Source
Supported Languages	A scene is saved in a special V-REP format. Support various programming options.	A scene is saved as an XML file. Functionality can be programmed either as compiled C++ plug-ins or via ROS programs.	A scene is saved as an XML file Robots can be programmed either through Lua scripts or in C++.
Performance	POOR	EXCELLENT	MODERATE
Suitability	Best choice for high-precision modelling of robotic applications	Best choice for large swarm robotics experiments	Best choice for simulations of swarm robotics tasks

V-REP is known for its abundance of functionality and resources. From the information provided by [67], it is able to perform high precision simulation due to the vast number of extremely detailed robot models it provides. According to [68], V-REP support numerous manipulation such as dragging and dropping, visual editing and simplify meshes were made available to sub-component [67]. In addition, it has extensive documentation, tutorial, code examples and a huge community. The user interface is very accessible, a well-organized model library that is always available even without internet connectivity. The most notable advantage of using V-REP will be the zero-freezing issue. However, the disadvantage of choosing V-REP will be its flexibility, it only allows the user to save the project scene in V-REP format which prevents the user to open the saved scene in a different format. Moreover, it is complicated to use and is only free for educational purposes as mentioned in Table 2-4.

GAZEBO has the advantage of being the most flexible and open-source robotic simulator among the three as per Table 2-4. [68] notes that GAZEBO has become a standard simulator in the ROS framework as there is a specific package built available to support this operation from the official ROS repository. [68] state that the user interface provided by GAZEBO is less complex and it could handle large simulation scenes with excellent performance. It has a customizable user interface that was constructed by QT API which allows the user to add extra functionality to the interface and the project scene is saved as an XML file. This facilitates a simple upgrade tool to migrate the old robot model description versions to a new version. Besides that, it has detailed documentation, tutorials, a large user community and a development road map shown on its official website that indicates GAZEBO is actively under development, constantly adding new features and fixing bugs along the way. It has an internal code and scene editor for the user to manipulate any models that were inserted into the scene and interaction could also be done during simulation. Due to the fact that GAZEBO is an open-source project that was developed by a team of talented individuals, most of the functionality was suggested and built upon by the community. Unfortunately, it still has several problems due to its characteristic, constant freezing of the user interface occurs when users try to edit robot models and while starting or stopping the simulation. Apart from that, an internet connection was essential to acquire the model library from its server as it was not ready locally and meshes that were imported to GAZEBO must be optimized with external software.

In [67] study, ARGoS has a lightweight infrastructure as it has minimal yet essential functionality of a robotic simulator. It has a simple default model which tolerates the handling of complex simulation. If compare to Gazebo, its responsive user interface does not have freezing issues. Individual robots or project scenes could add custom interfaces that were scripted with C++ and internet connection was not needed in order to retrieve robot models. Unfortunately, due to the minimal functionality it provides, a lot of capabilities was not included. It does not support importing external meshes that were not in its library, the internal model library has very limited resources. Moreover, physics engines were custom-built with minimal capabilities and it consists of a small community with no regular development updates.

2.6 HRI- Robot Motion and Path Planning

Motion or path planning plays a vital role in the field of robotics application, ranging from the industrial robot with a stationary base to mobile robots that is capable of moving around freely in a static or dynamic environment [65]. According to Mihai [69], robots are the machine that is accompanied by motion or manipulation capabilities. Motion planning can be counted as a problem of finding a collision-free path from a start location to a target location by considering the surrounding obstacles, and the robot, as an entity with complex and varying shapes [69][70]. According to Hilario et al. [71], motion planning techniques are generally separated into four major groups: sampling, graph search, numerical optimization and interpolation. Among various path planning techniques, this paper only considers the interpolation-based method because it is extensively used in engineering and robotics applications [71].

2.6.1 The Parametric Interpolation curves

The term ‘interpolation’ in path planning is defined as a mathematical method used to identify the values at any position between the data points [72]. An interpolating curve is a curve that travels across each control point while the control points are a set of points used to regulate the shape of a curve. Points that are naturally joined by straight line segments and every segment (attached with two data points) can be interpolated separately is known as linear interpolation. Although linear interpolation is easy to compute, it tends to be discontinued at the edge of each point and is not suitable to be used in robot trajectory. The discontinuity of linear interpolation may result in jerky trajectories as the robot have to stop and then resume a new linear interpolation between the next segment. To construct a smooth, continuous and easy to manipulate path, a parametric interpolation curve is proposed. A parametric interpolation curve is described as a distinct set of points, often known as “control points”, alongside a collection of simple functions that is capable of improving the path planning algorithms of robot trajectories [71]. According to Hilario et al. [71], a parametric expression of a curve could be defined as:

$$\alpha : [a,b] \rightarrow \Re^n \,/\, \alpha(u) = (\alpha_1(u), \ldots, \alpha_n(u)); u \in [a,b] \qquad (1)$$

The coordinates of a parametric curve are expressed as u, while a curve in Rn is expressed in (4) and the coordinate functions are labelled as αi. The expression α(u) is the parametrization of α. The three common types of parametric curves in computer graphics are Cubic Hermite, Bezier and B-spline. The following section provides a mathematical definition of parametric interpolation curves together with a table description of their properties.

2.6.1.1 Cubic Hermite Interpolation Curve

The Cubic Hermite interpolation curve developed by Charles Hermite is defined through a set $\{P_0, P'_0, P_1, P'_1..., P_n, P'_n\}$ of interpolation points P_i and their tangent vectors P'_I using the following formula:

$$P(u) = \sum_{i=0}^{3} C_i u_i , 0 \leq u \leq 1 \tag{2}$$

Every interpolation segment has 4 degrees of freedom (DoF) in 2D space and each segment is a 3rd-degree polynomial [73][74]. The Hermite curve passes only through the middle two control points, and the endpoints are used to determine the tangent of the middle control points [73]. Cubic Hermite is considered the easiest in terms of computation and is particularly employed in computer geometric modelling to obtain curves that pass via defined points of the plane in 3D space. The mathematical background behind Hermite curves is considered fundamental to understanding the whole family of curves or splines. According to Table 2-5, the main drawback of such a curve is that it cannot control more than 4 data points and is considered less smooth if compare to Bezier and B Spline.

2.6.1.2 Bezier Interpolation Curve

The Bezier Interpolation Curve is a parametric curve consisting of a minimum of three points which is the origin, endpoint and at least one control point to compute a smooth continuous line. The Bezier Curve was developed in the late 1970s by Pierre Bezier interpolate the first and last control points [75]. The parametric curve p (u) depicted with Bezier's formulation can be deduced by studying a set of $m - n + 1$ control points, $\{P_0, P_1..., P_{m-n+1}\}$ where P_0 is the origin and P_{m-n+1} is the endpoint using the following formula [73]:

$$p\,(u) = \sum_{j=0}^{m-n} b_{j,n}\,(u)\,P_j \qquad t_n \le u \le t_{m-n+1} \tag{3}$$

Where u is the positional parameter, m represents the number of knots, n is the polynomial degree of the blending functions while bj,n(u) are the blending functions. Recently, the Bezier interpolation curves have been applied in various smooth robot path planning applications (see [70] [71] [76] [77]) because they are easy to compute, easy to use in higher dimensions (3D and up) and very stable. Moreover, they are extensively used in 3D game animation to model curved paths such as racing lines, flight paths or robot trajectories [78]. Bezier interpolation curves present certain advantages and disadvantages, apart from being easy to use and possess a higher level of continuity and stability at a lower degree of control points [75], it often requires more computation time if compare to Cubic Hermite and have greater global control over the shape of the curve. Luckily, its disadvantages can be overcome easily by B-splines.

2.6.1.3 B-Spline Interpolation Curve

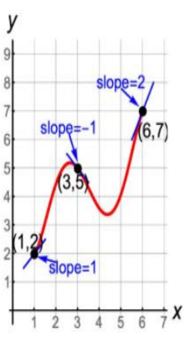
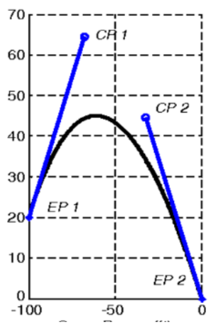
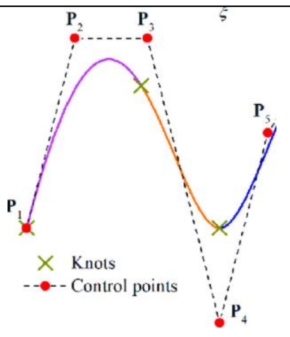
The term 'spline' is described as a mathematical representation that allows a user to design and control the shape of complex curves and surfaces. A B-spline with P_i control points and n denotes the number of control points can be expressed as:

$$P(u) = \sum_{i=0}^{n} P_i N_i, k(u), 0 \le u \le u_{\max} \tag{4}$$

Where N_i,$K_{(u)}$ represents the p^th degree B-spline basis function. A B-spline curve can be called a Bezier curve and satisfies all the significant characteristics of the Bezier curves because the curve is an extended form of the Bezier curve that comprises Bezier curves as segments. Open knots are employed to guarantee the interpolation of the first and last points [79]. Unlike the Bezier curve as illustrated in Table 2-5, the B-Spline curve may or may not interpolate any of its control points, involving the first and last depending on the number of knots [80]. The number of knots that are required at the endpoint is dependent on the order of the curve. They are more flexible than Bezier curves since the degree of a B-spline curve is split from the number of control points. More precisely, we can alter the position of a control point without altering the shape of the whole curve (greater local control). Given that B-spline curves fulfil the strong convex hull property, they have better control in defining a complex shape or curve. Furthermore, there are other alternatives in manipulating and modifying the shape of a curve based on knot insertion [80]. The disadvantage of the B-spline will the computation time increases with the curve complexity.

Overall, parametric interpolation curves have characteristics of cheap computational cost, natural softness, high manipulability through control points, and universal approximation. On behalf of these reasons, parametric curves are not just applicable as interpolators but are lately being used together with many other algorithms for example in Ma et al. [70] research, genetic operator and Bezier curve are combined to discover a smooth and safe path for the mobile robot application.

Table 2-5. Cubic Hermite, Bezier and B-spline Comparison Table adapted from [81]–[83].

Parameter	Cubic Hermite	Bezier	B-Spline
Definition	3rd degree polynomial with 4 data points and 4 coefficients	The curve of nth degree polynomial with n+1 = number of data points	Bezier curve with varying degree
Characteristics	Interpolate all control points	Interpolate first and last control points	Not guaranteed to interpolate control points
Formula	$P(u)=\sum_{i=0}^{3} C_i u_i, 0 \le u \le 1$	$p(u)=\sum_{j=0}^{m-n} b_{j,n}(u)\, P_j \quad t_n \le u \le t_{m-n+1}$	$P(u)=\sum_{i=0}^{n} P_i N_i, k(u), 0 \le u \le u_{max}$
Advantages	Easy to compute	The curve has a higher degree of continuity	Greater local control. The degree of the curve is independent of data points
Disadvantages	Cannot control more than 4 data points. Less smooth than Bezier and B-Spline	Require more computation time. Greater global control over the shape of the curve.	Computation time increases with curve complexity
Figures			

2.6.2 The Inverse Kinematic Solvers

Inverse kinematic equations are essentially needed in different fields, including, but not limited to, robotics, engineering and computer graphics[84]. They calculate the joint's position for the end-effector to be at a specific point in space. Inverse Kinematics is a method to calculate the collision-free robot arm motion [85]. According to the compact equation (5) where the inverse function, f^{-1} of Cartesian end effector coordinate transformations represented as $X_{1,\ldots,k,}$, is used to calculate the joint angles θ, needed to reposition the end-effector to the target location [84]. The position of a robot arm with reference to its joint angles in 3D space is defined as "Joint Space" while "Cartesian Space" stands for the robot pose with regard to its end effector [86]. The solution of inverse kinematics relates to the mapping of the desired joint coordinate in Cartesian space back to its corresponding joint space [84].

$$\theta = f^{-1}(X_{1,\ldots,k}) \tag{5}$$

2.6.2.1 Heuristic method

The term 'Heuristic' in computer science perspective can be considered as a shortcut approach for finding an estimated solution to a problem when traditional methods fail to discover any accurate solution. It may not be the optimal or perfect solution but is sufficient to solve a problem. It could also be used as a method to solve a problem more rapidly when traditional methods are too slow [87]. The objective of a heuristic method for inverse kinematics, in this case, is to repeatedly approximate the desirable joint updates with geometric calculations [84]. The Forward and Backward Reaching Inverse Kinematics (FABRIK) algorithm is one of the heuristic approaches to solve the Inverse Kinematics problem for articulated robots. FABRIK is a method reported in [88] to solve the problem of inverse kinematics. The FABRIK consists of two successive phases in a repetitive forward and backwards reaching fashion rather than a single direction along the kinematic chain. The combination of forward (6) and backward (7) computations provided by [84] continually adjusts each joint along the way starting from the end effector to reach the target position. Instead of computing the rotational joint updates, the interconnected positions are solved by discovering points on lines. The n joint positions of the kinematic chain can be described as $J_i = (x_i, y_i, z_i)$ where i = 1, 2, ..., n. The root joint, in this case, will be J_1 at coordinates (0, 0, 0) and J_n will be the end effector. According to [88], the algorithm converges in fewer iterations, lower computational cost and generates more realistic and smoother poses. Constraints can also be easily integrated within FABRIK.

$$J_i = (1 - \lambda_i)J_{i+1} + \lambda_i J_i \tag{6}$$

$$J_{i+1} = (1 - \lambda_i)J_i + \lambda_i J_{i+1} \tag{7}$$

2.6.2.2 Gradient-Based method

Gradient-based methods are among the most broadly applied methods for solving inverse kinematics especially in the field of robotics according to Starke et al. [84]. They require estimating the first or second-order derivatives and are more computationally expensive if compare to the previously mentioned heuristic algorithm. Nevertheless, they are more flexible in terms of constraints insertion and can directly operate in joint space as well as calculating the full-pose goals. The gradient-based methods are normally based on solving the Jacobian matrix [89]. The Jacobian matrix estimates the first-order partial derivatives of the individual flexible joint to obtain a linear estimation of the following end-effector velocities in Cartesian space [84]. The Jacobian matrix can be written as follow:

$$J(\theta)_{ij} = \left(\frac{\delta \mathcal{X}_i}{\delta \theta_j}\right) \tag{8}$$

The partial derivatives in joint space concerning Cartesian space are defined by inverting the Jacobian, J^{-1}. The inverse kinematics function is then written in the form of:

$$q_{next} = q_{prev} + J^{-1} p_{err}, \tag{9}$$

where q_{next} is the Forward Kinematics used to compute the new value of p_{err}. When all elements of p_{err} drop below a stopping standard, the present joint vector q is the returned IK solution [90]. The Orocos KDL framework is an extended version of the Jacobian matrix and is frequently being used together with the Robot Operating System (ROS). The commonly-used Orocos' KDL implementation of pseudoinverse Jacobian IK for robots with joint limits has several problems when applied to real-world robotic configurations. These particular problems can luckily be overcome by implementing the improved TRAC-IK algorithm. TRAC-IK is an alternative Inverse Kinematics solver to the standard inverse Jacobian methods in Orocos' KDL (Kinematics and Dynamics Library).

TRAC-IK simultaneously runs two IK implementations. The first one is a simple extension to KDL's Newton-based merging algorithm that identifies and alleviates local minima because of joint limits through arbitrary jumps. The second is a Sequential Quadratic Programming (SQP) nonlinear optimization approach which utilizes quasi-Newton methods to manage the joint limits better [91]. In Beeson et al. [90] analysis, TRAC-IK outperforms the traditional KDL IK algorithms tested on the two long manipulation chains (14 & 15 DoF chains) in terms of solve rate. TRAC-IK improve the overall runtime that took only 0.63 ms per solve with a 99.45% solve rate. While the traditional KDL took an average of 0.65 ms per solve with a 97.1% solve rate. The author concluded, the higher the success rate of KDL, the longer it took to solve the manipulation chains. The author obtains a further 99.59% IK success rate from TRAC-IK when running over 180,000 random samples while improving on the average time needed for a single IK solution to converge but obtain only an 85.86% success rate for the conventional KDL. In brief, TRAC-IK is said to be the best overall algorithm if compare to other IK methods.

3 SYSTEM ARCHITECTURE AND TECHNOLOGY STACK

The proposed HRI system architecture is presented in Fig.3-1. On the left is a Microsoft HoloLens 2 user interface with Unity 2019.4.21f1 (64-bit) game engine. The Microsoft HoloLens 2 AR HMD is employed because of its built-in localization and mapping (SLAM) abilities in addition to its self-supporting nature. It also currently represents state of the art in AR HMD technology. In the Centre, is a ROS (for more information on the topic, see Chapter 3.2) system running on a virtual Ubuntu16.04 LTS ROS Kinetic virtual machine.

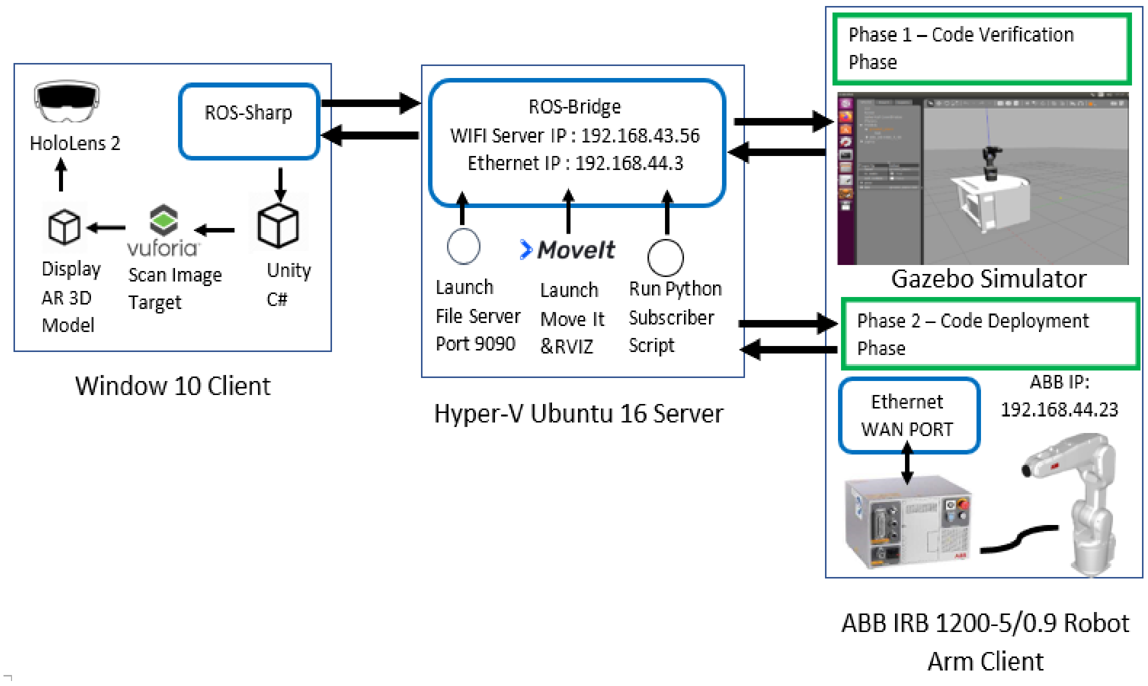
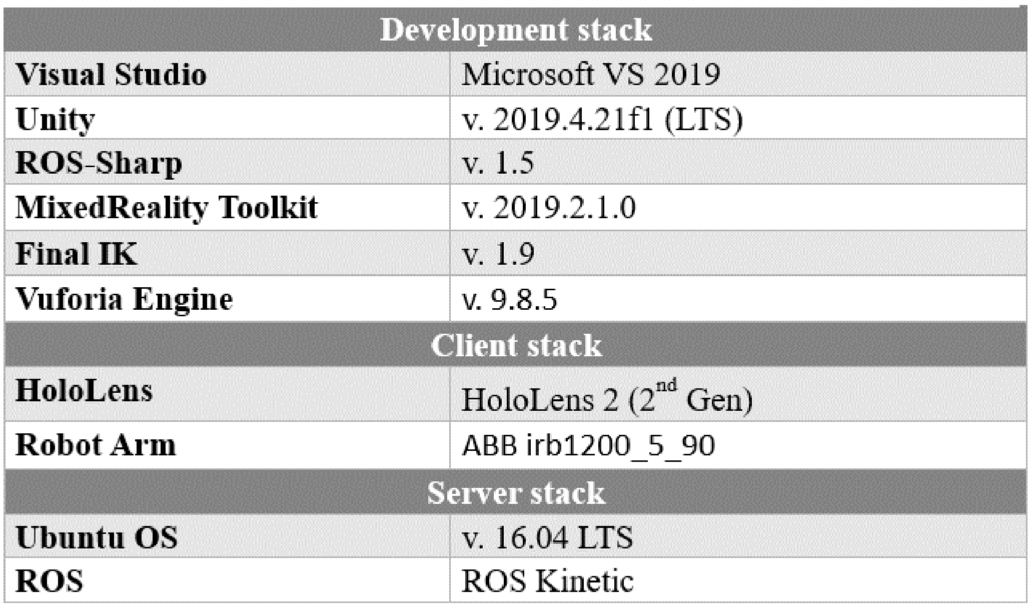

HoloLens 2
ROS-Sharp
Display AR 3D Model
vuforia
Scan Image Target
Unity C#
Window 10 Client
ROS-Bridge
WIFI Server IP : 192.168.43.56
Ethernet IP : 192.168.44.3
MoveIt
Launch File Server Port 9090
Launch Move It &RVIZ
Run Python Subscriber Script
Hyper-V Ubuntu 16 Server
Phase 1 – Code Verification Phase
Gazebo Simulator
Phase 2 – Code Deployment Phase
Ethernet WAN PORT
ABB IP: 192.168.44.23
ABB IRB 1200-5/0.9 Robot Arm Client

Figure 3-1. End-to-end RoSTAR System Architecture

On the right is our physical ABB robot arm connected to the ROS server over an ethernet cable. The integration of ROS and Unity 3D is enabled via ROS-Sharp, through a WebSocket connection on port 9090. All development tools used in this project are summarized in Table 3-1. On the HoloLens 2 client-side, the Mixed Reality Toolkit (MRTK) library [92] was imported to control the user hand gestures on a HoloLens 2. The Final IK asset attached to the holographic robot arm consisted of a powerful Inverse Kinematics solution (FABRIK algorithm is chosen) for advanced robot animation. Vuforia Image Target is used to correctly overlay the holographic robot arm on top of the real robot arm. On the server-side, two virtual switches were created and connected to the

Hyper V machine at the same time. A file server will be the first to launch opening port 9090, waiting for the HoloLens Client to connect over the mobile Wi-Fi hotspot. Secondly, both MoveIT and Rviz are launch together. MoveIT is mainly for Motion Planning. Basically, it creates a sequence of movements that all joints have to perform in order for the end effector to reach the desired position. While Rviz is for visualization purposes. On the ABB client-side, a WAN port is chosen to connect the ABB controller over the ROS server.

Table 3-1. Proposed System Development Tools



Development stack	
Visual Studio	Microsoft VS 2019
Unity	v. 2019.4.21f1 (LTS)
ROS-Sharp	v. 1.5
MixedReality Toolkit	v. 2019.2.1.0
Final IK	v. 1.9
Vuforia Engine	v. 9.8.5
Client stack	
HoloLens	HoloLens 2 (2nd Gen)
Robot Arm	ABB irb1200_5_90
Server stack	
Ubuntu OS	v. 16.04 LTS
ROS	ROS Kinetic

The integration of ROS and Unity 3D is enabled by implementing the open-source ROS-Sharp library[93]. A WebSocket connection is established on port 9090 between Unity and the ROS arm simulator. It allows ROS-Sharp to serialize a topic (for more information on the topic, see Chapter 3.2). A pose stamped publisher script in Unity will create a topic. The topic is activated each time when Unity tries to send end-effector pose information as JSON [94] to ROS. Within the script, the Unity end effector Euler rotation x, y and z will automatically be converted into quaternion x, y, z and w before sending.

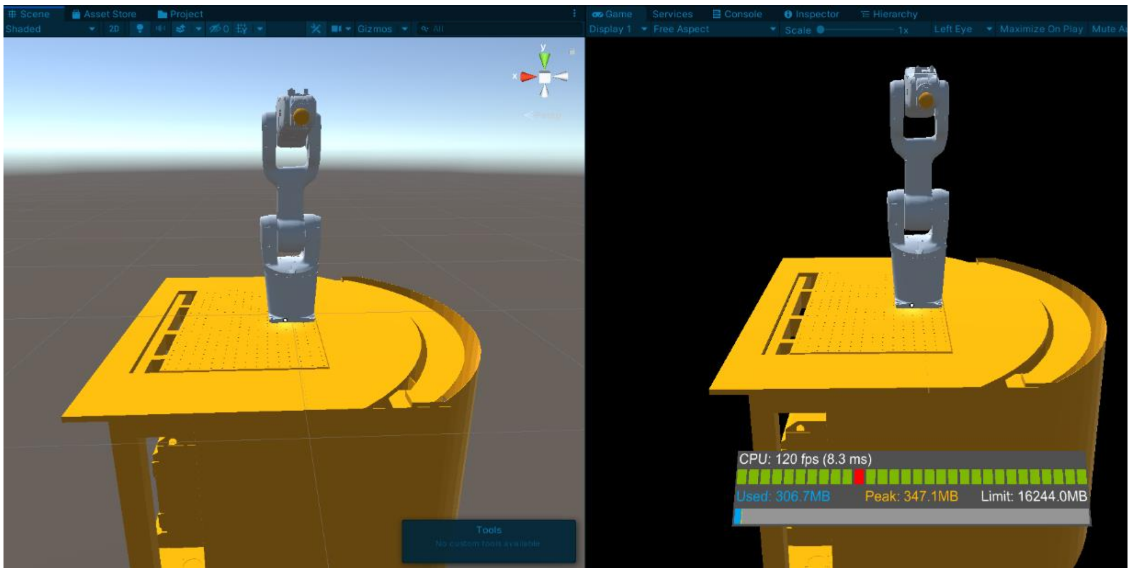
In this case, Unity will act as a publisher, while the ROS simulator is the subscriber. Unity converts C# data structures into JSON messages, and the pose data is passed across the bridge. The rosbridge library which contains the core rosbridge package then retrieves the pose data through the same topic by converting the JSON message back into geometry messages.

3.1 CLIENT-SIDE STACK

The Client-Side stack mainly responsible for handling the user's request and interaction. This section covers the client-side technology such as the Unity game engine, the Mixed Reality Toolkit (MRTK) and finally, the Vuforia SDK used to build the application on a HoloLens 2.

3.1.1 Unity Game Engine

Unity3D is a multi-platform game development tool developed by a Danish company named Over the Edge Entertainment (OTEE) but they rebranded to Unity Technologies in 2007 [95] [96]. It is a user-friendly platform that allows the creation of AR or VR applications. A wide variety of free assets and templates are available for use. Details about Unity3D is easily obtainable due to the early release to the domestic market. Its main purpose is to develop video games, 3D animation, architectural environment and models.



```
CPU: 120 fps (8.3 ms)
Used: 306.7MB    Peak: 347.1MB    Limit: 16244.0MB
```

Figure 3-2. Unity Editor with ABB GameObject Scene View displayed on the left and Game View on the right.

A Unity3D project consists of multiple hierarchies of scenes that contains a collection of multimedia elements referred to as GameObject, these components are structured by C#, Javascript etc. Manoeuvring and manipulating the GameObject happens within the Scene View on the left as shown in Fig 3-2. The ABB GameObject appeared on the game View on the right as illustrated in Fig3-2 was capture by the camera object when the Editor Play Mode button is pressed. Unlike the Scene View, any changes made in Game View is provisional and will be reset back to the previous settings when the user exit the Play Mode [97].

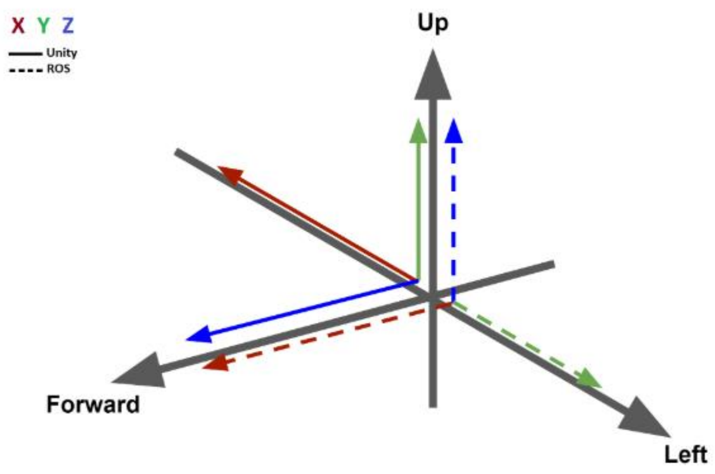


Figure 3-3. Coordinate systems of Unity and ROS Source: Adapted from [93]

Most AR applications evaluate the robot coordinates reference point and virtual objects in a left-handed coordinate system (LHS). The is due to the fact that the Unity3D frame axes follow the left-hand rule. In this case, y-axes are pointing up. ROS follows the right-handed system (RHS) with z-axes pointed up, as shown in Fig. 3-3. On account of the differences in world axes, custom messages that include a quaternion or 3D vector should obey the ROS world frame system [93]. Before publishing the messages to ROS, the Unity2Ros() method converts the Unity quaternion or 3D vector messages to the ROS coordinate system. Hence, both robot arms are aligned in terms of orientation and rotation.

In addition, Inverse kinematics were applied on the Unity robot arm. They calculate the position the joints should be in, in order for the end-effector to be at a specific point in space. Inverse Kinematics is a method to calculate the collision-free robot arm motion [85]. The FABRIK algorithm was chosen to control ABB 6 DOF robotic arm in this case. Please refers to Chapter 2.6.2 to obtain more information on the FABRIK algorithm. The Final IK asset [98] available in the Unity asset store allows Unity developers to integrate the FABRIK algorithm easily into their projects [98].

3.1.2 The Mixed Reality Toolkit (MRTK)

The Unity3D game engine is deployed to create the AR application. A suite of development tools is provided to create different experiences, and in this case, the Mixed Reality Toolkit (MRTK) [92] was used. MRTK is a Microsoft- motivated open-source development kit that offers a set of components and features used to accelerate application development for the Microsoft HoloLens 2 [99]. The MRTK in-editor hand input simulation allows the developer to test a scene in the Unity game view. The MRTK supports rapid AR application development. MRTK automatically sets up a default scene for a new project when the “Add to Scene and Configure” is selected. The HoloLens 2 implemented the style buttons provided by the MRTK example package to create a set of tag-along buttons as shown in Fig. 3-4(a). A Radial View script is attached to the parent button. This solver script allows the buttons to follow the user's head movement.

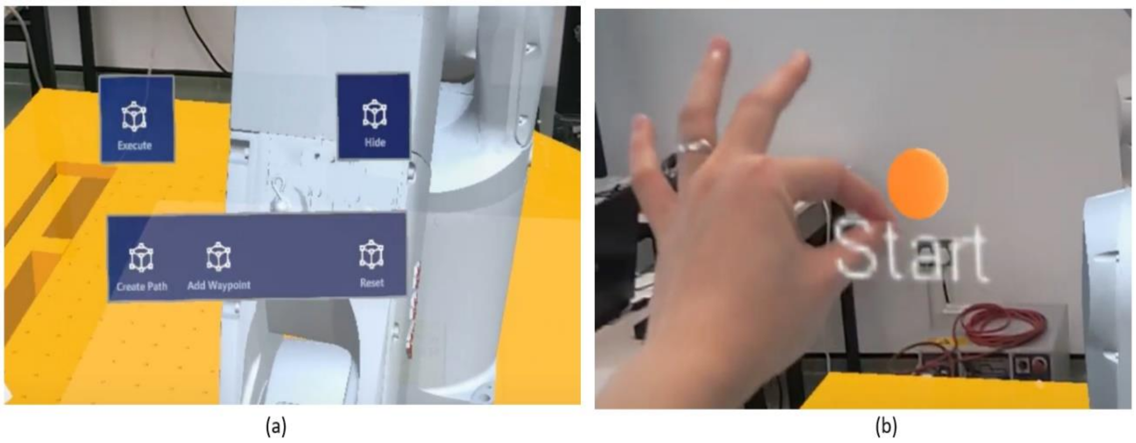


Figure 3-4. (a) Virtual Tag-Along Buttons Created with MRTK Examples package (b) Draggable 3D waypoint with MRTK articulated hand tracking input.

The Manipulation Handler and Near Interaction Grabbable scripts were attached to the waypoint GameObject as shown in Fig. 3-4(b) to make it draggable. When the InteractableOnGrabReceiver() detect the grab interaction performed by the user, an Interactable script will call the Component Manager script to switch the default material from white to orange. The Component Manager script consists of the MRTK_Standard_Glowing_Orange material that uses the MRTK Standard Shader. The MRTK Standard Shader is recommended since the MRTK Standard shader performs considerably less computation than the Unity Standard shader [100]. Safety feature such as the ‘Preview’ virtual button is designed to identify and avoid possible collision between the robot arm and its workspace. For example, the 3d model of the workspace or desk below the robot arm will turn red when a collision is detected in the preview mode.

3.1.3 Vuforia SDK

Vuforia was previously founded by Qualcomm in 2006 and is currently owned by PTC since November 2015. It utilizes camera vision technology to identify and track real-time 2D images as well as 3D objects [101]. Moreover, it is more accessible to a wide audience, from Android, iOS to UWP (Universal Windows Platform) and offers various tools for creating leading-edge AR experiences [60]. The Vuforia Engine package can also be easily added to a Unity project via a Git URL. Vuforia Image Target is utilized in our project to track and register the virtual robotic arm at a fixed position via the Hololens2 front-facing camera's coordinate system. The image target can be created at runtime in both native and Unity Editor besides uploading images to the Vuforia Target Manager [102].

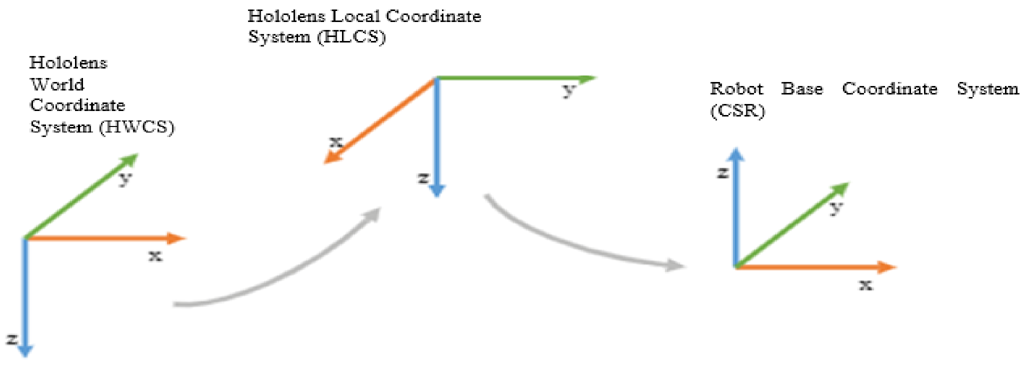

Hololens Local Coordinate System (HLCS)
Hololens World Coordinate System (HWCS)
Robot Base Coordinate System (CSR)
y
x
z
y
x
z
z
y
x

Figure 3-5. Coordinate transformation between the HoloLens 2 and the ABB robot arm adapted from [63]

In order to correctly teleoperate the ROS-enabled ABB robotic arm with the virtual robot arm that sits on top of the table, they must be transformed based on Fig. 3-5. Fundamentally, the robot arm pose is specified by the HoloLens 2 in a world coordinate system (HWCS). Hence, the start and goal coordinates are first transformed into the local coordinate system (HLCS) defined on the Vuforia image target. The start and goal end-effector pose will then be transformed into the robot's base coordinate system (CSR). The conversion between the left-handed Unity-coordinate system and right-handed ROS-coordinate system is performed next before the end-effector poses are sent to the ROS-enabled ABB robot from the HoloLens 2.

3.2 Server -side Stack

The Server-Side stack mainly responsible for receiving the user's request and feedback from the Client-Side. This section covers the server-side technology such as the Robot Operating System (ROS), MoveIT, ROSBridge and finally, the Robot Model used to teleoperate the physical ABB robot arm.

3.2.1 Robot Operating System (ROS) Settings

ROS is a versatile, multilingual middleware. It provides a collection of tools and libraries for robotic developers to simplify their effort of designing complicated and robust robot actions [103]. ROS is open-source and is language agnostic. It has become the common automation language for robotics and research focus in recent years. It is widely used in various robotic applications. Gazebo, as shown in Fig. 3-6(a), serves as a robotic simulator while ROS is the interface to the robot. The key benefit of using Gazebo is its ready integration with ROS. Similar to ROS, Gazebo is also open source. It is one of the most popular robotic simulators [68]. RViz, as shown in Fig. 3-6(b), is not a simulator but a 3D visualisation tool for ROS applications. It offers real-time sensor information from robot sensors and a view of the robot model. It can be used to display 2D and 3D point clouds as well as camera or lasers data [104].

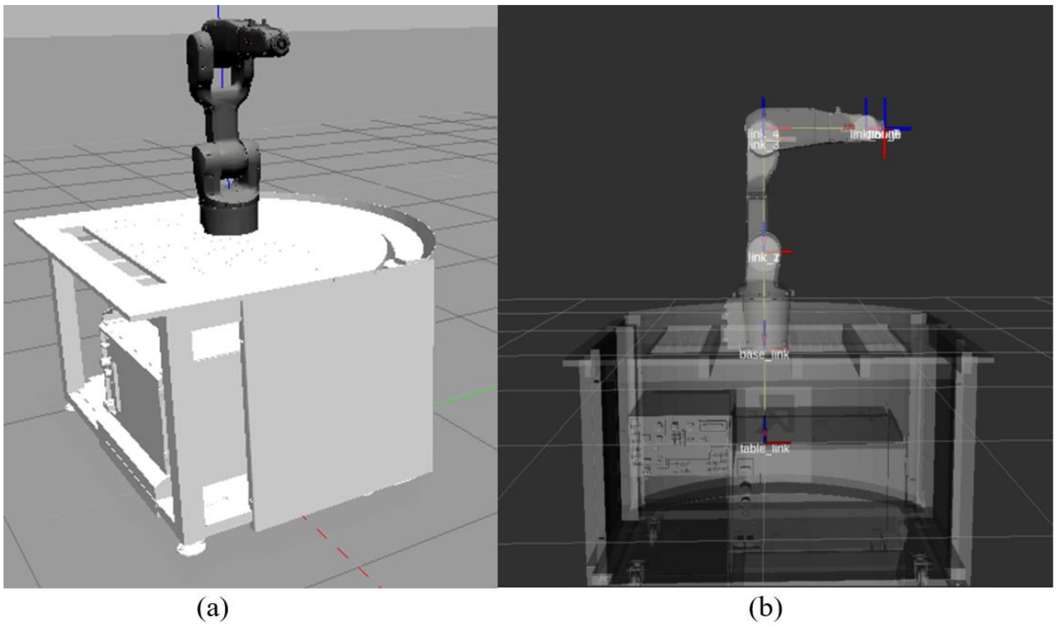


Figure 3-6. (a)ABB robot arm in Gazebo simulator (b) ABB robot arm in RViz with transform frames.

A node (as per Fig. 3-7) in ROS represents a separate program running a specific task. It can be written in any programming language, although it is primarily developed in C++ and Python [105]. Each standalone task can be divided into different nodes which communicate with each other through unique named buses. These so-called unique named buses are described as topics in [105].

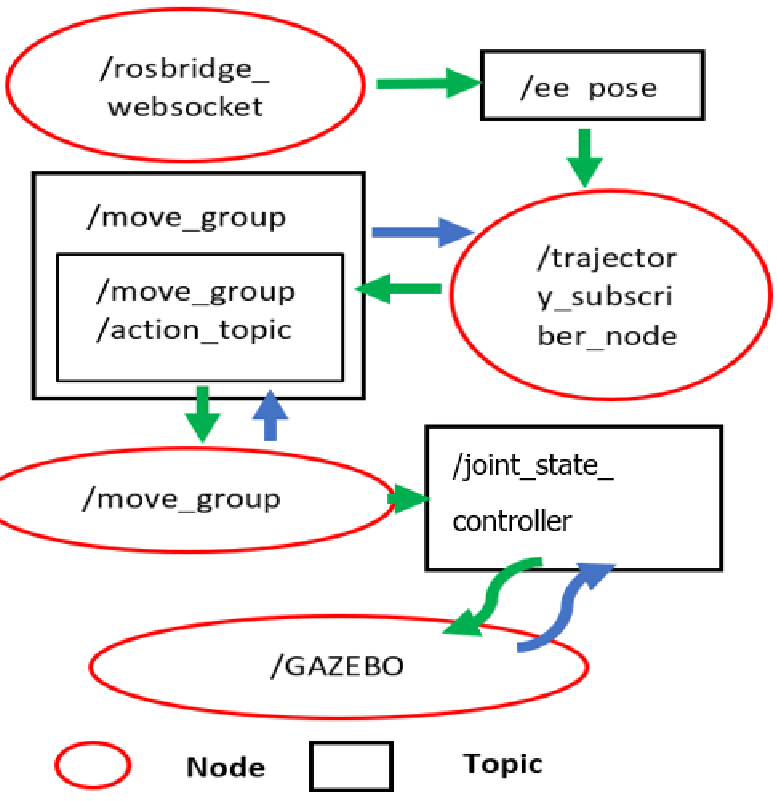


Figure 3-7. RQT graph showing the ROS Gazebo system in terms of nodes and topics

A publisher can broadcast messages into a ROS topic, whilst a subscriber listens and retrieves the messages from a topic. In order to successfully manipulate the robot arm in Gazebo as shown in Fig. 3-6(a), the first terminal will launch 6 nodes, namely: the file server; Gazebo simulator server; Gazebo simulator GUI client; ROS API libraries; ROSBridge WebSocket; and finally, the Gazebo URDF model node. The URDF model node will spawn the ABB robot arm model in the Gazebo simulator.

The controller manager node and the robot state publisher node are launched. The ROS control framework is defined inside a YAML[106] configuration file. It is used to implement and manage the simulated gazebo robot controllers. An action server uses FollowJointTrajectory type messages for controlling the joints of the robotic arm. The robot state publisher node listens to /joint_states messages from the joint_state_controller and publishes the transforms to /tf. The transform or /tf package keeps track of all 3D coordinate frames such as: world frame; base frame; shoulder frame; and the end-effector frame that changes over time to identify the robot's current location in the world. For example, it records the pose of the target at the end effector relative to the base coordinate frame [107]. Running both terminals at this stage will start a controlled simulated environment of the whole robotic system. All topics need to be verified as active and available by executing a "rostopic list" command. This ensures that the simulated robotic system is running correctly.

3.2.2 MoveIT

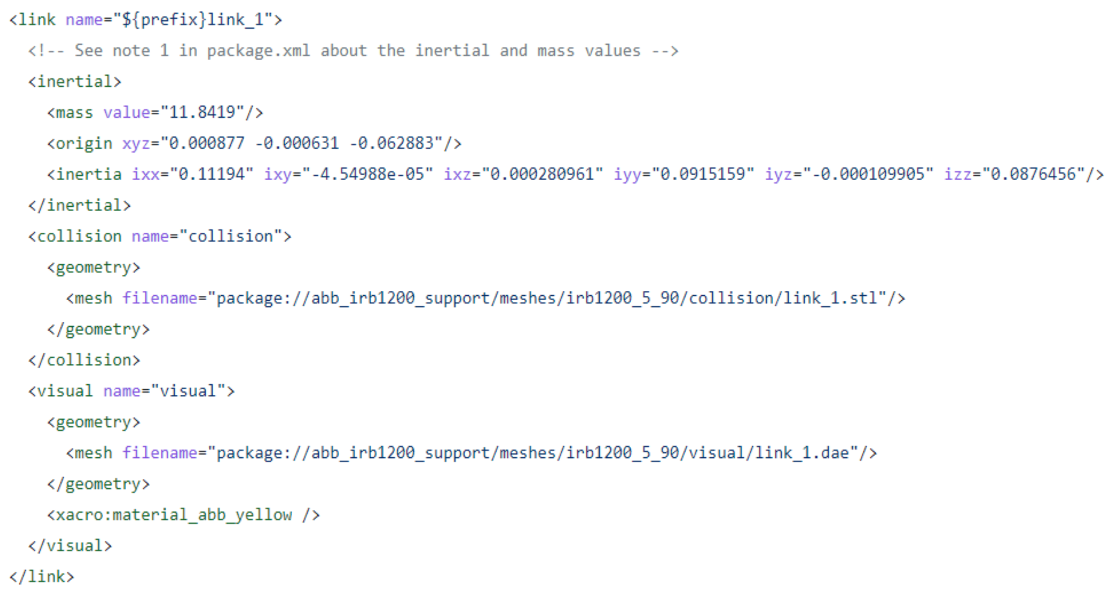
The MoveIt RViz environment is a ROS-based motion planning framework that provides extra inverse kinematics solvers called TRAC-IK plugins. In order to implement MoveIt, a MoveIt package needs to be configured using the built-in Setup Assistant tool. The robot arm's Universal Robotic Description Format (URDF) file is used to automatically generate MoveIt packages that contain all the configuration and launch files necessary for the robot model [108]. The URDF file (For more information, see Chapter 3.2.4) is an XML file. It contains metadata on the current state of the system, arm joint limits, the known position and the shape of the already assembled structure [109].

3.2.3 ROSBridge

The ROSBridge suite is responsible for establishing a connection link between the ROS nodes and any other application capable of parsing JSON (in our case the HoloLens 2 application connected over the internet) [110]–[112]. Serialized JSON messages (convert complex messages into byte strings) are sent over to the client via the WebSocket. The JSON messages are then deserialised (recover the original messages from the byte string) by the client into messages format supported by ROS. The client makes use of the communication tool provided by ROS to pass messages to the matching nodes and vice versa. Nevertheless, ROSBridge is used for just messages exchange purpose only.

A python script was developed in ROS to subscribe to the end-effector pose. When the Unity ROS Connector script is activated, the subscriber script is run. The trajectory_subscriber node continuously waits for Unity pose messages from /rosbridge_websocket node through /ee_pose topic. This was created earlier by the Unity pose_stamped_publisher. Based on the flowgraph as per Fig. 3-5, once the subscriber node has successfully acquired the updated robot arm positions and rotations, the python script directs the robot arm to reach the target position by calling the Move Group Interface class. This class wrapper is part of the MoveIt motion planning framework that provides all functionalities that are necessary to control the arm [108]. The interface communicates across ROS topics, services, and actions to the /move_group node. The node talks to the / joint_state_controller using the FollowJointTrajectoryAction interface and executes the robot trajectory on Gazebo simulator.

3.2.4 Robot Model



```
<link name="${prefix}link_1">
  <!-- See note 1 in package.xml about the inertial and mass values -->
  <inertial>
    <mass value="11.8419"/>
    <origin xyz="0.000877 -0.000631 -0.062883"/>
    <inertia ixx="0.11194" ixy="-4.54988e-05" ixz="0.000280961" iyy="0.0915159" iyz="-0.000109905" izz="0.0876456"/>
  </inertial>
  <collision name="collision">
    <geometry>
      <mesh filename="package://abb_irb1200_support/meshes/irb1200_5_90/collision/link_1.stl"/>
    </geometry>
  </collision>
  <visual name="visual">
    <geometry>
      <mesh filename="package://abb_irb1200_support/meshes/irb1200_5_90/visual/link_1.dae"/>
    </geometry>
    <xacro:material_abb_yellow />
  </visual>
</link>
```

Figure 3-8. A part of the URDF model of the ABB robot.

The fundamental design of the proposed HRI system is the open-source ABB IRB1200_5_90 robot model. It was provided by [113] in the form of a Unified Robot Description File (URDF) [109]. The ROS sharp folder [93] contains a RosBridge library and the UrdfImporter. The UrdfImporter is responsible for converting the URDF Robot model as shown in Fig. 3-8 into a Unity game object before it is imported into Unity via the RosBridge WebSocket. The URDF file consists of a detailed robot model description. The components of the robot such as links and joints are specified within the URDF file using the <link> and <joint> tags. Parameters such as the model weight, dimensions of the robot parts, inertia and collision geometry could be defined using the <mass>, <box size>, <inertia>, <collision> and finally, the < geometry> tags [114][115].

3.3 PHYSICAL ABB IRB1200 ROBOT ARM

An ABB IRB1200 5/0.9 robotic arm as illustrated in Fig. 3-9 was installed in the college's robotic lab. There are two versions of this robotic arm, ABB IRB1200 5/0.9 and ABB IRB1200 7/0.7. Each of these versions has two types, type A robot calibrates with Axis Calibration, type B robots inherit type A characteristics with SafeMove 2 as an additional feature that differentiates the two types. SafeMove 2 is a feature that provides functionality to create an efficient, flexible environment to achieve safety-certified tracking of speed limitation, standstill supervision, robot motion, and tool [116]. This robotic arm has 6 joints, each joint has 6 degrees of freedom, the maximum distance it could reach is 900 mm, it weighs 54 kg and it could carry a payload up to 5kg [117].

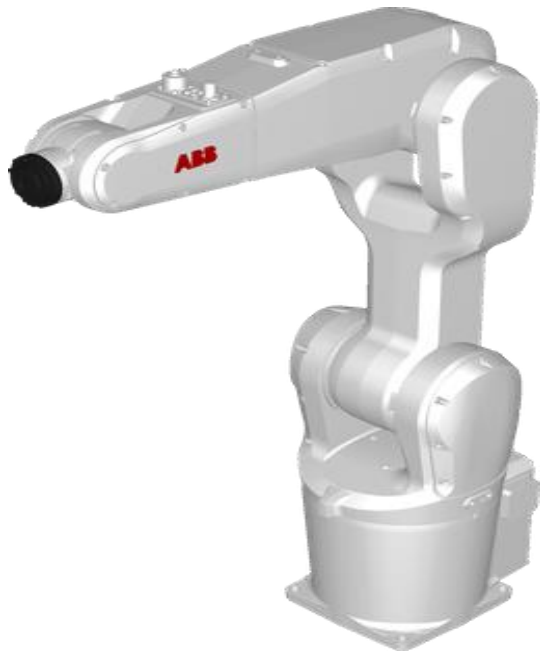


Figure 3-9. ABB IRB1200 5/0.9 robotic arm

3.3.1 The RAPID programming language

RAPID is a high-level programming language that is only exclusive to ABB industrial robots and it has a similar framework compare to common programming languages such as C, Python, etc, it has procedures, routines, error handling, multi-tasking, and it can run a maximum of 10 threads simultaneously [118]. According to [119], the

IRC5 controller is responsible to execute the RAPID code. It is a robot controller that features speed and accuracy to perform tasks, the ability to reduce cycle time with QuickMove and deliver precise path accuracy with TrueMove. QuickMove maintains maximum acceleration by optimizing cycle time to the minimum possible, TrueMove directs motion path according to programmed path disregard of the speed of the robot, these two features are the standard motion technology that prebuilt into IRC5 controller [120]. To establish a connection from the ABB robot to ROS, the ABB ROS server written in RAPID needs to be implemented onto the IRC5 controller.

3.3.2 ROS-Industrial package

The ROS-Industrial package is an additional project that includes libraries for varieties of robot manufacturers such as Fanuc, ABB, Kuka, etc, with this package manipulation of physical robots, became possible. It supports several essential functions that establish a communication channel for industrial robots to sends and receive instructions [121]. The three major functions are described in detail as follow:

- **Position Streaming:** The controller is constantly executing trajectory with minimal delay as it does not wait for the full path trajectory, the velocity will be controlled by the controller.
- **Trajectory Downloading:** The full path trajectory joint position and velocity constraint will be downloaded from ROS to the robot controller to be executed. After the latest acquired trajectory has been executed, the robot will remain stationary until the next trajectory arrived.
- **Trajectory Streaming:** This interface has similar traits as Position Streaming, except the velocity constraint is not managed by the controller.

The ROS- Industrial packages consist of an ABB sub-package called abb_experimental package. According to [122], the package includes nodes that enabled communication between ROS and the ABB industrial robot controllers. The URDF models and ROS MoveIt packages are associated with this package. The ABB controller drivers (provided also by the ABB package) must be installed on the robot controller to facilitate such communication.

4 EXPERIMENT AND RESULT DISCUSSION

The main design of the experiment as illustrated in Fig. 4-1 consists of a computer running Ubuntu16, HMD Microsoft HoloLens 2, and an ABB robot arm version IRB 1200-5 /0.9. We will place an image target marker on a table 15cm in front of the physical robot arm. The data collection flowchart of the RoSTAR system is shown in Fig. 4-2. The evaluation of the proposed RoSTAR system was achieved by running two experiments. It is the same person that performs the testing sequences throughout the first and second experiments.

4.1 TEST SCENARIOS

A series of objective measures for the ABB robot arm was conducted to test the robot path accuracy, end-effector's position accuracy and path completion time.

1. **Path Accuracy:** Path accuracy was calculated as the number of correct trajectories executed by the robot divided by the total number of trials (10 trials). We treated the robot arm movement as timeout when the robot arm does not move as predicted/ spinning 180 degrees along the pre-specified trajectories (Singularities). The path accuracy results will be plotted in the form of graphs.

2. **Absolute Position Accuracy:** The position measurements across four different complexities are stored only after the end-effector arrived at each pose (irrespective of the path taken by the robot to reach the goal position). The robot arm will execute the same trajectories 10 times after receiving the pose data.

3. **Average Path Completion Time:** Each trial began when the robot moves from start to goal position. The time measure was calculated as the average time of the interaction across all 10 trials.

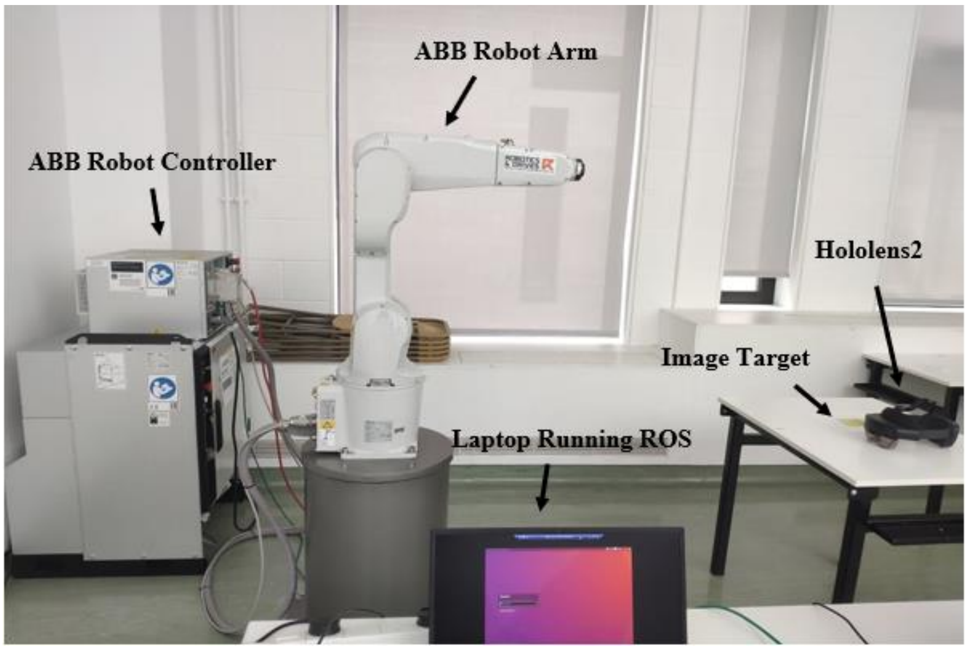


Figure 4-1. Experiment Setup

The performance of the HoloLens 2 in the task was evaluated by monitoring the frame rate at four different stages mainly: Active Tracking Stage, Fixed Position Stage, Trajectory Execution Stage, End Trajectory Stage. The objective measures for the HoloLens 2 are frame rate and path completion time.

1. **Frame Rate:** The frame rate is calculated by averaging the total frame rates received every second at each stage. The HoloLens 2 frame rate are collected every second at the start up and classified into four main stages: The tracking stage, planning stage, execution stage and reset the stage.
2. **Average Path Completion Time:** Each trial began when the virtual robot arm moves from start to goal position. The time measure was calculated as the average time of the interaction across all 10 trials.

The average path completion time across 4 levels of trajectory complexity for both HoloLens 2 virtual robot arm and real ABB robot arm based on the number of waypoints are compared via a 3D-clustered column chart. A shorter path completion time indicates fast inverse kinematics solve times.

4.1.1 Experiment One:

The first experiment is mainly designed to evaluate the performance difference between the virtual and the real telerobotic arm. In addition, to identify the capability of the HoloLens 2 in visualising the robot movement. During start-up, a script is launched to activate a timer and record the headset frame rate. It continuously collects the frame rate and time data in milliseconds until the app is closed. The frame rate and time data are stored inside a .csv file. The file can be found in the persistent data directory and accesses via the HoloLens Window Device Portal. The HoloLens 2 camera is activated to scan the Vuforia image target. Vuforia tracking will be disabled once the holographic robotic arm that sits on top of a large virtual desk is rendered and the “Stop Tracking” button is pressed. This keeps rendered virtual robotic arm fixed at the last position in 3D space. (As illustrated in Fig.4-2 and Fig.4-3). A 3-dimensional robotic path can be formed by dragging and dropping the waypoints to a specific location in 3D space and pressing

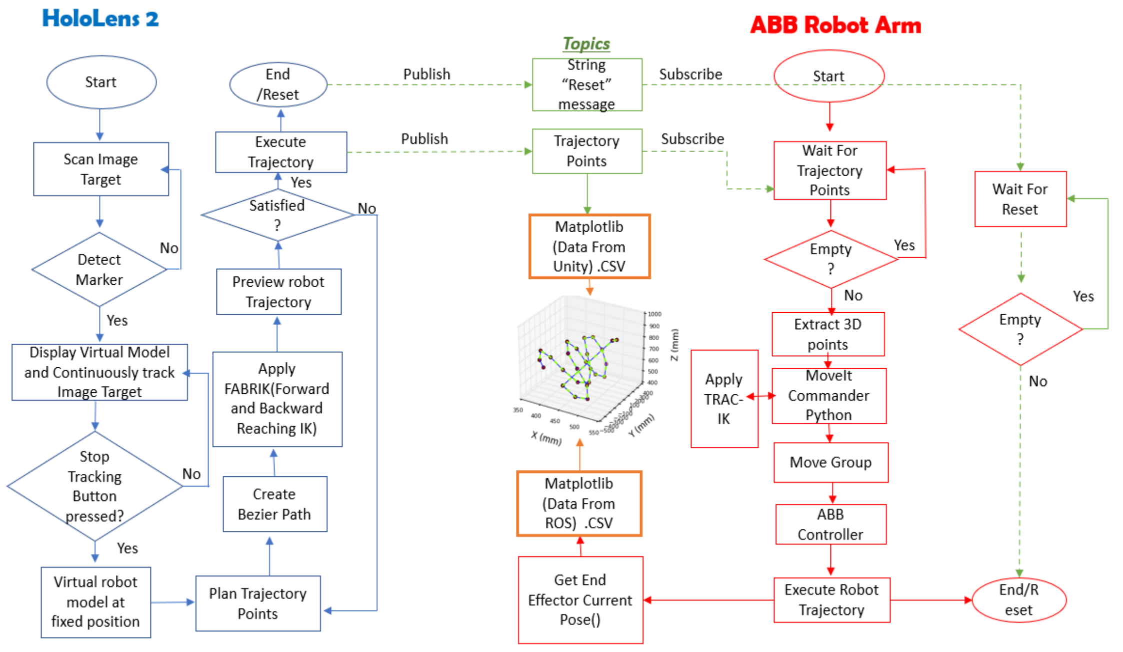


Figure 4-2. The logic flowchart of the RoSTAR System for data collection.

the “Create Path” button. The robot motion in the 'Preview' mode is used to identify whether or not a robot arm motion would collide with the desk. For safety reasons, the “Execute” button is disabled and the user is required to run the “Preview” mode at least once to activate the 'Execute' button. When the “Execute” button is pressed, the robot end-effector coordinates are saved in an array that merges into a single string. The string is then published to a ROS topic over the internet.

The ABB robot arm constantly waits for the trajectory points from Unity. Once it detects that the topic is not empty, it subscribes and extracts the end-effector trajectory pose consecutively. A 3D graph of Unity published trajectory will be plotted via the Matplotlib library. At the same time, the extracted 3D trajectory points are sent to the

MoveIt Commander for motion planning. Within the MoveIt library, TRAC-IK inverse kinematics solver is applied and the Move Group interface will create the motion plan according to the pose given by the MoveIt Commander. The ABB controller received the planned motion and execute the robot arm trajectory. As shown in Fig. 4-3, the level of trajectory complexity is based on the number of waypoints assigned: 4 waypoints (Low), 8 waypoints (Medium), 12 waypoints (High) and 16 waypoints (Extreme). The number of waypoints assigned will indirectly indicate the shape complexity of the robot trajectory.

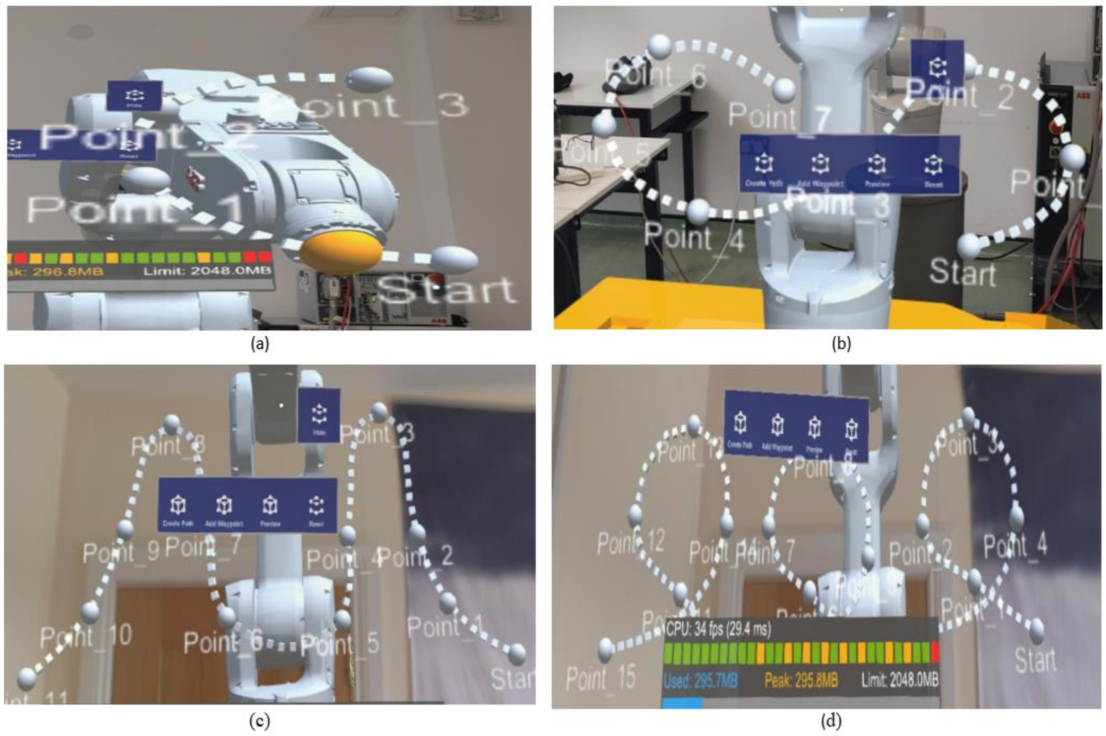


Figure 4-3. The level of trajectory complexity is based on the number of waypoints assigned. (a) 4 waypoints (Low), (b) 8 waypoints (Medium), (c) 12 waypoints (High), (d) 16 waypoints (Extreme)

The whole experiment process is repeated 10 times across each test type. During robot execution, Matplotlib will get the current end-effector's pose every second/ in real-time and save the trajectory data in .csv format. The ROS trajectory is plotted with Matplotlib. The data is then compared with the Unity trajectory on the same 3D graph. The path completion time (from start to goal) of the ABB robot arm is recorded simultaneously within the python script.

4.1.2 Experiment Two:

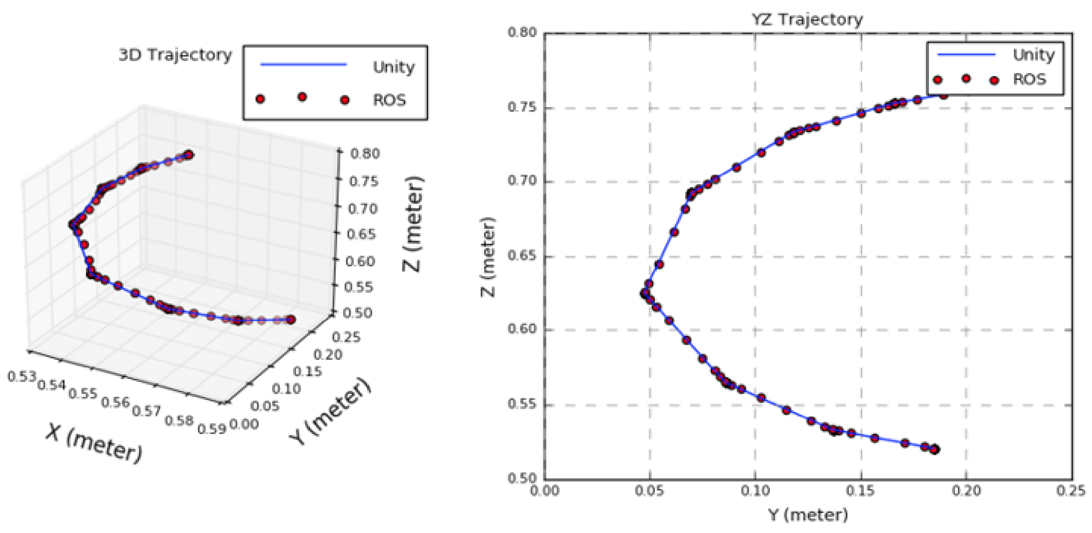
The second experiment is mainly designed to evaluate the absolute position accuracy of the robot's end-effector. The whole experiment procedure on the HoloLens 2 side is the same as the first experiment but instead of plotting a total of 10 trajectories across four different complexities, each trajectory complexity is plotted only once with the HoloLens 2 and all three translational axes, x, y, and z are then sent to the real robot arm. The real robot arm will then subscribe the pose as usual from the same topic as illustrated in Fig. 4-2. A separate python script is written to store the subscribed pose. Next, the robot arm will execute the same trajectories 10 times after receiving the pose data. To assess the absolute position accuracy of the robot end-effector, the position measurements are stored only after the end-effector arrived at each pose (irrespective of the path taken by the robot to reach the goal position). Both the virtual and robot translational axes, x, y, and z are stored in .csv format. Rotational experiments are not performed in this study. The Mean Absolute Percentage Error (MAPE) across the translation axes will be computed to measure the difference between the actual and predicted value, also known as the forecast accuracy.

4.2 RESULTS AND DISCUSSION

Based on the test scenarios in Section 4.1, the presentation of the measurement results is subdivided into three sub-sections: the accuracy result in Section 4.2.1, the path completion time results in Section 4.2.2 and finally the frame rate results in Section 4.2.3.

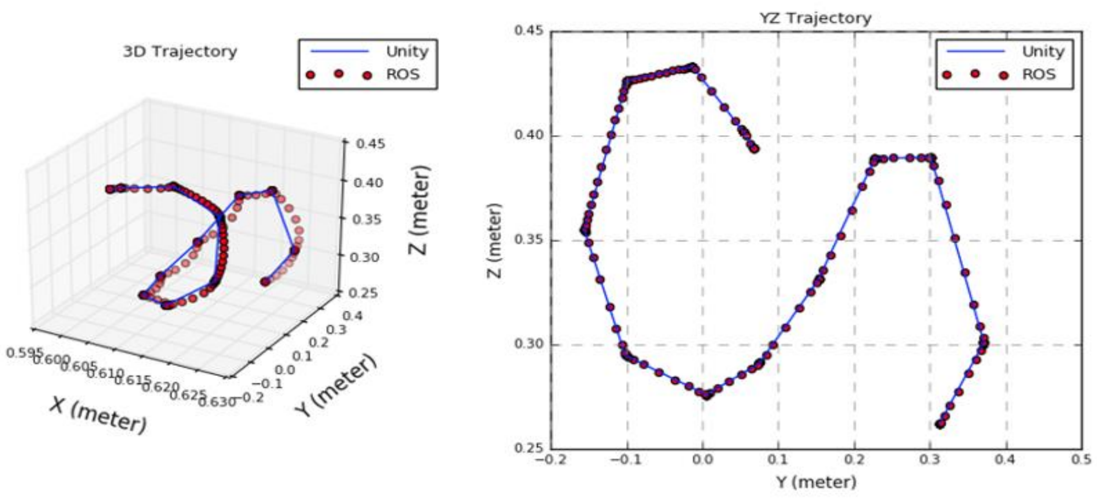
4.2.1 Accuracy

In this Section, path accuracy and positional accuracy between the HoloLens 2 virtual robot arm and the physical real robot arm are compared. Here, we detail the path accuracy graph plotted during our experiments across different trajectory complexity levels (see Fig. 4-4 to Fig. 4-7). The blue straight line represents the virtual robot trajectory while the red dotted line represents the ROS-enabled physical abb robot arm trajectory.



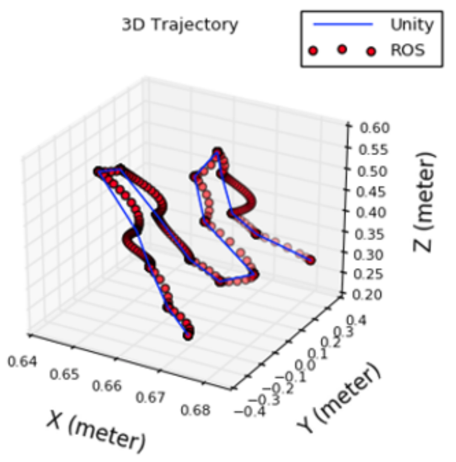
(a) (b)

Figure 4-4. (a) Low Complexity robot end-effector trajectory in a 3-D plane. (b) Low Complexity robot end-effector trajectory in yz- plane.

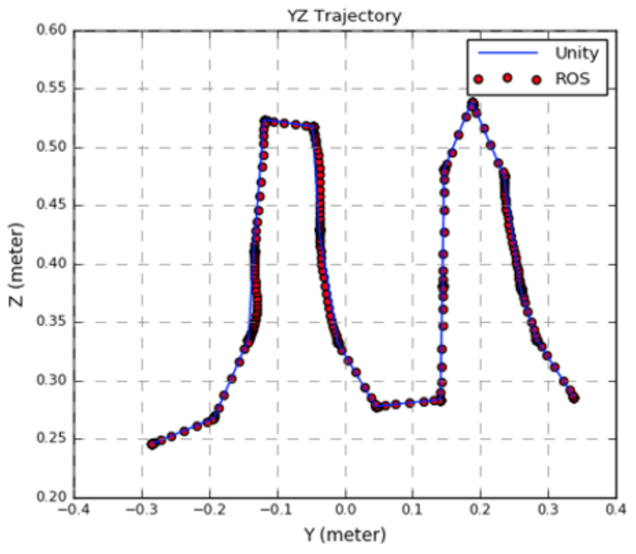


(a) (b)

Figure 4-5. (a) Medium Complexity robot end-effector trajectory in the 3-D plane. (b) Medium Complexity robot end-effector trajectory in yz- plane.

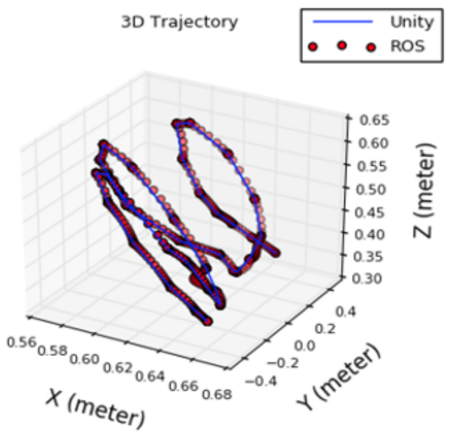


(a)

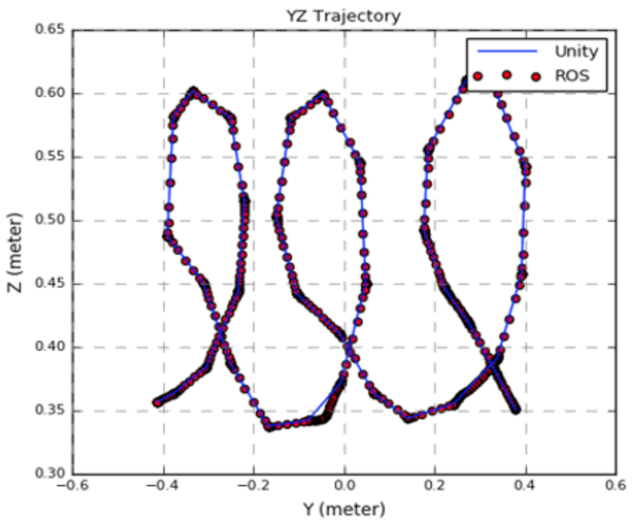


(b)

Figure 4-6. (a) High Complexity robot end-effector trajectory in the 3-D plane. (b) High Complexity robot end-effector trajectory in yz- plane.



(a)



(b)

Figure 4-7. Extreme Complexity robot end-effector trajectory in the 3-D plane. (b) Extreme Complexity robot end-effector trajectory in yz- plane.

The Average Path Accuracy across 10 trials for each robot trajectory is plotted in the form of clustered column chart as illustrated in Fig.4-8. where the number of correct trajectories executed by the robot is divided by the total number of trials (10 trials) and multiply by 100%. We treated the robot arm movement as timeout when the robot arm singularities were detected. Singularities are also caused by the inverse kinematics of the robot. When placed at a singularity, there may be an infinite number of solutions for the kinematics to accomplish the same tip position of the robot. If the best solution is not chosen, the robot joints could be instructed to move in an uncontrollable and unintended way. The most common and frequent encountered singularity in our system will be the wrist singularity. Wrist singularity happens when the axes of joints 4 and 6 become parallel and this led to a condition that corresponds to θ5 = 0° [123]. The most obvious way to detect a wrist singularity is through the plotted graph as shown in Fig. 4-6 (a). The red dotted line robot's end-effector trajectory which shows a sharp rise and slightly off track from the Unity end-effector trajectory are signs of wrist singularity where the wrist joint is spinning 180 degrees toward the next point along the path.

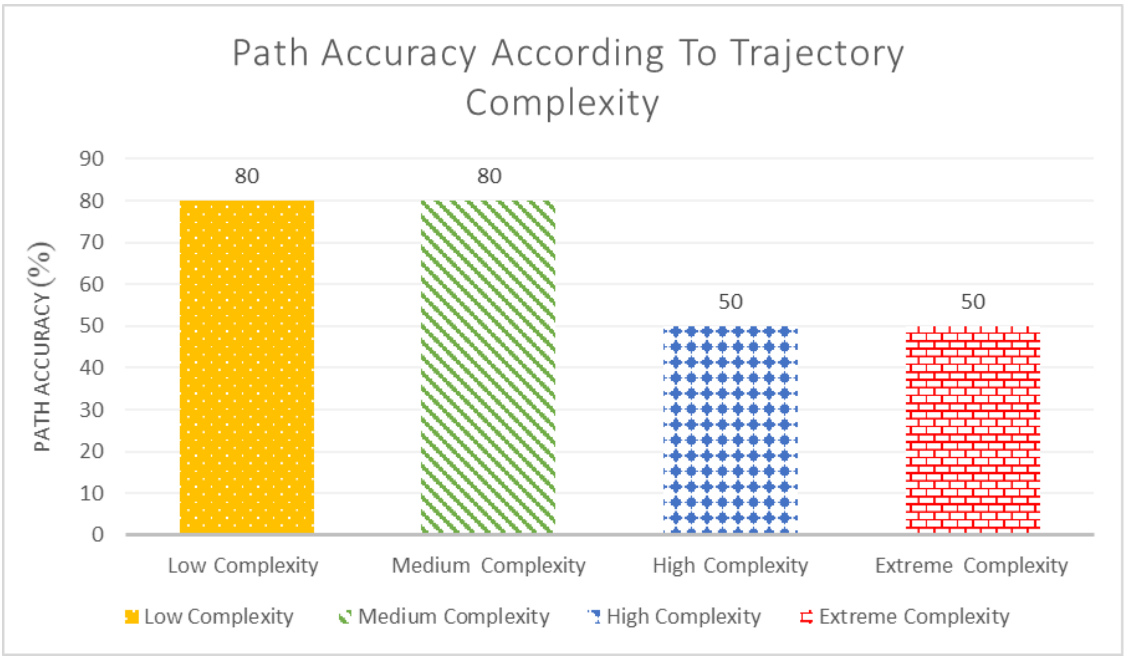


Figure 4-8. Average Path Accuracy graph across different trajectory complexity.

A relevant question is if the level of trajectory complexity influences the path accuracy result. According to the chart in Fig. 4-8, trajectory paths with low and medium complexity which in this case, below 8 user assigned waypoints, have greater accuracy if compared to the robot trajectory with high and extreme complexity. In conclusion, the lower the trajectory complexity, the higher the path accuracy.

According to [124], the position accuracy of the robot arm from a statistical point of view can be defined as the distance between the desired end-effector position and the actual end-effector position which is achieved after repetitive movements of the end-effector toward the goal position. In terms of computation, absolute accuracy is the collection of the composed errors for each of the x, y, z positional errors in cartesian space. The mean absolute percentage error (MAPE) is the most common and easy to understand measure used to forecast error. Mean Absolute Percentage Error (MAPE) is defined as a formula used to measure the accuracy of a forecast system in terms of percentages [125]. The smaller the value of MAPE the better the forecast. The formula adapted from [125] can be expressed as:

$$\mathrm{M} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \tag{10}$$

where A_t denotes the actual value, the actual value will be the physical robot arm end-effector x, y and z value and F_t is the forecast value, in our case will be the Unity end-effector x, y and z value. Σ is summation notation.

Table 3-2. Mean Absolute Percentage Error (MAPE) across 10 trials per trajectory complexity level

Trajectory Complexity	Average x MAPE (%)	Average y MAPE (%)	Average z MAPE (%)
Low	0.0064	0.0476	0.0104
Medium	0.0060	0.1428	0.0121
High	0.0065	0.0384	0.0092
Extreme	0.0059	0.0350	0.0113

Table 3-2 shows the result of Mean Absolute Percentage Error (MAPE) across 10 trials per trajectory complexity level. According to Swanson et al. [126], a MAPE forecast which is less than 5% is assumed to be highly accurate. It is notable that all MAPE percentages across x, y and z are very small, this indicates a high predictive accuracy between the physical ABB and the HoloLens 2 virtual robot arm. The Average x-position in terms of MAPE across four different trajectory complexity is around 0.0059%– 0.0065%. For the y- position, the MAPE value is slightly higher, which is around 0.035%- 0.1428% while the z-position MAPE value is around 0.009%- 0.012%. By referring to Table II, the end-effector achieved the highest accuracy at the x-position of the extreme trajectory level. However, the least accurate end-effector position is the y-position where MAPE value is 0.1428% while executing the medium level trajectory complexity.

4.2.2 Time

In this Section, the trajectory completion time for both HoloLens 2 and ROS enabled ABB robots are recorded and compared as shown in Figure 4-9. It is the most straightforward measure index of performance in terms of teleoperation control. The time is recorded (in seconds) from when the robot starts to move until it reaches the last user assigned waypoint. The bar chart illustrates the average path completion time of HoloLens 2 virtual robot arm and ROS- enabled robot arm across different trajectory complexity levels.

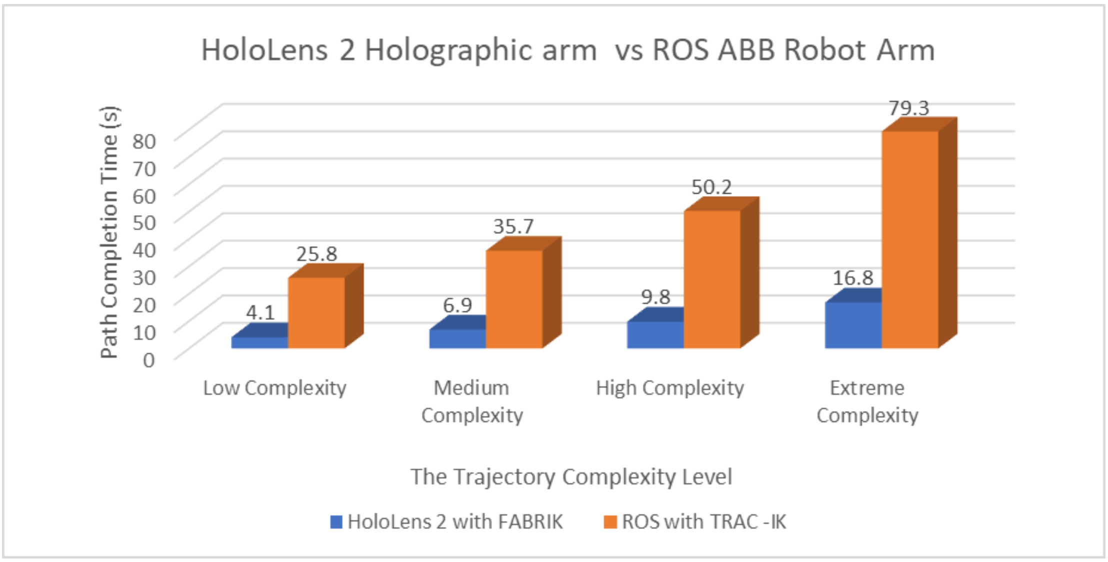


Figure 4-9. Path Completion Time of HoloLens 2 virtual arm vs the ROS enabled ABB robot arm.

It can be seen that the path completion time for the HoloLens 2 with Forward and Backward Reaching Inverse Kinematics (FABRIK) increases slowly over the trajectory complexity level. While the path completion time for the ROS-enabled ABB robot arm increased dramatically with a steep upward trend when the number of assigned waypoints increases. It could be noted that with the increased trajectory complexity level the path completion time will increase somewhat. This happens because longer inverse kinematics solve time is needed if more waypoints are assigned. The longest path completion time taken by the HoloLens 2 at an extreme complexity level is 16.8 seconds on average. On the other hand, the ROS-enabled robot arm with the TRAC-IK algorithm took an average of 79.3 seconds to complete the extreme path trajectory. The efficiency that the FABRIK inverse kinematic solver provide can be seen in the HoloLens 2 end-effector trajectories of the test sequences. For example, in Fig. 4-9 we see that the HoloLens 2 robot arm trajectory has a shorter path completion time (fast solving speed) overall with the FABRIK solver if compare to the ROS-enabled ABB robot arm with TRAC-IK solver. A shorter path completion time indicates fast inverse kinematics solve time. We, therefore, concluded that the FABRIK solver outperformed the TRAC-IK solver in terms of path completion time.

4.2.3 Frame Per Second

In this section, we describe the experiments conducted for assessing the stability of the Microsoft HoloLens 2. The frame per second (fps) across four different stages on the HoloLens 2 was recorded and evaluated in the form of line graphs. The frame rate is an important parameter for evaluating the visual performance of the HoloLens 2. The target frame rate of the Hololens 2 is set to 60 fps.

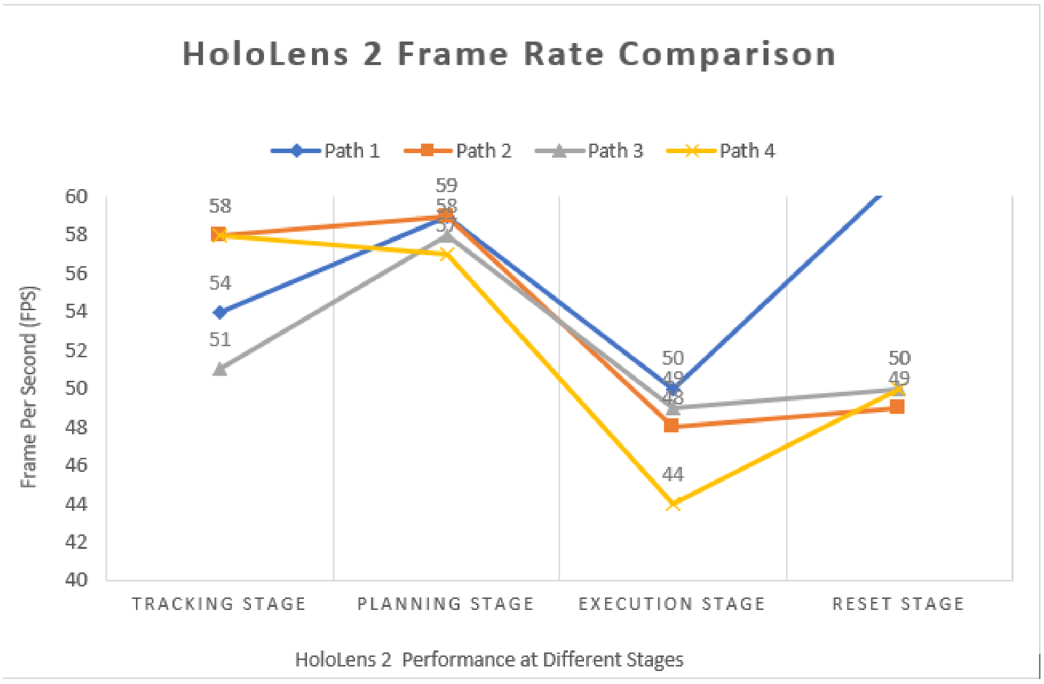


Figure 4-10. HoloLens 2 performance across four different stages: The tracking stage, planning stage, execution stage and finally the reset stage.

According to the result shown in Fig. 4-10, the HoloLens 2 entered the Vuforia tracking stage on start-up. The fps is pending from 51 to 58 when the HoloLens 2 camera was actively tracking the image target. The HoloLens 2 camera is switched off while entering the planning phase in which the virtual arm is located firmly at a fixed position when the ‘stop tracking’ button is pressed. The fps throughout the planning stage was constant and maintained at a very high fps between 57 to 59 fps across different trajectory complexity. The result shows that the HoloLens 2 is capable of visualizing high amounts of game objects. Nevertheless, the evaluation has shown that visualization of constantly outgoing data (publishing data over the internet to ROS) clearly affects the overall HoloLens 2 performance. This can be seen when both the execution stage and the reset stage as shown in Fig.4-10 involved the transferring of data that leads to a significant drop in fps if compare to the previous planning stage. However, the robot trajectory with the least waypoints assigned or low complexity shows a quick rebound from 50 back to constant 61fps during the reset stage. Nonetheless, only a little increase in fps have been observed when resetting the medium, high and extreme complexity robot trajectory. All robot trajectories fps will restore to a constant fps of 57-60 again at the next planning cycle. Overall, the evaluation results show that the HoloLens 2 is capable of displaying a great amount of 3D objects without any visual and performance downscale. The drop in fps during the HoloLens 2 data transfer is within the acceptable range (44 -61fps).

5 CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

In this thesis, we presented a novel open-source HRI system based on the Robot Operating System (ROS) and Augmented Reality (AR). In a nutshell, all objectives have been achieved in this project. We demonstrated that the HoloLens 2 is capable of visualising the robot movement and safety zone within a cutting-edge AR device. We propose a prototype application that successfully displayed key parameters of user-defined robot trajectory within the Microsoft HoloLens 2. The latter was evaluated in terms of performance to give significant insight into the feasibility of HMD-AR integration into robotics. Evaluation results show that the HoloLens 2 is capable of maintaining a high performance of 55–60 fps while performing simple to complex path planning tasks. By comparing the HoloLens 2 robot arm FABRIK solver with the ROS-enabled ABB robot arm TRAC-IK algorithm for mapping the movement between the robots, we can prove that the FABRIK solver outperformed the TRAC-IK solver in terms of path completion time. The proposed HRI system achieved a very high positional accuracy between the HoloLens 2 simulated and ROS-enabled ABB robot arm with a MAPE forecast between 0.0059%– 0.0065% (which is far below the standard5%). The trajectory path between the virtual and the real robot arm showed very realistic results, however, the path accuracy decreases when the trajectory complexity increases due to the existence of kinematics singularities.

5.2 LIMITATIONS

The limitations of our robotic system are as follows. Our robotic system which consists of 6 six revolute joints has singularities when using cartesian-space motion commands. There are scenarios where the ABB robot arm may have problems moving to a position, either because it is physically not possible for the robot to reach the required position, or because it is not possible to get there from the current position of the robot joints. Singularities are also caused by the inverse kinematics of the robot. When placed at a singularity, there may be an infinite number of solutions for the kinematics to accomplish the same tip position of the robot. If the best solution is not chosen, the robot joints could be instructed to move in an uncontrollable and unintended way. Due to time limitations, this problem has not yet been solved. In future, joint constraints could be added to let the planner know that we could never make the robot go to that configuration [127].

5.3 FUTURE WORK

In the future, the technical implementation could have been improved by installing the welding torch at the end of a robotic arm. The installation of such a tool could be highly advantageous in making the HRI scenario more practical and lifelike. Furthermore, we have also pointed out the kinematics singularities issue that must be addressed in order to achieve a higher path accuracy. Future work on HoloLens 2 must also include tracking the CPU usage when the HoloLens 2 was in use. Subjective assessments such as a Likert scale will be carried out in future to calculate the Mean Opinion Score (MOS) of participants. A cross-scale Repeated-Measures Analysis of Variance (RMANOVA) will be applied to investigate the difference in mean scores under three different conditions (Bezier curve, Cubic Hermite and B-spline). System Usability Scale and NASA Task Load Index questionnaires could also be used to assess the system's workload, usability, enjoyment.

BIBLIOGRAPHY

[1] A. C. Pereira and F. Romero, ‘A review of the meanings and the implications of the Industry 4.0 concept’, *Procedia Manuf.*, vol. 13, pp. 1206–1214, Jan. 2017, doi: 10.1016/j.promfg.2017.09.032.

[2] D. Kerpen, M. Löhrer, M. Saggiomo, M. Kemper, J. Lemm, and Y.-S. Gloy, ‘Effects of cyber-physical production systems on human factors in a weaving mill: Implementation of digital working environments based on augmented reality’, in *2016 IEEE International Conference on Industrial Technology (ICIT)*, Mar. 2016, pp. 2094–2098. doi: 10.1109/ICIT.2016.7475092.

[3] J. Egger and T. Masood, ‘Augmented reality in support of intelligent manufacturing – A systematic literature review’, *Comput. Ind. Eng.*, vol. 140, p. 106195, Feb. 2020, doi: 10.1016/j.cie.2019.106195.

[4] D. Feil-Seifer and M. Mataric, ‘Human-Robot Interaction’, in *Encyclopedia of Complexity and Systems Science*, 2009, pp. 4643–4659. doi: 10.1007/978-0-387-30440-3_274.

[5] M. Goodrich and A. Schultz, ‘Human-Robot Interaction: A Survey’, *Found. Trends Hum.-Comput. Interact.*, vol. 1, pp. 203–275, Jan. 2007, doi: 10.1561/1100000005.

[6] T. Williams, D. Szafir, T. Chakraborti, and E. Phillips, ‘Virtual, Augmented, and Mixed Reality for Human-Robot Interaction (VAM-HRI)’, in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, Mar. 2019, pp. 671–672. doi: 10.1109/HRI.2019.8673207.

[7] L. H. Manring *et al.*, ‘Augmented Reality for Interactive Robot Control’, 2020. doi: 10.1007/978-3-030-12243-0_2.

[8] G. Niemeyer, C. Preusche, and G. Hirzinger, ‘Telerobotics’, in *Automatica*, vol. 25, 2008, pp. 741–757. doi: 10.1007/978-3-540-30301-5_32.

[9] T. Haidegger, J. Sándor, and Z. Benyó, ‘Surgery in space: the future of robotic telesurgery’, *Surg. Endosc.*, vol. 25, no. 3, pp. 681–690, Mar. 2011, doi: 10.1007/s00464-010-1243-3.

[10] T. Kot and P. Novák, ‘Application of virtual reality in teleoperation of the military mobile robotic system TAROS’, 2018, doi: 10.1177/1729881417751545.

[11] D. W. Hainsworth, ‘Teleoperation User Interfaces for Mining Robotics’, *Auton. Robots*, vol. 11, pp. 19–28, Jul. 2001, doi: 10.1023/A:1011299910904.

[12] G. Liu, X. Geng, L. Liu, and Y. Wang, ‘Haptic based teleoperation with master-slave motion mapping and haptic rendering for space exploration’, *Chin. J. Aeronaut.*, vol. 32, no. 3, pp. 723–736, Mar. 2019, doi: 10.1016/j.cja.2018.07.009.

[13] N. Small, K. Lee, and G. Mann, ‘An assigned responsibility system for robotic teleoperation control’, *Int. J. Intell. Robot. Appl.*, vol. 2, no. 1, pp. 81–97, Mar. 2018, doi: 10.1007/s41315-018-0043-0.

[14] A. Perkis *et al.*, *QUALINET White Paper on Definitions of Immersive Media Experience (IMEx)*. 2020.

[15] D. Krupke, F. Steinicke, P. Lubos, Y. Jonetzko, M. Görner, and J. Zhang, *Comparison of Multimodal Heading and Pointing Gestures for Co-Located Mixed Reality Human-Robot Interaction*. 2018. doi: 10.1109/IROS.2018.8594043.

[16] L. Peppoloni, F. Brizzi, C. A. Avizzano, and E. Ruffaldi, 'Immersive ROS-integrated framework for robot teleoperation', in *2015 IEEE Symposium on 3D User Interfaces (3DUI)*, Mar. 2015, pp. 177–178. doi: 10.1109/3DUI.2015.7131758.

[17] E. Dima *et al.*, 'View Position Impact on QoE in an Immersive Telepresence System for Remote Operation', in *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*, Jun. 2019, pp. 1–3. doi: 10.1109/QoMEX.2019.8743147.

[18] S. Kortekamp, S. Werning, I. Ingmar, and O. Thomas, *THE FUTURE OF DIGITAL WORK - USE CASES FOR AUGMENTED REALITY GLASSES*. 2019.

[19] A. Monferrer and D. Bonyuet, *Cooperative robot teleoperation through virtual reality interfaces*. 2002, p. 248. doi: 10.1109/IV.2002.1028783.

[20] S. Green, M. Billinghurst, X. Chen, and J. Chase, 'Human Robot Collaboration: An Augmented Reality Approach A Literature Review And Analysis', Jan. 2007, doi: 10.1115/DETC2007-34227.

[21] 'Human-machine interface | computing | Britannica'. https://www.britannica.com/technology/human-machine-interface (accessed Jan. 21, 2021).

[22] K. Rana, 'Output Devices of Computer [Explanation with Examples]', *ArtOfTesting*, Jan. 28, 2021. https://artoftesting.com/computer-output-devices-example (accessed Feb. 21, 2021).

[23] 'What is an Input Device? - Definition from Techopedia', *Techopedia.com*. http://www.techopedia.com/definition/2344/input-device (accessed Feb. 21, 2021).

[24] M. Bellis, 'Why Your Computer Keyboard Has a QWERTY Layout', *ThoughtCo*. https://www.thoughtco.com/history-of-the-computer-keyboard-1991402 (accessed Feb. 21, 2021).

[25] D. Bachmann, F. Weichert, and G. Rinkenauer, 'Review of Three-Dimensional Human-Computer Interaction with Focus on the Leap Motion Controller', *Sensors*, vol. 18, no. 7, Art. no. 7, Jul. 2018, doi: 10.3390/s18072194.

[26] 'Are joysticks just for Gaming?' https://www.rs-online.com/designspark/are-joysticks-just-for-gaming (accessed Jul. 21, 2020).

[27] H. Liu and L. Wang, 'Gesture recognition for human-robot collaboration: A review', *Int. J. Ind. Ergon.*, vol. 68, pp. 355–367, Nov. 2018, doi: 10.1016/j.ergon.2017.02.004.

[28] A. Mohamad, B. Abdulbaqi, and N. Jumaa, 'Hand Motion Controlled Robotic Arm based on Micro-Electro-Mechanical-System Sensors: Gyroscope, Accelerometer and Magnetometer', *Commun. Appl. Electron.*, vol. 7, Jul. 2017, doi: 10.5120/cae2017652621.

[29] K. Nymoen, M. R. Haugen, and A. R. Jensenius, ‘MuMYO - Evaluating and Exploring the MYO Armband for Musical Interaction’, in *Proceedings of the international conference on New Interfaces for Musical Expression*, Baton Rouge, Louisiana, USA, May 2015, pp. 215–218.

[30] K.-Y. Chen, S. N. Patel, and S. Keller, ‘Finexus: Tracking Precise Motions of Multiple Fingertips Using Magnetic Sensing’, in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, San Jose California USA, May 2016, pp. 1504–1514. doi: 10.1145/2858036.2858125.

[31] T. Guzsvinecz, V. Szucs, and C. Sik-Lanyi, ‘Suitability of the Kinect Sensor and Leap Motion Controller—A Literature Review’, *Sensors*, vol. 19, no. 5, Art. no. 5, Jan. 2019, doi: 10.3390/s19051072.

[32] A. Vit and G. Shani, ‘Comparing RGB-D Sensors for Close Range Outdoor Agricultural Phenotyping’, *Sensors*, vol. 18, no. 12, Art. no. 12, Dec. 2018, doi: 10.3390/s18124413.

[33] S. Giancola, M. Valenti, and R. Sala, ‘A Survey on 3D Cameras: Metrological Comparison of Time-of-Flight, Structured-Light and Active Stereoscopy Technologies’, 2018. doi: 10.1007/978-3-319-91761-0.

[34] H. Chandel, W. Shimla, S. Vatta, and W. Shimla, ‘Occlusion Detection and Handling: A Review’, 2015.

[35] F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler, ‘Analysis of the Accuracy and Robustness of the Leap Motion Controller’, *Sensors*, vol. 13, pp. 6380–6393, May 2013, doi: 10.3390/s130506380.

[36] ‘Structure by Occipital - Give Your iPad 3D Vision’. https://structure.io/structure-core/specs (accessed Feb. 12, 2019).

[37] ‘Project North Star - Project North Star’. https://docs.projectnorthstar.org/ (accessed Sep. 21, 2019).

[38] ‘Depth Camera D435i – Intel® RealSense™ Depth and Tracking Cameras’. https://www.intelrealsense.com/depth-camera-d435i/ (accessed Sep. 21, 2021).

[39] ‘Intel’s New RealSense D435i Camera Adds Integrated Tracking - ExtremeTech’. https://www.extremetech.com/computing/280535-intels-new-realsense-d435i-adds-imu-for-integrated-tracking (accessed Jan. 21, 2021).

[40] A. Robotics, ‘Assemtica Robotics’, *Assemtica Robotics*. https://assemtica.com/ (accessed Sep. 21, 2021).

[41] P. Cipresso, I. A. C. Giglioli, M. A. Raya, and G. Riva, ‘The Past, Present, and Future of Virtual and Augmented Reality Research: A Network and Cluster Analysis of the Literature’, *Front. Psychol.*, vol. 9, p. 2086, 2018, doi: 10.3389/fpsyg.2018.02086.

[42] R. Yung and C. Khoo-Lattimore, ‘New realities: a systematic literature review on virtual reality and augmented reality in tourism research’, *Curr. Issues Tour.*, vol. 22, no. 17, pp. 2056–2081, Oct. 2019, doi: 10.1080/13683500.2017.1417359.

[43] ‘Best VR headsets in 2020: Standalone and PC-ready picks from Oculus, HTC and more’, *Wareable*, Nov. 07, 2019. https://www.wareable.com/vr/best-vr-headsets-7749 (accessed Sep. 21, 2021).

[44] M. Bekele and E. Champion, *Redefining Mixed Reality: User-Reality-Virtuality and Virtual Heritage Perspectives*. 2019.

[45] P. Milgram and F. Kishino, ‘A Taxonomy of Mixed Reality Visual Displays’, *IEICE Trans Inf. Syst.*, vol. E77-D, no. 12, pp. 1321–1329, Dec. 1994.

[46] I. E. Sutherland, ‘A head-mounted three dimensional display’, in *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, New York, NY, USA, Dec. 1968, pp. 757–764. doi: 10.1145/1476589.1476686.

[47] J. Swearingen, ‘Virtual Reality Lets You Escape the World. Augmented Reality Improves It.’, *Intelligencer*, Jan. 21, 2019. https://nymag.com/intelligencer/2019/01/augmented-reality-vs-virtual-reality-the-future-of-tech.html (accessed Sep. 21, 2021).

[48] G. Group, ‘Augmented Reality is Trending Across the Most Unexpected Industries’, *Medium*, Sep. 14, 2018. https://arvrjourney.com/augmented-reality-is-trending-across-the-most-unexpected-industries-d82c9e20d4a9 (accessed Sep. 21, 2021).

[49] ‘Industry 4.0: Digitalisation for productivity and growth - Think Tank’. https://www.europarl.europa.eu/thinktank/en/document.html?reference=EPRS_BRI%282015%29568337 (accessed Sep. 21, 2020).

[50] S. M. Chacko and V. Kapila, ‘An Augmented Reality Interface for Human-Robot Interaction in Unconstrained Environments’, in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 3222–3228. doi: 10.1109/IROS40897.2019.8967973.

[51] ‘Robot Programming Through Augmented Trajectories in Augmented Reality’. https://ieeexplore.ieee.org/document/8593700 (accessed Sep. 21, 2021).

[52] D. Ni, A. W. W. Yew, S. Ong, and A. Nee, ‘Haptic and visual augmented reality interface for programming welding robots’, 2017, doi: 10.1007/S40436-017-0184-7.

[53] ‘“Phantom Omni” – 6 DOF master device | Delft Haptics Lab’. https://delfthapticslab.nl/device/phantom-omni/ (accessed Sep. 21, 2021).

[54] A. Pacelli, ‘HoloLens 2 FAQ’, *SphereGen*, Mar. 11, 2020. https://www.spheregen.com/commonly-asked-questions-about-hololens-2/ (accessed Sep. 21, 2021).

[55] scooley, ‘HoloLens 2 hardware’. https://docs.microsoft.com/en-us/hololens/hololens2-hardware (accessed Sep. 21, 2021).

[56] D. Ungureanu *et al.*, ‘HoloLens 2 Research Mode as a Tool for Computer Vision Research’, *ArXiv*, 2020.

[57] L. Yi-bo, K. Shao-peng, Q. Zhi-hua, and Z. Qiong, ‘Development Actuality and Application of Registration Technology in Augmented Reality’, in *2008 International Symposium on Computational Intelligence and Design*, Oct. 2008, vol. 2, pp. 69–74. doi: 10.1109/ISCID.2008.120.

[58] C. M. Andrews, A. B. Henry, I. M. Soriano, M. K. Southworth, and J. R. Silva, ‘Registration Techniques for Clinical Applications of Three-Dimensional Augmented Reality

Devices’, *IEEE J. Transl. Eng. Health Med.*, vol. 9, pp. 1–14, 2021, doi: 10.1109/JTEHM.2020.3045642.

[59] R. Radkowski, ‘HoloLens Integration into a Multi-Kinect Tracking Environment’, in *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, Oct. 2018, pp. 138–142. doi: 10.1109/ISMAR-Adjunct.2018.00052.

[60] ‘Vuforia Developer Portal |’. https://developer.vuforia.com/ (accessed Sep. 18, 2020).

[61] ‘Build new augmented reality experiences that seamlessly blend the digital and physical worlds’, *Google Developers*. https://developers.google.com/ar (accessed Sep. 26, 2020).

[62] ‘Home’, *OpenCV*. https://opencv.org/ (accessed Sep. 21, 2020).

[63] S. Blankemeyer, R. Wiemann, L. Posniak, C. Pregizer, and A. Raatz, ‘Intuitive Robot Programming Using Augmented Reality’, *Procedia CIRP*, vol. 76, pp. 155–160, Jan. 2018, doi: 10.1016/j.procir.2018.02.028.

[64] C. Melchiorri, ‘Robotic telemanipulation system: An overview on control aspects’, *Proc 7 ThIFAC Symp. Robot Control*, vol. 36, Sep. 2003, doi: 10.1016/S1474-6670(17)33365-7.

[65] J. Spranger, R. Buzatoiu, A. Polydoros, L. Nalpantidis, and E. Boukas, ‘Human-Machine Interface for Remote Training of Robot Tasks.’, in *2018 IEEE International Conference on Imaging Systems and Techniques (IST)*, Oct. 2018, pp. 1–5. doi: 10.1109/IST.2018.8577081.

[66] Y. Su, M. Ahmadi, C. Bartneck, F. Steinicke, and X. Chen, ‘Development of an Optical Tracking Based Teleoperation System with Virtual Reality’, in *2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, Jun. 2019, pp. 1606–1611. doi: 10.1109/ICIEA.2019.8833835.

[67] L. Pitonakova, M. Giuliani, A. Pipe, and A. Winfield, *Feature and Performance Comparison of the V-REP, Gazebo and ARGoS Robot Simulators*. 2018. doi: 10.1007/978-3-319-96728-8_30.

[68] M. Santos Pessoa de Melo, J. Gomes da Silva Neto, P. Jorge Lima da Silva, J. M. X. Natario Teixeira, and V. Teichrieb, ‘Analysis and Comparison of Robotics 3D Simulators’, in *2019 21st Symposium on Virtual and Augmented Reality (SVR)*, Oct. 2019, pp. 242–251. doi: 10.1109/SVR.2019.00049.

[69] ‘Motion Planning Methods for Industrial Robots • Smashing Robotics’, *Smashing Robotics*, Jul. 20, 2012. https://www.smashingrobotics.com/motion-planning-methods-for-industrial-robots/ (accessed Sep. 22, 2020).

[70] J. Ma, Y. Liu, S. Zang, and L. Wang, ‘Robot Path Planning Based on Genetic Algorithm Fused with Continuous Bezier Optimization’, *Comput. Intell. Neurosci.*, vol. 2020, pp. 1–10, Feb. 2020, doi: 10.1155/2020/9813040.

[71] L. Hilario, M. Mora, N. Montes, and A. Falco, ‘Path Planning Based on Parametric Curves’, 2018. doi: 10.5772/intechopen.72574.

[72] ‘Interpolation methods’. http://paulbourke.net/miscellaneous/interpolation/ (accessed Sep. 22, 2020).

[73] S.-M. Hashemi-Dehkordi and P. P. Valentini, 'Comparison between Bezier and Hermite cubic interpolants in elastic spline formulations', *Acta Mech.*, vol. 225, no. 6, pp. 1809–1821, Jun. 2014, doi: 10.1007/s00707-013-1020-1.

[74] F. Oumellal and A. Lamnii, 'Curve and Surface Construction Using Hermite Trigonometric Interpolant', *Math. Comput. Appl.*, vol. 26, no. 1, Art. no. 1, Mar. 2021, doi: 10.3390/mca26010011.

[75] 'Bezier Curves'. https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV0405/DUGUID/bezier_curves.html (accessed Sep. 22, 2020).

[76] Z. Xu, S. Wei, N. Wang, and X. Zhang, 'Trajectory Planning with Bezier Curve in Cartesian Space for Industrial Gluing Robot', in *Intelligent Robotics and Applications*, Cham, 2014, pp. 146–154. doi: 10.1007/978-3-319-13963-0_15.

[77] J. Ogbemhe, K. Mpofu, and N. S. Tlale, 'CONTINUOUS TRAJECTORY PLANNING FOR WELDING OF COMPLEX JOINTS USING BEZIER CURVE', *Procedia Manuf.*, vol. 33, pp. 685–692, Jan. 2019, doi: 10.1016/j.promfg.2019.04.086.

[78] 'Bézier Path Creator | Utilities Tools | Unity Asset Store'. https://assetstore.unity.com/packages/tools/utilities/b-zier-path-creator-136082 (accessed Oct. 22, 2020).

[79] C. thai Hoang, 'DEVELOPMENT OF ISOGEOMETRIC FINITE ELEMENT METHODS', 2015. doi: 10.13140/RG.2.1.3861.2647.

[80] 'B-spline Curves: Important Properties'. https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/B-spline/bspline-curve-prop.html (accessed Sep. 22, 2020).

[81] L. Fowler and J. Rogers, 'Bézier Curve Path Planning for Parafoil Terminal Guidance', *J Aerosp Inf Syst*, 2014, doi: 10.2514/1.I010124.

[82] V. N. Chougule, R. Nirgude, K. Ghag, I. R. Madane, and A. Deshpande, 'Fitting Spline Curves through Set of Unorganized Point Cloud Data', 2013. https://www.semanticscholar.org/paper/Fitting-Spline-Curves-through-Set-of-Unorganized-Chougule-Nirgude/941ce29cfe96db23df1c585fba2758806df3e23c (accessed Jun. 23, 2020).

[83] D. Vo and P. Nanakorn, 'Geometrically nonlinear multi-patch isogeometric analysis of planar curved Euler-Bernoulli beams', *Comput. Methods Appl. Mech. Eng.*, vol. 366, May 2020, doi: 10.1016/j.cma.2020.113078.

[84] S. Starke, N. Hendrich, and J. Zhang, 'Memetic Evolution for Generic Full-Body Inverse Kinematics in Robotics and Animation', *IEEE Trans. Evol. Comput.*, vol. 23, no. 3, pp. 406–420, Jun. 2019, doi: 10.1109/TEVC.2018.2867601.

[85] 'What Is Inverse Kinematics?' https://www.mathworks.com/discovery/inverse-kinematics.html (accessed Apr. 2, 2020).

[86] ATOMOCLAST, 'Forward and Inverse Kinematics, an Introduction.', *Reality Bytes*, Jun. 16, 2017. https://realitybytes.blog/2017/06/16/forward-and-inverse-kinematics-an-introduction/ (accessed Jul. 22, 2020).

[87] 'What is Heuristic Programming? - Softjourn, Inc.', *Softjourn Inc*. https://softjourn.com/insights/heuristic-programming (accessed Sep. 16, 2019).

[88] A. Aristidou and J. Lasenby, 'FABRIK: A fast, iterative solver for the Inverse Kinematics problem', *Graph. Models*, vol. 73, no. 5, pp. 243–260, Sep. 2011, doi: 10.1016/j.gmod.2011.05.003.

[89] 'Jacobian matrix | Math Wiki | Fandom'. https://math.wikia.org/wiki/Jacobian_matrix (accessed Sep. 22, 2020).

[90] P. Beeson and B. Ames, *TRAC-IK: An Open-Source Library for Improved Solving of Generic Inverse Kinematics*. 2015. doi: 10.1109/HUMANOIDS.2015.7363472.

[91] 'TRAC-IK', *TRACLabs*. https://traclabs.com/projects/trac-ik/ (accessed Sep. 22, 2021).

[92] polar-kev, 'MRTK-Unity Developer Documentation - Mixed Reality Toolkit'. https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/ (accessed Sep. 22, 2021).

[93] 'Home · siemens/ros-sharp Wiki', *GitHub*. https://github.com/siemens/ros-sharp (accessed Sep. 22, 2021).

[94] 'JSON'. https://www.json.org/json-en.html (accessed Sep. 12, 2020).

[95] Y. Xue, L. Zhao, M. Xue, and J. Fu, 'Gesture Interaction and Augmented Reality based Hand Rehabilitation Supplementary System', in *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Oct. 2018, pp. 2572–2576. doi: 10.1109/IAEAC.2018.8577476.

[96] C. Li and B. Tang, 'Research on The Application of AR Technology Based on Unity3D in Education', *J. Phys. Conf. Ser.*, vol. 1168, p. 032045, Feb. 2019, doi: 10.1088/1742-6596/1168/3/032045.

[97] V. Kumar, D. K. Sharma, and V. K. Mishra, 'Visualizing Big Data with Mixed Reality', in *2020 9th International Conference System Modeling and Advancement in Research Trends (SMART)*, Dec. 2020, pp. 85–90. doi: 10.1109/SMART50582.2020.9337072.

[98] 'Final IK | Animation Tools | Unity Asset Store'. https://assetstore.unity.com/packages/tools/animation/final-ik-14290 (accessed Jul. 22, 2019).

[99] 'HoloLens 2—Pricing and Options | Microsoft HoloLens'. https://www.microsoft.com/en-us/hololens/buy (accessed Sep. 02, 2019).

[100] Y. Park (박동윤), 'MRTK 101 / FAQ: How to use Mixed Reality Toolkit Unity for Basic Interactions (HoloLens 2…', *Medium*, Jun. 23, 2021. https://dongyoonpark.medium.com/mrtk-101-faq-how-to-use-mixed-reality-toolkit-unity-for-basic-interactions-hololens-2-88dbcb9758f9 (accessed Sep. 22, 2020).

[101] X. Liu, Y.-H. Sohn, and D.-W. Park, 'Application Development with Augmented Reality Technique using Unity 3D and Vuforia', vol. 13, no. 21, p. 4, 2018.

[102] 'Image Targets | VuforiaLibrary'. https://library.vuforia.com/features/images/image-targets.html (accessed Aug. 22, 2020).

[103] ‘ROS.org | About ROS’. https://www.ros.org/about-ros/ (accessed May. 21, 2020).

[104] ‘rviz - AWS RoboMaker’. https://docs.aws.amazon.com/robomaker/latest/dg/simulation-tools-rviz.html (accessed Aug. 22, 2019).

[105] ‘ROS Basics 1 - Nodes, Topics, Services & Actions’, *Tarang Shah - Blog*, Apr. 01, 2017. http://tarangshah.com/blog/2017-04-01/ros-basics-1-nodes-topics-services-actions/ (accessed Sep. 04, 2019).

[106] ‘The Official YAML Web Site’. https://yaml.org/ (accessed Jun. 22, 2020).

[107] ‘tf - ROS Wiki’. http://wiki.ros.org/tf (accessed Jun. 20, 2020).

[108] ‘MoveIt Tutorials — moveit_tutorials Kinetic documentation’. http://docs.ros.org/en/melodic/api/moveit_tutorials/html/index.html (accessed Sep. 22, 2020).

[109] ‘Gazebo : Tutorial : URDF in Gazebo’. https://gazebosim.org/tutorials?tut=ros_urdf (accessed Mar. 22, 2019).

[110] B. Yan, D. Shi, J. Wei, and C. Pan, ‘HiBot: A generic ROS-based robot-remote-control framework’, in *2017 2nd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, Jun. 2017, pp. 221–226. doi: 10.1109/ACIRS.2017.7986097.

[111] ‘rosbridge_suite - ROS Wiki’. http://wiki.ros.org/rosbridge_suite (accessed Sep. 12, 2020).

[112] M. Sadowski, ‘ROS web tutorial part 1 - rosbridge server and roslibjs’, *msadowski blog*. https://msadowski.github.io/ros-web-tutorial-pt1/ (accessed Sep. 26, 2019).

[113] *ABB*. ROS-Industrial, 2020. Accessed: Sep. 22, 2020. [Online]. Available: https://github.com/ros-industrial/abb

[114] Y. Niu, H. Qazi, and Y. Liang, ‘Building a Flexible Mobile Robotics Teaching Toolkit by Extending MATLAB/Simulink with ROS and Gazebo’, in *2021 7th International Conference on Mechatronics and Robotics Engineering (ICMRE)*, Feb. 2021, pp. 10–16. doi: 10.1109/ICMRE51691.2021.9384836.

[115] ‘urdf/XML/link - ROS Wiki’. http://wiki.ros.org/urdf/XML/link (accessed Sep. 22, 2020).

[116] ‘Robot Safety - Safe Move 2 Ensuring The Highest Levels Of Safety’, *RMGroup*. https://rmgroupuk.com/abb-safe-move-2/ (accessed Jul. 22, 2019).

[117] ‘Technical data for the ABB IRB 1200 industrial robot | ABB Robotics’, *Robotics*. https://new.abb.com/products/robotics/industrial-robots/irb-1200/irb-1200-data (accessed Sep. 22, 2020).

[118] ‘abb_driver - ROS Wiki’. http://wiki.ros.org/abb_driver (accessed Sep. 20, 2020).

[119] ‘IRC5 Controller | ABB Robotics’, *Robotics*. https://new.abb.com/products/robotics/controllers/irc5-overview/irc5 (accessed Oct. 22, 2020).

[120] ‘IRC5 Standard features’, *Robotics*. https://new.abb.com/products/robotics/controllers/irc5-overview/irc5/irc5-standard-features (accessed Sep. 22, 2019).

[121] ‘Industrial/supported_hardware - ROS Wiki’. http://wiki.ros.org/Industrial/supported_hardware (accessed Sep. 4, 2020).

[122] Z. Forgo, M. A. Villanueva Portela, A. Hypki, and B. Kuhlenkoetter, ‘Dual arm robot control by hands gestures using ROS’, in *ISR 2020; 52th International Symposium on Robotics*, Dec. 2020, pp. 1–6.

[123] ‘What are singularities in a six-axis robot arm?’ https://www.mecademic.com/en/what-are-singularities-in-a-six-axis-robot-arm (accessed Aug. 22, 2019).

[124] ‘What are Accuracy and Repeatability in Industrial Robots?’ https://blog.robotiq.com/what-are-accuracy-and-repeatability-in-industrial-robots (accessed Sep. 22, 2021).

[125] Stephanie, ‘Mean Absolute Percentage Error (MAPE)’, *Statistics How To*, Apr. 25, 2021. https://www.statisticshowto.com/mean-absolute-percentage-error-mape/ (accessed Apr. 28, 2021).

[126] D. A. Swanson, ‘On the Relationship among Values of the Same Summary Measure of Error when it is used across Multiple Characteristics at the Same Point in Time: An Examination of MALPE and MAPE’, p. 15, 2015.

[127] A. Owen-Hill, ‘5 Tips to Avoid Singularity Problems in Robot Welding’, *RoboDK blog*, May 07, 2019. https://robodk.com/blog/robot-welding-singularity-problems/ (accessed Jan. 5, 2020).

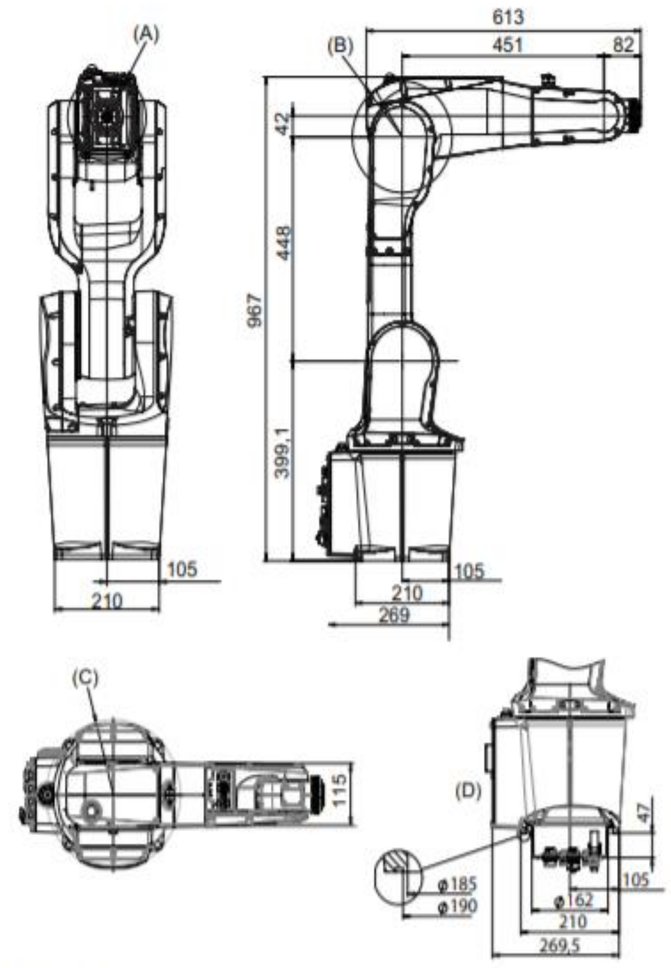
APPENDICES

A1. ABB IRB 1200 TECHNICAL SPECIFICATIONS

1 Description

1.1.2 The robot
Continued

Dimensions IRB 1200-5/0.9



xx1400000339

Pos	Description
A	Minimum turning radius axis 4 R=79 mm
B	Minimum turning radius axis 3 R=111 mm
C	Minimum turning radius axis 1 R=138 mm
D	Valid for option Robot cabling routing, 966-1 From below

A2. The code and data sets for creating this document can be found at:

https://github.com/ShamaineChung/ROS_ABB_Unity

A3. The video showcasing how the system works can be found at:

https://www.youtube.com/watch?v=7X5X7D9xEQE