

UNIVERSITY OF TARTU  
Faculty of Science and Technology  
Institute of Technology  
Robotics and Computer Engineering

Gaurav Garg

# Digital Twin for Industrial Robotics

Master's Thesis (30 ECTS)

Supervisor(s): Prof G. Anbarjafari  
Dr. Vladimir Kuts

Tartu 2021

# Digital Twin for Industrial Robotics

## **Abstract:**

This thesis aims to develop a Digital Twin model for robot programming that incorporates Virtual Reality (VR). The digital twin is a concept of creating a digital replica of a physical object (such as a robot), which is similar to establishing a model simulation along with some additional functionalities. Simulation software, such as robot operating system (ROS) or other industrial-owned simulation platforms, simulates a robot operation and sends the details to the robot controller. In contrast to this, the Digital Twin model establishes a two-way communication channel between the digital and the physical models. In this thesis, the proposed Digital Twin model can help in online/remote programming of a robotic cell with high accuracy by creating a 3D digital environment of a real-world physical setup, which is similar to watching a movie with 3D glasses. To create a Digital Twin model, the gaming platform is used that comes with specialized plugins for virtual and augmented reality devices. One of the main challenges in any robotic system is writing a code for a defined path and modifying it for future requirements. Programming robot via this traditional approach requires a lot of time and often disturbs the running process. Whereas, in the case of a Digital Twin model, the program can be adjusted or regenerated without disturbing the execution cycle of a physical robot.

In this work, I have considered the FANUC robot model M-10iA/12 and created a 3D virtual environment using a gaming engine (Unity), that describes a complete Digital Twin model. After establishing a Digital Twin, we can see the motion replica in a physical robot with benefits of VR environment, where we can see more precise details of the trajectory w.r.t to the working components. The simulation analysis gives the latency of approximately 40ms (milliseconds) with a mean error of joint movements is around  $0.28^\circ$ . Based on the results, it can be concluded that the model developed under this thesis is suitable for industrial applications.

## **Keywords:**

Digital Twin, FANUC, Robot, KAREL language, Unity, Blender, Virtual Reality, Tool Center Point (TCP)

## **CERCS:**

T120 System Technology, Computer Technology, T125 Automation, robotics, control technology, T130 Production Technology

## **Digitaalne kaksik tööstusrobotiks**

### **Lühikokkuvõte:**

Käesoleva lõputöö eesmärk on arendada robotite programmeerimiseks virtuaalreaalsust (VR) hõlmav digitaalse kaksiku (ingl Digital Twin) mudel. Digitaalne kaksik on mõiste, mis tähistab füüsilise objekti (nagu näiteks roboti) digitaalse koopia loomist, sarnanedes mudelipõhisele simulatsioonile koos mõningate lisafunktsionaalsustega. Simulatsioonitarkvarad nagu näiteks Robot Operating System (ROS) või muud tööstuslikud simulatsiooniplatvormid simuleerivad roboti tööd ning saadavad detailsed andmed roboti kontrollerile. Seevastu digitaalse kaksiku mudeli puhul luuakse kahe-suunaline suhtluskanal digitaalse ja füüsilise mudeli vahel. Selles lõputöös välja pakutud digitaalse kaksiku mudel võimaldab üle võrgu robotraku programmeerimist kõrge täpsusega, luues selleks 3D digitaalse keskkonna päris-elu füüsilisest keskkonnast, sarnanedes 3D prillidega filmi vaatamisele. Digitaalse kaksiku mudeli loomiseks kasutati mänguarendusplatvormi, mis hõlmas spetsiaalseid pistikprogramme virtuaal- ja täiendreaalsusseadmete jaoks. Üks põhilisi väljakutseid robotsüsteemides on lähtekoodi kirjutamine etteantud trajektoori jaoks ning selle modifitseerimine tuleviku nõuete täitmiseks. Sellisel traditsioonilisel moel programmeerimine on ajakuluks ning tekitab katkestusi protsessi töös. Digitaalse kaksiku mudeli puhul on aga võimalik programmi muuta või taasluua ilma füüsilise roboti käitustsükli häirimata.

Töös kasutasin FANUC roboti mudelit M-10iA/12 ning Unity mängumootoriga loin 3D virtuaalse keskkonna, mis kirjeldab terviklikku digitaalse kaksiku mudelit. Peale digitaalse kaksiku loomist saame näha liikumise replitseerimist füüsilisel robotil, sealjuures lisanduvad VR keskkonna eelised, kus saame näha liikuvate komponentide trajektoori detailsemaid andmeid. Simulatsiooni analüüs näitas, et latentsusaeg on ligikaudu 40 millisekundit ning keskmine liigeste hälve on  $0.28^\circ$ . Tulemustest võib järeldada, et dissertatsiooni raames välja töötatud mudel on kasutatav tööstuslikes rakendustes.

### **Võtmesõnad:**

Digitaalne kaksik, FANUC, robot, KAREL-keel, Unity, Blender, virtuaalne reaalsus, tööriista keskpunkt (TCP)

### **CERCS:**

T120 System Technology, Computer Technology, T125 Automation, robotics, control technology, T130 Production Technology

# Contents

<b>Glossary</b>	<b>8</b>
<b>1 Introduction</b>	<b>9</b>
1.1 Thesis Highlights . . . . .	10
1.2 Contribution . . . . .	10
1.3 Thesis Organization . . . . .	10
<b>2 Related Work</b>	<b>12</b>
2.1 Digital Twin Model . . . . .	12
2.1.1 Digital Twin and Virtual Reality for Multi-Robot Manufacturing Cell Commissioning . . . . .	12
2.1.2 Knowledge Driven Digital Twin Manufacturing Cell . . . . .	13
2.2 Virtual Reality (VR) . . . . .	14
<b>3 Model Description</b>	<b>16</b>
3.1 Robot . . . . .	16
3.2 VR Controller . . . . .	16
<b>4 Creation of Digital Model</b>	<b>18</b>
4.1 3D Model Rigging . . . . .	18
4.2 Import 3D model in Unity . . . . .	20
4.2.1 Defining Tool Center Point (TCP) . . . . .	21
4.3 Inverse Kinematic (IK) for Digital Model . . . . .	22
4.3.1 Unity Coordinate System . . . . .	23
4.3.2 Tranform Matrix between Unity and Physical Robot . . . . .	25
4.4 Communication Channel . . . . .	26
4.4.1 Source Code . . . . .	27
<b>5 User Interface (UI)</b>	<b>28</b>
5.1 Virtual Reality (VR) Interface . . . . .	28
5.1.1 Setting VR Environemnt in Unity . . . . .	28
5.2 Physical Device Interface via Teach Pendent . . . . .	29
<b>6 Results and Analysis</b>	<b>33</b>
6.1 Joint Movement Analysis . . . . .	33
<b>7 Discussion</b>	<b>37</b>
7.1 Generalized Solution For FANUC Robots . . . . .	37
7.2 Use Case Scenarios . . . . .	37
7.2.1 Trajectory Planning For Complex Edges . . . . .	37

7.2.2	Program Creation In Space Constrained Surroundings . . . . .	38
7.2.3	Training robotic operations via Machine learning (ML) . . . . .	38
7.2.4	Predictive Maintenance & Error Detection . . . . .	39
<b>8</b>	<b>Conclusion</b>	<b>40</b>
	<b>Acknowledgement</b>	<b>42</b>
	<b>References</b>	<b>46</b>
	<b>Licence</b>	<b>47</b>

## List of Figures

1	Multi-Robot Manufacturing Cell . . . . .	13
2	Knowledge-drive Robot Simulation model . . . . .	14
3	Medical Application with digital twin . . . . .	15
4	M-10iA/12 Robot Specification . . . . .	16
5	3D Model in Blender Software . . . . .	19
6	Blender Rigging . . . . .	20
7	Pivot Point Creation . . . . .	21
8	Robot Setup and its environment . . . . .	22
9	Tool Center Point . . . . .	23
10	Inverse Kinematic Packages . . . . .	24
11	Unity Coordinates System . . . . .	25
12	Coordinates System . . . . .	25
13	Communication Setup . . . . .	27
14	Controlling Twin model via Virtual reality controller . . . . .	29
15	Control cycle between Digital and Physical model . . . . .	30
16	Robot Status Data . . . . .	31
17	Online/Offline Mode . . . . .	31
18	Robot Trajectory (axis values in mm) . . . . .	33
19	Range of Movement of Individual Joints . . . . .	34
20	Difference Between Commanded (IK) and Executed (FK) Joint Angles	34
21	Joint Angle movement for a random trajectory . . . . .	35
22	Difference Between Commanded (IK) and Executed (FK) Joint Angles	35
23	VR Environment for programming an Engine component . . . . .	38
24	Robotic Operation in constrained space . . . . .	39

## List of Tables

1	Glossary . . . . .	8
2	VR Comparison with other Softwares. . . . .	13
3	HTC Vive Specifications . . . . .	17

## Glossary

Abbreviation/Word	Meaning
ROS	Robot Operating System
VR	Virtual Reality
AR	Augmented Reality
MR	Mixed Reality
w.r.t.	with respect to
ms	milliseconds
AI	Artificial Intelligence
fbx	filebox 3D file format
dae	collado file format
UI	User Interface
KDTMC	Knowledge-driven digital twin manufacturing cell
RMS	Root Mean Square
KMs	Kilometers
Kg	Kilograms
HMD	Head-Mounted Device
FOV	Field of View
SDK	software development kit
STL	standard triangular language
AR	Augmented Reality
MR	Mixed Reality
TCP	Tool Centre Point
IK	Inverse Kinematic
FK	Forward Kinematic
UOP	User Operator Panel
Unity	Gaming Engine
Roboguide	FANUC owned robot simulation software, it supports only FANUC robots simulation
KAREL	Programming language based on PASCAL

Table 1. Glossary



# 1 Introduction

Game Engines are proven to be excellent in terms of graphics displays [1]. The level of details that can be shown or designed on gaming platforms is one of the missing components from existing industrial simulation software present in the market. Enhanced graphics is one of the key features of game engines. In addition to their enhanced graphics, these game engines provide an excellent environment for generating synthetic data. For example, gathering data for an artificial intelligence (AI) model for pick and place applications under different illumination intensities is a time-consuming and challenging task in a physical environment. However, on a gaming engine, all it takes is an understanding of ray-light interaction, and a large amount of training data for the AI model can be generated in a short period of time.

On the other hand, nearly all industrial simulation software is based on an offline simulation, where we set up the simulation environment and generate a program, which can be transferred to a physical system. However, the programs developed through offline simulation environments lack accuracy, which results in time-consuming rework overhead on physical models. To overcome this limitation, the digital twin comes with the Online Simulation, allowing one to connect with the robot and move it along the desired path on the simulation screen and actual model. However, one of the challenging tasks of these digital twin models is to create and connect them with the actual model, as it requires a well-defined digital 3D structure. Generally, for industrial purposes, CAD software for 3d modeling is used. However, at present, CAD software does not support file formats such as filmbox (.fbx), collado (.dae) that is required for gaming engines. This requires additional 3D modeling software for rendering and converting the models in suitable formats.

In this thesis, I have worked on creating a digital setup of a physical robot. The developed digital model has the capability to generate the program for the actual robot setup. One can argue that there are already many simulation software present in the industry that can help to create a program, but it comes with an additional cost or is limited to a single robot type. To overcome these limitations, gaming engines for industrial applications have been utilized, which comes with prepacked packages for integration of Virtual/Augmented and mixed reality devices from different manufacturers that are missing in the available simulation software.

Robot Operating System (ROS) is one of the powerful open-source simulation platforms. However, it is missing the components of actual deployment or for production-ready solutions. Due to this, ROS2 is being introduced, which explicitly considers the development of a real-world production environment. The main difference between ROS and ROS2 is the internal architecture[2], but the simulation tools/environment like a gazebo, rviz remains the same. However, integrating the packages with other simulation tools such as Unity can solve today's demand for high precision simulations. The unity platform has the capability to handle high-resolution components; thus, training the

robotic system inside Unity environment along with some physics and motion controlling packages from ROS like MoveIt, mapping makes a complete solution.

## 1.1 Thesis Highlights

In this thesis, I have established a Digital Twin setup for an industrial robot (FANUC) and covers the following:

1. **Digital Twin model:** Developed a digital 3D model for gaming engine (Unity) that generates a program for the physical robot. In particular, I have used FANUC robot model M-10iA/12 and tested the program on the physical robot (Section 4).
2. **Communication Setup :** Established a communication channel between Digital and Physical robot for exchanging required data such as joint angles and actual robot status signals (Section 4.4).
3. **Virtual reality Interface :** Created a virtual reality environment on the Unity platform. For visualizing the VR environment, I have used the HTC VIVE VR device (Section 5.1).

## 1.2 Contribution

The contribution of this thesis work is three-fold:

1. The key contribution of this thesis is that I have developed a Digital Twin model of an industrial robot, which helps in controlling a physical robot in synchronization with a digital model for a desired robot trajectory which is usually done either offline or trajectory points are stored in robot memory via real-time communication setup [3, 4, 5].
2. Next, during the survey, participants found that the created virtual environment is interactive and easy to use for robotic operations. On average, participants rated three on a scale of 0-10, where 0 means very easy, and 10 represents highly difficult.
3. Finally, while previous research [6, 7] has worked on robots that support the same environment as the gaming engine, here communication across different platforms (Dot Net to KAREL) is developed.

## 1.3 Thesis Organization

The rest of the thesis is as follows. Next, we discuss the related work. Then the hardware description of the model is explained in Section 3. Section 4 presents the digital twin

model creation and its user interface (UI) is discussed in Section 5. The results and analysis of the experimentation are projected in Section 6. In Section 7, I have discussed the generalization of the developed solution, and some use case scenarios. Finally, conclusion with a discussion of future directions is covered in Section 8.

## 2 Related Work

For the last two decades, researchers have been working on the development of digital twin models. However, it has recently attracted a lot of attention, and work is being done in a variety of research industries ranging from medical to industrial manufacturing because, unlike other simulation environments, it provides two-way communication between physical and digital robot model. In this section, we discuss relevant literature with respect to digital twin models, which revolve around the intersection of digital twin and virtual reality for multi-robot manufacturing. We broadly divide the related works into the following: (1) Digital Twin model development and (2) Digital Twin Applications.

### 2.1 Digital Twin Model

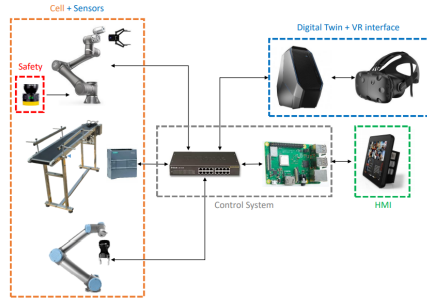
Digital Twin maps a physical object or process with digital environment that enables simulation, prediction and optimization [5, 8] and helps in product designing and simulating manufacturing systems [9, 10, 11] and processes [12].

Robotic applications [13], are the specific area of digital twin domain. Over the past few years many researchers are being conducted to utilize the capability of digital twin for various robotic tasks [7, 14, 15, 16], in issues of human-robot collaboration [17], or in issues related to predictive maintenance [18]. Industry 4.0 [19, 20] is highly used with robotics to generate a flexible solution with an ability to adapt and reconfigure systems. Additional to this, it makes a system intractable with its environment through extensive sensory systems.

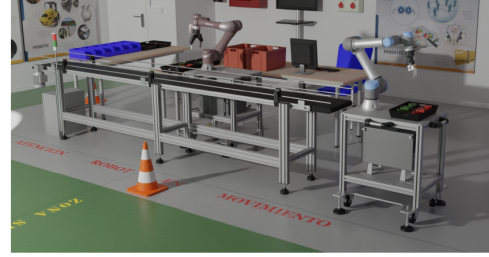
#### 2.1.1 Digital Twin and Virtual Reality for Multi-Robot Manufacturing Cell Commissioning

The research group from the University of Oviedo and the Idonial center of Technology has created a digital twin of a manufacturing cell [21]. The cell consists of two robots, a conveyor belt coupled with HTC Vive via control system as shown in Figure 1a. The digital environment, which is designed in the Unity Engine platform, is a replication of the physical setup as shown in Figure 1b. The simulation environment is designed by capturing 3D cloud data of the actual environment and parsing it through blender software. The task of the robotic cell is to detect the missing component on a tray, which is placed by an operator, and complete it. The training and testing are accomplished in a digital environment. Here the task of filling the tray is done in a virtual environment via an HTC Vive controller.

The effectiveness of Digital twin-based virtual reality is compared with simulation tools from robot manufacturers and commercial simulation software with virtual reality integration, shown in Table 2[21]. Comparison is made on various parameters such



(a) Digital Twin Design Approach



(b) Simulation Environment

Figure 1. Multi-Robot Manufacturing Cell

	Robot Manufacturers	Commercial Sim. Tools with VR	DT Based on VR
Low investment	2	1	3
Multi-robot	1	3	3
Human-robot collab.	1	1	3
Immersive	1	3	3
Customization	1	2	3
Training	1	2	3
Versatility	1	2	3

Table 2. VR Comparison with other Softwares.

as acquisition costs (labeled as *Low investment* in the table), robots from different manufacturers (*Multi-robot*), human-robot collaboration (*Human-robot collab*), virtual reality (*Immersive*), environment customization (*Customization*), usability for training (*Training*), and versatility to include new functionalities (*Versatility*). The scale is divided from 1–3, where “1” indicates the worst or not supported, and “3” means the best.

### 2.1.2 Knowledge Driven Digital Twin Manufacturing Cell

A digital twin is not limited to the creation of a digital model and perceiving online control. Research is moving towards the creation of intelligent and resilient systems. The study shows an implementation of a knowledge-driven digital twin manufacturing cell (KDTMC) [22]. KDTMC approach targets a smart manufacturing system that supports autonomous manufacturing by integrating perception, simulation, prediction, optimization, and controlling strategy.

Figure 2 [22], shows a digital twin setup for KDTMC, here a communication channel is established to collect data from the physical robot and store it to optimize the path

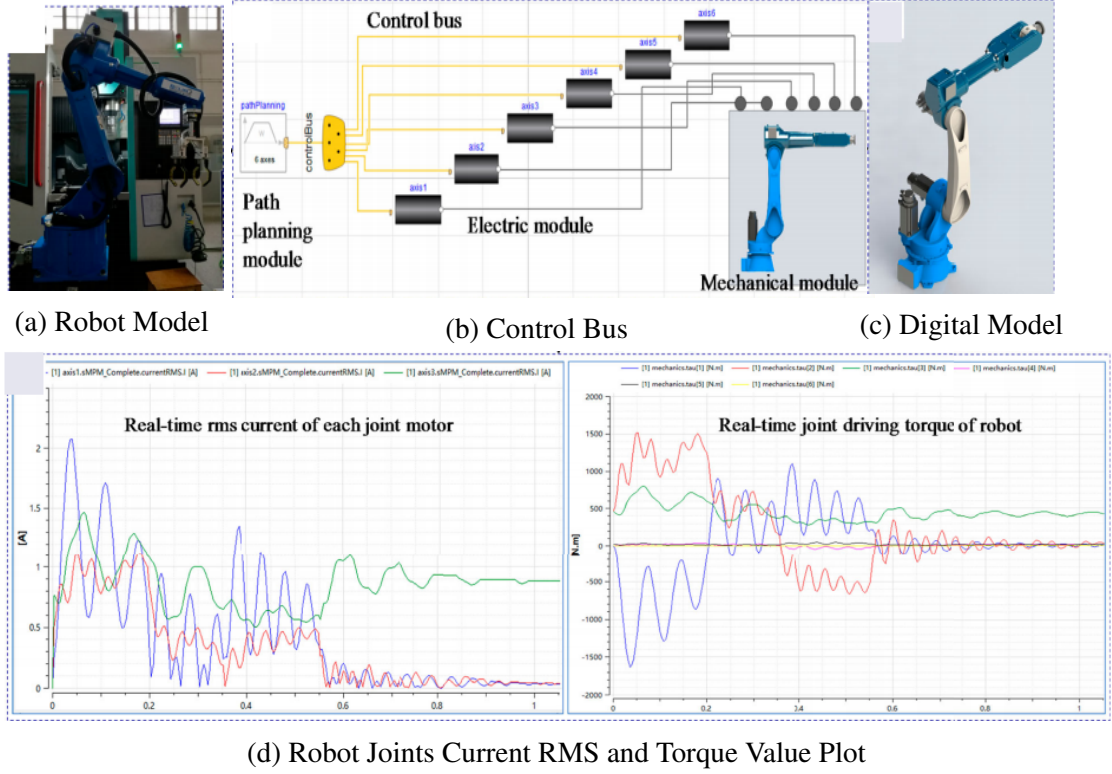


Figure 2. Knowledge-drive Robot Simulation model

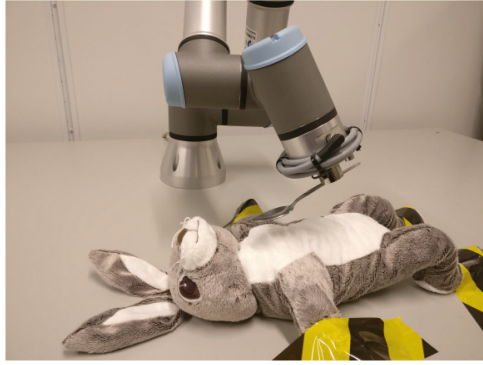
based on the torque and root mean square (RMS) of the current values of individual robot arm joints. Figure 2a shows an actual robot, Figure 2b shows control bus setup along with communication established between actual and digital model, Figure 2c explains the Digital model of robot, and Figure 2d indicates the root mean square (RMS) value of current at each joint angle along with torque values for a particular trajectory. This data can be used to optimize the planning path algorithm, predictive maintenance, or calculation of life expectancy on the basis of running conditions [23].

## 2.2 Virtual Reality (VR)

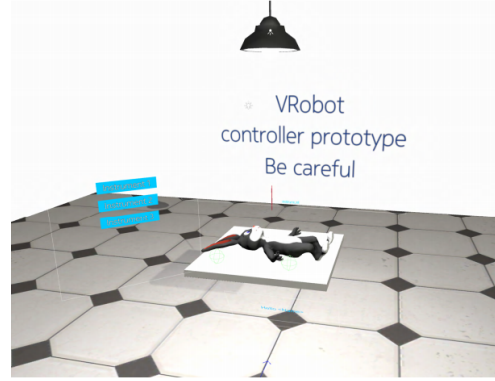
Virtual Reality (VR) is a major component of the Digital Twin setup. In a virtual environment, the feedback from the physical system is captured so that the user gets a sense of immersion while operating in a virtual setup [24]. VR has found its application in extensive sectors such as flight simulator [25] and surgical simulators [26]. With the help of VR setup, the workforce such as pilots, doctors can be pre-trained before using the actual machines and setup.

**Medical sector application:** In [27], the researchers utilize the digital twin of the

Universal robot for remote surgery. Figure 3 shows the setup for experimentation of remote surgery using digital twin via VR interface [27]. A distance of approximately 10KMs separated the physical robot and digital twin. The communication between physical and digital setup was established over 4G network. The time lapse or delay reported for the commanded movement was in the range of 50ms-150ms.



(a) Actual Remote Surgery Setup



(b) Digital Twin for Remote Surgery

Figure 3. Medical Application with digital twin

In this thesis, I control a physical robot via a digital model that operates in the virtual environment. There are certain robot status signals that are being captured and are necessary for any twin model, which are not mentioned in the existing literature. Additionally, this work has elaborated more on the controlling algorithms used in the physical and digital setup, compares the joint movement, and visualizes the errors between guided and executed joint angles.

### 3 Model Description

Here, experimental setup is explained that includes two components. First component is a Robot. In this experimentation, I have used FANUC robot. Second main component of this setup is the VR controller, for which HTC Vive VR device has been used.

#### 3.1 Robot

In this work, the FANUC robot model M-10iA/12 is used for experimentation. It is a hollow arm design that provides space for cable integration inside robot arm links rather than outside [28]. Such design protects the cables from wear and tear, making maintenance more manageable and cuts down maintenance costs. Another benefit of the hollow arm is the high productivity and shortened cycle times due to the rigid arm and servo motors, by allowing increasing speed during operation. The specifications and other information related to robot joint speed and robot motion range are shown in Figure 4 [28].

Robot Information		
Robot Specifications	Robot Motion Speed	Robot Motion Range
Axes:.....6	J1.....230°/s (4.01 rad/s)	J1.....+340° - 360°
Payload:.....12 kg	J2.....225°/s (3.93 rad/s)	J2.....±250°
H-Reach:.....1420 mm	J3.....230°/s (4.01 rad/s)	J3.....±447°
Repeatability: ±0.08 mm	J4.....430°/s (7.5 rad/s)	J4.....±380°
Robot Mass: 130 kg	J5.....430°/s (7.5 rad/s)	J5.....±380°
Structure: .....Articulated	J6.....630°/s (11 rad/s)	J6.....±720°
Mounting: .....Floor, Inverted, Angle		

Figure 4. M-10iA/12 Robot Specification

Figure 4 column-1 shows the robot specifications like the number of axes is 6, the mass of the robot body being 130Kg, its lifting capacity is 12Kg, etc. The lifting capacity is a sum of (end effector/gripper weight+ component weight). The end effector/gripper is an attachment to the 6th axes of a robot, and its design depends on the type of application a robot has to handle. And the component weight is the object weight that the robot needs to lift using its end-effector/gripper. Individual joints speed limits in mm/s and rotation limits in degree are also defined in Figure 4 column-2 and 3 respectively.

#### 3.2 VR Controller

The next main component of this experimental setup is a VR controller. In particular, we are utilizing the HTC Vive VR device [29], due to its availability in the lab. However,



any VR model can be used for a virtual environment projection. The two main units of any VR device are:

1. **A central Head-Mounted Device (HMD) unit equipped with a camera:** This unit helps in immersing a digital view into the real world. It acts as an eye in a virtual environment and helps to visualize the virtual scenes.
2. **Handheld controllers:** To perform any action in VR world, controllers act as a human hand. The actions defined in SDK of vive are defined similarly to our daily routine work with human hands such as grab, hold, button push, etc.

Display refresh rate	90HZ
Resolution each eye	1080x1200
Field of View (FOV)	110°
HMD refresh rate	225HZ
Controller Pose refresh rate	250HZ

Table 3. HTC Vive Specifications

Table 3 shows the specifications of the VR device [30]. The "Field of View (FOV)" defines the range covered with cameras that are mounted in HMD. In this device, single camera is embedded in HMD. However, in the newer version, two cameras are used to increase the FOV. HMD refresh rate defines the virtual view capturing rate and the "pose refresh rate" defines the rate of capturing pose actions performed in virtual environment using handheld controller which is 4 ms (milliseconds). To create a specific actions or to interact with objects in VR environment, SteamVR is used. SteamVR is an interface which comes along with its software development kit (SDK) that provides an extensive library to assign and read controllers actions, such as grabbing components, hitting an object, teleporting and many other human body actions can be implemented by using this SDK inside the VR environment [31].

## 4 Creation of Digital Model

In the last section, an introduction to the physical/hardware components (robot and VR controller) required for creating the digital twin model is provided. I also mentioned the specifications of these components in detail. In this section, a list of software components is defined that can be used for digital 3D model creation along with a communication channel set up between a digital model and the physical robot that will be used to exchange the required data. In this section, I will start by explaining the procedure of creating a digital model along with the coding part for establishing the communication pathway.

To create an operational digital model in the Unity platform, some details have to be added to the CAD model, such as rigging structure, pivot point, and child-parent relation between links. Tools or software used for creating digital model are:

1. **SolidWorks (2019):** SolidWorks is a 3D modeling software used widely in the industry for creating drawings of machines and components. Most of the industrial simulation software supports SolidWorks file formats such as STL, AMF, SAT, STEP, and many more. In this experimentation, the STL format is considered as an output to get highly detailed model. STL or standard triangular language is a file format supported by many animation character drawing software like Blender, 3D Maya. It represents the model as a combination of triangles and maintains the edges and curves of a 3D drawing.
2. **Blender (version:2.92.0):** Blender is an open-source animation character modeling software. There are other similar software, but I am considering this for conversion from the 3D CAD model to 3D animation-ready model as I have some experience in handling blender tools.
3. **Unity Game Engine (version:2021.1.1f1):** It provides a vast amount of packages and assets to add animation to 3D animation models which are created in Blender and integrate VR (Virtual Reality)/AR (Augmented Reality)/MR (Mixed Reality) interfaces. Another widely used gaming engine is Unreal Engine which is based on C++. Due to my experience in C#, I preferred the Unity platform.

### 4.1 3D Model Rigging

The first step towards creating an animation model or making it movable in a game engine is rigging. Rigging is a process of adding bones to a 3D CAD model, just like a human body [32]. But before adding the bones to a model, a relationship between joints and links has to be defined. Like, if I rotate my waist, do I want to turn my neck along with it or not. This characteristic is added by establishing the parent-child relationship between links and links.

In this experimentation setup, a FANUC Robot model M-10iA/12 is used (already discussed in Section 3.1). The CAD file of the robot is exported from SolidWorks in stl format, which is accepted by blender software. In blender software, the *Parent-Child* relationship is defined, where the *Base* is a parent, and consecutive links are the child to it. Figure 5a shows a 3D model in the blender and all the six-axis/links (Axis-1, Axis-2, Axis-3, Axis-4, Axis-5, and Axis-6) are marked. In Figure 5a, "Base" is also marked, which is a parent to all the links. "Axis-1" is a child to "Base" and parent to "Axis-2," and a similar parent-child relationship exists among other axis/links. This parent-child relationship is established in blender software. A tree structure of parent-child relationships among links is shown in Figure 5b. The concept behind establishing this relationship is to control the movement. For example, if we rotate "Axis-1", then rest all links from "Axis-2" to "Axis-6" will move along with it, as "Axis-1" is defined as a parent to "Axis-2" to "Axis-6", but it won't affect the movement of the Base. However, if we move Base, the entire model will move along with it.

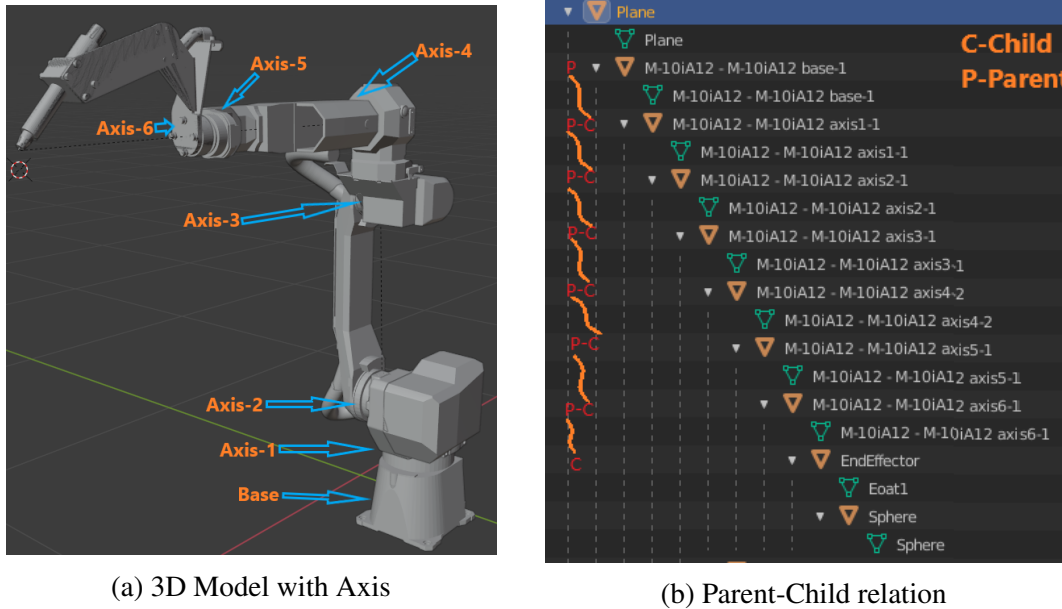
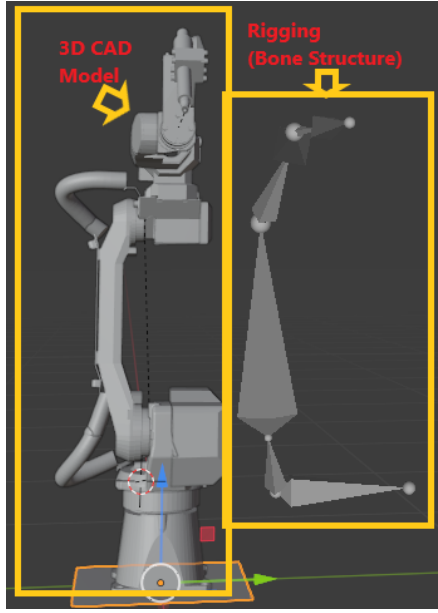


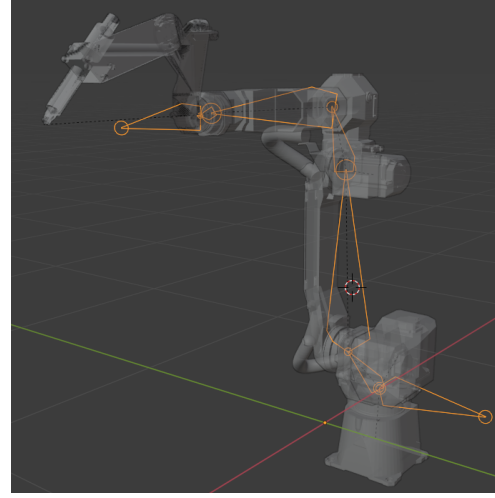
Figure 5. 3D Model in Blender Software

After establishing the relationship between links, a rigging structure is attached to the 3D model. By following the joints along a model, a rig structure is created that consist of bones linking to each other at angles specified by joint arrangement in the 3D model. The rig/bone structure created for the robot CAD model is shown in Figure 6a. Figure 6b shows the integration of rig model with 3D model. Now, the animation is done on rigged model and actual links of robot will move along with it.

In addition to the information mentioned above, the pivot point of each link must be



(a) Rigged Bone Structure



(b) 3D Rigged Model

Figure 6. Blender Rigging

specified. A pivot point, also known as a joint central point, is a point around which the motion of a link is defined [33]. Thus, the parent-child relation specifies the movement of the links, and the pivot point defines the joint movement, forming the complete model. Finally, the defined model is exported from the blender software in fbx (film box) format.

By default the links are pivoted w.r.t its center of origin of an individual link. This results in wrong movement as the movements are programmed for this point. So, the pivot point position is corrected using Blender as shown in Figure 7, for all six joints of robot arm.

## 4.2 Import 3D model in Unity

Once the model is exported from blender software in fbx format, it is ready for animation and scripting inside Unity platform. After importing the model in the Unity platform, I added some additional components to the robot environment such as floor, robot bench etc. From simulation perspective, this environment should replicate the actual industrial setup considering the space around operational component such as bench or mounting table, walls/fences, etc. The programming accuracy is highly depends on this additional details of height of robot table, and the height and distance of component with respect to a robot.

Figure 8a shows an unity environment setup. It also shows a setup of robot with

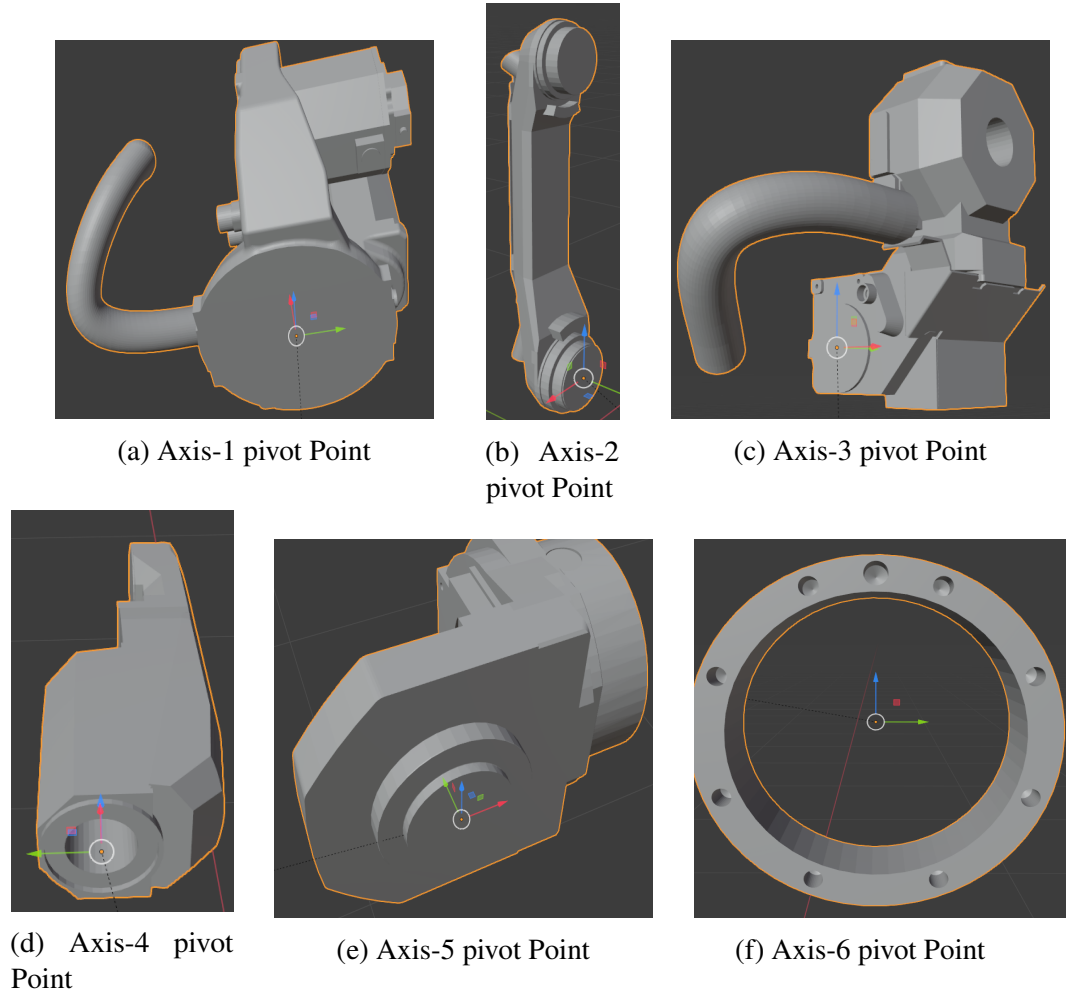


Figure 7. Pivot Point Creation

respect to its operating table (component bench). Similar setup is shown in a Figure 8b, which reflects an actual robotic cell environment.

#### 4.2.1 Defining Tool Center Point (TCP)

The robot model is associated with certain frames of reference in a 3D space such as World Frame and tool Center frame known as tool center point (TCP) [34]. The robot movement is defined with respect to its TCP, which is typically located at the center of the end-effector in most applications. The end-effector is an operational tool attached to the robot structure and performs a desired task such as grabbing a component, welding or some other application specified tasks. The end-effector changes as per the application. In this experimentation, I have considered a welding end-effector and a TCP, which is

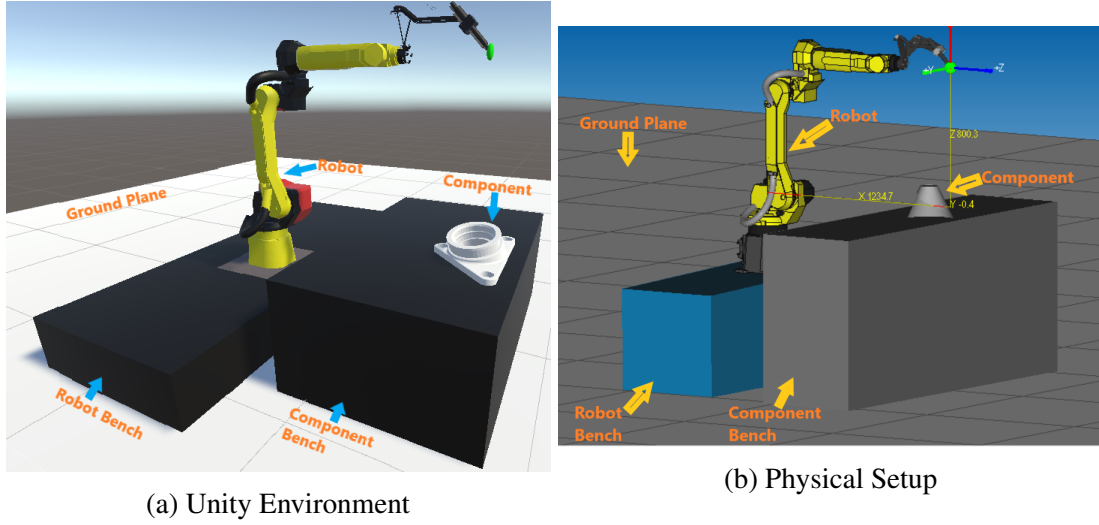


Figure 8. Robot Setup and its environment

defined at the tip of end-effector as shown in Figure 9b.

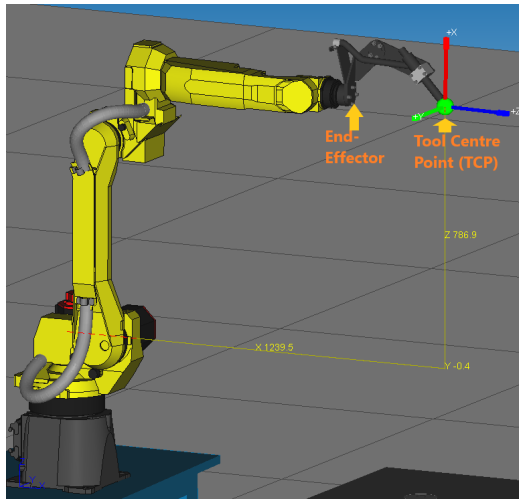
Additional to this, in some cases a User Frame is defined for components. The commanded coordinates for the movement of the end effector joint can be either defined relative to the TCP or user frame. The User frame is a frame attached to the working area of the component, where components are placed for robotic operations. In this case, I have tested the robotic movement with respect to its TCP. TCP of the digital model should be as close as possible to the TCP of the physical model, the closer the value, the higher the accuracy between them. Figure 9a shows a TCP of Physical robot and 9b reflects digital model TCP.

Although, TCP is an imaginary point that is defined as per the application and end-effector design. In this case, since the target is to control and program in the digital model, I have attached a component in green color shown in 9b for visualization purposes.

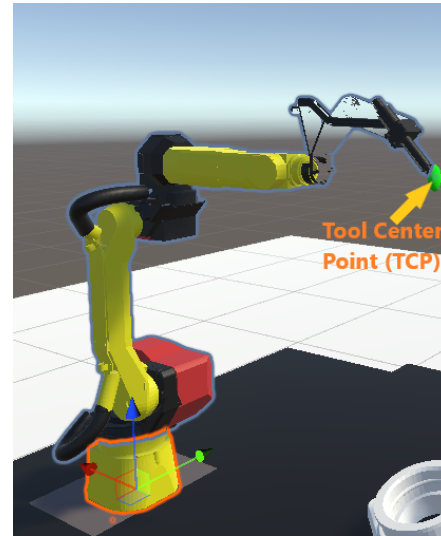
### 4.3 Inverse Kinematic (IK) for Digital Model

Inverse Kinematic is one of the major research fields for industrial applications [35]. With the advancement in machine learning, researchers are trying to create inverse kinematics with high accuracy [36]. For any mechanical structure, starting from a 2-link to n-link robot arm, an algorithm is required to calculate its joint angles to reach a specific point in 3D space, which is done using IK.

As already discussed in Section 3.1, in this experimentation, I have used a 6-link robot arm. To calculate its joint angles, I have used BioIK, a part of the Unity package manager. Starting from the Unity release 2020.3.3f1, it has provided many packages for solving inverse kinematics. The selection of algorithm depends on the number of links



(a) Robot Tool Center Point



(b) Robot Tool Center Point

Figure 9. Tool Center Point

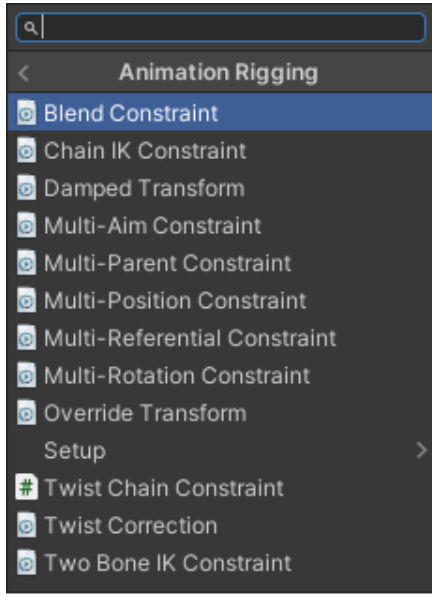
associated with mechanical structure as shown in Figure 10a.

Different solutions or packages from Figure 10a can be added to a single model, such as "Chain IK Constrained" combined with "Two Bone IK Constrained" to create a single solution. By combining different packages, we can get joint angle data via "Chain IK Constrained" with joint movement limitations via "Two Bone IK Constrained", as described in Figure 4, Column 3, where individual limits are defined for a robot. However, these Unity packages don't produce accurate results for industrial applications and some positions/coordinates in digital 3D space, it violates the joint angle limitations and results in error/accident in the Physical robot.

BioIK is a more dynamic single solution package, which can be plugged to any model. Figure 10b shows the parameter settings for a model definition. The main parameters for this model are velocity and acceleration that may varies for specific robot model. In this case, I have refereed Figure 4, column 2 (shown in Section 3.1), this shows individual joint speed in radians per second (rad/s). In this case, I have considered an average of 5 rad/s and it generates a data considering this value for all joints

#### 4.3.1 Unity Coordinate System

In order to control a physical system by capturing data from digital environment, it is important to understand the data representation system of Unity. There are different functions in Unity library, to read position and rotation of a component. These functions are as follows:



(a) Standard Unity Packages for IK



(b) BioIK Parameters

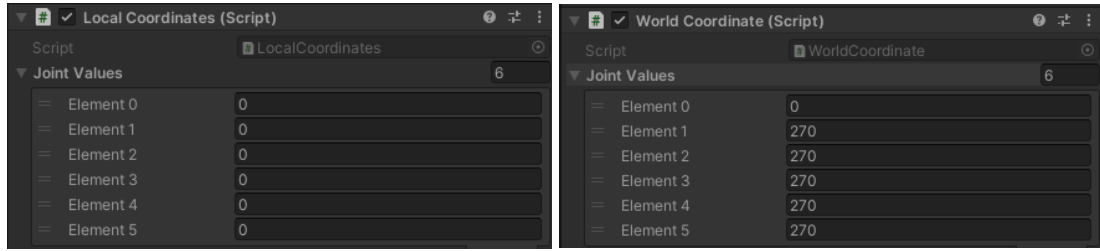
Figure 10. Inverse Kinematic Packages

1. World and Local Euler's
2. World and Local Quaternion

Before we dive into Euler and Quaternion coordinates, let's first understand what does world and Local coordinate system means in context of the Unity environment. Whenever we add a component in the Unity environment, two types of values are associated with it: (1) **World Coordinate system:** All the components inside the gaming engine can be located with a world frame. But when we try to get details of the Parent-child component as defined in Figure 5b, we consider the local coordinate system. (2) **Local coordinate system:** The local coordinate system gives value relative to another component rather than with respect to the world frame. At initial step when there is no motion, all the angles should be at  $0^\circ$  as shown in Figure 11a, which shows local coordinate system values for all joints of robot from J1-J6. On the other hand, if we look at Figure 11b without any movement, it shows  $270^\circ$  as a joint value with respect to the world coordinate system. However, when we use world values for controlling physical robot, it leads to wrong motion path and some values goes beyond the range of robot specifications as indicated in Figure 4, column-3.

Now, we see into difference between the Quaternion and Euler coordinate system, as most 3D mathematics uses Quaternion system for the calculation of movement positions and angles. To describe the rotation of a component it uses four variables X, Y, Z and





(a) Local Euler Values

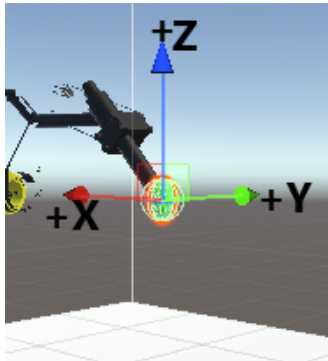
(b) Global Euler Values

Figure 11. Unity Coordinates System

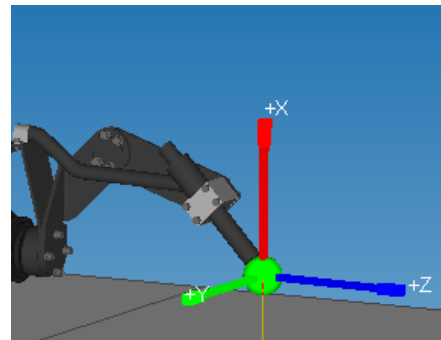
W. Whereas Eulers system only uses  $X$ ,  $Y$  and  $Z$  variables. Euler system is easy to understand, therefore, unity shows data in Euler system but internally uses Quaternion system.

#### 4.3.2 Transform Matrix between Unity and Physical Robot

As, we got a gist of different coordinate system in the previous section. Let us get into the type of data that is exchanged between physical model and the digital model. The model in Unity and the physical robot may have a different coordinate system, but Unity uses a left-handed coordinate system ( $Y$  points upward,  $Z$  points to the right, and  $X$  toward the viewer with positive rotation in a clockwise direction). However, the FANUC robot follows differnt orientation  $Z$  points Upward,  $Y$  to the right, and  $X$  toward the viewer. The coordinates of both digital model and physical model is shown in Figure 12. Here, Figure 12a shows Unity coordinate where  $Z$  is pointing upwards and Figure 12b shows  $X$  coordinate points upwards.



(a) Coordinates in Unity Space



(b) Coordinates in Physical Space

Figure 12. Coordinates System

With the help of the transform matrix, the Cartesian coordinates of Unity model

can be transformed to cartesian coordinates of a physical robot system using Hadamard product [37, 38, 39] as shown in equation 1:

$$\begin{bmatrix} J1_{ux} & J1_{uy} & J1_{uz} \\ J2_{ux} & J2_{uy} & J2_{uz} \\ J3_{ux} & J3_{uy} & J3_{uz} \\ J4_{ux} & J4_{uy} & J4_{uz} \\ J5_{ux} & J5_{uy} & J5_{uz} \\ J6_{ux} & J6_{uy} & J6_{uz} \end{bmatrix} \circ \begin{bmatrix} 0 & 0 & -1 \\ 0 & -1 & -0 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} J1_{fx} & J1_{fy} & J1_{fz} \\ J2_{fx} & J2_{fy} & J2_{fz} \\ J3_{fx} & J3_{fy} & J3_{fz} \\ J4_{fx} & J4_{fy} & J4_{fz} \\ J5_{fx} & J5_{fy} & J5_{fz} \\ J6_{fx} & J6_{fy} & J6_{fz} \end{bmatrix} \quad (1)$$

where,  $J1_{fx}$ ,  $J1_{fy}$ , and  $J1_{fz}$  shows the X, Y and Z components of J1 joint for FANUC robot respectively, and  $J1_{ux}$ ,  $J1_{uy}$ , and  $J1_{uz}$  indicates X, Y and Z components of J1 joint of Unity digital model. With the help of element-wise multiplication or Hadamard product, transformed the Unity cartesian coordinates for J1-J6 joint to Physical robot joint coordinates.

#### 4.4 Communication Channel

After establishing the digital model in the gaming engine, next comes selecting a communication link between the Digital and Physical model, which is the backbone of the digital twin.

FANUC robots support various communication protocols, starting from Modbus TCP/IP to Dot Net-based APIs. However, the types of data that can be accessed using these communication protocols are minimal. Most of the protocols support position registers, as well as digital input and output registers. These communication methods are easy to implement as a base code for this is provided by the company.

There exist another communication protocol known as socket messaging, which is based on Client-Server architecture. It is one of the oldest client-server-based protocols and is used in many industrial devices. In the case of FANUC robots, they provide a KAREL programming language compiler, and by using this compiler, the required socket messaging channel is written (for both Client and Server) with customized functions for motion execution.

Talking about the KAREL programming language, KAREL is an educational programming language designed for beginners by Richard E. Pattis in 1981 and adapted to use in programming robots and is named after Karel Čapek. He introduced the word "Robot" [40]. The benefits of using the KAREL programming language are two-folded. First, the KAREL programming language is native to FANUC robots. Second, FANUC KAREL enlists the functions to handle robot motion commands and provides access to all data registers starting from system variables to digital signals. Thus, I have implemented a socket messaging server with KAREL usage that receives joint angle data from the socket client and shares robot status data to its client, and calls the motion execution function.

On the unity platform, the socket messaging client is written in C# script. Now, both physical and digital models are ready to communicate via socket channel. After a successful handshake between Unity and Robot Controller as shown in Figure 13, required data can be exchanged between them.

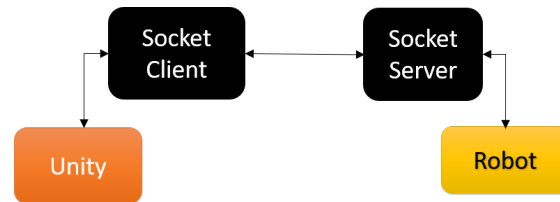


Figure 13. Communication Setup

**Programming for Communication Setup:** The programming part is designed considering the modularity of adding features to the robot both on the physical and digital sides. Many industrial robots support a single-threaded application. In this case, also, FANUC robots support synchronized functions. In this experimentation, the Unity scripts are divided into four parts:

1. **Socket server client:** Responsible for initiating a connection with socket server written on the robot side.
2. **Reading and sending data:** This script reads the robot joints data, which is generated by IK and send to the robot over socket messaging port.
3. **Update status:** In this experimentation, certain robot status signals are programmed. Next, these status signals are pushed by the socket server to its connected client. The list of signals is explained in Section 5.2.
4. **Data recording:** If the digital twin is not connected to the physical robot, a provision is created to temporarily store the data inside the Unity environment in a text file format. In addition to this, work has also been done to save the data sent to the robot via socket client in physical robot memory for its repetitive operations.

#### 4.4.1 Source Code

The code for this thesis work is available via google drive [41], the folder structure for the code is as follows:

1. DigitalTwinVR : Unity Project files
2. WeldTwinModel : FANUC robot files
3. Roboguide: Roboguide Version 9 (FANUC robot simulation software)

## 5 User Interface (UI)

As, digital model has been created and a communication channel (for interaction between digital and physical robot) is established. Now, let us take a look at the creation of a user interface (UI) that acts as a medium to control and perform desired operations between digital and physical robot.

### 5.1 Virtual Reality (VR) Interface

VR interface is present in the gaming industry for a long time. Nowadays, companies and researchers have been trying to explore the possibilities to utilize this technology in different sectors as well. The manufacturing industry is one of them, and they are exploring the benefits and use cases of VR interface. In this thesis, I have integrated VR with an industrial robot, with the scope of easing the programming and monitoring robotic actions. In the traditional approach, to program a robot for a specific task, we teach the intermediate points via teach pendant or by guiding a robot by holding its axes (also known as collaborative robots) and save these intermediate points inside robot memory. However, to successfully guide and generate an effective path requires training and experience for the operators.

#### 5.1.1 Setting VR Environment in Unity

In contrast to the traditional approach, the VR interface provides a simplified way for programming the desired trajectory. In this work, HTC Vive is used to interact with the robot. This interaction is facilitated using the robot's digital model in a virtual environment that is created on a Unity platform.

In a virtual environment, the VR controller generates a new position by holding the TCP of the digital model when it is around the component/region of interest. The data of the updated position is sent to an inverse kinematic algorithm, and the algorithm generates the corresponding joint angles for the robot joints to reach the updated position of TCP. By doing so, the tip of the end-effector follows the TCP and generates an entire program from the initial position to the complete trajectory. Figure 14 shows controlling of twin model in a virtual environment using VR controller.

After generating the list of joint angles for the required trajectory, these joint details are sent to the physical body (robot). Now, to replicate the path of the digital robot, I have used a forward kinematics algorithm on the physical robot side. Forward and Inverse Kinematics are a complement of each other, where IK calculates the joint angles to reach a position. On the other hand, forward kinematics calculate the position on the basis of joint angles. The controlling mechanism between the digital twin and physical model is shown in Figure 15.

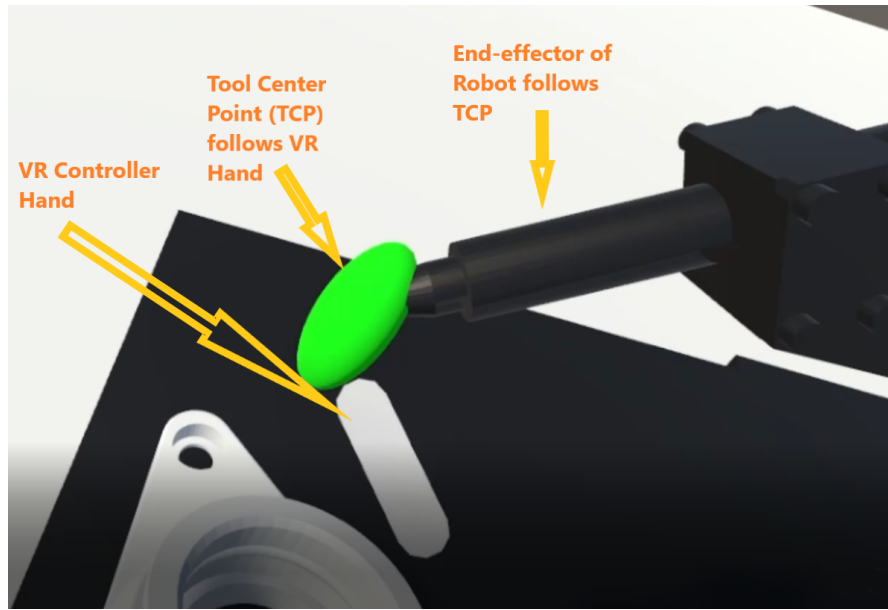


Figure 14. Controlling Twin model via Virtual reality controller

## 5.2 Physical Device Interface via Teach Pendant

The robot controller comes with a predefined set of screens where we can monitor and configure the details of a robotic operation. In this experimentation, over the socket messaging channel, the data exchange between digital and physical mode is bi-directional. To control the physical model through the VR interface, I am reading the robot's status, such as whether the communication channel is healthy or not, is there any fault on the robot side? Or if a robot is in an operating state or in an idle state. For most industrial robots, there is a list of status signals related to robotic operations that are internally mapped to binary registers. In FANUC robots, these status signals are available through User Operator Panel (UOP) signals. So, I read them in Unity and mapped them to the digital model. Figure 16 shows the list of data that is available on the UOP screen of the robot controller.

As we can observe from Figure 16, there is a vast amount of data available in a robot controller, but for this experimentation, I am considering three main signals as listed below:

1. **Communication Status:** Communication status (communication channel) is the heart of the complete setup. Every second it sends a beat/flag with a "0" value and receives back "1". If the value remains unchanged "0" for more than one second, it raises a red flag indicating that there exists a problem with communication line and sets the simulation mode to offline. This status bit is being read by other scripts (such as "Parallel Motion" and "Record and Send") that requires online status for

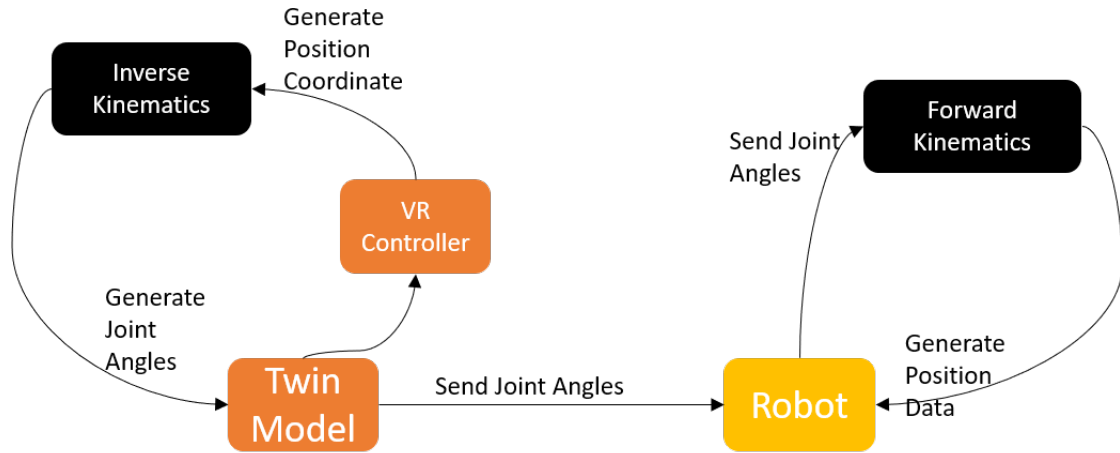


Figure 15. Control cycle between Digital and Physical model

its operation. If communication is broken, then it disables the options for selection by the user. Although, in this state, we can still create a offline simulation program and later transfer the recorded file to the robot.

2. **Fault:** Fault status of the physical robot is checked/read with every communication heart beat signal. If the robot is in a fault mode, then running the robot in any mode (Online/Offline) is not possible. To retain any operation with robot, the fault has to be resolved manually.
3. **Running/Stop:** This signal indicates whether a robot is in motion or in a stop state. This is a binary bit signal, therefore, if robot is in running state then Parallel motion option will be disabled.

On the basis of the feedback or status signals from the physical robot, the robot is operated in two different modes via digital twin:

1. **Parallel Motion/Online Mode:** Verifying the robot's movement is one of the main advantages of the twin model, unlike other simulation software, which works in offline mode, where the program is generated and sent to a physical device. It is convenient in some cases where the path followed by the robot is fixed. However, it requires precise information about the physical setup, which is very difficult and often leads to rework in the actual environment. However, in this model, I am using a simultaneous control approach to control the physical body and the simulated model. Due to this, the manual adjustments can be removed and provides an accurate model for immediate implementation.

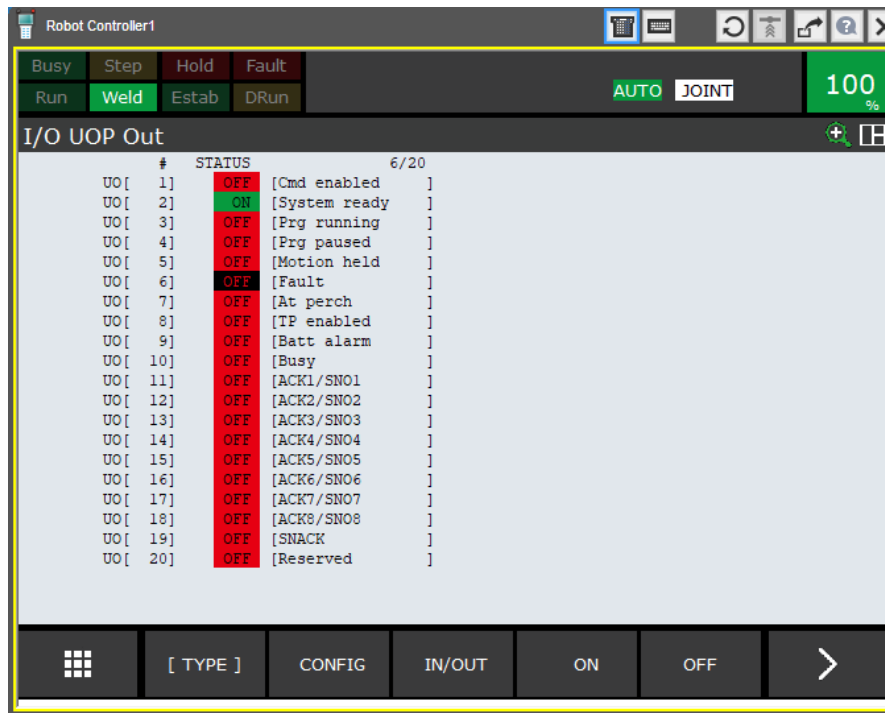


Figure 16. Robot Status Data

To operate the robot in parallel motion. The twin model first checks the status of communication, which should be healthy, and if it is idle or not. If both conditions agree, the physical model can follow the commands sent from the VR interface and record the joint angle data for future repetitive work. Figure 17a shows the set of signals which are captured from the robot and mapped to a virtual environment. The signals are Connection, Fault, and Running. If all three indicate a green signal, we can operate for synchronized movements.

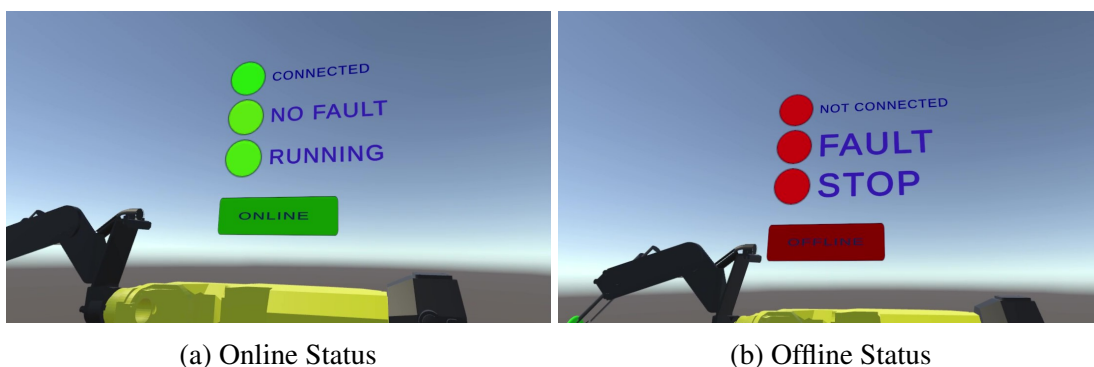


Figure 17. Online/Offline Mode

2. **Record and Send/Offline Mode:** The record and Send operating mode is simple compared to parallel motion mode, as it does not require an online connection with the robot. In this case, the user can visualize the robotic operation w.r.t its component in the virtual environment and generates the joint data which is saved in a local space (Unity). Once the robot comes online, the required file can be transferred.

Additional to this, if a robot is busy in performing a predefined task, the new task cannot be defined considering the safety concerns. However, a user can create an offline program that records the joint values in a text file format. Later, when the robot comes to an idle status, the file can be sent to the robot, and the motion program can be executed. The offline status screen of VR is shown in Figure 17b, in this situation all the signals shows red color, and stores the motion path data in local directory.



## 6 Results and Analysis

The first step in determining the system's responsiveness is to examine the time-lapse to exchange data. As per the setup of the experiment, the time-lapse for transferring joint angles (six joint angles in this experimentation) data of the digital model to the physical robot model is observed. It is observed that by using socket messaging, it takes approximately 40ms (milliseconds) to transfer the joint data and read robot status signals in a digital setup. This is a considerable improvement compared to 150ms that is reported in research [27].

### 6.1 Joint Movement Analysis

Now, let us dive into the accuracy of robot movement attained in a digital twin process. Figure 18 shows a trajectory path created via VR controller movement in the virtual environment. The path traced by the TCP of a robot is plotted in a 3D graph. Here the movement in 3D space is in mm (millimeters).

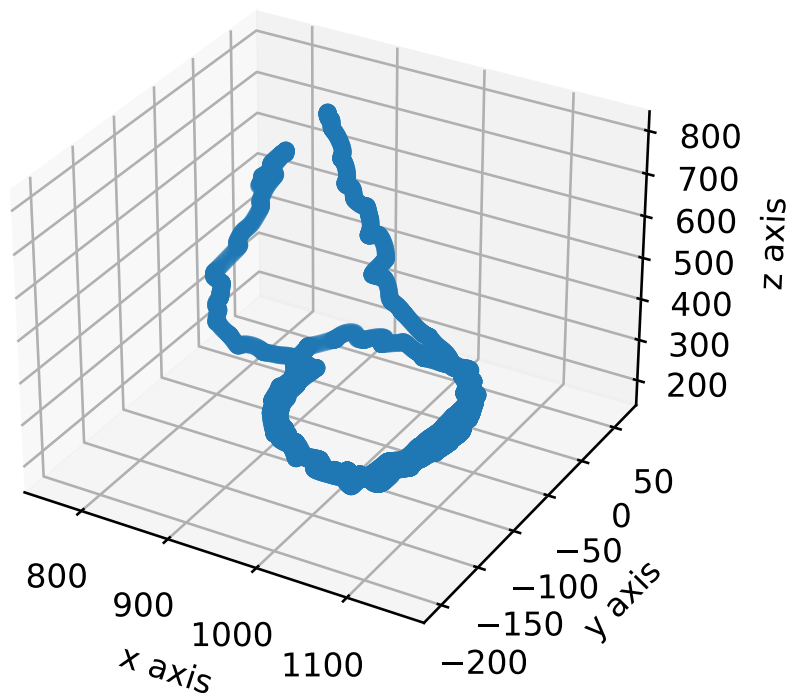


Figure 18. Robot Trajectory (axis values in mm)

For the above trajectory, Figure 19, shows the range of movement of joints J1-J6 (in degree) of a robot arm to attain the intermediate points across the motion path. At every step, when the TCP position is updated, the IK algorithm generates a new set of Joint angles (J1-J6).

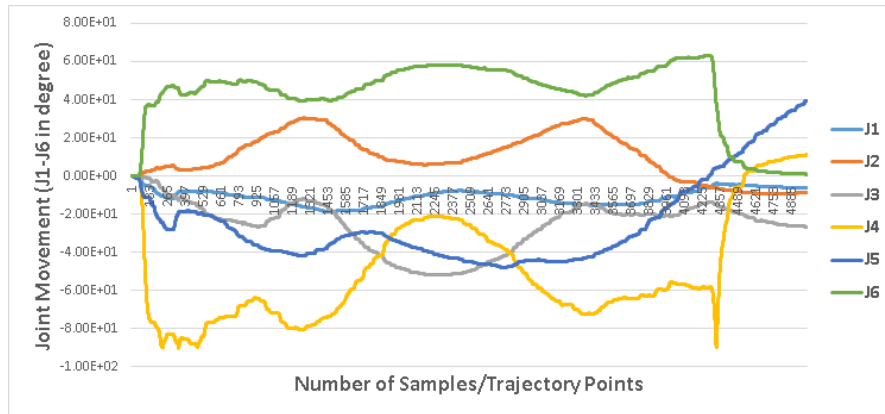


Figure 19. Range of Movement of Individual Joints

After looking into the trajectory and joint angle movement, let us compare the joint movement data of digital and physical robot, as we discussed earlier in Section 4.3 that on digital and physical robot model, we are using IK and forward kinematics (FK) algorithms. As a result, there is a possibility of some difference in Commanded and Executed Joint movements.

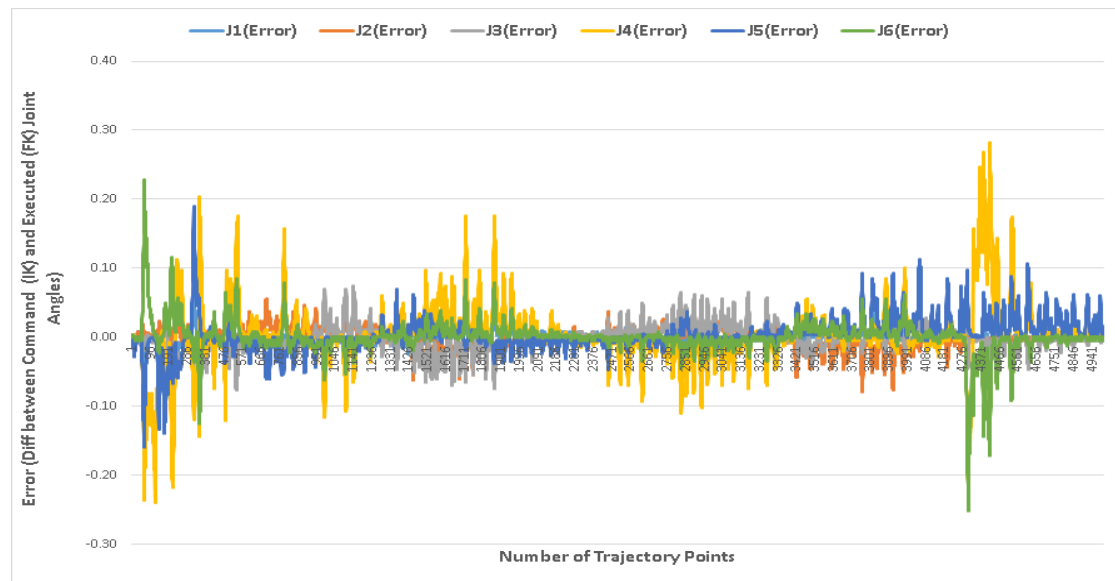


Figure 20. Difference Between Commanded (IK) and Executed (FK) Joint Angles

Figure 20 shows the difference between commanded joint angles, which is the data generated from the digital robot model and executed motion in a physical robot. It is observed that the value of error across each joint J1-J6 varies from  $0.01^\circ$  and  $0.28^\circ$ . The error is at its peak at the start and end of a trajectory. There can be two different factors

contributing to this error, (1) error can be due to acceleration and deceleration cycle across robot joints, (2) error induced by conversion process of IK to FK.

To analyse the cause of error, a random trajectory is generated in VR and analyzed the errors across the joints of the physical robot. Figure 21 shows the range of movement of a robot arm joint angles. The purpose of the random movement is to get more details on the error. Whether it increases with an increase in joint movement range or it depends on acceleration and deceleration cycle.

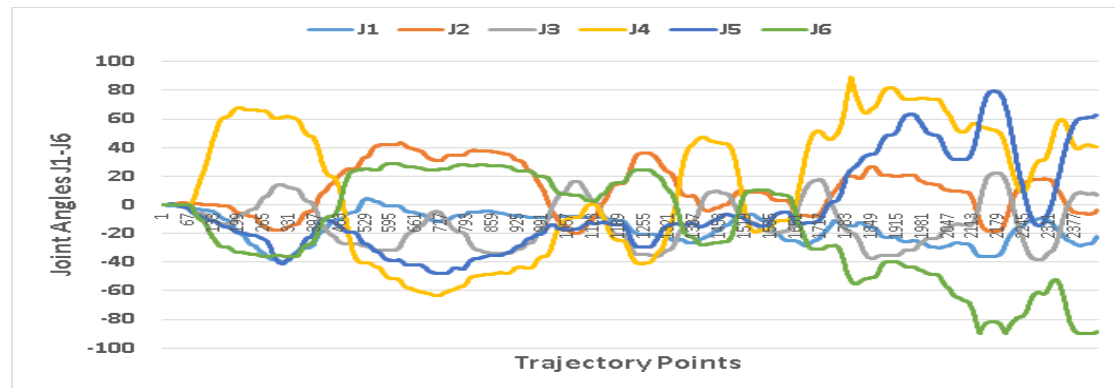


Figure 21. Joint Angle movement for a random trajectory

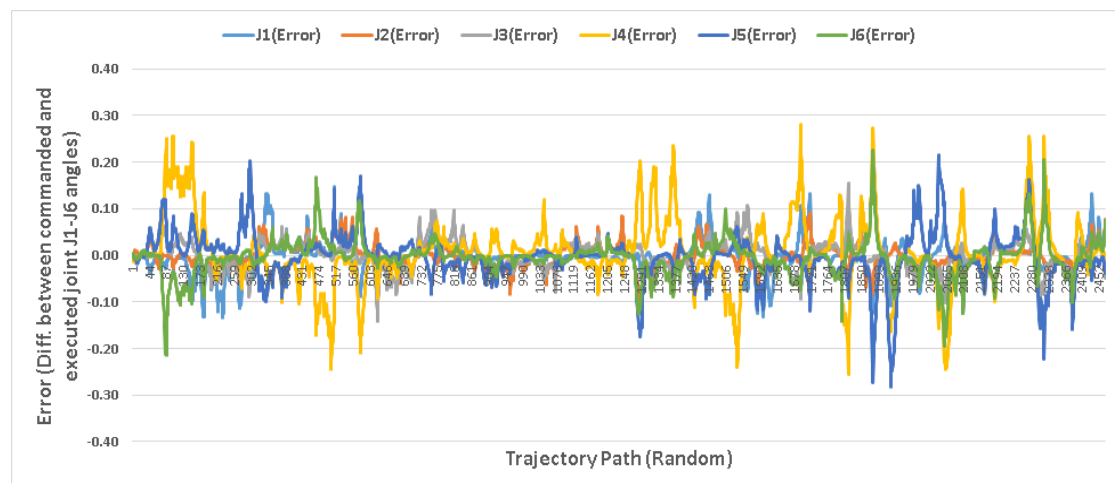


Figure 22. Difference Between Commanded (IK) and Executed (FK) Joint Angles

Figure 22, shows the same error range as compared to Figure 20, even during the random motion of TCP, the error range remains in the same window of  $0.1^{\circ}$  to  $0.28^{\circ}$ .

From the above error plots, there is a possibility that the deviation of joint movements is due to acceleration or deceleration values. As the joint moves from  $0^{\circ}$  to  $80^{\circ}$ , the error

also increases. However, after attaining a particular value, the error gradually decreases if the movements remain in that range. In addition to this, it is observed that the error across the J4-J6 joint is more as compared to J1-J3 joints, as the movement across J4-J6 varies more abruptly in the range of 0-100° as compared to J1 to J3 movement pattern, which is smooth and its range of motion is less (0-30°) compared to other joints.

## **7 Discussion**

Here, we are going to discuss that the solution/framework developed in this work is easy to adapt to almost all FANUC robots. This makes the framework more generalized in a way that it is independent of the robot model (see Section 7.1 for detail). Additionally, we showed some advantages of using VR interface for programming robot and how it can benefit in various challenging tasks using Used Case scenarios in Section 7.2.

### **7.1 Generalized Solution For FANUC Robots**

The software components under this experimentation are developed in such a way that we can generalize this implementation process for any FANUC robot. The first step towards Digital Twin is to develop a 3D animated compatible robot which can be done by following the step defined in Section 4.1 (Model Rigging).

After importing the 3D animated robot developed above, it can be imported in Unity and other scripts that are developed in this thesis can be simply plugged to any digital and physical robot models. This provides a generalized solution for entire range of FANUC robots. However, this solution has a limitation that it may not be compatible with other robotic manufacturer companies such as KUKA, NACHI, ABB, YASKAWA, etc.

### **7.2 Use Case Scenarios**

The concept of Digital Twin has applications in vast domains such as Industry Planning, production estimation, medical division for remote surgery, robot trajectory simulation, etc. In this thesis, I have discussed a few applications that give a more specific idea about VR environment utilization and the benefits of digital twin two-way communication. These applications include trajectory programming for complex edges and space-constrained environments, estimation of robot operation for bigger objects such as Jet Plane, and Predictive maintenance.

#### **7.2.1 Trajectory Planning For Complex Edges**

Pointing/guiding a physical robot's end-effector at a particular position and its orientation can be a difficult or time-consuming task via a manual programming method. One such setup is shown in Figure 23. The edges of the component that requires high accuracy take a lot of time and often lead to minor adjustments. This rework can be avoided by visualizing the component details in a VR environment, where we can clearly see the detailed 3D view of the region of interest and move the VR hand around it.

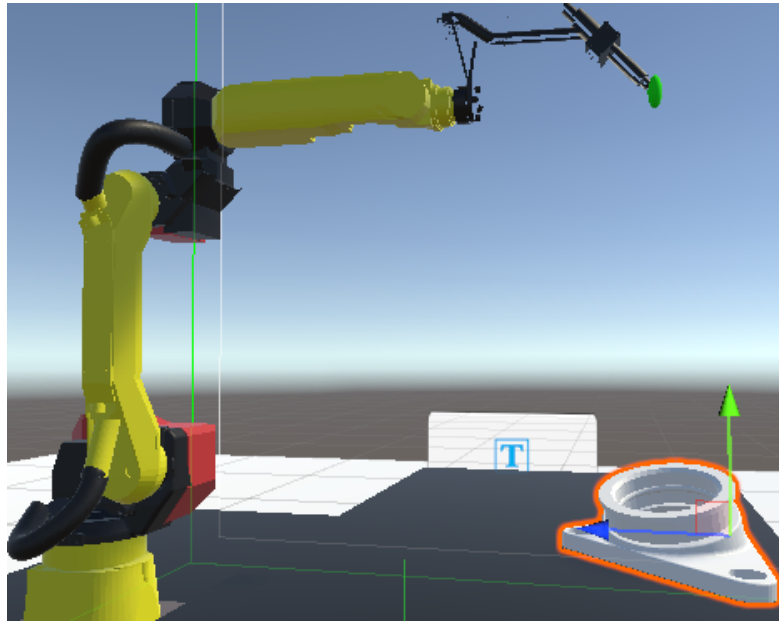


Figure 23. VR Environment for programming an Engine component

### 7.2.2 Program Creation In Space Constrained Surroundings

Space required or available for a new setup is always a challenge for factories and manufacturing units. This pushes the limits and demands a minimum space acquisition for setting up robotic/machine cells. Here, the capability of a virtual environment can be used to create a motion path in constrained spaces and simulate the feasibility of setting up a physical system.

Figure 24 shows a setup of a tank welding application. Here robot bench and component are closely placed. Such setup creates difficulties in creating or troubleshooting a robot's motion path. However, performing this task in a virtual environment provide a safe and well-defined environment for human-robot interaction.

### 7.2.3 Training robotic operations via Machine learning (ML)

Nowadays, as we are trying to deploy robots for a broad range of applications such as warehouse operations, agriculture, kitchen work, etc., the main challenge for making applications with a robot is to teach it for a dynamic handling environment. The process of teaching a robot can be done by creating synthetic data using a VR setup. Let us consider agriculture application. To operate robots in an open environment, we have to gather data for the robot vision system under different illumination intensities. This environment is easy to create in a gaming engine as compared to capturing data in a real scenario. Gathering data in real scenarios can be time-consuming and costly.

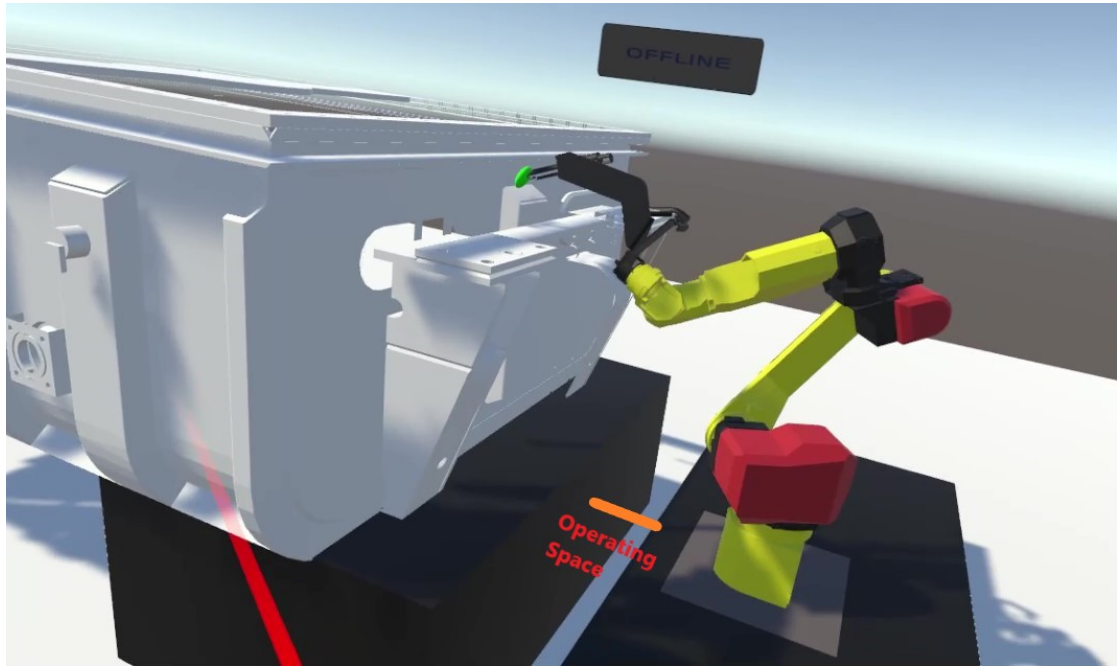


Figure 24. Robotic Operation in constrained space

#### 7.2.4 Predictive Maintenance & Error Detection

Predictive maintenance is a hot topic specially for manufacturing sector. As a single failure can lead to huge losses and interrupts the complete manufacturing line. Due to which manufacturers are focusing on capturing the data from devices and predicts the failure and maintenance requirements. For this purpose, Digital Twin can be utilized. As Twin model defines a two-way communication. So, over this communication channel, we can capture the robot data and fed the same to machine learning algorithms for prediction. This eliminates the dependency on any additional software for data capturing.

## 8 Conclusion

In this thesis, we have successfully created a Digital Twin model of an industrial robot. The developed digital model communicates with the physical robot using client-server architecture via socket messaging. By integrating VR in a digital environment, we have tested robot trajectory programs. The trajectories that are created in a virtual environment are easy to follow in a physical robot. In addition to this, we have analyzed the accuracy of the physical robot's trajectory compared to the data generated via a VR environment.

Based on the developed digital twin model for industrial robots, my contribution towards research enhancement can be summarized as follows: (1) Online/parallel motion between the virtual environment and physical robot for desired trajectories, and a generalized program is created on the physical robot for repetitive motion execution; (2) Ease of programming for complex edges and space-constrained environments. Based on the survey, we can confidently conclude that the VR environment is easy to operate and highly interactive for robot programming as compared to other available programming tools; and (3) Communication across different platforms (Dot Net and Karel) is established. The simulation analysis shows the latency of approximately 40ms (milliseconds) with a mean error of joint movements is  $0.28^\circ$ . Therefore, with this latency and accuracy, we can utilize this model effectively for industrial applications. However, for high precision control, there is a need to smooth out the joint movements by implementing a smoothing function or other algorithms.

**Limitations:** There are a few limitations of this work, as this work can only be copied/applied to FANUC robots. As on the robot side, the scripts are developed in KAREL language that is native to FANUC robots only. As per my knowledge, the KAREL platform is not being used by any other robot manufacturer company. However, this limitation can be removed with a bit of rework on establishing communication with other manufacturer robot libraries. Some of them provide Dot Net libraries or Java-based APIs to interact and establish communication and control joint movements.

**Future Work:** Gaming platforms are loaded with many features that can be extended to an industrial environment. One such example is the multiple user single-server virtual reality system. There are several networking games where players from different locations create their virtual avatar and interact in a simulated environment. The same concept can be implemented in the industrial environment via a game engine. The main advantage of this is the better understanding of the problem. For example, let suppose there is a robot that is installed at a remote location, and you being a service provider, have to listen and understand the customers and also help them to solve the problem, assisting over telephonic discussion, takes a lot of time just to get a gist of the problem as a client is very less likely to be comfortable with technical terms and may land on different understanding nodes. However, having a digital twin model eases the process of explaining and understanding the problem in a short span of time.

The accuracy of trajectories generated via VR environment depends on human hand

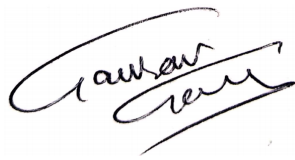


movement and most of the motion path seems rough compared to trajectories generated via simulation software. To overcome this, further work is required to smoothen the trajectory, this will help in utilizing the complete potential of the digital twin in a broad range of application areas. At present, there is not much work done that compares the motion path between Digital and Physical models. At present most of the focus is on interaction system development with VR/AR/MR.

## Acknowledgement

I am heartily thankful to my supervisors Professor G. Anbarjafari (University of Tartu) and Dr. Vladimir Kuts (Taltech University), for their guidance throughout the thesis work. Their advice helped me to stick to a plan and complete the work on time.

In addition, my former colleagues Yash Padia and Aravinda Rao, assisted me in learning more about FANUC robots and successfully conducting experiments.

A handwritten signature in black ink, appearing to read 'Gaurav Garg', with a stylized, cursive script.

Gaurav Garg

## References

- [1] David Eberly. *3D game engine design: a practical approach to real-time computer graphics*. CRC Press, 2006.
- [2] Dirk Thomas. Changes between ros1 and ros2. <https://design.ros2.org/articles/changes.html>.
- [3] Andrzej Burghardt, Dariusz Szybicki, Piotr Gierlak, Krzysztof Kurc, Paulina Pietruś, and Rafał Cygan. Programming of industrial robots using virtual reality and digital twins. *Applied Sciences*, 10(2):486, 2020.
- [4] Justin W Hart, Nick DePalma, Mitchell W Pryor, Bradley Hayes, Karl Kruusamäe, Reuth Mirsky, and Xuesu Xiao. Exploring applications for autonomous nonverbal human-robot interaction. In *Companion of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, pages 728–729, 2021.
- [5] Magdalena Muszyńska, Dariusz Szybicki, Piotr Gierlak, Krzysztof Kurc, Andrzej Burghardt, and Marek Uliasz. Application of virtual reality in the training of operators and servicing of robotic stations. In *Working Conference on Virtual Enterprises*, pages 594–603. Springer, 2019.
- [6] Vladimir Kuts, Tauno Otto, Toivo Tähemaa, and Yevhen Bondarenko. Digital twin based synchronised control and simulation of the industrial robotic cell using virtual reality. *Journal of Machine Engineering*, 19, 2019.
- [7] Ali Ahmad Malik and Arne Bilberg. Digital twins of human robot collaboration in a production setting. *Procedia manufacturing*, 17:278–285, 2018.
- [8] Haiwen Zhang, Lin Ma, Jiao Sun, Hansheng Lin, and Matthias Thürer. Digital twin in services and industrial product service systems:: Review and analysis. *Procedia CIRP*, 83:57–60, 2019.
- [9] Fei Tao, Jiangfeng Cheng, Qinglin Qi, Meng Zhang, He Zhang, and Fangyuan Sui. Digital twin-driven product design, manufacturing and service with big data. *The International Journal of Advanced Manufacturing Technology*, 94(9):3563–3576, 2018.
- [10] Yuqian Lu, Chao Liu, I Kevin, Kai Wang, Huiyue Huang, and Xun Xu. Digital twin-driven smart manufacturing: Connotation, reference model, applications and research issues. *Robotics and Computer-Integrated Manufacturing*, 61:101837, 2020.

- [11] Jinfeng Liu, Xiangmeng Du, Honggen Zhou, Xiaojun Liu, L Ei Li, and Feng Feng. A digital twin-based approach for dynamic clamping and positioning of the flexible tooling system. *Procedia CIRP*, 80:746–749, 2019.
- [12] Mariusz Oleksy, Grzegorz Budzik, Agnieszka Sanocka-Zajdel, Andrzej Paszkiewicz, Marek Bolanowski, Rafał Oliwa, and Łukasz Mazur. Industry 4.0 part i. selected applications in processing of polymer materials. *Polimery*, 63, 2018.
- [13] Chao Zhang, Guanghui Zhou, Jun He, Zhi Li, and Wei Cheng. A data-and knowledge-driven framework for digital twin manufacturing cell. *Procedia CIRP*, 83:345–350, 2019.
- [14] Sidharth Baskaran, Farbod Akhavan Niaki, Mark Tomaszewski, Jasprit Singh Gill, Yi Chen, Yunyi Jia, Laine Mears, and Venkat Krovi. Digital human and robot simulation in automotive assembly using siemens process simulate: a feasibility study. *Procedia Manufacturing*, 34:986–994, 2019.
- [15] Arne Bilberg and Ali Ahmad Malik. Digital twin driven human–robot collaborative assembly. *CIRP Annals*, 68(1):499–502, 2019.
- [16] Niki Kousi, Christos Gkournelos, Sotiris Aivaliotis, Christos Giannoulis, George Michalos, and Sotiris Makris. Digital twin for adaptation of robots’ behavior in flexible robotic assembly lines. *Procedia manufacturing*, 28:121–126, 2019.
- [17] Klaus Dröder, Paul Bobka, Tomas Germann, Felix Gabriel, and Franz Dietrich. A machine learning-enhanced digital twin approach for human-robot-collaboration. *Procedia Cirp*, 76:187–192, 2018.
- [18] P Aivaliotis, K Georgoulis, Z Arkouli, and S Makris. Methodology for enabling digital twin using advanced physics-based modelling in predictive maintenance. *Procedia Cirp*, 81:417–422, 2019.
- [19] Fei Tao, Qinglin Qi, Lihui Wang, and AYC Nee. Digital twins and cyber–physical systems toward smart manufacturing and industry 4.0: Correlation and comparison. *Engineering*, 5(4):653–661, 2019.
- [20] Ján Vachálek, Lukás Bartalský, Oliver Rovný, Dana Šišmišová, Martin Morháč, and Milan Lokšík. The digital twin of an industrial production line within the industry 4.0 concept. In *2017 21st international conference on process control (PC)*, pages 258–262. IEEE, 2017.
- [21] Luis Pérez, Silvia Rodríguez-Jiménez, Nuria Rodríguez, Ruben Usamentiaga, and Daniel F Garcia. Digital twin and virtual reality based methodology for multi-robot manufacturing cell commissioning. *Applied Sciences*, 10(10):3633, 2020.

- [22] Guanghui Zhou, Chao Zhang, Zhi Li, Kai Ding, and Chuang Wang. Knowledge-driven digital twin manufacturing cell towards intelligent manufacturing. *International Journal of Production Research*, 58(4):1034–1051, 2020.
- [23] Riccardo Pinto and Tania Cerquitelli. Robot fault detection and remaining life estimation for predictive maintenance. *Procedia Computer Science*, 151:709–716, 2019.
- [24] Grigore Burdea and Philippe Coiffet. Virtual reality technology, 2003.
- [25] SG Tzafestas, P Borne, DG Caldwell, T Fukuda, and S Monaco. Intelligent systems, control and automation: Science and engineering, 2016.
- [26] Anthony G Gallagher, E Matt Ritter, Howard Champion, Gerald Higgins, Marvin P Fried, Gerald Moses, C Daniel Smith, and Richard M Satava. Virtual reality simulation for the operating room: proficiency-based training as a paradigm shift in surgical skills training. *Annals of surgery*, 241(2):364, 2005.
- [27] Heikki Laaki, Yoan Miche, and Kari Tammi. Prototyping a digital twin for real time remote control over mobile networks: Application of remote surgery. *IEEE Access*, 7:20325–20336, 2019.
- [28] RobotWorx Team. Fanuc robot specifications. <https://www.robots.com/robots/fanuc-m-10ia-12>, 2017.
- [29] HTC VIVE Team. Htc vive specification specifications. <https://www.vive.com/eu/product/vive/#vive-spec>.
- [30] Htc vive review. <http://doc-ok.org/?p=1478>.
- [31] Lee Wasilenko. Getting started with steamvr and unity., 2019.
- [32] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3d characters. *ACM Transactions on graphics (TOG)*, 26(3):72–es, 2007.
- [33] Amir Jafari, Nikos G Tsagarakis, and Darwin G Caldwell. Awas-ii: A new actuator with adjustable stiffness based on the novel principle of adaptable pivot point and variable lever ratio. In *2011 IEEE International Conference on Robotics and Automation*, pages 4638–4643. IEEE, 2011.
- [34] M Abtahi, H Pendar, Aria Alasty, and Gh R Vossoughi. Calibration of parallel kinematic machine tools using mobility constraint on the tool center point. *The International Journal of Advanced Manufacturing Technology*, 45(5-6):531, 2009.

- [35] Onder Tutsoy, Duygun Erol Barkana, and Sule Colak. Learning to balance an nao robot using reinforcement learning with symbolic inverse kinematic. *Transactions of the Institute of Measurement and Control*, 39(11):1735–1748, 2017.
- [36] S Phaniteja, Parijat Dewangan, Pooja Guhan, Abhishek Sarkar, and K Madhava Krishna. A deep reinforcement learning approach for dynamically stable inverse kinematics of humanoid robots. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1818–1823. IEEE, 2017.
- [37] Maria Blanton, Ana Stephens, Eric Knuth, Angela Murphy Gardiner, Isil Isler, and Jee-Seon Kim. The development of children’s algebraic thinking: The impact of a comprehensive early algebra intervention in third grade. *Journal for research in Mathematics Education*, 46(1):39–87, 2015.
- [38] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.
- [39] Chandler Davis. The norm of the schur product operation. *Numerische Mathematik*, 4(1):343–344, 1962.
- [40] Richard Pattis, J Roberts, and M Stehlik. Karel the robot. *A gentele introduction to the Art of Programming*, 1981.
- [41] Gaurav Garg. Source code. [https://drive.google.com/drive/folders/1bhz9Jj0Lzbc8zXpIR8p0F8QQ6\\_ndcvo8?usp=sharing](https://drive.google.com/drive/folders/1bhz9Jj0Lzbc8zXpIR8p0F8QQ6_ndcvo8?usp=sharing), 2021.

## **Licence**

### **Non-exclusive licence to reproduce thesis and make thesis public**

I, **Gaurav Garg**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

#### **Digital Twin for Industrial Robotics,**

supervised by Prof. G. Anbarjafari and Dr. Valdimir Kuts.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.
3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.
4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Gaurav Garg

**20/05/2021**