QUEEN'S UNIVERSITY BELFAST

School of Mechanical and
Aerospace Engineering
Ashby Building
Stranmillis Road
Belfast
BT9 5AH

Final Project Report

(MEE4040)

Development of a Digital Twin of a surgical robot

Author: Michael Newbold [40231351]

Project supervisor: Dr. Yan Jin

Programme: MEng - Mechanical Engineering

Date: 21/03/2022

SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

Detailed project description form

Project title	Development of a Digital Twin of a surgical robot.	**Project ID number**	118
		Supervisor	Dr. Yan Jin
Student name	Michael Newbold	**Academic year**	2021-22
		Student number	40231351

Indicate the student cohort

Stage 3 (MEE3030)		*Stage 4 (AER4002/MEE4040)*		*Postgraduate (MEE7012)*	
BEng Aerospace Engineering (40 CATS)		MEng Aerospace Engineering (60 CATS)		MSc Mechanical Engineering with Management (60 CATS)	
BEng Mechanical Engineering (40 CATS)		MEng Mechanical Engineering (40 CATS)	**X**	MSc Materials Engineering (60 CATS)	
BEng Product Design Engineering (40 CATS)					

Project background

There are over 60 million people around the globe who suffer from some form of visual impairment with approximately a third caused by cataracts, which is in fact the leading contributor for blindness across both lower- and middle-income countries as well as the second leading contributor in the UK and US. This issue can be remedied with cataracts surgery, however there is a global deficiency of specialised eye surgeons which is especially prevalent across developing countries. The arguable reason for such a deficit is because this form surgery requires both thousands of hours of training and high precision fine motor ability, often not attainable through any level of training. In contrast, a robot can be over ten times more precise than the human hand and can be efficiently used by a doctor to provide high quality surgery with a fraction of the time in training.

Robotic surgery requires seamless pairing of the doctor and robot in tandem to be able to effectively respond to developments in the operating room with no delay. To ensure this strong pairing, the doctor will use a haptic input device to track their input movements along side digital twin technology to see both live visual feed from the operating room and a real time simulation using live feedback data of the surgery. Digital twin technology is the future of robot simulation and has been used across the automation sector in some instances to simulate and optimise entire factory production lines. Through the use of a digital twin, it is possible to accurately simulate the surgery before the patient has even stepped into the operating room. This technology also enables the ability to train surgeons without the use of the physical robot therefore not subtracting from the resources towards other surgeries.

This project has been set up to develop such a digital twin with the ability for the user to input both individual DOF movement commands as well as vector commands with cartesian coordinates.

QUEEN'S UNIVERSITY BELFAST | SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

Detailed project description form

Project aims and objectives

The aim of this project is to develop a digital twin with the ability to be used to accurately simulate cataracts surgery on the human eye.

Project Objectives:

- Review previous literature on both robotic eye surgery and digital twin technology
- Understand and integrate the inverse kinematics of the robot
- Gain knowledge of Solidworks API and programme (by Visual basic or C++) to create movement in the CAD model
- Develop a GUI in MATLAB
- Develop algorithmic links between the GUI and digital model
- Create code commands for movement within each DOF
- Create code commands for vector movement across the cartesian coordinate system
- Debug and optimise the system

Project deliverable(s)

- Construct a literature review of previous research in robotic eye surgery and digital twin technology
- Develop a working digital twin model in Solidworks
- Develop a GUI with implemented movement commands
- Develop an algorithmic links between the GUI and digital model
- Final report

Mitigation plans and management for risks including for COVID-related issues

It is important to outline how Covid-19 and other such setbacks may affect progress. This project is computer based and will not require use of labs, technicians or the reliance of any parts or components. Additionally, the primary computer used during this will not be on campus and so the risk of viral spread is even further reduced. If the primary personal computer were to fail, both the Solidworks software and MATLAB are available at the CBS labs. Should the public transport to QUB ever be distributed, the project can be completed from home. Should a campus visit be necessary, risk can be avoided by following Queen's University and government guidelines, e.g. wearing PPE, adequate social distancing and keeping up with the newest measures.

Supervisor feedback

The project description is very well presented, with clear aim and objectives. It shows a very good understanding of the whole project. The project plan is logical, reasonable and achievable, with milestones appropriately identified. The mitigation plan is feasible, and the risk is low due to the nature of the project.

Signatures

Student signature: [signature] Date: 20/10/2021

Supervisor signature: [signature] Date: 20/10/2021

Summary

The aim of this project is to develop a digital twin of a surgical robot for cataracts eye surgery to help ease the demand for trained eye surgeons across the globe and allow for a most cost-effective method of testing design changes to robot. This report details the research and development of the digital twin, along with full verification and validation of design specifications as outlined by the design team of its physical prototype counterpart.

The digital twin was developed in Solidworks using the CAD model provided by the design team and was written in VBA using a macro. Both point to point (PTP) and individual parameter driven motion commands were implemented using inverse kinematic calculations and vector translations. A GUI was developed in VBA in conjunction with the twin and was linked within the macro. The GUI includes a nest function (NST), an RCM lock option, a haptic overwrite feature, an operation time control option, along with both real-time coordinate position updates of key points on the robot and real-time parameter values.

All functions of the digital twin have been developed for the integration of a haptic input device so that direct movements from a surgeon's hand can be used in place of inputs from the GUI. A solution to this haptic link has also been outlined within this report.

The digital twin developed was found to meet all specifications set by the design team of the physical prototype, and all additional functions were verified to be fully operational.

Acknowledgments

I'd like to thank my supervisor Dr Yan Jin for his continual positive feedback and inspiring excitement in my work. Both his lectures and time as a supervisor have driven my passion for robotics and mechatronics.

I'd also like to thank Yinglun Jian for his wealth of knowledge of the physical prototype and for being always available for advice regarding it.

Contents

Nomenclature

Symbols:

l	parameter length (mm)
ϑ	parameter angle (°)
r	radius of RCM mechanism (mm)
n	step variable (no units)
w	wait variable (s)
ε	timing error (s)
α	robot position angle (°)
T	operation time (s)
i	time interval between each data point received by DT (s)

Subscripts:

RCM	remote centre of motion
1	left-hand side of robot (from front facing)
2	right-hand side of robot (from front facing)
T	tool
A	joint A
B	joint B
C	joint C
D	joint D
E	joint E
OLD	old position
NEW	new position
SL	Solidworks

Abbreviations:

CAD	computer-aided design
VBA	Visual Basic for Applications

PTP	point-to-point
GUI	graphical user interface
NST	nest
QUB	Queens University Belfast
DT	digital twin
NHS	National Health Service
FDA	Federal Drug Agency
DoF	degrees of freedom
CAE	computer-aided engineering
3D	three-dimensional
NASA	National Aeronautics and Space Administration
VR	virtual reality
Tcp	tool central point
UI	user interface
ROS	Robot Operating System
MS	Microsoft
SQL	Structured Query Language
UN	United Nations
min	minimum
max	maximum

1.0 Introduction

1.1 Background

The lack of trained surgeons for cataracts surgery across the globe has led to waiting lists for the surgery of up to a median of 9 months within the NHS (National Health Service), with the length of these waits only increasing [1]. With the use of a robotic aid, surgeons can perform the surgery in a fraction of the training time and be 10 times as precise as non-robot aided surgeries [19].

In fact, 15.1% of all general surgeries in 2018 were robotic assisted, up from only 1.8% in 2012 [2]. Robots are no doubt the future for how surgeries will be carried out. This trend is the result of worldwide advancements in the combined fields of mechatronics and medicine; two multibillion-pound industries.

QUB (Queens University Belfast) is currently developing a surgical robot for cataracts surgery, as seen in figure 1. This robot will aid in the solution to the growing waiting list of vision impaired patients who require this procedure. A working prototype of this device has been manufactured and is currently being tested and optimised.

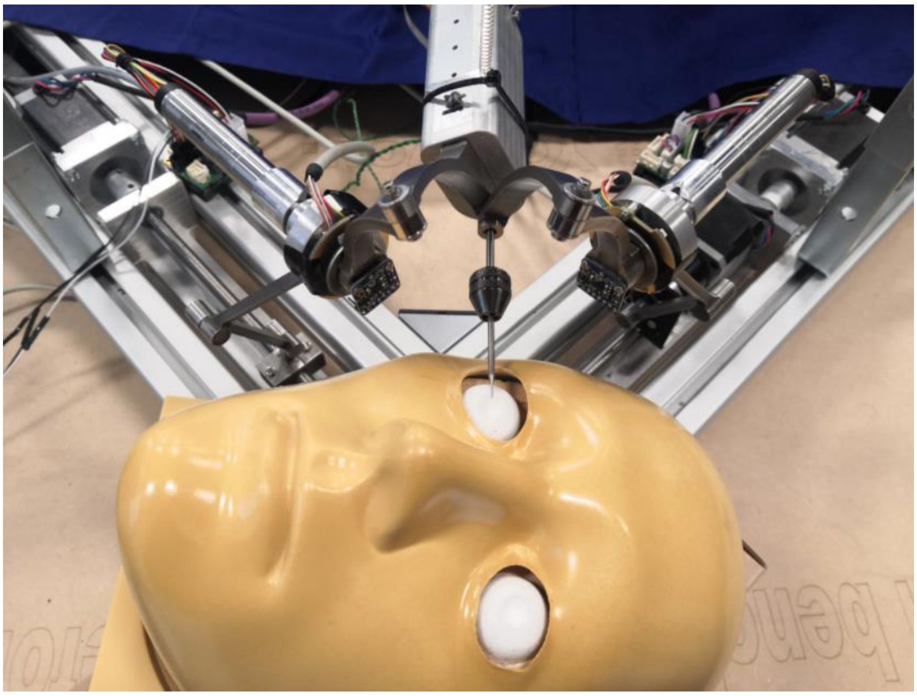


Figure 1 QUB prototype of cataracts surgical robot.

Although this prototype is a fully functional proof of concept and design, it does not have a digital twin. A DT (digital twin) is a virtual representation that serves as the real-time simulation of a process. In this case it is a virtual simulation of the robot, patient, and operating table; allowing the surgeon real-time visual and potentially haptic feedback of the surgery. Without this piece of technology surgeries cannot be simulated, increasing costs of testing any amendments to the design as manufacturing of new parts would be required to run such tests.

Additionally, the current movement commands to drive the prototype are not intuitive and would not be suitable for a surgeon to carry out an efficient surgery. A solution to these drawbacks will the focus of this report.

1.2 Project Aims and Objectives

As the rapid advancements of robotic technology in surgery progress, the DT technology must be developed in tandem. For QUB's prototype surgical robot, the development of both a digital twin and GUI (Graphic User Interface) will be discussed in detail in this report, as well as its full verification and validation.

The project aims have been set as such to meet the specifications of the QUB surgical robot design team. These aims are as follows:

- Develop fully controllable cartesian driven PTP (point to point) motion for the robot end effector and for locating the RCM (remote centre of motion) of the spherical parallel mechanism, both within a maximum tolerance of 20 microns.
- Develop fully controllable parameter driven motion.
- Design an intuitive GUI with cartesian and parameter motion inputs, displaying current positions coordinates for key points along with current parameter position.
- Develop a NST (nest) function for the twin.
- Design the logic in such that it allows for the future integration for a haptic input device.

Note: This DT is **not** intended for conducting eye surgery on live human patients. It has been developed solely for testing of the physical prototype and surgeon training purposes.

2.0 Literature Review

A review of relevant literature has been carried out to illustrate the scope of the innovations in the field of robot assisted eye surgery with particular focus on DT technology and the revolutionary progress on the implementation of it to the medical field.

2.1 Medical Robots

It was only natural that mankind would try to integrate the reliability and fine motor skills of modern machinery to the operating room, it was only a matter of when. It was as far back as the late 1980s when this vision would become a reality, when a standard PUMA 560 assembly robot was modified for use in a delicate neurosurgical biopsy, a non-invasive, non-laparoscopic surgery [3].

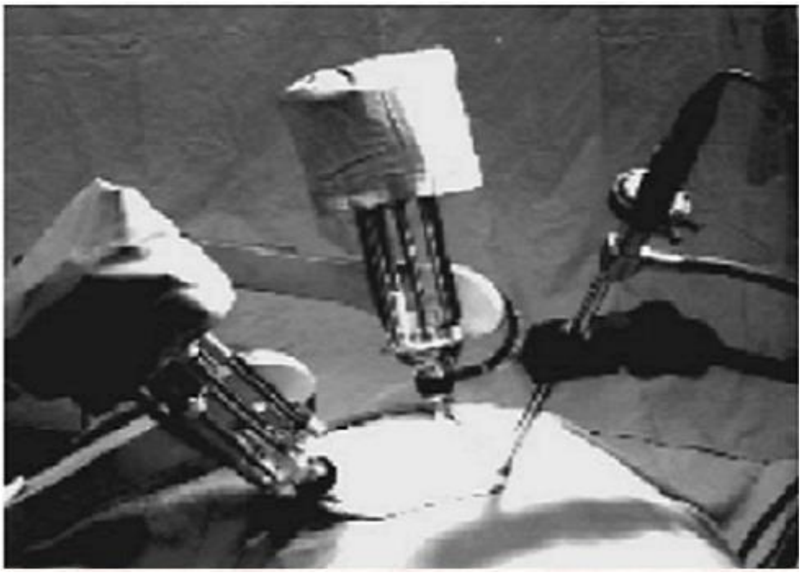


Figure 2 Modified PUMA 560 for neurosurgical biopsy. [3]

As is the case for many technological advances, further development of this concept was spearheaded by the United States Department of Defence. There was much effort to create a device capable of performing long distance trauma surgeries in battlefield settings. In collaboration with many innovative start-ups and research agencies, the first prototype for the world renown da Vinci Surgical System was developed [4].

Figure 3 Intuitive Surgical Inc's da Vinci Surgical System. [4]

Original prototypes were designed for highly specialised surgeries; however, this design quickly became ubiquitous for minimally invasive surgery and after receiving FDA (Federal Drug Agency) approval in 2000 it remains the most prominently available multipurpose surgical robot, being found in operating rooms across the globe.

2.2 Robot for Eye Surgery

The concept of introducing this technology to surgery of the human eye has only recently started to be explored. It was as recent as 2016 that Oxford University conducted the first robot assisted surgery on the eye. This trial was performed with a 7 DoF (degree of freedom) parallel robot (Figure 4) which carried out vitreoretinal surgery on 12 patients with age-related muscular degeneration of the retina.

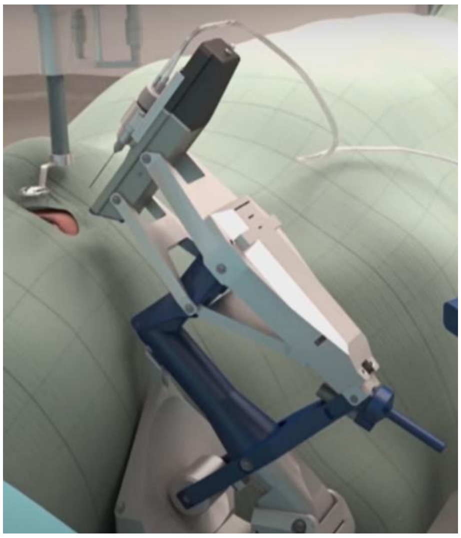


Figure 4 The PRECEYES Surgical System. [5]

Similar to minimally invasive cataracts surgery, an incision of only 1 mm was required. The trials clearly demonstrated the ability to remove unwanted tremor from a surgeon's hand and concluded that robot assisted surgeries could be performed with a precision of 1000th of a millimetre [5].

2.3 Digital Twin Technology

A digital twin can be most simply defined as a virtual model or simulation designed to accurately represent an object or system in the physical world.

Before the dawn computing and CAE (computer-aided engineering), if an engineer wanted to test the feasibility of their design they would have to manufacture a physical prototype. If they wanted to make a change to that design, the hand calculations, the engineering drawing, the manufacture, and the testing would all be carried out again. This process was both costly and time consuming, and thus the multibillion-pound industry of CAE modelling software was born.

Creating an accurate 3D (3 dimensional) representation of a design in a modelling software is just the start of a DT's potential. Entire supply chains, factory workflows and human biological systems can all be simulated virtually. With modern simulation software, a design can be tested virtually in every situation limited to your computing power, which continues to be rapidly advancing.

The first recorded use of the term 'digital twin' in the present context was by John Vickers of NASA (National Aeronautics and Space Administration) as recently as 2010 although the concept is much older. The concept is first seen as far back as the late 1960s during the Apollo 13 program. NASA's engineers were presented with a unique dilemma as astronauts must account for the rapid changes to their vehicle while exposed to the extremely hostile environment of space. After life support failed, NASA decided they could no longer base any corrective decisions on an original model and thus the first primitive DT was born (figure 5), in a combination of physical and mathematical models [6].

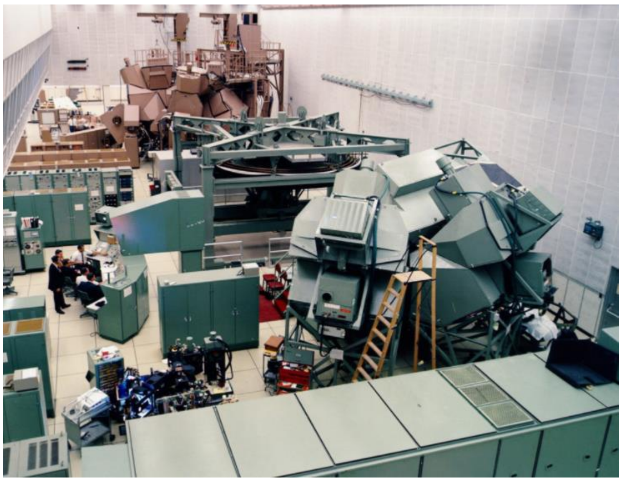


Figure 5 Apollo simulators at Mission Control in Houston. [6]

2.3.1 Digital Twin for Robots

It wasn't until 2002 when Dr. Michael Grieves introduced DT technology to manufacturing that this concept would be first implemented to simulate an entire system. In the following decades, the supporting information technology to both maintain and develop these systems has exploded [7].

By utilising a DT of an assembly line and its product, the time taken to both set up and validate a robotic system is significantly reduced. Thus, robust and reliable results are achieved faster and at much lower cost [8].

2.3.2 Digital Twin for Surgical Robots

Digital twin technology is used for simulation throughout every corner of the medical field, from improving hospital supply chain resilience to creating virtual organs [9 & 10]. As is the focus of this report, this technology is also being used to simulate surgical robots for the purpose of reducing the required skill level of a surgeon performing minimally invasive robotic surgery. In fact, a pilot study performed on the DLR MiroSurge system was conducted to examine if haptic feedback obtained from a DT during training influenced a novice surgeon's ability [11].

The surgeons who participated in haptic assisted training were found to perform more accurate motions when completing robotic assisted pick and place operations. Thus, it was concluded that there were positive trends to suggest haptic DT training would reduce time taken to train a surgeon for minimally invasive robotic surgery.

This DT technology is the baseline for further innovation. With an accurate model, hundreds of surgeons can be trained simultaneously on different machines, thousands of miles away from the physical robot. In fact, research is already being carried out for conducting real-time remote-control surgery with a digital twin over a mobile network with VR (virtual reality) integration [12].

3.0 Kinematic Model of QUB Surgical Robot

Before the virtual development is discussed, the physical counterpart must first be outlined. This section will detail the architecture and kinematic derivation of the QUB Surgical Robot. Both the prototype design and derivation were carried out by the design team based at QUB [14].

3.1 Architecture of Robot

For minimally invasive eye surgery, a surgeon would typically have two tools, which are inserted through incision points on either side of the eye, to destroy the malfunctioned lens and replace it with the new artificial one. For each tool, 4 DoF motion is required, including 3 rotations (3R) about the incision point and one translation (1T) through it. For a robot to be a suitable replacement for a human hand, it must be designed to maintain the same 3R1T DoFs after locating the point of incision. Throughout this report, this point of incision will be referred to as the RCM. This RCM also must be a virtual point, as to avoid collision with the eye. With these specifications in mind, the QUB surgical robot was designed as a 7 DoF 2-PRRRRR (P & R representing prismatic and rotational joints respectively) parallel robot with a remote centre of motion, depicted in figure 6. To limit robot tool motion in 3R1T DoFs, a Parallel spherical RCM mechanism was used. This mechanism was used due to its advantage in high precision, light moving structures and high stiffness [13 & 14].

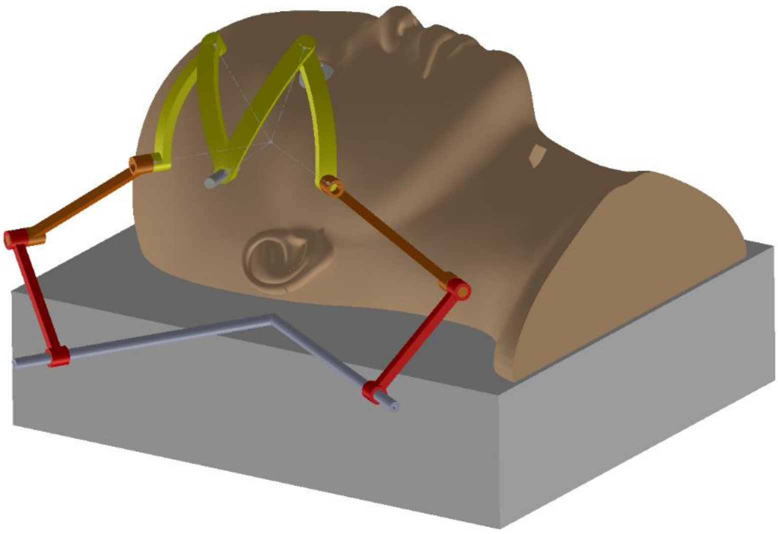


Figure 6 CAD depiction of QUB surgical robot with patient.

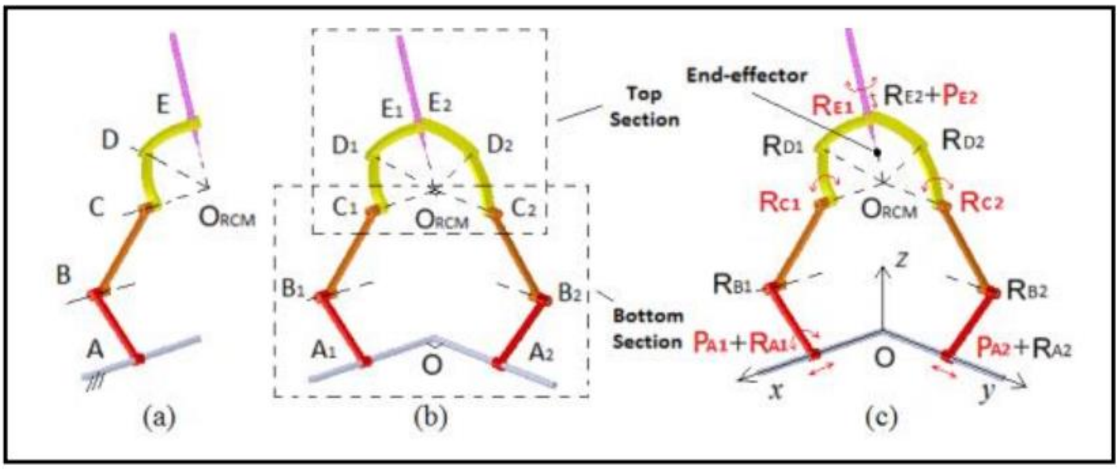


Figure 7 Schematics of QUB surgical robot. **(a)** One leg. **(b)** Two sections. **(c)** Passive and actuated joints. [14]

As seen in figure 7(b), OA_1 and OA_2 are perpendicular to each other to decouple the two translational DoFs at the base. OA_1 and $O_{RCM}C_1$ are parallel. OA_2 and $O_{RCM}C_2$ are also parallel, hence $O_{RCM}C_1$ and $O_{RCM}C_2$ are perpendicular to each other. Lengths and orientations of $O_{RCM}C_1$ and $O_{RCM}C_2$ are both set when all three actuated base joints are fixed. As a result, the relative position between C_1 and C_2 is constant, allowing the top section to be regarded as a five-bar parallel spherical RCM mechanism. The bottom section is responsible for 3 DoF positioning to locate the RCM [14].

Figure 7(c) highlights the passive and actuated joints, where the actuated joints are highlighted in red. The three actuated base joints, P_{A1}, P_{A2} and R_{A1} provide 3T DoFs along x, y and z axes to position the RCM at the incision point. P_{A1} and P_{A2} are solely responsible for movement along the x and y axis respectively. Movement along the z axis requires actuation of both P_{A2} and R_{A1}. When the y coordinate of O_{RCM} is first determined, the z coordinate of O_{RCM} is then only dependant on R_{A1}, thus all three DoFs are partially decoupled [14].

After O_{RCM} is positioned, P_{A1}, P_{A2} and R_{A1} are locked. The remaining actuated joints, R_{C1}, R_{C2}, R_{E1} and P_{E2} manipulate the end-effector to control the 3R1T DoF motion. R_{C1} and R_{C2} control the tool orientation with 2R DoF while R_{E1} and P_{E2} provide 1R1T DoFs along the tool axis. The mobility of these actuators is depicted in figure 8 [14].

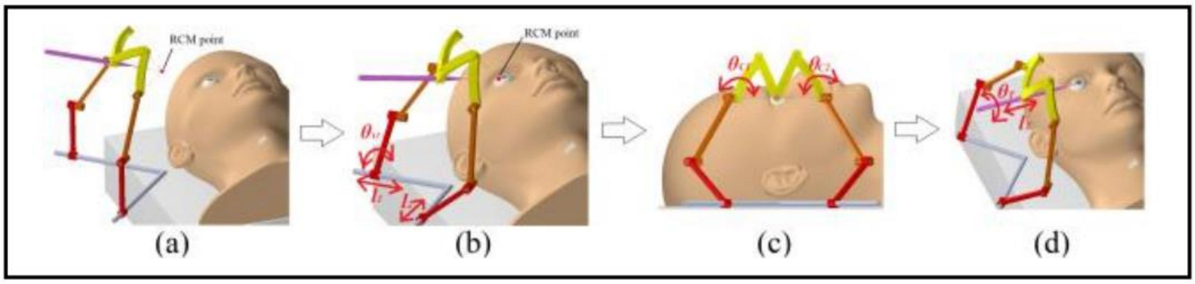


Figure 8 **(a)** NST position. **(b)** 3T motion. **(c)** 2R motion. **(d)** 1R1T motion. [14]

3.2 Kinematic Model

Table 1 Actuators and kinematic input parameters.

Actuators	P_{A1}	P_{A2}	R_{A1}	R_{C1}	R_{C2}	R_{E1}	P_{E2}
Parameters	l_1	l_2	θ_{A1}	θ_{C1}	θ_{C2}	θ_T	l_T

Table 1 shows the kinematic input parameters of all seven actuators. The positioning of the robot is solely dependent on these parameters.

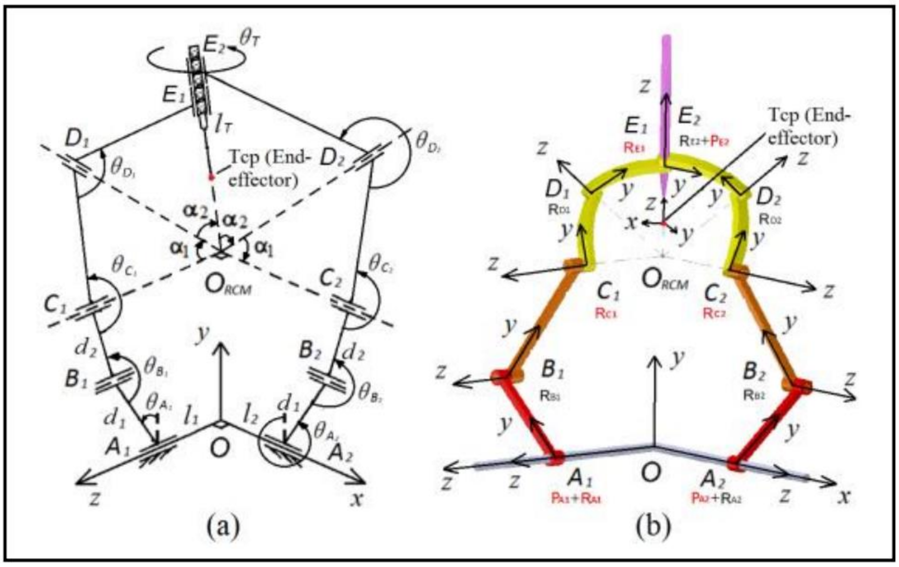


Figure 9 **(a)** Schematics of QUB surgical robot with kinematic parameters. **(b)** 3D model with kinematic parameters. [14]

Figure 9 illustrates all kinematic parameters. All angles are anticlockwise positive. For the O frame, the z axis is along OA_1, and the x axis is along OA_2. l_1 is the distance of OA_1; l_2 is the distance of OA_2; ϑ_{A1} is the angle from the y axis to axis A_1B_1; ϑ_{B1} is the angle from linkage A_1B_1 to linkage B_1C_1; ϑ_{C1} is the angle from linkage B_1C_1 to linkage C_1D_1; ϑ_{D1} is the angle from linkage C_1D_1 to linkage D_1E_1; ϑ_{A2} is the angle from y axis to axis A_2B_2; ϑ_{B2} is the angle from linkage A_2B_2 to linkage B_2C_2; ϑ_{C2} is the angle from linkage B_2C_2 to linkage C_2D_2; ϑ_{D2} is the angle from linkage C_2D_2 to linkage D_2E_2. d_1 is both the length of linkage A_1B_1 and linkage A_2B_2; d_2 is both the length of linkage B_1C_1 and linkage B_2C_2. r is all the length of $O_{RCM}C_1$, $O_{RCM}C_2$, $O_{RCM}D_1$, $O_{RCM}D_2$ and $O_{RCM}E_1$. α_0 is the angle from $O_{RCM}C_1$ to $O_{RCM}C_2$. α_1 is both the angle from $O_{RCM}C_1$ to $O_{RCM}D_1$, and the angle from $O_{RCM}C_2$ to $O_{RCM}D_2$; α_2 is both the angle from axes $O_{RCM}D_1$ to $O_{RCM}E_1$, and the angle from $O_{RCM}D_2$ to $O_{RCM}E_2$. The tool central point (Tcp/End-effector) is the tool tip point. l_T is the distance of E_1Tcp [14].

The Denavit-Hatenberg (D-H) frame assignment method was used to assign frames to the mechanism joints where the z axis of each frame points outward along the joint, and the x axis of each frame runs along the linkages that links the next joint.

The inverse kinematic model is used to drive three key points on the robot as seen in figure 9; O_{RCM}, E_1 and end effector (Tcp), which will be referred to as the tool tip throughout this report. RCM coordinates will be referred to in terms of the O coordinate system, while E_1 and Tcp in terms of the O_{RCM} coordinate system.

3.3 NST Position

The NST position, as seen in figure 8(a), serves as the home/initial point for the robot and has been pre-determined by the design team as seen in table 2.

Table 2 NST position parameters values.

Parameters	l_1	l_2	ϑ_{A1}	ϑ_{C1}	ϑ_{C2}	ϑ_T	l_T
Values	120 mm	120 mm	45 °	230 °	130 °	0 °	40 mm

As the parameter values are known, the forward kinematic model was used to determine the initial positions for the three key points O_{RCM}, E_1 and Tcp. These values can be seen in table 3.

Table 3 NST position O_{RCM}, E_1 and Tcp coordinates.

	Value/mm		**Value/mm**		**Value/mm**
x (O_{RCM})	0	**x (E_1)**	49.221	**x (Tcp)**	21.095
y (O_{RCM})	98.995	**y (E_1)**	49.221	**y (Tcp)**	21.095
z (O_{RCM})	0	**z (E_1)**	7.388	**z (Tcp)**	3.169

4.0 Digital Twin Modelling Methodology

4.1 Defining Functionality

From the design specifications previously outlined, a set of required functions for the DT were defined. These functions can be separated into two groups, those that are mandatory for meeting the design specifications (primary), and those that provide extra quality of life and more flexibility to the user (secondary). The purpose of this separation was to ensure development focused primarily on meeting the design specifications. These functions are outlined as follows:

Primary Functions:

- O_{RCM} PTP driven motion
- Tcp PTP driven motion
- Parameter driven motion of all 7 parameters
- RCM lock capability
- NST function
- Haptic overwrite option button

Secondary Functions:

- E_1 PTP driven motion
- Time control option
- Pre-set parametric adjustment buttons
- Current position display data for key points and all parameters

4.2 Software Selection

There are many viable digital twin modelling software packages available on the market. igus® Robot Control Software is well renowned for its intuitive UI (user interface) and range of kinematic control. ROS (Robot Operating System) was also a consideration; a Linux based system commonly used across the robot hobbyist community and industry. Similarly, RoboDX is a software widely used throughout the manufacture industry when developing simulations for articulated robots and assembly lines. It is however locked behind a licence not available at QUB.

Solidworks is also commonly used for DT modelling. It has a built-in macro tool which allows the user to run code written in Microsoft's VBA that is able to directly interact with an assembly model. VBA allows for a fully customisable GUI and can accurately compute kinematic calculations. Flexibility was the biggest factor in software selection. The CAD model for the QUB Surgical Robot was developed in Solidworks and so it was the perfect choice for the digital twin as any change to a dimension in a part file by the design team would immediately update the assembly and thus, the DT.

4.3 GUI Design

An intuitive GUI is crucial when designing any digital twin as it serves as the only interactive link between the user and model. In the high pressure setting of an operation especially, a cluttered and unintuitive interface may cause a delay in a surgeon's decision-making ability. The final iteration of the GUI that has been developed for this model can be seen in figure 11. It was created using VBA's highly customisable UserForm feature. This feature creates a direct link between any user input to the main body of code.

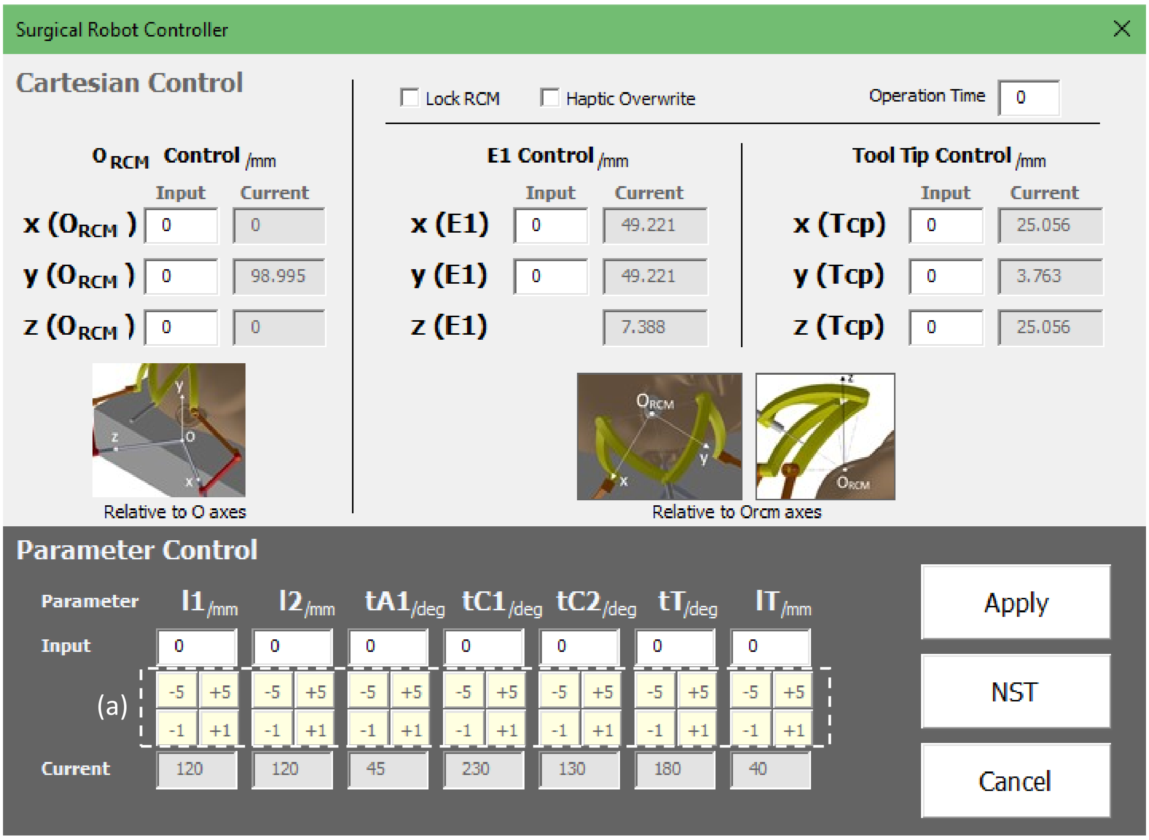


Figure 11 GUI at initial conditions. **a)** Parametric input buttons.

All cartesian and parameter controls are labelled and sectioned, along with user input boxes labelled 'Input' and current positions boxes labelled 'Current'. Graphics displaying the axes of motion have been included to aid the user in what axis each cartesian input refers to. All primary and secondary functions have been implemented into the GUI. For the individual parametric input buttons as seen in figure 11(a), values of +/-1 and +/-5 were chosen, however these can be edited within the code should the user want to further customise their function. The three key command buttons, Apply, NST and Cancel were positioned together and enlarged to further aid the user. These buttons function as follows:

Apply - Set all coded cartesian and parameter input variables to their respective user input.

NST - Run NST Function.

Cancel -Terminate the macro.

4.4 Range of Motion Limits

To prevent the user from breaking the model by inputting a position value outside of the range of motion of the robot, range of motion limits have been coded to alert the user and prevent the motion. These ranges for the three key points O_{RCM}, E_1 and Tcp, as well as all seven parameters have been outlined in tables 3 and 4 of Appendix A.

4.5 Functional Modules

The full VBA main code for the digital twin can be found in Appendix A.

In order to allow the VBA macro to interact with the Solidworks CAD model, a link was first established, as seen in figure 11. This code sets the active assembly model as the workstation for the DT, thus enabling any subsequent function to interact with its properties.

```
Set swApp = Application.SldWorks
Set Part = swApp.ActiveDoc
Set swModelDoc = swApp.ActiveDoc
Set swAssemblyDoc = swModelDoc
```

Figure 11 Defining Solidworks assembly as virtual workstation.

As discussed previously, the seven parameters determine the physical robot's position. For the DT these parameters have been defined as mates within the CAD model, and have been declared as variables DimL1, DimtL2, DimtA1, Dimt1, Dimt2, DimtT and DimlT, respective to their order in table 2. VBA is able to link these variables with their respective mate using the Solidworks property manager as seen in figure 12, with the example of the l_1 parameter.

```
Set CusPropMgr = swModelDoc.Extension.CustomPropertyManager("")
Set DimL1 = Part.Parameter("D1@L1m")
```

Figure 12 Link l_1 parameter mate to DimL1 variable.

After establishing the link, the parameter variables are set to the NST conditions, the model is updated and the GUI is created. These NST conditions are then uploaded onto the GUI in the corresponding current position fields. as seen in figure 13.

```
DimL1.SystemValue = 0.12                            (a)

Part.EditRebuild                                    (b)

myform.Caption = "Surgical Robot Controller"        (c)

myform.TextBox21.Value = 0.12                       (d)
```

Figure 13 **(a)** l_1 parameter set to NST position. **(b)** Model update. **(c)** GUI created. **(d)** l_1 current position set in GUI.

This section of the code will be referred to as function 1 (F1) and can be represented in flow form as seen in figure 14.

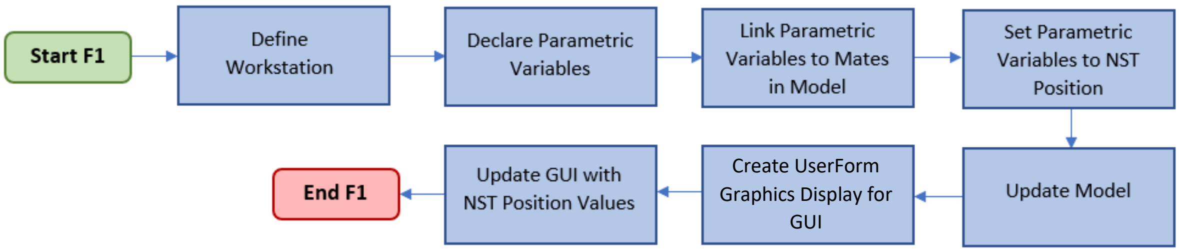


Figure 14 Function 1 logic - define workstation and initial position.

A function to detect if the RCM lock is currently active was then developed. An IF Statement is used to detect this from an option box on the GUI. If it is enabled, function 2 (as seen in figure 15) will overwrite any input to the parameters l_1, l_2 and ϑ_{A1}, as well as any cartesian inputs for x, y or z (O_{RCM}). The overwrite sets these values to 0, thus preventing RCM motion.

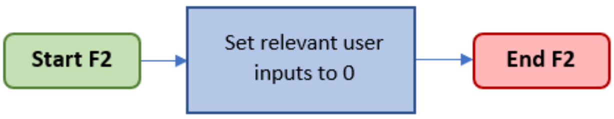


Figure 15 Function 2 logic – RCM lock detection.

A function for cartesian driven O_{RCM} motion was then developed. An IF Statement is used to detect any values in any of the three O_{RCM} cartesian inputs. If true, then function 3 (as seen in figure 16) will commence and those values will be converted to SI units. Inverse kinematic calculations will then be carried out to compute the new parameter values for l_1, l_2 and ϑ_{A1}. With these new values, the digital twin O_{RCM} motion will be carried out and the model updated.

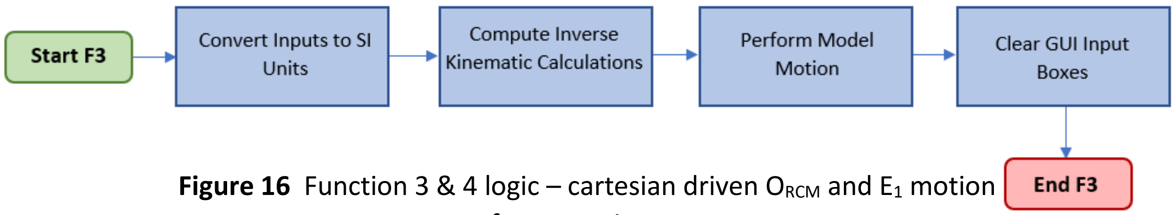


Figure 16 Function 3 & 4 logic – cartesian driven O_{RCM} and E_1 motion from user input.

Similar to the cartesian O_{RCM} input detection, an IF Statement is used. However instead of checking the RCM boxes, function 4 is checking the x and y (E_1) input boxes. It is also computing a different set of inverse kinematic equations in order to determine the new parameter values for ϑ_{C1} and ϑ_{C2}. In terms of flow logic representation, functions 3 & 4 are identical.

A function for parameter driven motion was then developed. As discussed previously, in order to drive the digital twin, only parameter values are needed. If the user were to input a desired change in parameter value into the GUI, the new parameter value can be calculated as the sum of the previous value and the change, as seen in figure 17. Therefore, unlike for cartesian functions 3 & 4, to achieve parametric motion, no kinematic computations are required.

$$\texttt{DimL1}_{new} = \texttt{DimL1}_{old} + ((\texttt{myform.L1inp} * 0.001) * n)$$

Figure 17 Parameter change in l_1 from parametric user input (for n =1).

Similar to functions 3 & 4, an IF Statement is used, model motion is performed, and the GUI inputs are cleared. Function 5 is outlined as seen in figure 18.

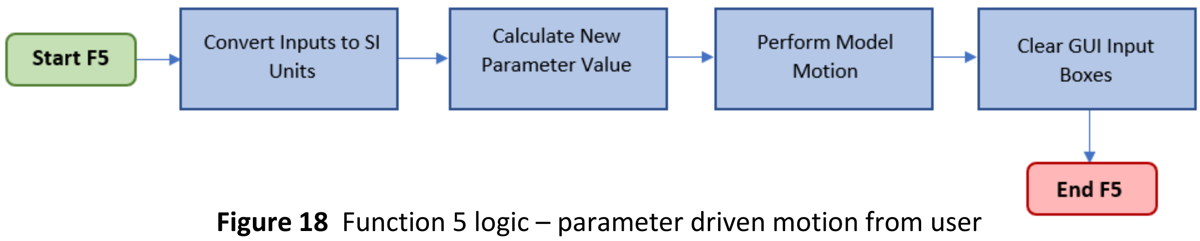


Figure 18 Function 5 logic – parameter driven motion from user input.

The parametric input command buttons, as seen in figure 11(a) are programmed similarly to function 5. However, they do not need to grab an input from the GUI as the values are predetermined within the code corresponding to their label. For the purpose of outlining the overall flow logic, this will be referred to as function 5.5 (as seen in figure 19).

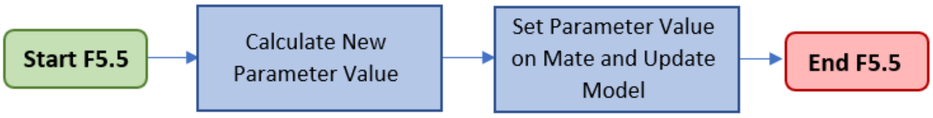


Figure 19 Function 5.5 logic – parameter driven motion from pre-set GUI command buttons.

An NST function was then developed. This function uses an IF Statement to detect activation of the NST command button on the GUI. If true, then it will calculate the necessary change in parameter values required in order to reach initial conditions. It achieves this by subtracting the current value from the pre-set initial value.

$$\mathrm{DimL1}_{new} = \mathrm{DimL1}_{old} + ((0.12 - \mathrm{DimL1}_{old}) * n)$$

Figure 20 Parameter change in l_1 to NST value (for n =1).

Thus, similar to function 5, no inverse kinematic computations are required, and the motion can be performed. However, should the user have the RCM locked, another check is required as to prevent the RCM from returning to its initial conditions. The logic for NST (function 6) is as outlined in figure 21.

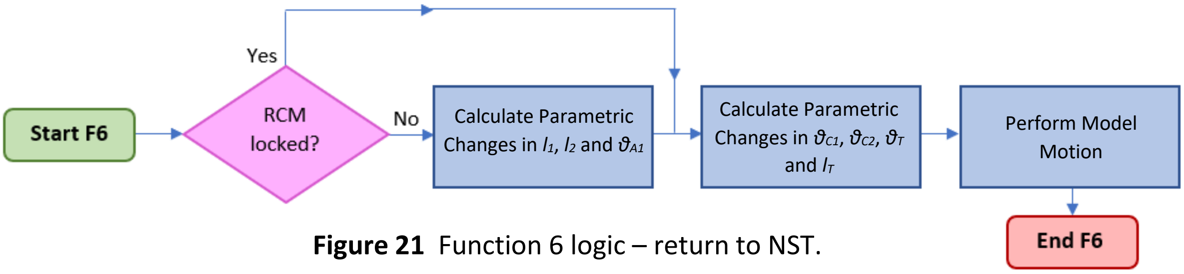


Figure 21 Function 6 logic – return to NST.

A function to update all current coordinate and parameter position to the GUI was then developed. The parameter values are read directly from the model mates and sent to the GUI. The coordinate values are obtained using the Solidworks coordinate reading feature and are then put in terms of their relevant axes using translation matrices. These updated positions are sent to the GUI, as described in figure 22 (Function 7).

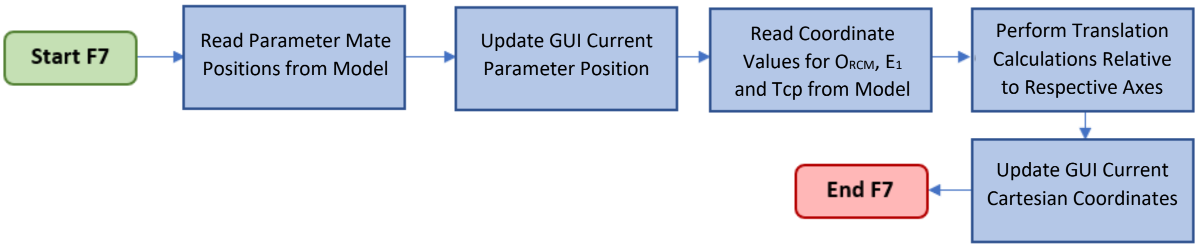


Figure 22 Function 7 logic – read & update current positions to GUI.

A function for cartesian driven Tcp motion was then developed. After the O_{RCM} and any E_1 motion is complete, the instrument can then be controlled. Motion is attained in the x, y and z (Tcp) axes through the use of vector translations and the same kinematic equations as used in function 4. An IF Statement is used to detect user input which is then used to calculate a direction vector which is then translated across the RCM to obtain the required E_1 position for the motion. This updated E_1 value is then used in kinematic calculations to determine the updated ϑ_{C1} and ϑ_{C2} parameters. The updated l_T configuration is also obtained from vector calculations. New current positions for x, y, z (tool) and l_T are then sent to the GUI. This is outlined in figure 23 (Function 8).

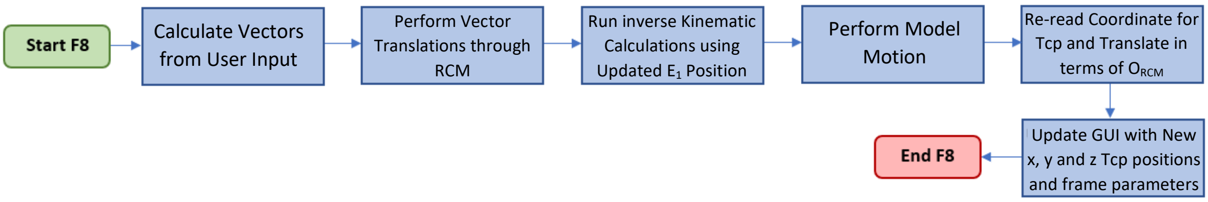


Figure 23 Function 8 logic - cartesian driven Tcp motion from user input.

4.6 Overall Digital Twin Logic

With the functional modules outlined, the overall logic of the twin can be described as seen in figure 24. IF Statements are used to detect any user command button presses and Boolean (binary true/false) variables are used to detect an NST or parameter increment activation. Figure 24(a) denotes now implementation of a haptic device could be carried out.

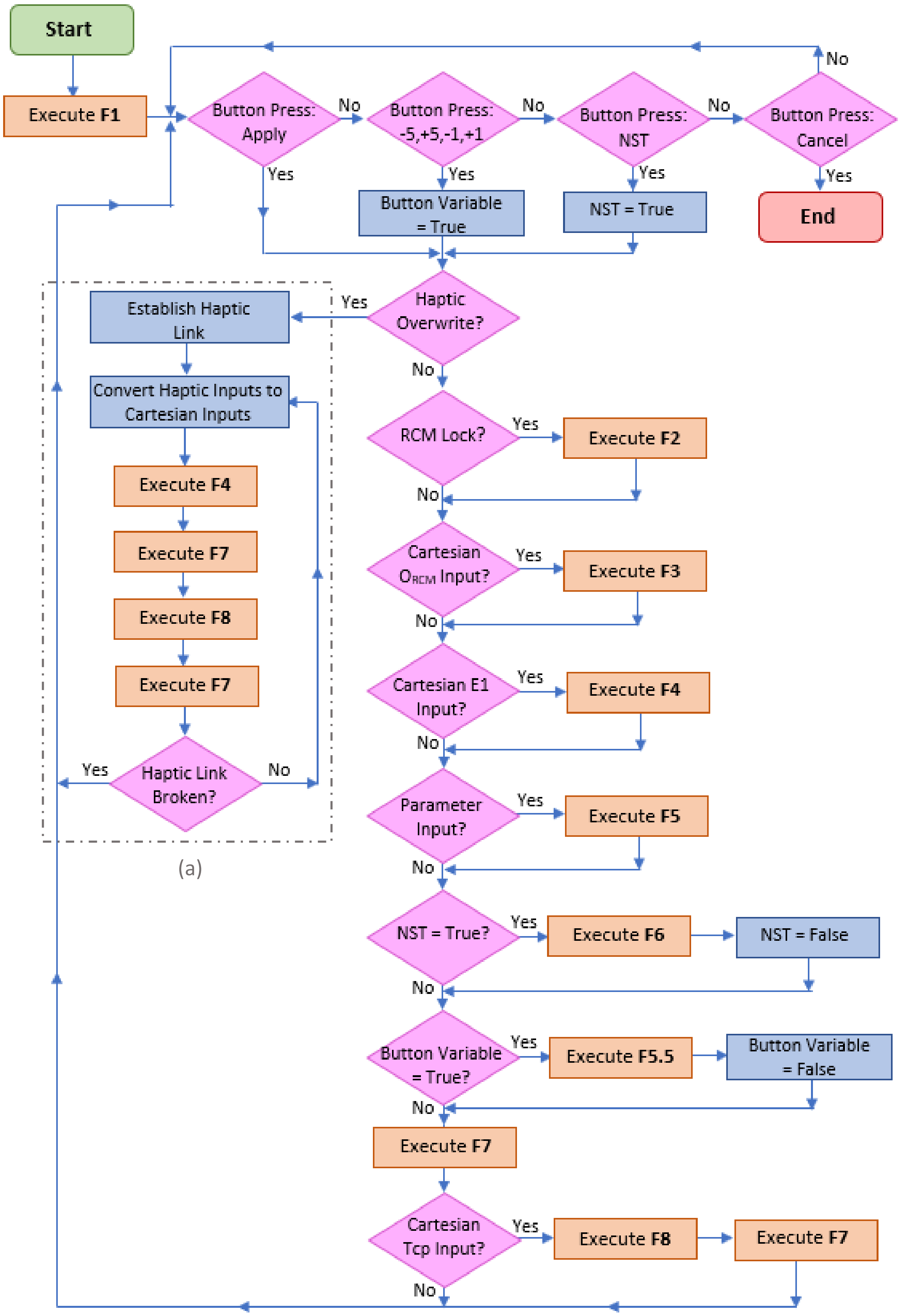


Figure 24 Digital twin logic. **a)** Potential haptic integration solution.

5.0 Digital Model Development

5.1 Motion Development

While directly updating the parameter mates on the model is sufficient to set the twin to its new position, without any intermediate steps, motion will not be simulated. To create such intermediate steps a VBA Step function was used with ten intervals, as seen in figure 25 for an example of l_1 driven motion.

```
For n = 0 To 1 Step 0.1                                  (a)
       DimL1new = DimL1old + ((L1 - DimL1old) * n)       (b)
       boolstatus = Part.EditRebuild3()                  (c)
       DoEvents
Next n
```

Figure 25 (a) Creating intervals. **(b)** l_1 parameter update. **(c)** Update model mate.

Figure 25(a) shows how the ten intervals are created. The initial position is defined as $DimL1_{old}$, the final position $DimL1_{new}$, the change L1-$DimL1_{old}$, and *n* the step variable. The change is incrementally added to the initial state (figure 25(b)) and the model is set to an intermediate state with a model update (figure 25(c)). The string of intermediate point updates between initial and final positions simulates the motion of the robot.

A single step loop can contain multiple parameter updates. For example, the DT motion for point O_{RCM} contains all three base parameters l_1, l_2 and ϑ_{A1}, meaning multi-axes motion can be simulated for any position in the workable area within one operation, in ten interval steps.

For theoretical haptic input motion, no intermediate steps would be required. New position coordinates for E_1 and Tcp would be sent from the device to the code, kinematic calculations would be computed, and parameter mates would be updated directly.

5.1.1 A Comment on Time

Wait variable, *w*, was added to each model update in order to allow the user to vary the total time taken to perform a movement operation between two points ($T_{Theorteical}$), with equation 1. The GUI input ladled "Operation Time" corresponds to $T_{Theorteical}$. This feature would only apply when using non-haptic input motion, as $T_{Theorteical}$ would equate to the time taken for the user to physically move the haptic device between the two points.

$$T_{Theorteical} = w * n \qquad \textbf{(1)}$$

Theoretically with infinite processing power and w = 0, both calculations and model updates would happen instantaneously. However, using a VBA Timer function, it was recorded that the time taken to complete both kinematic calculations and ten interval model updates was approximately 4.65 s on the computer that the digital twin was developed on. In contrast, on a more powerful computer, such as those in Ashby's CLB, this recorded time was only 0.67 s. Therefore, the true total time (T_{True}) is influenced by an error, $\mathcal{E}$, due to processing time. This relationship is described in equation 2.

$$T_{True} = T_{Theorteical} + \mathcal{E} \tag{2}$$

5.2 Point Position Reading

In order to determine the real-time positions of the three key points, a VBA GetPoint function is used. With the established link between the macro code and the CAD model, this reads points O_{RCM}, E_1 and Tcp based on their label on the feature tree and stores recorded coordinate data in a matrix array. However, the position data collected is relative to the Solidworks global origin (O_{SL}) axes and so must be translated to their relative axes O and O_{RCM}. For SL to O, translations were performed (180 ° about the x axis and α ° about the y axis. Translation matrix $^{SL}\boldsymbol{T}_O$ was calculated as seen in equation 3.

$$^{SL}_{O}\boldsymbol{T} = \begin{bmatrix} cos(\alpha) & 0 & -sin(\alpha) & 0 \\ 0 & -1 & 0 & 0 \\ -sin(\alpha) & 0 & -cos(\alpha) & 0 \\ ^{SL}_{O}x & ^{SL}_{O}y & ^{SL}_{O}z & 1 \end{bmatrix} \tag{3}$$

Coordinates for O were recorded relative to O_{SL} were recorded from the model as x = 213.35, y = 11.96 and z = 197.15. Angle α is a constant defined as the angle between the base of the robot and the operating table. It was measured to be approximately 31.69 ° as seen in figure 26.

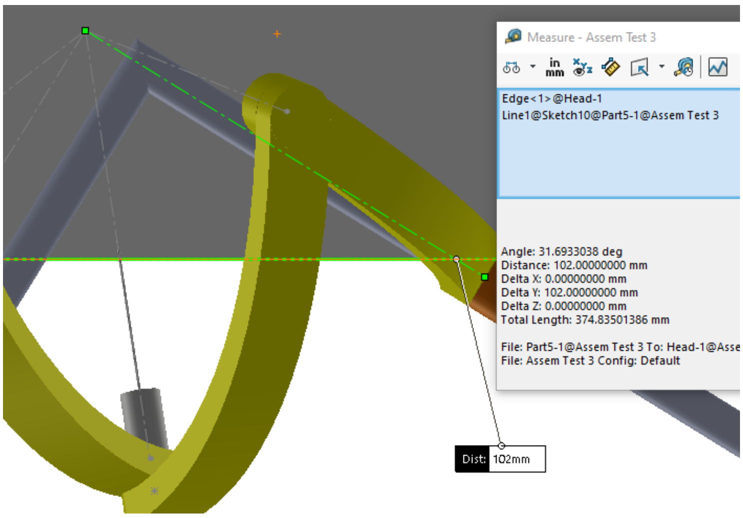


Figure 26 Measurement of α from CAD model.

Another translation was carried out to set position data for E_1 and Tcp relative to O_{RCM}. The translation matrix for ${}^{SL}\boldsymbol{T}_{ORCM}$ is as described in equation 4.

$$ {}^{SL}_{Orcm}\boldsymbol{T} = \begin{bmatrix} -sin(\alpha) & 0 & -cos(\alpha) & 0 \\ 0 & -1 & 0 & 0 \\ cos(\alpha) & 0 & -sin(\alpha) & 0 \\ {}^{SL}_{Orcm}x & {}^{SL}_{Orcm}y & {}^{SL}_{Orcm}z & 1 \end{bmatrix} \quad \textbf{(4)} $$

Following the coordinate translations, the data is sent to the GUI to display current positions and is used for vector and kinematic calculations for end effector manipulation.

5.3 Cartesian End Effector Manipulation

Figure 27 describes how for any movement of Tcp position within the negative x and y axes, the corresponding E_1 position can be determined with which inverse kinematic calculations will be carried out.

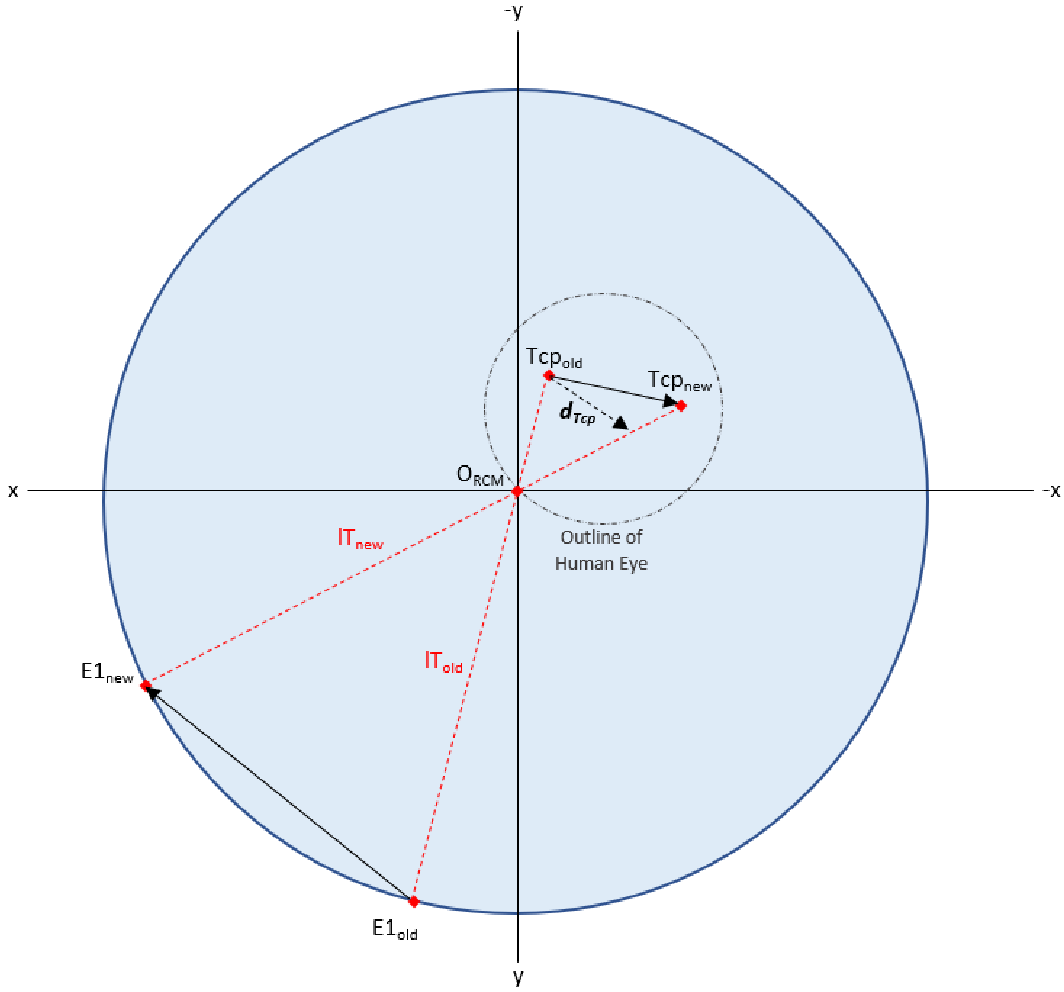


Figure 27 Determining E1$_{new}$ from change in tool tip position.

The magnitude of $\boldsymbol{Tcp_{old}}$ is determined prior by subtracting the value of parameter lT_{old} from the E_1 radius (70 mm). The magnitude of $\boldsymbol{Tcp_{old}Tcp_{new}}$ is the distance moved by the end effector. It is defined using equation 5 with the GUI cartesian inputs for x, y and z (Tcp).

$$|\boldsymbol{Tcp_{old}Tcp_{new}}| = \sqrt{x_{Tcp}{}^2 + y_{Tcp}{}^2 + z_{Tcp}{}^2} \qquad \textbf{(5)}$$

$\boldsymbol{Tcp_{new}}$ is found by summing $\boldsymbol{Tcp_{old}}$ with the x, y and z (tool) inputs. The magnitude of which is determined with equation 6.

$$|\boldsymbol{Tcp_{new}}| = \sqrt{(x_{old}^{Tcp} + x_{Tcp})^2 + (y_{old}^{Tcp} + y_{Tcp})^2 + (z_{old}^{Tcp} + z_{Tcp})^2} \qquad \textbf{(6)}$$

The change in l_T is determined in equation 7.

$$\boldsymbol{lT_{new}} - \boldsymbol{lT_{old}} = |\boldsymbol{Tcp_{new}}| - |\boldsymbol{Tcp_{old}}| \qquad \textbf{(7)}$$

In order to determine the position of $E1_{new}$, direction vector $\boldsymbol{E1_{old}E1_{new}}$ must be known. This is obtained by creating parallel vector $\boldsymbol{d_{Tcp}}$ and translating it through O_{RCM}. $\boldsymbol{d_{Tcp}}$ is defined in equation 8.

$$\boldsymbol{d_{Tcp}} = \left(\frac{\boldsymbol{Tcp_{new}}}{|\boldsymbol{Tcp_{new}}|} * |\boldsymbol{Tcp_{old}}|\right) - \boldsymbol{Tcp_{old}} \qquad \textbf{(8)}$$

The translation of $\boldsymbol{d_{Tcp}}$ to $\boldsymbol{E1_{old}E1_{new}}$ is defined in equation 9.

$$\boldsymbol{E1_{old}E1_{new}} = -\left(\frac{\boldsymbol{d_{Tcp}}}{|\boldsymbol{d_{Tcp}}|}\right) * 70 \qquad \textbf{(9)}$$

Finally, the coordinates of $E1_{new}$ can be found by summing $\boldsymbol{d_{Tcp}}$ with the coordinates of $E1_{old}$.

$$\begin{pmatrix} x_{new}^{E1} \\ y_{new}^{E1} \\ z_{new}^{E1} \end{pmatrix} = \boldsymbol{d_{Tcp}} + \begin{pmatrix} x_{old}^{E1} \\ y_{old}^{E1} \\ z_{old}^{E1} \end{pmatrix} \qquad \textbf{(10)}$$

This new $E1_{new}$ position is then used within the inverse kinematic model to return updated parameter values ϑ_{C1} and ϑ_{C1} [reference 14; equations 44 – 54].

6.0 Verification & Validation

6.1 Verification

Initial testing found that the coordinate reading for the tool tip position was inaccurate by approximately 0.001 mm, which although was above the acceptable error, it could potentially lead to larger inaccuracies across a motion cycle of several points. The Tcp within the model was defined as a reference point mated with the centre of the face of the instrument. It was found that upon any motion involving changes in the ϑ_{C1} and ϑ_{C2} parameters, the Tcp position on the updated model would be out of position. This is likely due Solidworks not being optimised for such small tolerances when motion is introduced. To eliminate this error, l_T parameter updates of + and - 0.01 nm were applied at the end of any motion operation as a means of refreshing the reference point. Table 3 highlights this error when the model was moved -8 ° and +8 ° from NST in the ϑ_{C1} and ϑ_{C2} parameters respectively.

Table 5 Examining Tool Tip Error from θ_1 and θ_2 Driven Motion

	Distance Between O_{RCM} and Tool Tip (mm)	**Percentage Error (%)**
Without Tool Tip Refresh	30.00147172	0.0049057
With Tool Tip Refresh	30.00000000	0.0000000

In order to verify the functionality of all functional modules, a simplified simulation of a cataract surgery was tested on the DT, as outlined in figure 29. This test was designed as to engage all seven parameters with both cartesian and parameter driven motion of O_{RCM} and Tcp, as well as to verify functionality of both the RCM lock and NST functions.

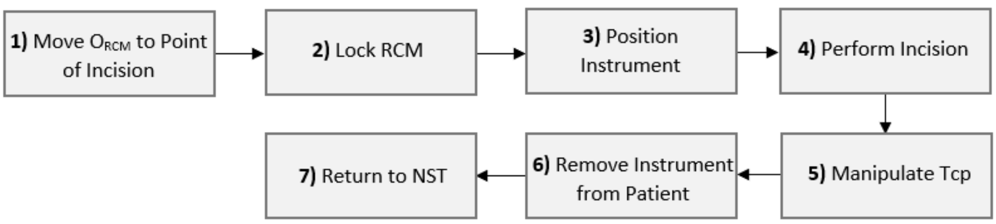


Figure 29 Simplified steps of a cataract surgery.

For this test, the point of incision was decided to be at coordinates (-46.8, 115.195, -45.6) relative to origin O. To make O_{RCM} concentric with the point of incision from NST, the change in position of O_{RCM} was calculated by subtracting the NST coordinates from the incision point coordinates. This change was found to be (-46.8, 16.2, -45.6) and was entered into the GUI input for x, y and z (O_{RCM}) respectively. Figure 30 highlights the relevant segment of the GUI and the DT, both prior and after the user inputs were applied for step 1.

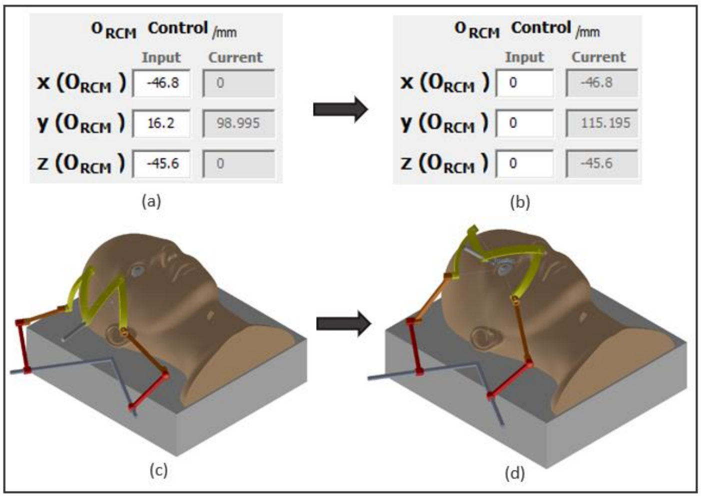


Figure 30 Verification of cartesian driven O_{RCM} motion. **(a)** GUI segment before applied input. **(b)** GUI segment after applied input. **(c)** DT before applied input. **(d)** DT after applied input.

Figure 30 demonstrates that the O_{RCM} motion calculations were completed correctly as the position of O_{RCM} was moved equal to the coordinates of the previously defined point of incision.

The RCM lock feature was verified in step 2. For this step, the RCM lock was engaged and arbitrary values of +10 were entered into both the cartesian control for x, y and z (O_{RCM}) as well as the parameter control for l_1, l_2 and ϑ_{A1}, ensuring the verification that for any input that drives O_{RCM}, no motion occurs. Figure 31 highlights both relevant segments of the GUI and the DT, both prior and after the user inputs were applied.

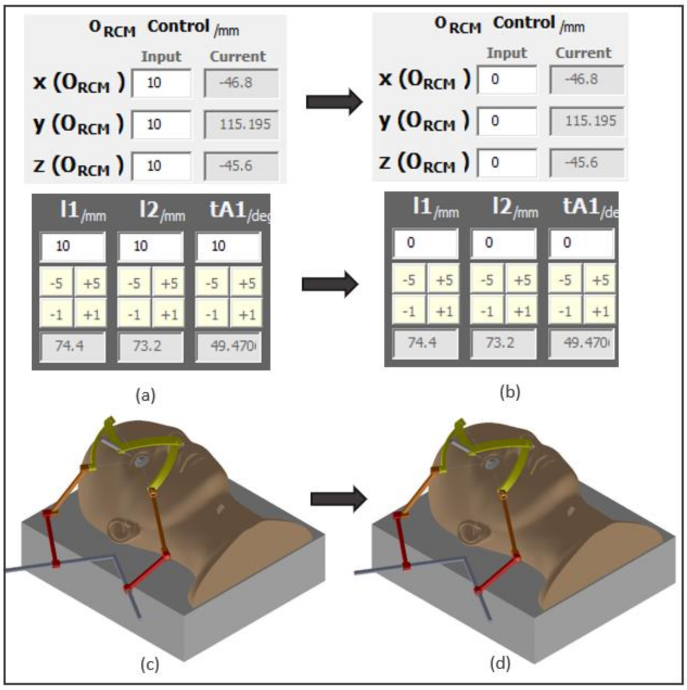


Figure 31 Verification of RCM Lock. **(a)** GUI segments before applied input. **(b)** GUI segments after applied input. **(c)** DT before applied input. **(d)** DT after applied input.

Figure 31 demonstrates that no motion occurred as the current position values of both O_{RCM} and parameter values remain the same after input was applied.

Parameter driven motion of the instrument was verified in steps 3 & 4. Arbitrary values of -15 ° and +15 ° were input into the GUI for ϑ_{C1} and ϑ_{C2} respectively to position the instrument. The incision was then performed with a +35 mm parameter input for l_T. Figure 32 highlights the relevant segments of the GUI and the DT, both prior and after the user inputs were applied.

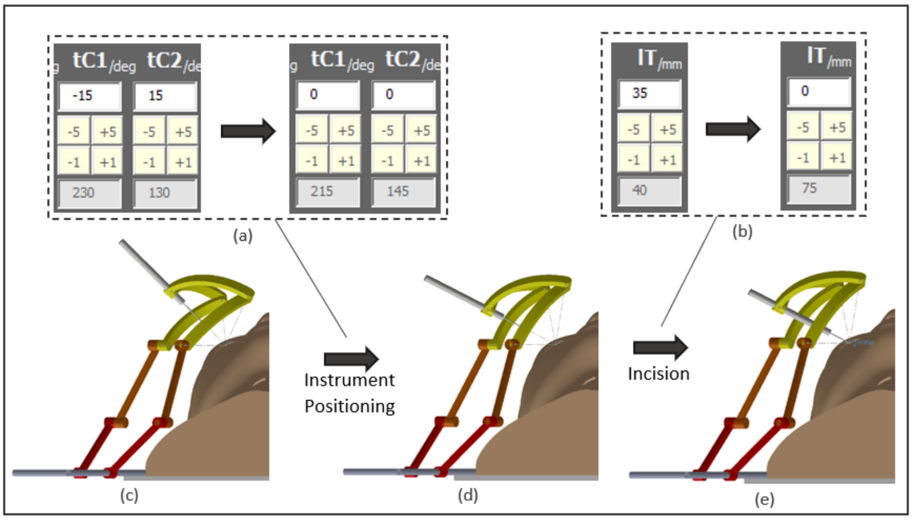


Figure 32 Verification of parameter driven motion. **(a)** GUI segments before & after instrument positioning. **(b)** GUI segments before & after incision. **(c)** DT before instrument positioning. **(d)** DT before incision. **(e)** DT after incision.

Figure 32 demonstrates that the DT parameter priven motion was applied correctly for steps 3 & 4 as the updated current parameter values are equal to the sum of previous parameter values and their respective inputs.

Cartesian driven Tcp motion was verified in step 5. Arbitrary values of -8 mm were input into the GUI for x, y and z (Tcp). Figure 33 highlights the relevant segments of the GUI and the DT, both prior and after the user inputs were applied.

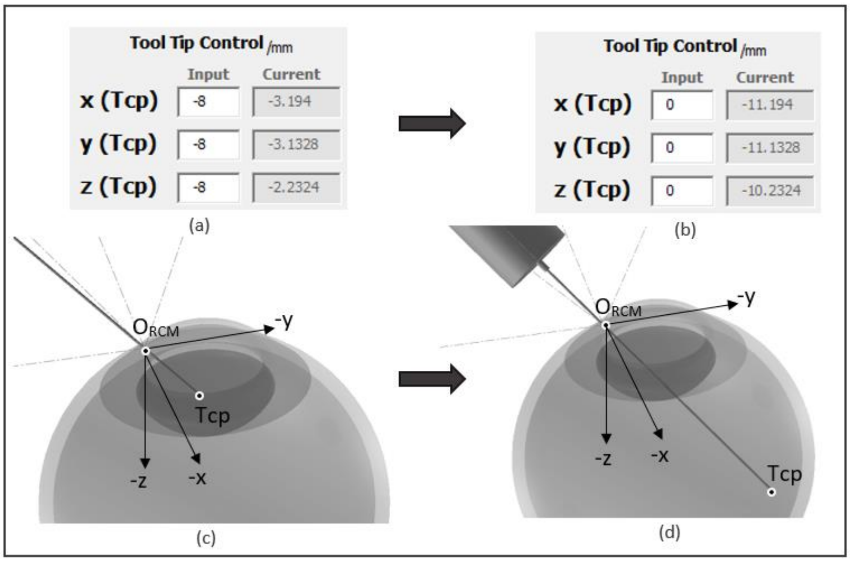


Figure 33 Verification of cartesian driven Tcp motion. **(a)** GUI segment before applied input. **(b)** GUI segment after applied input. **(c)** DT before applied input. **(d)** DT after applied

Figure 33 demonstrates that the cartesian driven Tcp motion calculations for step 5 were completed correctly as the updated current coordinates of Tcp values are equal to the sum of previous coordinates and their respective inputs.

Step 6 involves another parameter driven change in l_T to remove the instrument from the eye. This type of motion operation has been previously verified in step 4 however, for the clarity of the simulated surgery, the input value for l_T for this step was -50 mm.

Finally, the NST function was verified after the removal of the instrument. Figure 34 shows the current coordinates of O_{RCM}, E_1 and Tcp in a segment of the GUI as well as the DT, both prior and after the NST button was pressed.

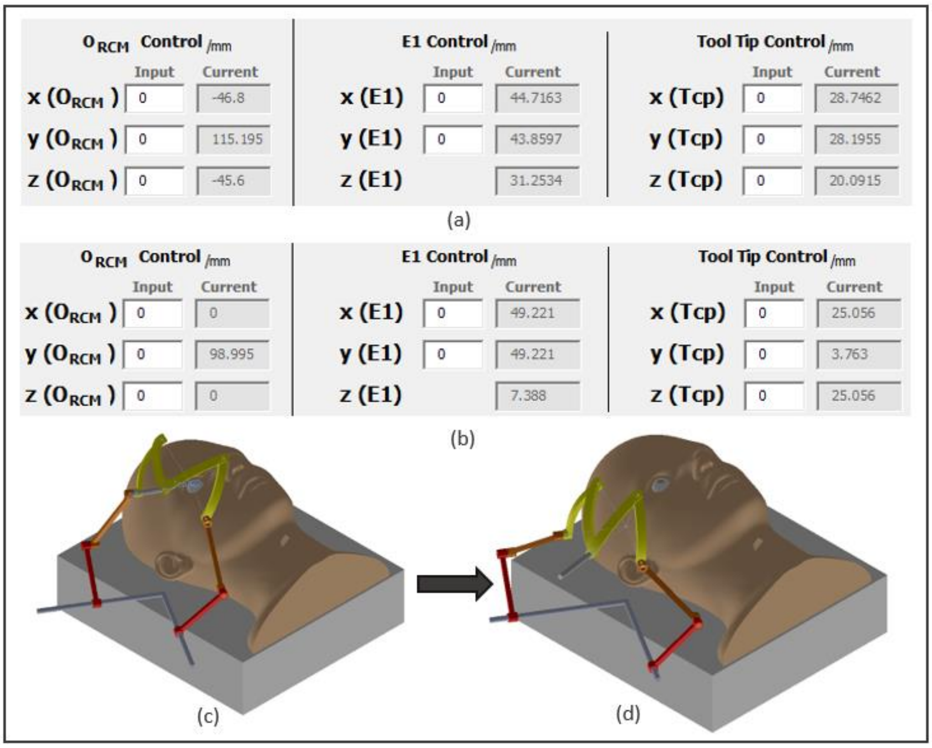


Figure 34 Verification of NST function. **(a)** GUI segment before NST activation. **(b)** GUI segment after NST activation. **(c)** DT before NST activation. **(d)** DT after NST activation.

Figure 34 demonstrates that the NST function is working correctly as the updated current coordinates of O_{RCM}, E_1 and Tcp are all equal to the NST position coordinates previously outlined in section 3.3.

After full successful verification a simple surgery cycle, it was at this point that any compilation bugs in the code were fixed, the code was condensed as much as possible and relevant annotations were added as to aid any future developers of this digital twin in understanding its operation.

6.2 Haptic Integration Potential

The design team of the robot requested that this digital twin be compatible with a haptic input device. To accommodate this, certain design considerations were made.

Firstly, a haptic device overwrite option was added to the GUI to allow the user to switch to haptic control on demand. This is useful to the surgeon as it allows them to first accurately locate the point of incision (O_{RCM}) mathematically using cartesian inputs, then switch to haptic control to make an incision and perform the surgery with scaled hand motion.

Figure 35 (a) 3D SYSTEMS® Touch™ Haptic Device. **(b)** Defining key coordinates virtually on haptic device.

The haptic device that is to be linked to the digital twin is a 3D SYSTEMS® Touch™, as seen in figure 35(a). Using Visual Studio, virtual boundaries can be defined within the device that provide the user haptic feedback or restrict motion entirely [15]. Point coordinates can also be defined relative to the posture of the device as seen in figure 35(b). The haptic end effector acts as a Tcp and the workable area of E_1 can be defined on a virtual sphere. Relative coordinate data of these points can also be read by the drivers and sent to an MS Access or SQL (structured query language) database, which can then be accessed by the digital twin macro. In theory, this solution also enables the physical robot to move in sync with the DT as coordinate data can simultaneously be accessed by the physical robot, as seen in figure 36.

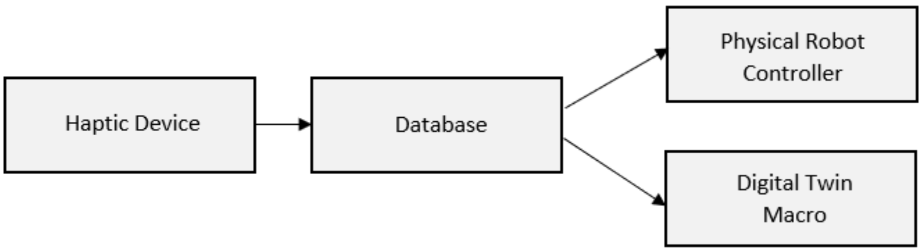


Figure 36 How haptic Input could drive the physical robot and digital twin simultaneously.

Before selecting Solidworks as the base for the digital twin, it was first confirmed that it was possible to link a Solidworks macro with such a database. In fact, VBA is able to access both types of databases. However, MS Access is optimal as it can be read with fewer permissions and is less resource intensive [16].

Figure 24(a) highlights the proposed solution to haptic integration within the VBA macro logic.

6.3 RCM and Tool Tip Accuracy Testing

All variables within the VBA macro used for calculations are known as Double variables. This form of variable can hold up to 14 decimal places, consuming only 8 bytes of system memory. While this may appear almost perfectly accurate for single calculation operations, the digital twin macro involves multi-stage calculations such as the previously described translations and inverse kinematics. This may cause an increased inaccuracy beyond the designer's requested error tolerance of within 20 microns (0.02 mm).

Two tests were designed to measure inaccuracies in both the cartesian driven O_{RCM} and Tcp motion. Parameter driven motion can be assumed to be more accurate as there are no kinematic calculations required and thus were not tested. In both tests, the model was first set to its NST position and motion was carried out using the GUI. Coordinates were then read and translated from the model using the coordinate reading function previously outlined. These values were then compared against the expected values – the sum of the NST coordinates and the cartesian GUI inputs. As these values are obtained in the virtual world and are solely the product of calculations with one solution, the test was not repeated for each point as repeatability would yield the same result.

For testing the O_{RCM} PTP motion accuracy, three points on the extremities of the y, -x and -z axes from the origin were chosen (a, c and d respectively) in order to test tolerances on each axis in isolation. An additional point on the extremities of the -x, -y, and -z axes was chosen (b) to examine the tolerance of the combined axes. Figure 37 outlines these points. The model was reset to NST by restarting the macro to setup for the next point.

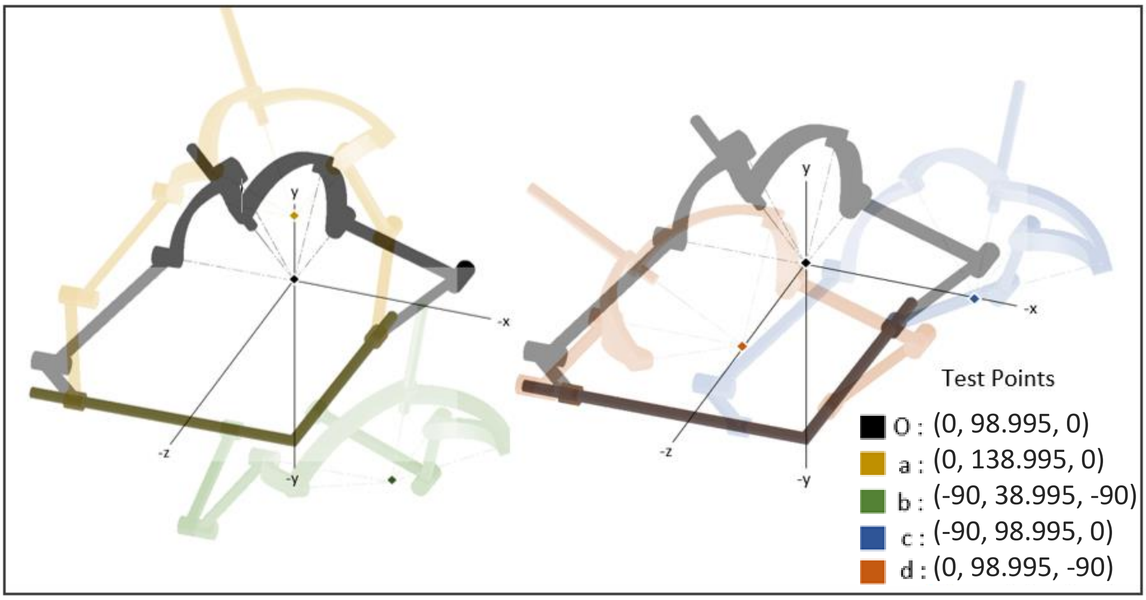


Figure 37 RCM positions at extremity points for tolerance testing.

To test the accuracy of the Tcp PTP motion, a simplified 8-point trajectory was designed to mimic the tool tip trajectory of the breaking and removal of a lens during manual cataract surgery. Unlike the O_{RCM} motion accuracy test, the macro was not reset in between each point in order to examine any accumulative error that may be more apparent during a full cycle of 7 motion operations. Figure 38 depicts this trajectory.

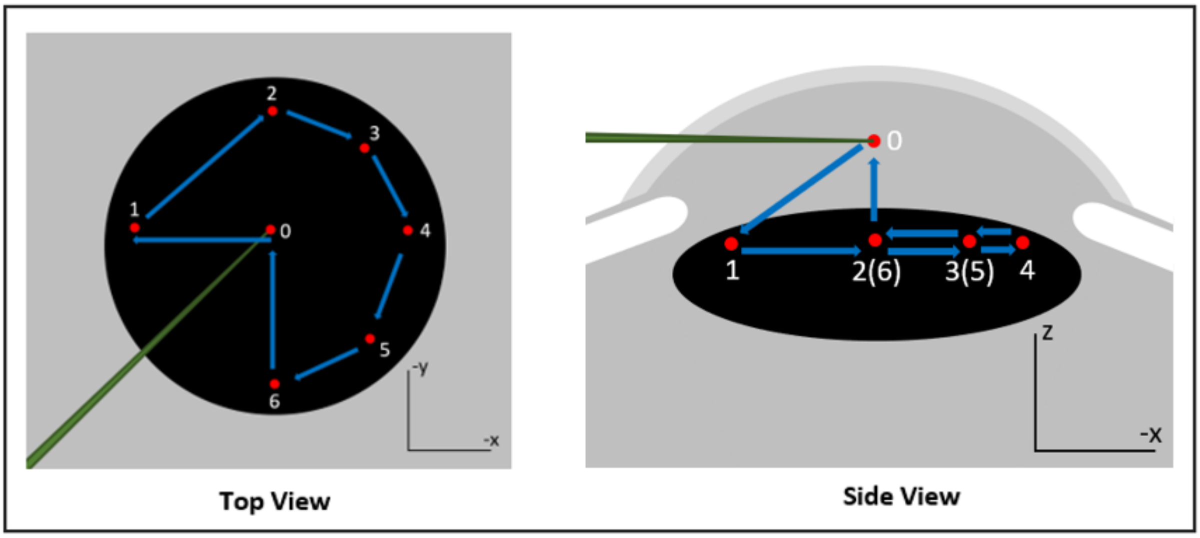


Figure 38 8-Point trajectory of Tcp for tolerance testing.

Initially the tool tip was set from the NST point to point 0 of the trajectory. To do this, the values -8.3984 and +8.3984 were input into the GUI for parameter motion in ϑ_{C1} and ϑ_{C2} respectively, setting the instrument horizontal to the base. It was the request of the designer that any tool tolerance test be conducted at 10 mm from the RCM as this is general estimate of the horizontal distance between the RCM and the centre point of the lens for a typical cataract surgery. Therefore, the parameter l_T was set to 80 mm. The average diameter of a human lens is approximately 10 mm [17], thus the radius of centre point 0 to all other points was set at 5 mm.

It was predicted that the error value of Tcp PTP motion would be greater than that of O_{RCM} PTP motion, as calculations for the Tcp position are more complex and the model was not reset to NST before each point motion.

The maximum combined error of O_{RCM} and Tcp motion were found to be 6.111e-6 µm and 2.985e -5 µm respectively. While the prediction was correct, both values are magnitudes smaller than the acceptable error of 20 µm and thus the digital twin motion can be assumed perfect, with negligible error. Full tabular results to both tests can be found in tables 6 and 7 of Appendix A.

7.0 Discussion

7.1 Potential Issues to be Considered

The rate at which current position data is sent from the haptic device to the DT is yet to be determined. As the time taken to complete a motion operation is dependent on a processing time error ($\mathcal{E}$) – as previously outline in equation 2 – when a haptic link is established, the time interval (i) between each current position data received by the twin cannot exceed this value ($\mathcal{E}$) to allow time for the model to update before it receives its next input, else the digital twin will lag behind the haptic input. Therefore, the rate at which haptic data is collected by the drivers should be less than or equal to the maximum rate at which the digital can process the data. This may cause the simulation to stutter, however for a powerful machine, $\mathcal{E}$ is low (<0.1 s) and thus i can be set at a decreased value (<0.1 s).

Additionally, although a solution to haptic integration was proposed, and the digital twin was designed to accommodate it, such a link is yet to be established and proof of concept should be the subject of further development.

7.2 Future Development

Once a haptic link to the DT has been established, further avenues of software development can be pursued. Firstly, a scale down control option can be implemented. This would for example convert a 10 mm movement of the haptic device to a 1 mm movement in the DT, allowing for increased precision during the surgery. To further increase precision a tremor mitigation algorithm could be developed, using position data from the haptic device and correcting any tremors in the motion before sending the data to the DT or the physical robot.

Virtual boundaries within the haptic space could be defined as such to give the user a vibration alert if the Tcp left the virtual space of the lens of the eye, ensuring the surgeon is immediately aware of a potential mistake in their movement.

A link between the DT and physical prototype could also be established, allowing the physical robot to be driven by the GUI and allowing real-time motion comparison between the real and virtual worlds.

7.3 Sustainability

This project touches on two of the three fundamental pillars to sustainability: social equality and economic development. Primarily, the driving force behind this project is to help the growing number of people in the world who suffer from cataracts. As fitting with UN sustainability goal 3 [18], a digital twin of a surgical robot for cataracts surgery was developed as a solution to reduce the training time and skill level needed for an eye surgeon to perform this surgery, filling the demand for trained eye surgeons, thus granting lower and middle-income countries access to the much need procedure.

Filling the demand for trained eye surgeons not only helps those on the waiting list for cataracts, but also helps the economy, in line with UN sustainability goal 7 [18]. With the equivalent loss of over 255 million full-time jobs from the COVID-19 pandemic, decreasing entry requirements for such an in-demand job allows for more eligible applicants, increasing employment within the healthcare sector.

8.0 Conclusions

The digital twin developed within this report has met all specifications set by the QUB prototype designer and encompasses all previously outlined primary and secondary functions.

- Fully controllable cartesian driven PTP motion of O_{RCM}, E_1 and Tcp within their respective workable area.
- O_{RCM} and Tcp motion maximum combined error less than 20 microns of 6.111e-6 µm and 2.985e -5 µm respectively.
- Fully controllable parameter driven motion.
- Pre-set parameter controls to allow posture change without motion simulation.
- Intuitive GUI linked to VBA macro with labelled boxes for user input along with real time position information of O_{RCM}, E_1, Tcp and all seven parameters.
- Theoretical total time to complete motion is variable from the GUI.
- Additional features include NST function, RCM lock and haptic overwrite.
- Developed to accommodate the proposed solution to integration of 3D SYSTEMS Touch™ haptic input device.

References

1. **Martin, E.** *Average cataract surgery wait time increases in NHS.* **Healio – Ophthalmology News Journal, (2021).** [online] Available at: https://www.healio.com/news/ophthalmology/20211007/average-cataract-surgery-wait-time-increases-in-nhs#:~:text=The%20median%20wait%20time%20for,specific%20procedures%20being%2011%20weeks.
2. **Sheetz, K.H., Claflin, J. and Dimick, J.B.** *Trends in the Adoption of Robotic Surgery for Common Surgical Procedures.* **JAMA Network Open, (2020).** [online] Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6991252/
3. **Dietl, R., Levene, J.B. and Dominique, D.** *Clinical Case Series with Post-Operative CT Analysis of Spine Surgery with Miniature Robotic Guidance.* **(2008).** [E-Poster presented at the 3rd German Spine Conference (DWG)].
4. **George, E.I., Brand, T.C., LaPorta, A., Marescaux, J. and Satava, R.M.** *Origins of Robotic Surgery: From Scepticism to Standard of Care* **JSLS Journal of The Society of Laparoscopic & Robotic Surgeons, (2018).** [Volume 22, Issue 4].
5. *Wold first for robot eye operation* **Oxford University – News and Events, (2016)** [online] Available at: https://www.ox.ac.uk/news/2016-09-12-world-first-robot-eye-operation
6. **Ferguson, S.** *Apollo 13: The First Digital Twin* **SIEMENS-Blog, (2020).** [online] Available at: https://blogs.sw.siemens.com/simcenter/apollo-13-the-first-digital-twin/
7. **Grieves, M.** *Digital Twin: Manufacturing Excellence through Virtual Factory Replication* **ResearchGate, (2015).** [online] Available at: https://www.researchgate.net/publication/275211047_Digital_Twin_Manufacturing_Excellence_through_Virtual_Factory_Replication
8. **Löcklin, A., Müller, M., Jung, T. and Jazdi, N.** *Digital Twin for Verifiaction and Validation of Industrial Automation Systems -a Survey* **ResearchGate, (2020).** [online] Available at: https://www.researchgate.net/publication/344452105_Digital_Twin_for_Verification_and_Validation_of_Industrial_Automation_Systems_-a_Survey
9. *Accenture Digital Health Technology Vision 2021 – Leaders Wanted* **accentuate, (2021).** [E-presentation] Available at: https://www.accenture.com/_acnmedia/PDF-156/Accenture-Digital-Health-Tech-Vision-2021.pdf#zoom=40
10. **Corral-Acero, J. *et al.*** *The 'Digital Twin' to enable the vision of precision cardiology* **European Heart Journal, (2020).** [online] Available at: https://academic.oup.com/eurheartj/article/41/48/4556/5775673

11. **Hagmann, K. *et al.*** *A Digital Twin Approach for Contextual Assistance for Surgeons During Surgical Robotics Training* **frontiers – in Robotics and AI, (2021).** [online] Available at: https://www.frontiersin.org/articles/10.3389/frobt.2021.735566/full
12. **Laaki, H., Miche, Y. and Tammi, K.** *Prototyping a Digital Twin for Real Time Remote Control Over Mobile Networks: Application of Remote Surgery* **IEEE, (2019).** [Volume 7, pp. 20325-20336]. [online] Available at: https://ieeexplore.ieee.org/abstract/document/8632888
13. **Hsu, J.K., Li, T. and Payandeh, S.** *On Integration of a Novel Minimally Invasive Surgery Robotic System* **IEEE, (2005).** [p 437-444]
14. **Jian, Y., Jin, Y., Price, M. and Moore, J.** *A new 7-degree-of-freedom 2-PRRRRR parallel remote center-of-motion robot for eye surgery* **8th IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics (BioRob), New York City, NY, USA, (2020).** [pp. 891-896]
15. *OpenHaptics Toolkit* **3DSYSTEM,** date accessed, **(2022).** [Version 3.3.0] [online] Available at: http://vizulab.com.au/wp-content/uploads/2017/08/OpenHaptics_ProgGuide.pdf
16. **Lysis.** *Microsoft Access Vs. SQL Server* **Techwalla,** date accessed, **(2022).** [online] Available at: https://www.techwalla.com/articles/microsoft-access-vs-sql-server
17. **Forrester, J., Dick, A., McMenamin, P. and Lee, W.** *The Eye : Basic Sciences in Practice* **London: W. B. Saunders Conpany Ltd.** [p 28] ISBN 0-7020-1790-6
18. *The 17 Goals* **UN, Department of Economic and Social Affairs,** date accessed, **(2022).** [online] Available at: https://sdgs.un.org/goals
19. **Meenink, H.C.M.** *Vitreo-retinal eye surgery robot : sustainable precision* **Eindhoven: Technische Universiteit Eindhoven, (2011).**

Appendix A

Range of Motion Limit Tables

Table 3 Table of range motion of positions for drivable points.

Point Name	Axis	Min Allowable Value (mm)	Max Allowable Value (mm)
O_{RCM}	x	-90	20
	y	38.995	138.995
	z	-90	20
E_1	x	0	70
	y	0	70
Tcp	x	-30	30
	y	-30	30
	z	-30	30

Table 4 Table of range of motion positions for parameters.

Parameter Name	Min Allowable Value	Max Allowable Value
l_1 (mm)	30	140
l_2 (mm)	30	140
ϑ_{A1} (°)	0	85
ϑ_{C1} (°)	100	330
ϑ_{C2} (°)	30	160
ϑ_T (°)	0	360
l_T (mm)	30	100

Tabular Results for Accuracy Testing

For both tests, combined error is calculated using equation 11.

$$\text{Combined Error} = \sqrt{x_{Error}^2 + y_{Error}^2 + z_{Error}^2} \qquad \textbf{(11)}$$

Table 6 Table of results for O_{RCM} driven motion accuracy testing.

Test Point	Axis	Expected Value	Attained Value	Error (µm)	Combined Error (µm)
a	x	0.000	-1.42E-13	-1.42E-10	1.4210855E-10
	y	138.995	138.995	0.00E+00	
	z	0.000	-1.28E-14	-1.28E-11	
b	x	-90.000	-90.000	4.32E-06	6.1111616E-06
	y	38.995	38.995	2.06E-10	
	z	-90.000	-90.000	-4.32E-06	
c	x	-90.000	-90.000	-1.99E-10	4.3213650E-06
	y	98.995	98.995	9.95E-11	
	z	0.000	-4.32E-09	-4.32E-06	
d	x	0.000	4.32E-09	4.32E-06	4.3209297E-06
	y	98.995	98.995	2.98E-10	
	z	-90.000	-90.000	9.95E-11	

Table 7 Table of results for Tcp driven motion accuracy testing.

Test Point	Axis	Expected Value	Attained Value	Error (µm)	Combined Error (µm)
1	x	-2.0710678122	-2.0710678245	1.2281220E-05	2.9857087E-05
	y	-7.0710678115	-7.0710678247	1.3222840E-05	
	z	-5.0000062556	-5.0000062318	-2.3786000E-05	
2	x	-7.0710678122	-7.0710678245	1.2249550E-05	2.9788737E-05
	y	-12.0710678115	-12.0710678247	1.3214050E-05	
	z	-5.0000062556	-5.0000062319	-2.3721430E-05	
3	x	-10.6066017182	-10.6066017303	1.2112901E-05	2.9582619E-05
	y	-10.3847046435	-10.3847046568	1.3255550E-05	
	z	-5.0000062556	-5.0000062321	-2.3509560E-05	
4	x	-12.2929648862	-12.2929648983	1.2102301E-05	2.9385046E-05
	y	-6.8491707375	-6.8491707507	1.3137869E-05	
	z	-5.0000062556	-5.0000062323	-2.3332630E-05	
5	x	-10.6066017182	-10.6066017303	1.2097900E-05	2.9252519E-05
	y	-3.3136368315	-3.3136368445	1.3013249E-05	
	z	-5.0000062556	-5.0000062324	-2.3238030E-05	
6	x	-7.0710678122	-7.0710678243	1.2102370E-05	2.9326457E-05
	y	-1.6272736635	-1.6272736766	1.3109829E-05	
	z	-5.0000062556	-5.0000062323	-2.3274580E-05	
7	x	-7.0710678122	-7.0710678243	1.2122390E-05	2.9555540E-05
	y	-6.6272736635	-6.6272736768	1.3240979E-05	
	z	-5.0000062556	-5.0000062321	-2.3478800E-05	
8	x	-7.0710678122	-7.0710678241	1.1876550E-05	2.9777213E-05
	y	-6.6272736635	-6.6272736770	1.3502150E-05	
	z	-0.0000062556	1.7478800E-02	-2.3734404E-05	

VBA Macro Main Code

Function 1 code **(lines 001-119)** - define workstation and initial position.

```
001	Option Explicit
002	Dim swApp As Object
003	Dim swAssemblyDoc As SldWorks.AssemblyDoc
004	Dim pIntMgr As SldWorks.InterferenceDetectionMgr
005	Dim swSelMgr As SldWorks.SelectionMgr
006	Dim swModelDoc As SldWorks.ModelDoc2
007	Dim swModelDocExt As SldWorks.ModelDocExtension
008	Dim Part, CurrentDoc As Object
009	Dim boolstatus As Boolean
010	Dim longstatus As Long, longwarnings As Long
011	Dim status As Boolean
012	Dim n, i, m, t, w, L1, L2, TA1, Lt
013	Dim TEST2 As Double
014	Dim swMathPt As SldWorks.MathPoint
015	Public xRCM, yRCM, zRCM, xE1, yE1, zTOOL, r1, Wait As Double
016
017	Sub main()
018	'Set SolidWorks Current Assembly as Active
019	    Set swApp = Application.SldWorks
020	    Set Part = swApp.ActiveDoc
021	    Set swModelDoc = swApp.ActiveDoc
022	    Set swAssemblyDoc = swModelDoc
023	'Setup the CustomPropertyManager to access and set Custom Properties
024	    Dim CusPropMgr As SldWorks.CustomPropertyManager
025	    Set CusPropMgr = swModelDoc.Extension.CustomPropertyManager("")
026	    Dim UpdateProp
027	'Setup the SelectionManager to select points on CAD model
028	    Dim selMgr As SelectionMgr
029	    Set selMgr = Part.SelectionManager
030	'Declare Variables and associate distance mates
031	    Dim DimL1, DimL2, DimTA1 As Object
032	    Set DimL1 = Part.Parameter("D1@L1m")
```

```
033          Set DimL2 = Part.Parameter("D1@L2m")
034           Set DimTA1 = Part.Parameter("D1@TA1m")
035
036           Dim Dimt1, Dimt2 As Object
037           Set Dimt1 = Part.Parameter("D1@t1m")
038           Set Dimt2 = Part.Parameter("D1@t2m")
039
040           Dim DimLt,DimtT As Object
041           Set DimLt = Part.Parameter("D1@Ltm")
042           Set DimtT = Part.Parameter("D1@tTm")
043
044       'Constants from CAD model
045               Dim d1,d2, PI As Double
046               d1 = 70 * 0.001
047               d2 = 70 * 0.001
048               PI = 3.14159265358979
049
050       'Create Motion Variables
051               Dim MotionL1, MotionL2, MotionTA1, Motiont1, Motiont2, MotionLt As Double
052
053       'NST Position Conditions [metres and radians]
054           'Base   [L1 = 120mm, L2 = 120mm, TA1 = 45deg]
055               r1 = 120 * 0.001
056               DimL1.SystemValue = r1
057               DimL2.SystemValue = r1
058               DimTA1.SystemValue = 45 * (PI / 180)
059
060               'Coordinates manually measured from model at NST
061               Dim xRCMold, yRCMold, zRCMold As Double
062               xRCMold = 0 * 0.001
063               yRCMold = 98.995 * 0.001
064               zRCMold = 0 * 0.001
065           'Frame   [Lt = 40mm, tT = 0deg]
066               Dim a0, a1, a2 As Double
067               a0 = PI / 2
068               a1 = (PI * 63) / 180
069               a2 = (PI * 69) / 180
070               Dim r2, r2metre As Double
071               r2 = 70 * 0.001
072               r2metre = r2 * 1000
073
074               'Values adjusted to represent CAD mates
075               Dimt1.SystemValue = (180 - 50) * (PI / 180)
076               Dimt2.SystemValue = (180 - 50) * (PI / 180)
077               DimLt.SystemValue = r2 - (10 * 0.001)
078               DimtT.SystemValue = PI
079
080               'Coordinates manually measured from model at NST
081               Dim xE1old, yE1old, zE1old As Double
082               xE1old = 49.22086905 * 0.001
083               yE1old = 49.22086905 * 0.001
084               zE1old = 7.388183083 * 0.001
085
086               'Variables for kinematic motion
087               Dim t1old, t2old As Double
088               t1old = -3.05452213081008
089               t2old = 3.05452236880807
090           'Tool   [Lt = 40mm, tT = 0deg]
091               Dim xTOOLold, yTOOLold, zTOOLold As Double
092               xTOOLold = 25.05551571 * 0.001
093               yTOOLold = 3.763438843 * 0.001
094               zTOOLold = 25.05551572 * 0.001
095           Part.EditRebuild
096
097       'Create GUI
098           Dim myform As New UserForm1
099           myform.Caption = "Surgical Robot Controller"
100
101       'Load Initial Parameter & Positions to GUI & change to mm/degrees
102           myform.TextBox7.Value = r1 * 1000
103           myform.TextBox15.Value = r1 * 1000
104           myform.TextBox16.Value = 45
105           myform.TextBox17.Value = 360 - (180 - 50)
106           myform.TextBox18.Value = (180 - 50)
107           myform.TextBox19.Value = 180
108           myform.TextBox20.Value = (r2 * 1000) - 30
109
110           myform.TextBox21.Value = 0
111           myform.TextBox22.Value = 98.995
112           myform.TextBox23.Value = 0
113           myform.TextBox24.Value = 49.221
114           myform.TextBox33.Value = 49.221
115           myform.TextBox25.Value = 7.388
116           myform.TextBox30.Value = Round((xTOOLold * 1000), 3)
117           myform.TextBox31.Value = Round((yTOOLold * 1000), 3)
118           myform.TextBox26.Value = Round((zTOOLold * 1000), 3)
119
120       '======= Creating an infinite loop for GUI staying on screen =======
121       Do While i = 0
122       i = 0
123       'Load GUI
124           myform.Show
125       '--------------------------Haptic Overwrite----------------------------
126           If myform.CheckBox2.Value = True Then
127           'Establish Haptic Link
128               MsgBox "Establish Haptic Link"
129           Else
130           End If
131       '--------------------------RCM Lock Check----------------------------
132           If myform.CheckBox1.Value = True Then
133
134           'Overwright RCM inputs
135               myform.xRCM = 0
```

Haptic Overwrite
(lines 125-130)

Function 2 code
(lines 131-156) –
RCM lock detection.

```
136             myform.yRCM = 0
137             myform.zRCM = 0
138             myform.L1inp = 0
139             myform.L2inp = 0
140             myform.tA1inp = 0
141 
142         'Overwrite RCM parameter inputs
143             myform.B4 = False
144             myform.B5 = False
145             myform.B6 = False
146             myform.B7 = False
147             myform.B8 = False
148             myform.B9 = False
149             myform.B18 = False
150             myform.B19 = False
151             myform.B20 = False
152             myform.B21 = False
153             myform.B22 = False
154             myform.B23 = False
155         Else
156         End If
157     '----------------Base Movement Inverse Kinematics---------------------
158         If myform.xRCM <> 0 Or myform.yRCM <> 0 Or myform.zRCM <> 0 Then
159 
160             'Converting Input to metres
161             Dim xRCMinc, yRCMinc, zRCMinc As Double
162             xRCMinc = myform.xRCM * 0.001
163             yRCMinc = myform.yRCM * 0.001
164             zRCMinc = myform.zRCM * 0.001
165 
166             'Old Coordinates + Change = New Coordinates
167             Dim xRCM, yRCM, zRCM As Double
168             xRCM = xRCMold + xRCMinc
169             yRCM = yRCMold + yRCMinc
170             zRCM = zRCMold + zRCMinc
171 
172         'Run Kinematic Calculations EQ42
173             L1 = zRCM + r1
174             L2 = xRCM + r1
175             Dim acos As Double
176 
177             'acos function using atn
178             Dim E As Double
179             E = (d1 ^ 2 - d2 ^ 2 + xRCM ^ 2 + yRCM ^ 2) / (2 * d1 * Sqr(xRCM ^ 2 + yRCM ^ 2))
180 
181                 If E = 1 Then
182                     acos = 0
183             ElseIf E = -1 Then
184                     acos = 4 * Atn(1)
185                 Else
186                     acos = Atn(-E / Sqr(-E * E + 1)) + 2 * Atn(1)
187                 End If
188                 If xRCM = 0 Then
189                     TA1 = acos
190                 Else
191                     TA1 = acos + Atn(yRCM / xRCM) - ((Abs(xRCM) * PI) / (2 * xRCM))
192                 End If
193 
194             'Part.EditRebuild
195             For n = 0 To 1 Step 0.1
196                 DimL1.SystemValue = DimL1.SystemValue + ((L1 - DimL1.SystemValue) * n)
197                 DimL2.SystemValue = DimL2.SystemValue + ((L2 - DimL2.SystemValue) * n)
198                 DimTA1.SystemValue = DimTA1.SystemValue + ((TA1 - DimTA1.SystemValue) * n)
199                 Part.ClearSelection2 True
200                 If myform.Time <> 0 Then
201                     myform.MotionTimer
202                 Else
203                 End If
204                 boolstatus = Part.EditRebuild3()
205                 DoEvents
206             Next n
207 
208         'Set new inital conditions for next input value
209             xRCMold = xRCM
210             yRCMold = yRCM
211             zRCMold = zRCM
212 
213         'Clear GUI
214             myform.TextBox1.Value = 0
215             myform.TextBox2.Value = 0
216             myform.TextBox3.Value = 0
217         Else
218         End If
219     '---------------------Frame Movement Inverse Kinematics------------------------
220         If myform.xE1 <> 0 Or myform.yE1 <> 0 Then
221 
222             'Converting Input to metres
223             Dim xE1inc, yE1inc As Double
224                 xE1inc = myform.xE1
225                 yE1inc = myform.yE1
226             Dim xE1metre, yE1metre, zE1metre As Double
227                 xE1metre = xE1old * 1000
228                 yE1metre = yE1old * 1000
229                 zE1metre = zE1old * 1000
230 
231             'Old Coordinates + Change = New Coordinates [+convert to ratio of r]
232             Dim xE1, yE1 As Double
233             xE1 = ((((r2metre) ^ 2 - (zE1metre) ^ 2 - (yE1metre + yE1inc) ^ 2) ^ 0.5) + xE1inc) / r2metre
234             yE1 = ((((r2metre) ^ 2 - (zE1metre) ^ 2 - (xE1metre + xE1inc) ^ 2) ^ 0.5) + yE1inc) / r2metre
235 
236         'Implementing Model from MatLab
237             Dim t3, t4, zE1 As Double
238             zE1 = Sqr(1 - xE1 ^ 2 - yE1 ^ 2)
```

Function 3 code **(lines 157-218)** – cartesian O_{RCM} driven motion from user input.

Function 4 code **(lines 219-306)** – cartesian O_{RCM} driven motion from user input.

```
239
240             Dim cost3 As Double
241             cost3 = (Cos(a1) * Cos(a2) - xE1) / (Sin(a1) * Sin(a2))
242             Dim F As Double
243                 If cost3 = 1 Then
244                     F = 0
245                 ElseIf cost3 = -1 Then
246                     F = 4 * Atn(1)
247                 Else
248                     F = Atn(-cost3 / Sqr(-cost3 * cost3 + 1)) + 2 * Atn(1)
249                 End If
250             t3 = -F
251
252             Dim cost1 As Double
253             cost1 = (yE1 * Sin(t3) * Sin(a2) - zE1 * (Cos(t3) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2))) / ((Sin(t3)
                *Sin(a2)) ^ 2 + (Cos(t3) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2)) ^ 2)
254             Dim sint1 As Double
255             sint1 = (zE1 * Sin(t3) * Sin(a2) + yE1 * (Cos(t3) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2))) / ((Sin(t3) *
                Sin(a2)) ^ 2 + (Cos(t3) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2)) ^ 2)
256             Dim cost4 As Double
257             cost4 = (Cos(a1) * Cos(a2) - Cos(a0) * xE1 - Sin(a0) * yE1) / (Sin(a1) * Sin(a2))
258             Dim G As Double
259                 If cost4 = 1 Then
260                     G = 0
261                 ElseIf cost4 = -1 Then
262                     G = 4 * Atn(1)
263                 Else
264                     G = Atn(-cost4 / Sqr(-cost4 * cost4 + 1)) + 2 * Atn(1)
265                 End If
266             t4 = G
267
268             Dim cost2 As Double
269             cost2 = (-xE1 * Sin(t4) * Sin(a2) - zE1 * Sin(a0) * (Cos(t4) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2)) +
                Cos(a0) * Sin(t4) * Sin(a2) * (Cos(a1) * Cos(a2) - Sin(a1) * Sin(a2) * Cos(t4))) / ((Sin(a0)) * (Sin(t4) *
                Sin(a2)) ^ 2 + (Sin(a0)) * (Cos(t4) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2)) ^ 2)
270             Dim sint2 As Double
271             sint2 = (zE1 * Sin(a0) * Sin(t4) * Sin(a2) - xE1 * (Cos(t4) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2)) +
                Cos(a0) * (Cos(t4) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2)) * (Cos(a1) * Cos(a2) - Sin(a1) * Sin(a2) *
                Cos(t4))) / ((Sin(a0)) * (Sin(t4) * Sin(a2)) ^ 2 + (Sin(a0)) * (Cos(t4) * Cos(a1) * Sin(a2) + Sin(a1) *
                Cos(a2)) ^ 2)
272
273             'Adjusting for current TA1 position
274             Dim t1 As Double
275             Dim t2 As Double
276             t2 = Atn(sint2 / cost2) + PI
277             t1 = Atn(sint1 / cost1) + PI
278
279             'Calculate t1&t2 change
280             If t1old < 0 Then
281                 t1old = t1old + (2 * PI)
282             Else
283             End If
284
285             'Part.EditRebuild
286                 Motiont1 = Dimt1.SystemValue
287                 Motiont2 = Dimt2.SystemValue
288             For n = 0 To 1 Step 0.1
289                 Dimt1.SystemValue = Motiont1 + ((t1old - t1) * n)
290                 Dimt2.SystemValue = Motiont2 + ((t2 - t2old) * n)
291                 Part.ClearSelection2 True
292                 If myform.Time <> 0 Then
293                     myform.MotionTimer
294                 Else
295                 End If
296                 boolstatus = Part.EditRebuild3()
297                 DoEvents
298             Next n
299         'Update t1&t2 old
300             t1old = t1
301             t2old = t2
302         'Clear GUI
303             myform.TextBox4.Value = 0
304             myform.TextBox32.Value = 0
305         Else
306         End If
307     '-------------------------Parameter Input Updates---------------------------
308         If myform.L1inp <> 0 Or myform.L2inp <> 0 Or myform.tA1inp Or myform.tC1inp <> 0 Or myform.tC2inp <> 0 Or
            myform.tTinp <> 0 Or myform.lTinp <> 0 Then
309
310         'Calculations & Part.EditRebuild
311                 MotionL1 = DimL1.SystemValue
312                 MotionL2 = DimL2.SystemValue
313                 MotionTA1 = DimTA1.SystemValue
314                 Motiont1 = Dimt1.SystemValue
315                 Motiont2 = Dimt2.SystemValue
316                 MotionLt = DimLt.SystemValue
317                 MotiontT = DimtT.SystemValue
318             For n = 0 To 1 Step 0.1
319
320                DimL1.SystemValue = MotionL1 + ((myform.L1inp * (0.001)) * n)
321                DimL2.SystemValue = MotionL2 + ((myform.L2inp * (0.001)) * n)
322                DimTA1.SystemValue = MotionTA1 + ((myform.tA1inp * (PI / 180)) * n)
323                Dimt1.SystemValue = Motiont1 - ((myform.tC1inp * (PI / 180)) * n)
324                Dimt2.SystemValue = Motiont2 + ((myform.tC2inp * (PI / 180)) * n)
325                DimLt.SystemValue = MotionLt + ((-myform.lTinp * (0.001)) * n)
326                DimtT.SystemValue = MotiontT + ((myform.tTinp * (PI / 180)) * n)
327
328             'CAD Update
329                 Part.ClearSelection2 True
330                 boolstatus = Part.EditRebuild3()
331                 DoEvents
332             Next n
333         'Clear GUI
```

Function 5 code **(lines 307-342**) – parameter driven motion from user input.

```
334             myform.TextBox8.Value = 0
335             myform.TextBox9.Value = 0
336             myform.TextBox10.Value = 0
337             myform.TextBox11.Value = 0
338             myform.TextBox12.Value = 0
339             myform.TextBox13.Value = 0
340             myform.TextBox14.Value = 0
341         Else
342         End If
343     '---------------------------NST Button Press-------------------------------
344         If myform.NST = True Then
345         myform.NST = False
346 
347         'Revert to NST Position & Part.EditRebuild
348             For n = 0 To 1 Step 0.1
349             'Base motion
350                 'Check for RCM Lock
351                 If myform.CheckBox1.Value = False Then
352 
353                     DimL1.SystemValue = DimL1.SystemValue + ((r1 - DimL1.SystemValue) * n)
354                     DimL2.SystemValue = DimL2.SystemValue + ((r1 - DimL2.SystemValue) * n)
355                     DimTA1.SystemValue = DimTA1.SystemValue + (((45 * (PI / 180)) - DimTA1.SystemValue) * n)
356                 Else
357                 End If
358 
359             'Frame Motion
360                 Dimt1.SystemValue = Dimt1.SystemValue + ((((180 - 50) * (PI / 180)) - Dimt1.SystemValue) * n)
361                 Dimt2.SystemValue = Dimt2.SystemValue + ((((180 - 50) * (PI / 180)) - Dimt2.SystemValue) * n)
362             'Tool Motion
363                 DimLt.SystemValue = DimLt.SystemValue + (((r2 - (10 * 0.001)) - DimLt.SystemValue) * n)
364                 DimtT.SystemValue = DimtT.SystemValue + (((PI) - DimtT.SystemValue) * n)
365             'CAD Update
366                 Part.ClearSelection2 True
367                 If myform.Time <> 0 Then
368                     myform.MotionTimer
369                 Else
370                 End If
371                 boolstatus = Part.EditRebuild3()
372                 DoEvents
373             Next n
374         Else
375         End If
376     '-------------------------Parameter User Button Updates--------------------------
377         'Check RCM Lock
378     If myform.CheckBox1.Value = False Then
379     'L1 -5,+5,-1,+1
380         If myform.B4 = True Or myform.B5 = True Or myform.B18 = True Or myform.B19 = True Then
381         myform.B4 = False
382         myform.B5 = False
383         myform.B18 = False
384         myform.B19 = False
385             DimL1.SystemValue = DimL1.SystemValue + myform.c
386             Update
387     'L2 -5,+5,-1,+1
388         ElseIf myform.B6 = True Or myform.B7 = True Or myform.B20 = True Or myform.B21 = True Then
389         myform.B6 = False
390         myform.B7 = False
391         myform.B20 = False
392         myform.B21 = False
393             DimL2.SystemValue = DimL2.SystemValue + myform.c
394             Update
395     'TA1 -5,+5,-1,+1
396         ElseIf myform.B8 = True Or myform.B9 = True Or myform.B22 = True Or myform.B23 = True Then
397         myform.B8 = False
398         myform.B9 = False
399         myform.B22 = False
400         myform.B23 = False
401             DimTA1.SystemValue = DimTA1.SystemValue + myform.c
402             Update
403         Else
404         End If
405     Else
406     End If
407     'tC1 -5,+5,-1,+1
408         If myform.B10 = True Or myform.B11 = True Or myform.B24 = True Or myform.B25 = True Then
409         myform.B10 = False
410         myform.B11 = False
411         myform.B24 = False
412         myform.B25 = False
413             Dimt1.SystemValue = Dimt1.SystemValue + (myform.c * -1)
414             Update
415     'tC2 -5,+5,-1,+1
416         ElseIf myform.B12 = True Or myform.B13 = True Or myform.B26 = True Or myform.B27 = True Then
417         myform.B12 = myform.B13 = myform.B26 = myform.B27 = False
418             Dimt2.SystemValue = Dimt2.SystemValue + myform.c
419             Update
420     'tT -5,+5,-1,+1
421         ElseIf myform.B14 = True Or myform.B15 = True Or myform.B28 = True Or myform.B29 = True Then
422         myform.B14 = myform.B15 = myform.B28 = myform.B29 = False
423             DimtT.SystemValue = DimtT.SystemValue + (myform.c * -1)
424             Update
425     'Lt -5,+5,-1,+1
426         ElseIf myform.B16 = True Or myform.B17 = True Or myform.B30 = True Or myform.B31 = True Then
427         myform.B16 = False
428         myform.B17 = False
429         myform.B30 = False
430         myform.B31 = False
431             DimLt.SystemValue = DimLt.SystemValue + (myform.c * -1)
432             Update
433         Else
434         End If
435     '-------------------------GUI Update Current Position--------------------------
436     'Parameter Updates
```

Function 6 code **(lines 343-375)** – return to NST

Function 5.5 code **(lines 376-434)** – parameter driven motion from pre-set GUI command buttons.

Function 7 code **(lines 435-502 & 707-733)** – read & update current positions to GUI.

```
437             myform.TextBox7.Value = DimL1.SystemValue * 1000
438             myform.TextBox15.Value = DimL2.SystemValue * 1000
439             myform.TextBox16.Value = DimTA1.SystemValue * (180 / PI)
440             myform.TextBox17.Value = 360 - Dimt1.SystemValue * (180 / PI)
441             myform.TextBox18.Value = Dimt2.SystemValue * (180 / PI)
442             myform.TextBox19.Value = 360 - DimtT.SystemValue * (180 / PI)
443             myform.TextBox20.Value = 70 - (DimLt.SystemValue * 1000) + 30
444
445     'Coordinate Updates
446         'RCM in terms of O
447             'Read Coord from model
448             CoordRCM
449
450             'Difference between RCM and O in terms of O[SL]
451             Dim xOSL, yOSL, zOSL As Double
452             xOSL = (swMathPt.ArrayData(0) * 1000)
453             yOSL = (swMathPt.ArrayData(1) * 1000)
454             zOSL = (swMathPt.ArrayData(2) * 1000)
455
456             Dim xDIF, yDIF, zDIF As Double
457             xDIF = xOSL - 213.353485562132
458             yDIF = yOSL - 11.9568330831327
459             zDIF = zOSL - 197.151841638515
460
461             'Apply Translation Matrix O[SL] to O
462             Dim xNEW, yNEW, zNEW As Double
463             xNEW = (xDIF * 0.850872515872724) + (yDIF * 0) + (zDIF * -0.525372212562123)
464             yNEW = (xDIF * 0) + (yDIF * -1) + (zDIF * 0)
465             zNEW = (xDIF * -0.525372212562123) + (yDIF * 0) + (zDIF * -0.850872515872724)
466
467             'GUI Update with rounded values
468             myform.TextBox21.Value = Round(xNEW, 4)
469             myform.TextBox22.Value = Round(yNEW, 4)
470             myform.TextBox23.Value = Round(zNEW, 4)
471
472             'Reset Inital State
473             xRCMold = xNEW * 0.001
474             yRCMold = yNEW * 0.001
475             zRCMold = zNEW * 0.001
476         'E1 in terms of Orcm
477             'Read Coord from model
478             CoordE1
479             'Difference between E1 and RCM in terms of O[SL]
480             Dim xOrcmSL, yOrcmSL, zOrcmSL As Double
481             xOrcmSL = (swMathPt.ArrayData(0) * 1000)
482             yOrcmSL = (swMathPt.ArrayData(1) * 1000)
483             zOrcmSL = (swMathPt.ArrayData(2) * 1000)
484             xDIF = xOrcmSL - xOSL
485             yDIF = yOrcmSL - yOSL
486             zDIF = zOrcmSL - zOSL
487
488             'Apply Translation Matrix O[SL] to Orcm
489             Dim xE1NEW, yE1NEW, zE1NEW As Double
490             xE1NEW = (xDIF * -0.525372212562123) + (yDIF * 0) + (zDIF * -0.850872515872724)
491             yE1NEW = (xDIF * 0) + (yDIF * -1) + (zDIF * 0)
492             zE1NEW = (xDIF * 0.850872515872724) + (yDIF * 0) + (zDIF * -0.525372212562123)
493
494             'GUI Update with rounded values
495             myform.TextBox24.Value = Round(xE1NEW, 4)
496             myform.TextBox25.Value = Round(yE1NEW, 4)
497             myform.TextBox33.Value = Round(zE1NEW, 4)
498
499             'Reset Inital State
500             xE1old = xE1NEW * 0.001
501             yE1old = zE1NEW * 0.001
502             zE1old = yE1NEW * 0.001
503     '------------------------------Tool Tip Catersian Motion-------------------------------
504         If myform.xTOOL <> 0 Or myform.yTOOL <> 0 Or myform.zTOOL <> 0 Then
505
506         'calculate new E1 coordinates using spherical E1 direction vector
507             xE1metre = xE1NEW
508             yE1metre = yE1NEW
509             zE1metre = zE1NEW
510         '[--Re-Running invers kinematic Calculations to find old t1&t2--]
511                 'New Coordinates [+convert to ratio of r]
512                 xE1 = (((r2metre) ^ 2 - (yE1metre) ^ 2 - (zE1metre + yE1inc) ^ 2) ^ 0.5) / r2metre
513                 yE1 = (((r2metre) ^ 2 - (yE1metre) ^ 2 - (xE1metre + xE1inc) ^ 2) ^ 0.5) / r2metre
514
515             'Implementing Model from MatLab
516                 zE1 = Sqr(1 - xE1 ^ 2 - yE1 ^ 2)
517                 cost3 = (Cos(a1) * Cos(a2) - xE1) / (Sin(a1) * Sin(a2))
518                     If cost3 = 1 Then
519                         F = 0
520                     ElseIf cost3 = -1 Then
521                         F = 4 * Atn(1)
522                     Else
523                         F = Atn(-cost3 / Sqr(-cost3 * cost3 + 1)) + 2 * Atn(1)
524                     End If
525                 t3 = -F
526
527               cost1 = (yE1 * Sin(t3) * Sin(a2) - zE1 * (Cos(t3) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2))) / ((Sin(t3) *
              Sin(a2)) ^ 2 + (Cos(t3) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2)) ^ 2)
528               sint1 = (zE1 * Sin(t3) * Sin(a2) + yE1 * (Cos(t3) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2))) / ((Sin(t3) *
              Sin(a2)) ^ 2 + (Cos(t3) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2)) ^ 2)
529               cost4 = (Cos(a1) * Cos(a2) - Cos(a0) * xE1 - Sin(a0) * yE1) / (Sin(a1) * Sin(a2))
530
531                     If cost4 = 1 Then
532                         G = 0
533                     ElseIf cost4 = -1 Then
534                         G = 4 * Atn(1)
535                     Else
536                         G = Atn(-cost4 / Sqr(-cost4 * cost4 + 1)) + 2 * Atn(1)
537                     End If
```

Function 8 code **(lines 503-700)** – cartesian driven Tcp motion from user input.

```
                t4 = G

            cost2 = (-xE1 * Sin(t4) * Sin(a2) - zE1 * Sin(a0) * (Cos(t4) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2)) + Cos(a0) * Sin(t4) * Sin(a2) * (Cos(a1) * Cos(a2) - Sin(a1) * Sin(a2) * Cos(t4))) / ((Sin(a0)) * (Sin(t4) * Sin(a2)) ^ 2 + (Sin(a0)) * (Cos(t4) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2)) ^ 2)
            sint2 = (zE1 * Sin(a0) * Sin(t4) * Sin(a2) - xE1 * (Cos(t4) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2)) + Cos(a0) * (Cos(t4) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2)) * (Cos(a1) * Cos(a2) - Sin(a1) * Sin(a2) * Cos(t4))) / ((Sin(a0)) * (Sin(t4) * Sin(a2)) ^ 2 + (Sin(a0)) * (Cos(t4) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2)) ^ 2)

                'Adjusting for current TA1 position
                t2 = Atn(sint2 / cost2) + PI
                t1 = Atn(sint1 / cost1) + PI

                'Calculate t1&t2 change
                If t1old < 0 Then
                    t1old = t1old + (2 * PI)
                ElseIf t2old < 0 Then
                    t2old = t2old + (2 * PI)
                End If

            'Update t1&t2 old
            t1old = t1
            t2old = t2
      '-------------------------------
          'tool in terms of Orcm
            'Read Coord from model
            CoordToolTip

            'Difference between ToolTip and RCM in terms of O[SL]
            Dim xToolTipSL, yToolTipSL, zToolTipSL As Double
            xToolTipSL = (swMathPt.ArrayData(0) * 1000)
            yToolTipSL = (swMathPt.ArrayData(1) * 1000)
            zToolTipSL = (swMathPt.ArrayData(2) * 1000)

            xDIF = xToolTipSL - xOSL
            yDIF = yToolTipSL - yOSL
            zDIF = zToolTipSL - zOSL

            'Apply Translation Matrix O[SL] to Orcm
            Dim xToolTipO, yToolTipO, zToolTipO As Double
            xToolTipO = (xDIF * -0.525372212562123) + (yDIF * 0) + (zDIF * -0.850872515872724)
            yToolTipO = (xDIF * 0) + (yDIF * -1) + (zDIF * 0)
            zToolTipO = (xDIF * 0.850872515872724) + (yDIF * 0) + (zDIF * -0.525372212562123)

            'Calculate new tool tip position based on GUI input
            Dim xToolTipNEW, yToolTipNEW, zToolTipNEW As Double
            xToolTipNEW = xToolTipO + myform.xTOOL
            yToolTipNEW = yToolTipO + myform.zTOOL
            zToolTipNEW = zToolTipO + myform.yTOOL

            'distance from ToolTip Old to ToolTip New
            Dim c As Double
                c = ((myform.xTOOL) ^ 2 + (myform.yTOOL) ^ 2 + (myform.zTOOL) ^ 2) ^ 0.5
            'distance from RCM to ToolTip New
            Dim b As Double
                b = ((xToolTipNEW) ^ 2 + (yToolTipNEW) ^ 2 + (zToolTipNEW) ^ 2) ^ 0.5
            'distance from RCM to ToolTip Old
            Dim a As Double
                a = ((xToolTipO) ^ 2 + (yToolTipO) ^ 2 + (zToolTipO) ^ 2) ^ 0.5

            'unit vector of ToolTip New multiplied by magnitude of ToolTip Old
            Dim xToolTipNEWu, yToolTipNEWu, zToolTipNEWu As Double
            xToolTipNEWu = (xToolTipNEW / b) * a
            yToolTipNEWu = (yToolTipNEW / b) * a
            zToolTipNEWu = (zToolTipNEW / b) * a

            'spherical direction vector (ToolTip Old to ToolTip New) & converted to unit vector
            Dim xToolTipDirec, yToolTipDirec, zToolTipDirec As Double
            xToolTipDirec = (xToolTipNEWu - xToolTipO) / a
            yToolTipDirec = (yToolTipNEWu - yToolTipO) / a
            zToolTipDirec = (zToolTipNEWu - zToolTipO) / a

            'invert spherical tool tip direction vector to create spherical E1 direction vector
            Dim xE1Direc, yE1Direc, zE1Direc As Double
            xE1Direc = (-1 * xToolTipDirec) * (r2 * 1000)
            yE1Direc = (-1 * yToolTipDirec) * (r2 * 1000)
            zE1Direc = (-1 * zToolTipDirec) * (r2 * 1000)

            'calculate new E1 coordinates using spherical E1 direction vector
            xE1metre = xE1NEW + xE1Direc
            yE1metre = yE1NEW + yE1Direc
            zE1metre = zE1NEW + zE1Direc

        '[--Re-Running invers kinematic Calculations to find new t1&t2--]
                'New Coordinates [+convert to ratio of r]
                xE1 = (((r2metre) ^ 2 - (yE1metre) ^ 2 - (zE1metre + yE1inc) ^ 2) ^ 0.5) / r2metre
                yE1 = (((r2metre) ^ 2 - (yE1metre) ^ 2 - (xE1metre + xE1inc) ^ 2) ^ 0.5) / r2metre

            'Implementing Model from MatLab
                zE1 = Sqr(1 - xE1 ^ 2 - yE1 ^ 2)
                cost3 = (Cos(a1) * Cos(a2) - xE1) / (Sin(a1) * Sin(a2))
                    If cost3 = 1 Then
                        F = 0
                    ElseIf cost3 = -1 Then
                        F = 4 * Atn(1)
                    Else
                        F = Atn(-cost3 / Sqr(-cost3 * cost3 + 1)) + 2 * Atn(1)
                    End If
                t3 = -F

            cost1 = (yE1 * Sin(t3) * Sin(a2) - zE1 * (Cos(t3) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2))) / ((Sin(t3) * Sin(a2)) ^ 2 + (Cos(t3) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2)) ^ 2)
```

```
            sint1 = (zE1 * Sin(t3) * Sin(a2) + yE1 * (Cos(t3) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2))) / ((Sin(t3) * Sin(a2)) ^ 2 + (Cos(t3) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2)) ^ 2)
            cost4 = (Cos(a1) * Cos(a2) - Cos(a0) * xE1 - Sin(a0) * yE1) / (Sin(a1) * Sin(a2))
                  If cost4 = 1 Then
                      G = 0
                  ElseIf cost4 = -1 Then
                      G = 4 * Atn(1)
                  Else
                      G = Atn(-cost4 / Sqr(-cost4 * cost4 + 1)) + 2 * Atn(1)
                  End If
              t4 = G

            cost2 = (-xE1 * Sin(t4) * Sin(a2) - zE1 * Sin(a0) * (Cos(t4) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2)) + Cos(a0) * Sin(t4) * Sin(a2) * (Cos(a1) * Cos(a2) - Sin(a1) * Sin(a2) * Cos(t4))) / ((Sin(a0)) * (Sin(t4) * Sin(a2)) ^ 2 + (Sin(a0)) * (Cos(t4) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2)) ^ 2)
            sint2 = (zE1 * Sin(a0) * Sin(t4) * Sin(a2) - xE1 * (Cos(t4) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2)) + Cos(a0) * (Cos(t4) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2)) * (Cos(a1) * Cos(a2) - Sin(a1) * Sin(a2) * Cos(t4))) / ((Sin(a0)) * (Sin(t4) * Sin(a2)) ^ 2 + (Sin(a0)) * (Cos(t4) * Cos(a1) * Sin(a2) + Sin(a1) * Cos(a2)) ^ 2)

              'Adjusting for current TA1 position
              t2 = Atn(sint2 / cost2) + PI
              t1 = Atn(sint1 / cost1) + PI

              'Calculate t1&t2 change
              If t1old < 0 Then
                  t1old = t1old + (2 * PI)
              ElseIf t2old < 0 Then
                  t2old = t2old + (2 * PI)
              End If

      'Calculating updated Lt
          Dim Ltchange As Double
              Ltchange = (b - a) / 1000
          'Part.EditRebuild
              Motiont1 = Dimt1.SystemValue
              Motiont2 = Dimt2.SystemValue
              MotionLt = DimLt.SystemValue

          For n = 0 To 1 Step 0.1
              Dimt1.SystemValue = Motiont1 + ((t1old - t1) * n)
              Dimt2.SystemValue = Motiont2 + ((t2 - t2old) * n)
              DimLt.SystemValue = MotionLt - (Ltchange * n)
              Part.ClearSelection2 True
              If myform.Time <> 0 Then
                  myform.MotionTimer
              Else
              End If
              boolstatus = Part.EditRebuild3()
              DoEvents
          Next n
      'Update t1&t2 old
          t1old = t1
          t2old = t2
      'Clear GUI & Update GUI
          myform.TextBox6.Value = 0
          myform.TextBox28.Value = 0
          myform.TextBox29.Value = 0
          myform.TextBox20.Value = 70 - (DimLt.SystemValue * 1000) + 30

          myform.TextBox24.Value = Round(xE1metre, 4)
          myform.TextBox25.Value = Round(yE1metre, 4)
          myform.TextBox33.Value = Round(zE1metre, 4)

          myform.TextBox17.Value = 360 - Dimt1.SystemValue * (180 / PI)
          myform.TextBox18.Value = Dimt2.SystemValue * (180 / PI)

          myform.TextBox30.Value = Round(xToolTipNEW, 4)
          myform.TextBox31.Value = Round(zToolTipNEW, 4)
          myform.TextBox26.Value = Round(yToolTipNEW, 4)
       Else
       End If
       'ToolTip correction
              MotionLt = DimLt.SystemValue
              DimLt.SystemValue = MotionLt + 0.01
              Part.ClearSelection2 True
              boolstatus = Part.EditRebuild3()
              MotionLt = DimLt.SystemValue
              DimLt.SystemValue = MotionLt - 0.01
              Part.ClearSelection2 True
              boolstatus = Part.EditRebuild3()
       'Updating Tcp in GUI
          'Read Coord from model
          CoordToolTip

          'Difference between ToolTip and RCM in terms of O[SL]
          xToolTipSL = (swMathPt.ArrayData(0) * 1000)
          yToolTipSL = (swMathPt.ArrayData(1) * 1000)
          zToolTipSL = (swMathPt.ArrayData(2) * 1000)

          xDIF = xToolTipSL - xOSL
          yDIF = yToolTipSL - yOSL
          zDIF = zToolTipSL - zOSL

          'Apply Translation Matrix O[SL] to Orcm
          xToolTipNEW = (xDIF * -0.525372212562123) + (yDIF * 0) + (zDIF * -0.850872515872724)
          yToolTipNEW = (xDIF * 0) + (yDIF * -1) + (zDIF * 0)
          zToolTipNEW = (xDIF * 0.850872515872724) + (yDIF * 0) + (zDIF * -0.525372212562123)

          'GUI Update with rounded values
          myform.TextBox30.Value = Round(xToolTipNEW, 4)
          myform.TextBox31.Value = Round(zToolTipNEW, 4)
          myform.TextBox26.Value = Round(yToolTipNEW, 4)
```

Sub code to perform coordinate reading from model **(lines 734-806**)

```
732         Loop
733         End Sub
734         Sub Update()
735             'Rebuild the model
736             Part.ClearSelection2 True
737                 boolstatus = Part.EditRebuild3()
738                 DoEvents
739         End Sub
740         'Function to grab coordinate data from CAD Model
741
742         Function GetPoint(pointName As Variant) As RefPoint
743         Dim swFeat As SldWorks.Feature
744             status = swModelDocExt.SelectByID2(pointName & "@" & swModelDoc.GetTitle, "DATUMPOINT", 0, 0, 0, False, 1,
                Nothing, 0)
745             Set swFeat = swSelMgr.GetSelectedObject6(1, -1)
746             Set GetPoint = swFeat.GetSpecificFeature2
747         End Function
748
749         'Sub to set up coordinate data as MathPt
750
751         Sub GetPointCoordinates(myPoint As RefPoint)
752         'Dim swMathPt As SldWorks.MathPoint
753             If Not myPoint Is Nothing Then
754                 Set swMathPt = myPoint.GetRefPoint
755             Else
756                 MsgBox "Point not found"
757             End If
758         End Sub
759
760         'Sub to read coordinates of RCM relative to Solidworks origin
761         Sub CoordRCM()
762
763         Dim myPoints As Variant
764         Dim point As Variant
765
766             Set swModelDocExt = swModelDoc.Extension
767             Set swSelMgr = swModelDoc.SelectionManager
768
769             myPoints = Array("Orcm")
770
771             For Each point In myPoints
772                 Debug.Print point & "coordinates:"
773                 GetPointCoordinates GetPoint(point)
774             Next
775         End Sub
776
777         'Sub to read coordinates of E1 relative to Solidworks origin
778         Sub CoordE1()
779         Dim myPoints As Variant
780         Dim point As Variant
781
782             Set swModelDocExt = swModelDoc.Extension
783             Set swSelMgr = swModelDoc.SelectionManager
784
785             myPoints = Array("E1")
786
787             For Each point In myPoints
788                 Debug.Print point & "coordinates:"
789                 GetPointCoordinates GetPoint(point)
790             Next
791         End Sub
792
793         'Sub to read coordinates of ToolTip relative to Solidworks origin
794         Sub CoordToolTip()
795         Dim myPoints As Variant
796         Dim point As Variant
797
798             Set swModelDocExt = swModelDoc.Extension
799             Set swSelMgr = swModelDoc.SelectionManager
800
801             myPoints = Array("ToolTip")
802             For Each point In myPoints
803                 Debug.Print point & "coordinates:"
804                 GetPointCoordinates GetPoint(point)
805             Next
806         End Sub
```

Appendix B

Project Meetings

QUEEN'S UNIVERSITY BELFAST | SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

Individual Project Meeting Record

Project Title	Development of Digital Twins for a surgical robot.		
Supervisor	Dr. Yan Jin	Student	**Michael Newbold**
Date and time	27/09/21	Location	**Microsoft Teams**

Review of actions from previous meeting

N/A

Discussion, decisions, assignments

- Discussed background on cataracts surgery
- Watched demonstration video of the surgery
- Discussed development of current design with PHD student designer

Agreed actions and completion dates

- Read and took notes on relevant literature sent by supervisor
- Early stages of defining project aims

Date and time of next meeting	04/10/21 11:30am	Location of next meeting	**Microsoft Teams**
Supervisor signature		Student signature	

QUEEN'S UNIVERSITY BELFAST | SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

Individual Project Meeting Record

Project Title	Development of Digital Twins for a surgical robot.		
Supervisor	Dr. Yan Jin	Student	**Michael Newbold**
Date and time	04/10/21	Location	**Microsoft Teams**

Review of actions from previous meeting

- Discusses background on cataracts surgery
- Watched demonstration video of the surgery
- Discussed development of current design with Phd student designer, Yinglun

Discussion, decisions, assignments

- Discussed GUI development and possible software
- Laid out project aims
- Discussed possibility of implementing a model eye into the simulation

Agreed actions and completion dates

- Look into different GUI software
- Layout a project description
- Develop a rough project plan
- Continue with literature review

43min

Date and time of next meeting	11/10/21 11:30am	Location of next meeting	**Microsoft Teams**
Supervisor signature		Student signature	

QUEEN'S UNIVERSITY BELFAST | SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

Individual Project Meeting Record

Project Title	Development of Digital Twins for a surgical robot.		
Supervisor	Dr. Yan Jin	Student	**Michael Newbold**
Date and time	11/10/21	Location	**Microsoft Teams**

Review of actions from previous meeting

- Discussed GUI development and possible software
- Laid out project aims
- Looked into specifications of model eye dimensions

Discussion, decisions, assignments

- Discussed functions and advantages of a digital twin
- Confirmed use of Solidworks as primary software
- Reviewed a draft project plan Gantt Chart
- Reviewed a recently made video on what the digital twin should look like once in use during a surgery

Agreed actions and completion dates

- Develop draft project description and revised plan
- Begin to understand and review examples of using macros in Solidworks API
- Continue with literature review

27min

Date and time of next meeting	18/10/21 11:30am	Location of next meeting	**Microsoft Teams**
Supervisor signature		Student signature	

QUEEN'S UNIVERSITY BELFAST | SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

Individual Project Meeting Record

Project Title	Development of Digital Twins for a surgical robot.		
Supervisor	Dr. Yan Jin	Student	**Michael Newbold**
Date and time	18/10/21	Location	**Microsoft Teams**

Review of actions from previous meeting

- Draft of project plan completed
- Viewed macro tutorials for Solidworks API

Discussion, decisions, assignments

- Revised the draft of project description with supervisor feedback
- Further revised the project plan Gantt Chart
- Discussed how the inverse kinematic would be integrated into the API
- Reviewed the current CAD model and became familiar with its operation

Agreed actions and completion dates

- Complete final draft of project description and send to supervisor
- Look more in depth at Solidworks API and start to learn the function of movement and motor commands with macros
- Continue with literature review

29min

Date and time of next meeting	01/11/21 11:30am	Location of next meeting	**Microsoft Teams**
Supervisor signature		Student signature	

QUEEN'S UNIVERSITY BELFAST | SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

Individual Project Meeting Record

<table>
<tr><td>Project Title</td><td colspan="3">Development of Digital Twins for a surgical robot.</td></tr>
<tr><td>Supervisor</td><td>Dr. Yan Jin</td><td>Student</td><td>Michael Newbold</td></tr>
<tr><td>Date and time</td><td>01/11/21</td><td>Location</td><td>Microsoft Teams</td></tr>
<tr><td colspan="4"><u>Review of actions from previous meeting</u>

- Project description form submitted
- Began modelling motion in Solidworks

<u>Discussion, decisions, assignments</u>

- Discussed full feedback of submitted project description form
- Demonstrated movement macro on a test robot as proof of concept
- Showed initial design of GUI
- Reviewed the inverse kinematics for base movement of Orcm

<u>Agreed actions and completion dates</u>

- Continue with GUI development
- Fully integrate inverse kinematic model for base Orcm movement and link with GUI

17 min</td></tr>
<tr><td>Date and time of next meeting</td><td>08/11/21 11:30am</td><td>Location of next meeting</td><td>Microsoft Teams</td></tr>
<tr><td>Supervisor signature</td><td></td><td>Student signature</td><td></td></tr>
</table>

QUEEN'S UNIVERSITY BELFAST | SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

Individual Project Meeting Record

Project Title	Development of Digital Twins for a surgical robot.		
Supervisor	Dr. Yan Jin	Student	**Michael Newbold**
Date and time	08/11/21	Location	**Microsoft Teams**

Review of actions from previous meeting

- Continued development of GUI
- Fully functioning base Orcm movement with working kinematic calculations

Discussion, decisions, assignments

- Highlighted new code written
- Demonstrated full movement of base Orcm
- Discussed limits of work area and how to implement error message
- Discussed implementing inverse kinematic model for tool tip model

Agreed actions and completion dates

- Implement motion for tool tip
- Continue with GUI development
- Brainstorm any possible additional features

23min

Date and time of next meeting	25/11/21 11:30am	Location of next meeting	**Microsoft Teams**
Supervisor signature		Student signature	

QUEEN'S UNIVERSITY BELFAST | SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

Individual Project Meeting Record

Project Title	Development of Digital Twins for a surgical robot.		
Supervisor	Dr. Yan Jin	Student	**Michael Newbold**
Date and time	25/11/21	Location	**Microsoft Teams**

Review of actions from previous meeting

- Coding for implantation of tool tip movement
- Error found in draft kinematic model

Discussion, decisions, assignments

- Amendment to kinematic model discussed
- Highlighted initial stages of logic to the model
- Discussed how the new inverse kinematic would be integrated for the tool tip
- Discussed possible haptic integration with model

Agreed actions and completion dates

- Complete implementation of motion for tool tip with amended kinematic model
- Think about how a haptic input would work with the API
- Continue writing code and troubleshooting errors

21min

Date and time of next meeting	07/12/21 11:30am	Location of next meeting	**Microsoft Teams**
Supervisor signature		Student signature	

Individual Project Meeting Record

<table>
<tr><td>Project Title</td><td colspan="3">Development of Digital Twins for a surgical robot.</td></tr>
<tr><td>Supervisor</td><td>Dr. Yan Jin</td><td>Student</td><td>Michael Newbold</td></tr>
<tr><td>Date and time</td><td>07/11/21</td><td>Location</td><td>Microsoft Teams</td></tr>
<tr><td colspan="4"><u>Review of actions from previous meeting</u>

- Fully integrated full inverse kinematic model and motion for tool tip
- Cleaned up layout of code with labels for logic and functions

<u>Discussion, decisions, assignments</u>

- Demonstrated working initial draft digital twin
- GUI with 6 inputs, controlling base Orcm and tool tip movement separately
- Discussed possible tasks for over the holidays
- Agreed to collect haptic device from Yinglun to learn how it functions for possible integration if all other task are complete

<u>Agreed actions and completion dates</u>

- Haptic device collected and its drivers installed
- Tasks for over the holidays:
  ➢ Continue cleaning up code layout
  ➢ Brainstorm & develop new features for model
  ➢ Begin initial compilation of work for report
  ➢ Layout full logic flow chart of model

17 min</td></tr>
<tr><td>Date and time of next meeting</td><td>20/01/22 11:30am</td><td>Location of next meeting</td><td>Microsoft Teams</td></tr>
<tr><td>Supervisor signature</td><td></td><td>Student signature</td><td></td></tr>
</table>

QUEEN'S UNIVERSITY BELFAST | SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

Individual Project Meeting Record

Project Title	Development of Digital Twins for a surgical robot.		
Supervisor	Dr. Yan Jin	Student	**Michael Newbold**
Date and time	20/01/21	Location	**Microsoft Teams**

Review of actions from previous meeting

- Function to provide live coordinate updates of Orcm and tool tip were developed to allow for possible integration of haptics
- Translation matrix for these coordinates was developed
- Logic flow chart was developed
- Draft of summary and literature review for report
- Model and report submitted for progress report

Discussion, decisions, assignments

- Discussed progress and feedback
- Discussed direction of final report and areas of importance
- Reviewed flow chart for model

Agreed actions and completion dates

- Focus on report writing
- Create draft for introduction and literature review
- Organise all project files and compile project meeting forms, project plan and project description for addition to report

34min

Date and time of next meeting	07/02/22 10:00am	Location of next meeting	**Microsoft Teams**
Supervisor signature		Student signature	

QUEEN'S UNIVERSITY BELFAST | SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

Individual Project Meeting Record

Project Title	Development of Digital Twins for a surgical robot.		
Supervisor	Dr. Yan Jin	Student	**Michael Newbold**
Date and time	07/02/22 11:30am	Location	**Microsoft Teams**

Review of actions from previous meeting

- Draft of lit review and intro completed
- All project meeting files, and project files organised
- Further model development

Discussion, decisions, assignments

- Full draft report deadline agreed upon
- Future model development was discussed
- Clarified digital twin specifications with designer

Agreed actions and completion dates

- Draft report to be written
- Bugs in code to be fixed
- Validation testing to be carried out

27min

Date and time of next meeting	TBD	Location of next meeting	**Microsoft Teams**
Supervisor signature		Student signature	

Meeting 10 was approximately 22 minutes.

Project Plan

Project work packages have been outlined in the Gantt Chart as seen in figure 33. Overall, the project plan has been designed efficiently as all deadlines were met on schedule. However, the coding element of the project took more time than anticipated and bugs took longer to find and fix.

Ideally, all coding related work packages would be moved forward one week as to allow for bug fixing to begin before half term.

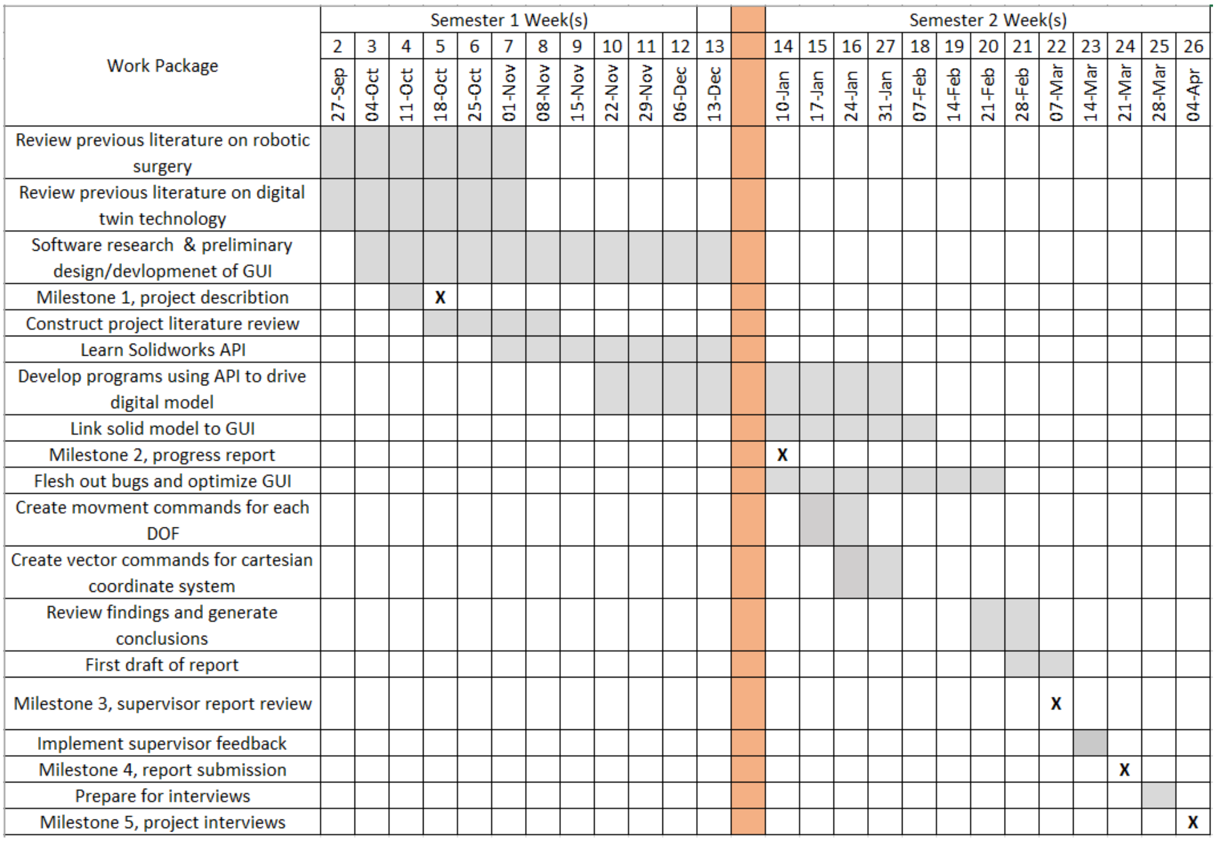
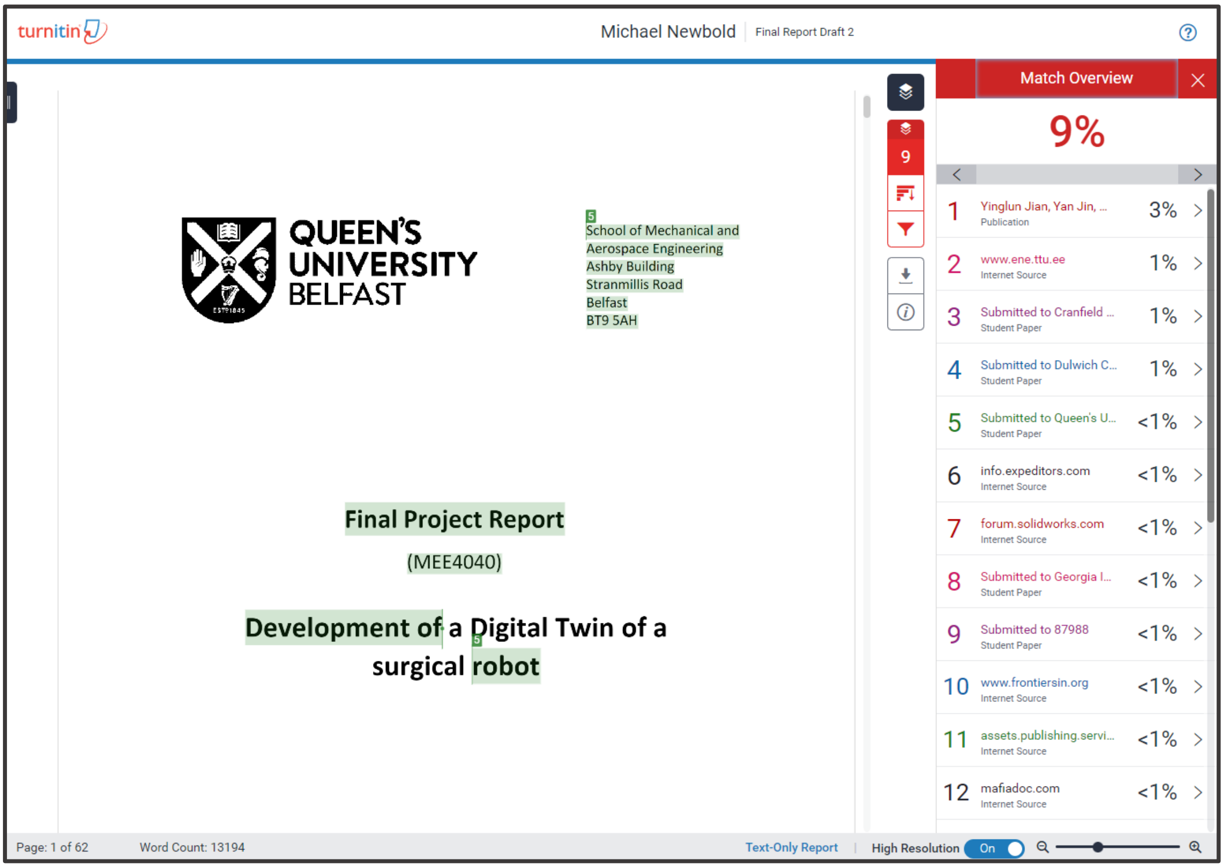


Figure 33 Project Gantt Chart

Turnitin Originality Screenshot



Submission Checklist

	TICK IF MET
Does the report meet the formatting stipulated in the module handbook and template provided?	✓
Line spacing (1.5).	✓
Font (Calibri 11 Pt).	✓
Margins: Top & Bottom (25 mm), Left (25 mm), Right (25 mm).	✓
Paragraphs are fully justified.	✓
Does the main body of the report meet the strict 30 page limit (40 page limit for MEE7012) (excluding Title Page, Table of Contents, Turnitin Summary, References and Submission Checklist)?	✓
Do the appendices meet the strict 10 page limit (15 page limit for MEE7012) (excluding meeting minutes and project management info)?	✓
Are all tables and figures numbered correctly, captioned and referenced if required?	✓
Has the report been checked using Turnitin?	✓
Is the Turnitin summary report included in the report?	✓
What is the similarity index provided by Turnitin?	9%
Pages are numbered.	✓
Has the project description (as submitted in Semester 1, with any subsequent changes agreed with your supervisor) been included at the start of the report? The Gantt chart should be saved in the project management appendix.	✓

Statement of originality

I hereby declare that this project is my own work and that it has not been submitted for another degree, either at Queen's University Belfast or elsewhere. Where other sources of information have been used, they have been acknowledged.

Signature: ....

Date: 21/03/22