VIRTUAL REALITY SIMULATION OF A SMART EATING TABLE FOR HANDICAPPED PEOPLE

DISSERTATION
zur Erlangung des akademischen Grades
eines Doktors der Ingenieurswissenschaften (Dr.-Ing.)

vorgelegt von

Salih Rashid Majeed

eingereicht bei der
Naturwissenschaftlich-Technischen Fakultät
der Universität Siegen

Siegen, 2019

Betreuer und erster Gutachter

Prof. Dr.-Ing. Hubert Roth

Universität Siegen

Zweiter Gutachter

Prof. Dr.rer.nat. Volker Blanz

Universität Siegen

Tag der mündlichen Prüfung

06. Oktober 2020

Acknowledgment

I would like to thank my god for supporting me through the PhD study.

Through the PhD study, I had many problems in my research, which couldn't be solved without the help and great ideas from Prof. Dr-Ing. Huber Roth and Prof. Dr-Ing. Klaus Dieter Kuhnert and his team (Dipl.-Inform. Jens Schlemper, M.Sc. Simon Hardt, Dipl.-Inform. Klaus Müller, Dipl.-Inform. Jan Kunze, Dipl.-Inform. Marc Steven Krämer). Many thanks to M.Sc.Marco Hutwol for his scientific advice through this research. Also, many thanks for Prof.rer.nat.Volker Blanz for his cooperation and his great advice. I would also like to thank Dr.Khaled Sailan for his idea for improving the work.

Many thanks go to Prof.Dr. Holger scheonherr, Prof.Dr. Ivor Fleck and Prof.Dr. Markus Lohery for their support and cooperation.

Many thanks to DAAD and Germany for giving me the opportunity to complete my PhD study in Germany and supporting me.

Kurzfassung

Diese Arbeit beschäftigt sich mit der simulierten Interaktion zwischen Mensch und Roboter und der Verwendung des Roboters in verschiedenen Lebensbereichen. Entsprechend der Entwicklung in der künstlichen Intelligenz und in der Robotik wird es möglich, neue intelligente Roboteranwendungen zu entwickeln. Die Idee zu diesem Projekt entstand durch die große Anzahl der behinderten Menschen in meinem Heimatland, die durch den dortigen Terrorkrieg verletzt worden sind. Die Herausforderung ist, ein neues intelligentes Robotersystem zu bauen, das den Menschen bei ihren alltäglichen Lebensaktivitäten - speziell beim Essen und Trinken - hilft. Aus vielen Gründen, die im ersten Kapitel erläutert werden, haben wir beschlossen, die Simulationssoftware zu verwenden, um ein solches System zu implementieren. Das vorgeschlagene simulierte System besteht aus drei Subsystemen: dem Robotersystem (einem JACO Manipulator Arm). Dem Seh-System (semantische Kamera), das verschiedene Umgebungsobjekte nach ihren Typen und Namen erkennen kann. Es liefert uns dann die Koordinaten für jedes einzelne Objekt. Das letzte Subsystem ist das Geräuscherkennungssystem, das die Wörter erkennt, die programmiert wurden, um den Prozesstyp Essen oder Trinken auch auf die Lebensmittelarten zu beziehen. Die gesamte Umgebung wurde mit Hilfe eines Simulationsprogrammes (MORSE) erstellt, das Python vollständig unterstützt und auf Blender basiert (3D-Zeichnungs und Animationsprogramm). Die Umgebung besteht aus dem Modell einer Person, einem Essenstisch, zwei Schüsseln mit verschiedenen Farben, um die verschiedenen Essensarten darzustellen, einer Küche mit einigen Pflanzen, Heizungen, Spüle und Tisch. Der Arm wurde mit den Designdaten des realen Armes implementiert. Er beinhaltet die gleichen Gelenke, Links und Funktionen. Der Roboterarm, der in diesem Projekt verwendet wurde, ist ein JACO Roboterarm. Dieser Arm wurde speziell für Behinderte entwickelt, die kleine Probleme mit Essen und Trinken haben. Wir haben diesen Arm durch Hinzufügen von zwei Systemen weiterentwickelt: diese Systeme sind Sehen und Hören. Dies wird den Leuten helfen, die ihre Hände nicht benutzen können. Das externe Mikrofon wird mit dem PC verbunden und startet seine Funktion, wenn das Programm startet. Das System wird unter Verwendung von robot operating system (ROS) gesteuert, dass die Sprache C ++ unterstützt. Das System beginnt wie folgt: Wenn der Mensch das gewünschte Wort sagt, das die Nahrungsmittelarten oder das Getränk beschreibt, wird es durch das Geräuscherkennungssystem erkannt. Das Sichtsystem wird die Koordinaten des erforderlichen Essens oder Getränks erkennen. Die Koordinaten des Essens oder des Getränkes werden gespeichert und dann an den Roboterarm übertragen. Der Roboterarm fährt zum Essenstisch oder zum Glas, nimmt es auf und bewegt sich dann zum Mund. Danach bringt er den Rest des Essens oder das Glas zurück an die erste Position. Dieses Projekt kann

sehr hilfreich sein für Menschen, die körperliche Probleme beim Essen oder Trinken haben. Also insbesondere für Behinderte mit Einschränkungen der Handhabung.

Dedication

To my country Iraq with love. You will be back better than before.

To my mother, and father. Without your encouragement, I would never have been able to complete my graduate studies.

To my wife and beautiful son, you always support me, I love you both and I appreciate everything that you have done for me.

To my brother and sisters, thank you for your love.

“"be", and it is!”

Abstract

This thesis deals with simulated human-robot interaction and how to use a robot in different everyday life activities. It is in line with the significant developments in the sector of artificial intelligent and robotic fields that have enabled the emergence of a new smart robot generation. The idea of this project came from the large number of disabled people, especially handicapped people. This number has been substantially increased by the terrorist war in my country. The challenge of this task is how to build a new smart robotic system that can help those people in everyday life activities, especially in eating and drinking. Due to many reasons that will be clarified in the first chapter, we decided to use simulation software to implement such a system. The proposed simulated system comprises three sub-systems: the robotic system (a JACO manipulator arm), the semantic camera system – which can detect the different environment objects according to their types and names, which finally gives us the coordinates for each one – and the sound recognition system, which will recognize the words from the person using it. It has been programmed to refer to the process type (eating or drinking), as well as the food types. The whole environment has been built using a simulation program (MORSE) that completely supports Python and is based on Blender (a 3D drawing and animation program). The simulated environment comprises a human model, a food table, two dishes with different colours to represent the food types, a kitchen room with some plants, heaters, sink, and tables. The robot arm has been implemented with the same design as the real arm. It includes the same joints, links, and functions. The robotic arm simulated in this project is a JACO robotic arm. This arm has been designed especially for disabled people who have minor problems with eating and drinking. We improved this arm by adding two systems, one for semantic camera system and one for sound. This will help people who cannot use their hands. An external microphone is connected to the PC and starts its function when the program starts, whereby the system is controlled by using a robot operation system (ROS) that supports C++. The system then starts with the human saying the required word, which represents the food type or a drinkable liquid (e.g. water). Each of the available food types has a specific word. When the person says this word, it will be recognized by the sound recognition system. Subsequently, the semantic camera system will detect the coordinates of the required food or drink, which will be stored and then transferred to move the robotic arm. The robotic arm will drive to the food dish or the water cup and pick up it, before moving to the mouth and finally returning the rest food or cup to the first position and continuing in this execution loop. This project will be very helpful to people who have physical problems with eating or drinking, especially for disabled and handicapped people.

Table of Content

List of Figures

List of Tables

1 Introduction

Robots today are involved in many different life fields, including the medical field, the automotive industry, human life activity and space, among others. Robots normally comprise mechanical parts, sensors and actuators. This research deals with the design of an intelligent system (robot arm, environment), while also dealing with the implementation of a control software that makes the robotic arm autonomously through the pick-up and release object operations, which will be associated with sound recognition and semantic camera systems. This project may be expansive because any error in the programming algorithms could affect the robotic arm hardware and subsequently may damage the arm joints and links. Moreover, our project deals with humans directly, which means that any error in this algorithm could harm people. One of the most suitable and effective solutions for conducting our task is by using simulation programs to carry out the system physically with all reality specifications. With the simulation, many features will be achieved, such as:

- ♦ low cost of implementation
- ♦ avoiding hardware defects (the arm simulation gives more flexibility through motion)
- ♦ increased safety for the user (it could be handled easily if a programming error occurs)
- ♦ more flexibility in using of the simulated intelligent semantic camera system

On the other hand, this solution also has side effects as follows:

- ♦ losing the real imagination for the system
- ♦ more programming complexity, given that many specifications have to be programmed as follows:

a. the robotic arm motion mechanism, such as inverse kinematic, trajectory, joints rotation limits, links length

b. the physical features for the environment (the border of the objects, gravity)

c. implementing a semantic camera system and sound system with different middle ware language

1.1 Motivation

The main idea for this research came from the negative security situation in my country (Iraq), where the numbers of handicapped and disabled people have increased due to terrorist attacks and previous wars. Moreover, the number of disabled people has also increased around the world and they need some devices to simplify their lives. At present, robots with artificial intelligence are entering all fields, including different life activities. The main target of this research is to explore how to implement an autonomous robotic eating system supported by a semantic camera system and a sound recognition system, using a simulation program to simulate a JACO robot arm that has six degrees of freedom (DOF). In the simulated environment, there is a table, eating dishes and a water cup. The arm moves according to the required food types, whereby each one of these types refers to a specific colour dish. The sound recognition system stores the colour names and then the semantic camera system inquiries about the coordinates of each object in the environment according to the internal data of these objects into blender environment which have been implemented and stored into MORSE data bank. When the human says the specific object-type (colour name), the arm will move directly to the coordinates of this dish and pick up the food, before moving it to the mouth coordinates. This will help disabled people, especially those who have physical problems with their hand, and it will also be an effective alternative to receiving external help from another person.

1.2 Research Objectives

This research is part of a larger project conducted at the Institute of Real-Time Learning Systems at the University of Siegen. This research aims to develop a new simulated intelligent robotic system for the disabled. Following momentous inspections, it has emerged that the building of such a simulation robotic arm with sound and semantic camera systems has never been attempted. This has led to various inquiries concerning the automation of such a system:

- How can a completely automated intelligent system be created for helping handicapped people when they are eating or drinking? This system should comprise three sub-systems: semantic camera system, sound and robotics systems.
- Can a JACO robotic arm be integrated in a simulation program with completely functioning hardware and specifications? The arm should be represented by links

length with the real specifications, the rotation angle (value and axis) of each joint and solve the inverse kinematic problem while skinning and planning a trajectory. These are important challenges when arm movement is simulated.

- How complex is software when combining three different language systems? Compatibility between the three systems is not easy, especially because each of these systems uses a different programming environment, which necessitates building connecting software bridges to combine their work.
- Will the software support the capacity to create reasonable sensor and input information that can be utilized by the control program? This should correspond in real time.
- Can the simulation give a graphic user interface that permits imagining the movements of the robotic joints (rotate or translate)? This should be easy to implement with the crashing and gripping process between robotic parts and objects.
- Can the software give an identical control interface as real robots? This interface should essentially comprise various topics that can be utilized to send control messages to the different robotic parts.
- Can the semantic camera system be checked to recognize the objects required and coordinate effectively with software? Will there be sufficient time for recognizing the sound and then sending the signal back to complete the task?
- Will the sound system complete its task easily and not be affected by repetition or background noise?

1.3 Main contributions

The main objective of this research is to create a complete intelligent robotic system for people with disabilities when they are eating or drinking, as shown in Figure (1). The main contributions pertaining to specialized technical issues in building this new system are listed briefly below, before explained in detail in chapters three, four and five:

- Studying, analysing and modelling the implantation systems used to create a new library in the simulation software for the whole system, which should work as one unit. When the user calls up this library, the system starts automatically, continuing until the end of the eating or drinking process.
- Analysing, programming and designing a sound recognition system based on the static language model, improving the threshold for each word before adding its

value and saving it in the ARPA text format to be used later in the sound control library. This has significantly improved the sound system.

- Implementing and programming the inverse kinematic calculations, adding the joint's rotational angles and link length. This is achieved by studying and analyzing the JACO robot arm through its kinematics and movement mechanisms.
- Modelling and analyzing a semantic camera system. Designing the environment objects, and then programming them as active parameters. The physical properties of each object (collision border, free fall) also have to be planned. After this, building a software approach for discovering the coordinates according to their nature, label and type in the semantic environment is necessary.
- Implementing a new skinning method for removing deformations through movement. This is achieved by studying the mesh distribution method and then adding the axis and the rotation parameters as new conditions to ensure the soft motion of the simulation.
- Studying, implementing and then programming a new trajectory control approach. This is undertaken by adding more additional control nodes to control the rotational joints, instead of using one control node that is responsible for giving the joint angle values. This improves the motion response time.
- Through studying the coordinates system, finding out a new transformation coordinates system to obtain the exact coordinates of the objects. The camera gives us the coordinates according to its environment rather than the general coordinates. Thus, a way of transferring the coordinates from the camera to the robot should be discovered.
- Designing and implementing a new end-effector position controller using a PID control system that reduces the error between the desired position coordinates and the actual coordinates for the robotic arm end-effector.

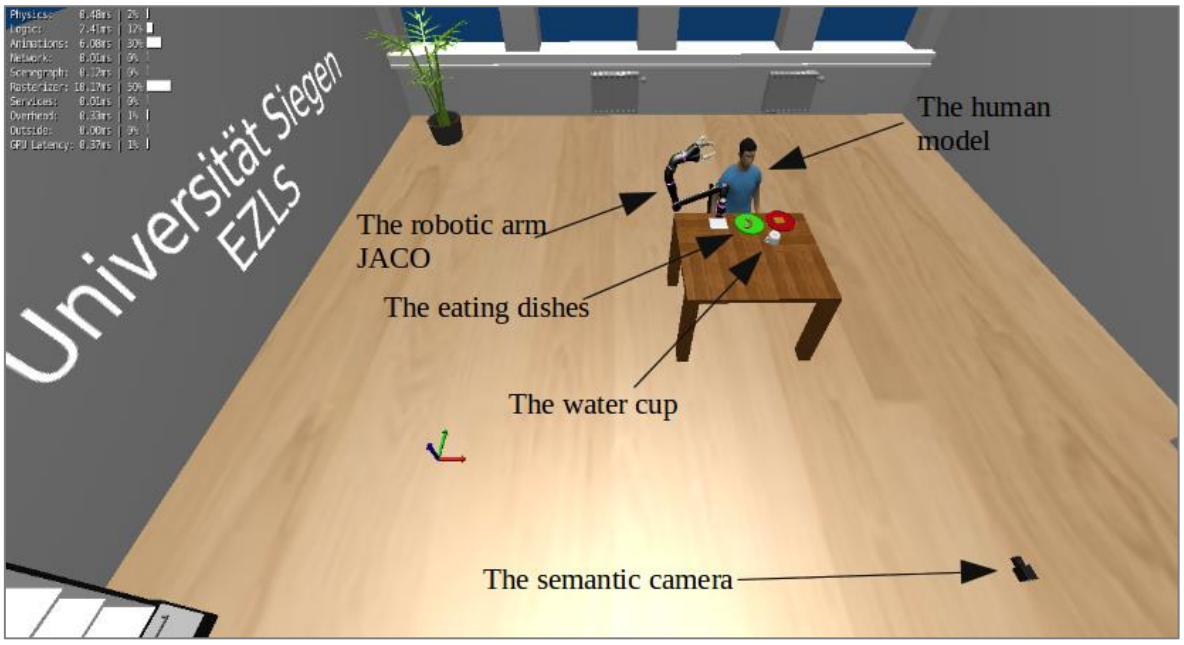


Figure 1: The simulation project.

1.4 The publications

During the PhD research period, the author has published a number of research papers in scientific journals and one workshop. This research is related to these papers, which are listed as follows:

1 Salih Rashid Majeed, Klaus D. Kuhnert. Simulation of Jaco robot arm for handicapped people. Proceedings of the 8th International Workshop on Human Friendly Robotics – HFR TUM 2015, October 21–23, 2015, in Munich, Germany.

2 Salih Rashid Majeed, Klaus D. Kuhnert. Automatic Skinning of the Simulated Manipulator Robot Arm. International Journal of Computer Graphics & Animation (IJCGA) Vol.6, No.1, January 2016.

3 Salih Rashid Majeed, Klaus D. Kuhnert. Simulation of the Inverse-Kinematics for JACO Manipulator Robot Arm. International Journal of Robotics and Automation Vol. 1, No. 1, 2016.

4 Salih Rashid Majeed, Klaus D. Kuhnert. Object Detection based on Semantic Camera for Indoor Environment. International Journal of Advance Robotics & Expert Systems (JARES) Vol.1, No.1, 2016.

5 Salih Rashid Majeed, Klaus D. Kuhnert. Trajectory Controller Building for the (KUKA, JACO) Simulated Manipulator Robot Arm Using ROS. International Journal of Robotics and Automation (IJRA) Vol.3, Issue 1, January 2017.

6 Salih Rashid Majeed, Klaus D. Kuhnert. Simulated Robotic Arm Control using Sound Recognition System Commands. International Robotics and Automation Journal (IRATJ) Vol.3, Issue 1, September 2017.

7 Salih Rashid Majeed. Improvement of the automatic skinning approach for the JACO robotic arm based on a new weight-vertex distribution method. International Robotics and Automation Journal,Volume 5, Issue 4 - 2019

1.5 Dissertation Outlines

This dissertation comprises nine chapters, which explain the main contributions and the results for each of the main systems as follow:

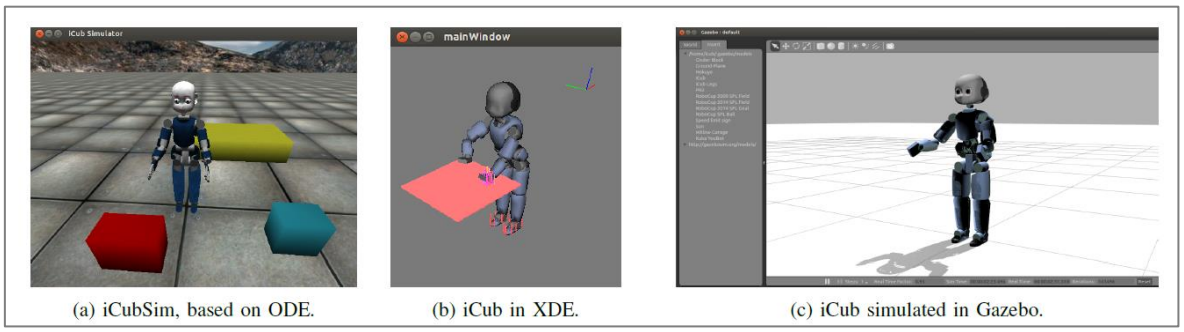
- Chapter two presents the research and papers used as references for our research, followed by the state of art of each sub-system.
- Chapter three displays the flow chart of the project, describes the control loop for the different systems and explains the communication between the systems.
- Chapter four discusses the implementation of the JACO robotic arm. It describes the arm specifications with the mechanical design and how to implement the real arm. It studies and analyses inverse kinematic calculations, as well as describing our new skinning approach and explaining it in detail. Our new control system for the arm is described in this chapter.
- Chapter five describes the semantic camera system implementation, as well as displaying the objects recognition method and how the camera works through the program. It explains the camera control unit with the ROS nodes and how to obtain the coordinates of the detected objects.
- Chapter six discusses the sound system, including its methodology and control unit. Our new improvement compared with the old system is discussed, including how to achieve a better recognition result.
- Chapter seven discusses the position control of the task space and how to control the end-effector position using the PID controller. Moreover, it discusses some theory and results regarding the actual and desired coordinates.
- Chapter eight describes the experimental results, as well as the safety case study.
- Chapter nine provides a summary of the dissertation provided and some suggestions and ideas for the future.

2 Literature survey

This chapter deals with the research that has been represented as a basis for this project. The research is classified into four different groups, each of which deals with a specific field. The first group displays the simulation of the environments and how to implement the different things or objects, while the second group involves the semantic camera and maps. The assistant robot research is presented in the third group, before the final group deals with research in the field of sound recognition.

2.1 Simulation tool background

The first challenge in this project was how to simulate the project into a real environment and then find suitable software to simulate the environment. Simulation softwares are widely used due to flexibility in the designing process and the facility through the implementing control process, the simulation gives the capability to build and test multiple prototypes until reaching the optimum one as in the nature life. As shown Figure (2) [1] there is a growing number of tools for dynamics simulation, ranging from dynamic solver libraries to systems simulation software, provided through either open or closed source code solutions, each more or less tailored to their expected domains of application[1].



(a) iCubSim, based on ODE. (b) iCub in XDE. (c) iCub simulated in Gazebo.

Figure 2: Simulators of iCub. From left to right: iCubSim, based on ODE, XDE and Gazebo. (credits for Gazebo: Silvio Traversaro)[1]

In our project, we encountered many problems during the simulation process. These included the physical properties for the static and dynamic objects, such as free fall and collision. In the robotic researches, simulation programs permit to build and test multiple virtual designs for the robots, also give the ability for implementing and testing many controlling theories such as the motion planning , PID controller, fuzzy logic controller and many other theories. [2] most simulators have been restricted to 2D worlds, and few have matured to the point where they are both highly capable and easily adaptable. [2] Gazebo is designed to fill this

niche by creating a 3D dynamic multi-robot environment capable of recreating the complex worlds that will be encountered by the next generation of mobile robots.

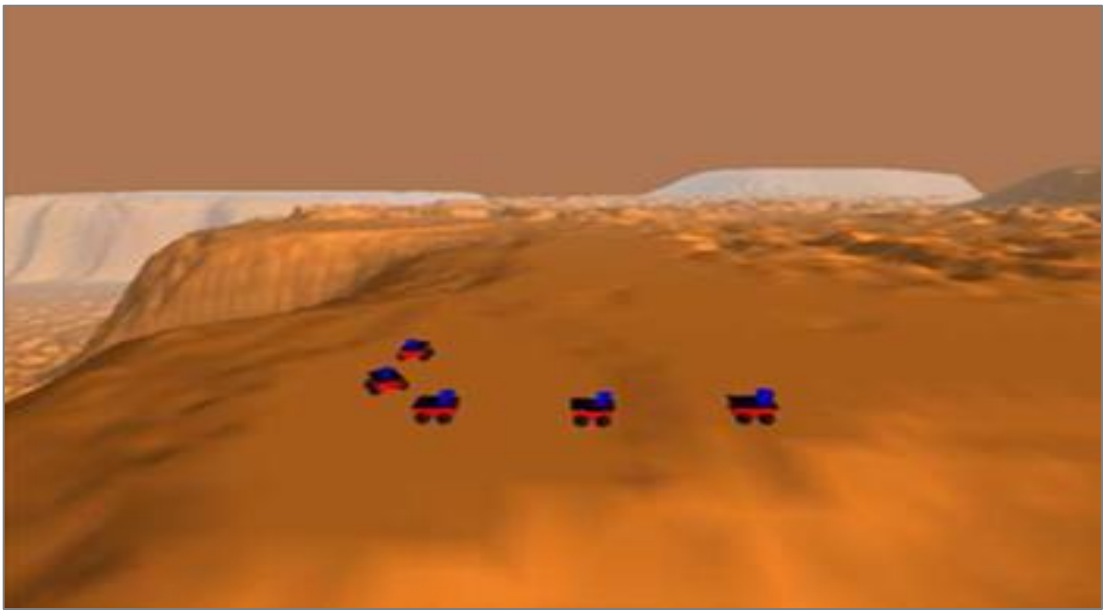


Figure 3: Terrain and Pioneer2 AT robots [2].

2.2 Semantic environments background

As one of the two main sub-systems the camera is based on semantic model principles, represented by the semantic camera in this research. The camera can be directed to detect objects by type in the simulation environment (further details will be provided later). The second challenge was recognizing multiple object coordinates and how to use them later in the motion task. Regardless of usage of the simulation data analysis due to the reasons in the introduction we displayed a literature of a real science data analysis as a reference for the future studies. In the past, different definitions have been given: the word 'winter' could also signify 'snow', ' sledging' or ' mulled wine,' [3]. It has also been suggested that the meaning of a word is only the entity in the world to which it refers, such as proper nouns like New York and Eiffel Tower.[3]

One important piece of research is provided by [4] their application involves detection of objects placed in a clutter and in tight environments, such as a shelf. In particular, given access to 3D object models, several aspects of the environment are simulated and the models are placed in physically realistic poses with respect to their environment to generate a labeled synthetic dataset, The 3D CAD models are generated and loaded in a calibrated environment on the simulator using Blender, a subset of the objects is chosen for generating a scene, the scenes are rendered from known camera poses and perspective projection is used to compute 2D bounding boxes for each object as shown in the Figure (4)[4].

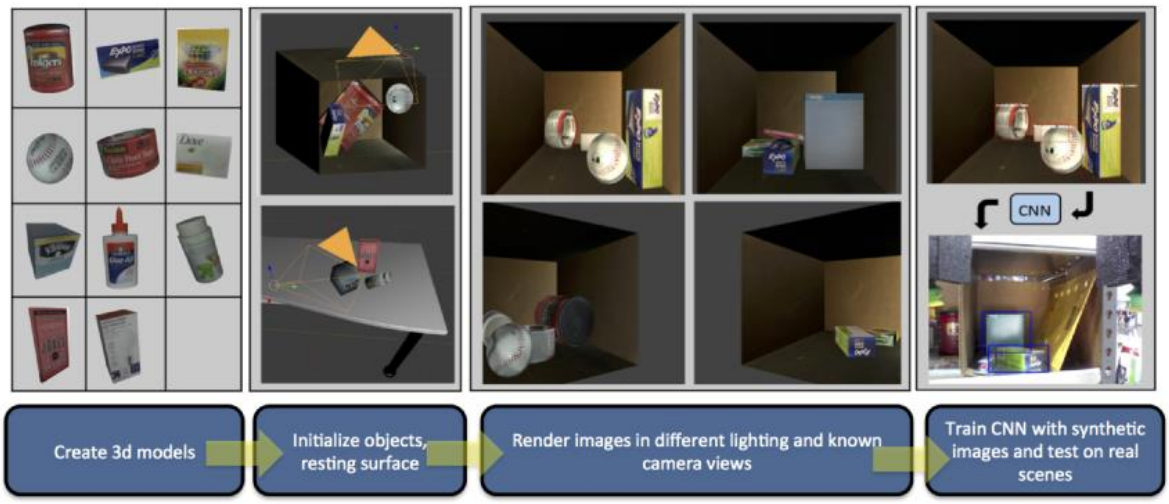


Figure 4: Pipeline for physics aware simulation[4].

[5] presented an approach that combines the robustness of CNNs with a fine-resolution instance-based 3D pose estimation, where the model is trained with fully-annotated synthetic training data, generated automatically from the 3D models of the objects. [5] Their results show that the proposed model can be trained only with synthetic renderings of the objects 3D models and still be successfully applied on images of the real objects.

[6] Proposing an algorithm that allows robots to efficiently learn human-centered environment models from descriptions of natural language, typical semantic mapping approaches add to metrical maps with high-level surroundings properties (e.g. place type, object locations) but do not use this information to improve the metric map. A semantic hierarchical classifier that uses the semantics of image labels to extract knowledge about inter-class relationships and integrates them into the visual appearance learning process reduces the classifier complexity in the number of classes, helping to learn about the visual similarities in the experimental section [7].

The new approach to semantical mapping proposed in [8] is object-class image segmentation, which is used to recognize objects in the pixel-wise RGB-D images shown in Figure (5). They include the depths and colour markers in the random forest decision classifier, normalizing the scale characteristics through depth measurements, based on trajectory estimates derived from the SLAM approach [8]. These use the image segmentation into a probabilistic 3D object-class map and demonstrate in experiments from two data sets that their approach not only provides a 3D segmentation of object classes but also significantly improves the quality of 2D segmentation [8]. Their approach operates directly within the original image measurements while fusing RGB-D measurements into a 3D map and classifying 3D volumes. [8]

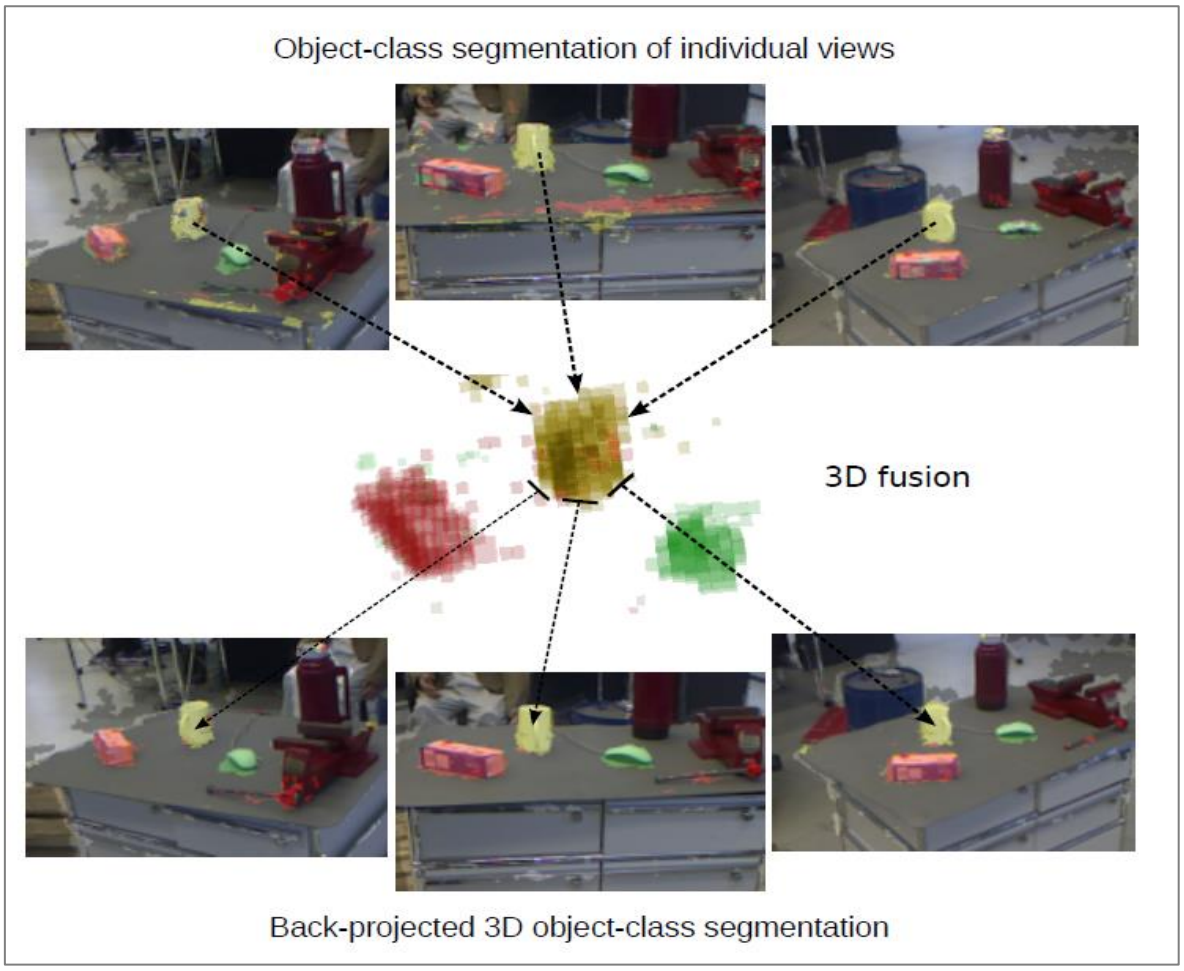


Figure 5: Learned object-class segmentation of various views fused in 3D in a Bayesian framework. They not only obtain 3D object-class maps: filtering in 3D from multiple views also reduces false positives and significantly improves segmentation quality. This is reflected in the crisp back-projection of the 3D object-class map into the images.[8].

2.3 Assistive robot background

The main part or execution of the motion uses a JACO robotic arm, which is part of a project for people with disabilities. This arm is well known with suitable material for human activities, whereby there are different types of assistive robots according to [9].

A. Four types of rehabilitation robots:

1) Static robots that operate in a structured environment (workstations) [9]: this type of robots is designed for disabled people, and they usually comprise a manipulator robotic arm fixed on the table or disk with no sensor. [9] This arm is programmed to collect some item from drawers (e.g. phone, book) such as DeVAR [10], ProVar [11], RAID [12] and Master [13]. In such types, the positions of the collected objects are known in advance. [9] There is also another type of workstation robots, such as the robotic arm that is designed with a spoon and fork for self-feeding for disabled or handicapped people. [9] This arm is usually fixed

on the food plate station to take the food quickly, and transfers it to the person's mouth. RAIL [14], Handy I [15] and MySpoon [16], [17] are some systems of this type. [9]

Figure 6: Workstation assistive robot. [13]

2) Stand-alone manipulators comprise [9] a robotic system fixed on the desk or any feeding station, whereby the positions of the required food or object is not known in advance and thus sensors should use for collecting the object position positions and sending it to the arm to transfer to the disabled people. This type has a limitation regarding the positions of the objects when the object is far from the arm and it cannot reach this point. Examples of these types are Tou robot [18] and the ISAC robot [19][9]. As an example, a simple robotic system with a dual-arm handling system allows Korean food such as boiled rice to be handled in an ordinary food container, as shown in Figure (8). [20] The first robotic arm (a spoon arm, arm #1) uses the spoon to transfer food from a container to the mouth of the user, and the second robot arm (a grab-arm, arm #2) picks up the food from a container and then places it on the spoon of a robotic arm. It divides the food processor into two sub-takes: picking up/releasing food and transferring it into the mouth of the user. [20] The two arms have different functions, whereby the design of the end-effectors of the two arms can be chosen to take up or release food stably, and the grab arm can use odd-shaped grippers as indicated by the bottom left-hand of Figure (7), as the gripper needs to stay away from the user's mouth. [20] If an end-effector has an unusual shape, it could pose a risk to the user when it comes near the person's face. A spoon arm has two degrees of freedom (DOF) to transfer the food to the spoon without altering the spoon's positioning: a grab arm includes a three-

DOF-SCARA planar motion array, and a one-DOF prismatic up and down motion joint and a gripper as shown in Figure (8) [20].

Figure 7: Assistive robot for self-feeding. Spoon arm (arm #1) uses a spoon to transfer the food from a container to a user's mouth. Grab arm (arm #2) picks up the food from a container and then loads it onto the spoon of arm #1 [20]

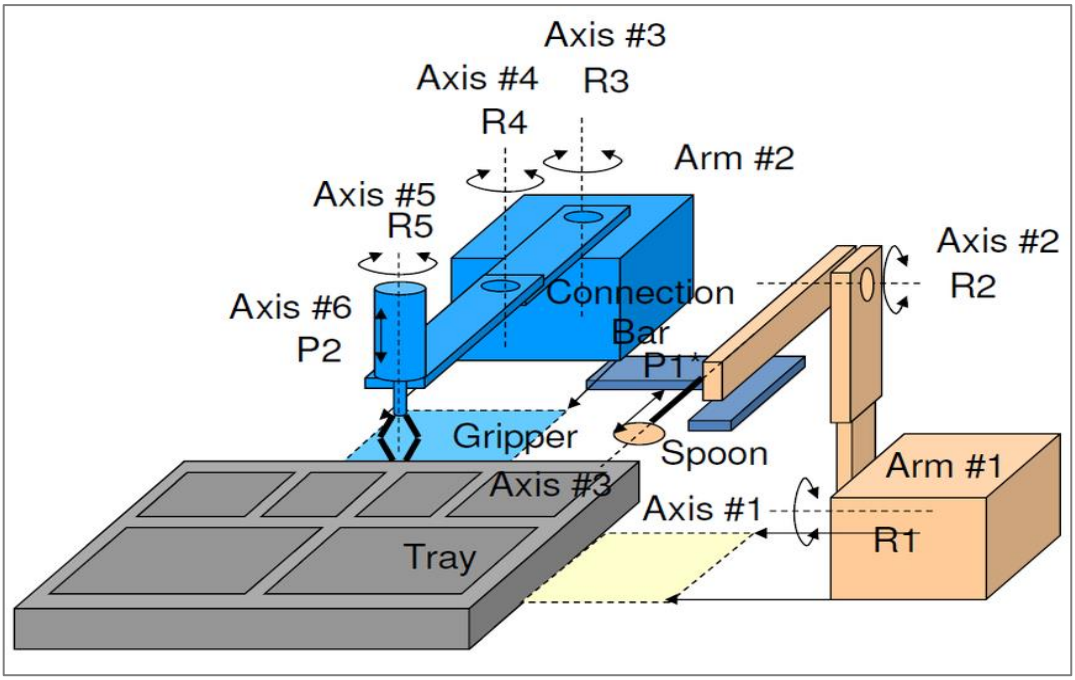


Figure 8: Joint configuration of a novel feeding robot for Korean foods. P1 (prismatic joint#1) is optionally applied. R = Revolute. P = Prismatic [20]

3) Wheelchair mounted manipulators [9] are represented by a wheelchair with a robotic arm, whereby this system allows disabled people to easily obtain objects such as food on a table that is far away from them. [9] These systems also have a limitation regarding the working area due to the fixed robotic arm on the wheelchair, which can be influenced on the arm working area. An example of this is MAUNS [21] [22], which uses a joystick, panel and voice commands, while another example is FRIEND with a voice commands system [23], AVISO [24], [25] and VICTORIA with a touch control screen [26]. Another type can be controlled using a keyboard and can also be controlled by using a joystick, namely a raptor arm [27][9].

Figure 9: Wheelchair with JACO robotic arm as a project for disabled people [28]

4) Mobile robots comprise a complete mobile robot with a robotic arm [9]. However, the arm can move independently from the movement of the wheelchair, which gives the disabled person more flexibility while carrying the object, and it also gives the person an option for sharing the object with another person. Examples of this type are WALKY [29], MOVAID [30], ARPH [31], HERMES [32], KARES II [33] and CARE-O- BOT[34].

As an example, [35] the robotic arm could appear in the briefcase of a laptop computer without removing any parts, with the user interface comprising a single web camera to receive the user’s eye movements, a computer that runs to the centre of the iris sensor and the student, and a display unit to indicate user feedback information and control boxes for detecting the user's eye motion [35]. The user interface comprises a web camera that offers VGA video and monitors the user's eye movements, a personal computer (PC) for photo-processing eye movements, and a display unit that provides simple feedback on a desk that is selected by turning LEDs on/off [35].

[36] Four LEDs provide feedback information with a four-bit binary value, and the position coordinates of the plate are then transmitted to the robotic arm when it is selected,

which then initiates a feeding program that allows the end effector of the robotic arm to reach the food on the select platform, collect and bring it to the mouth [36].

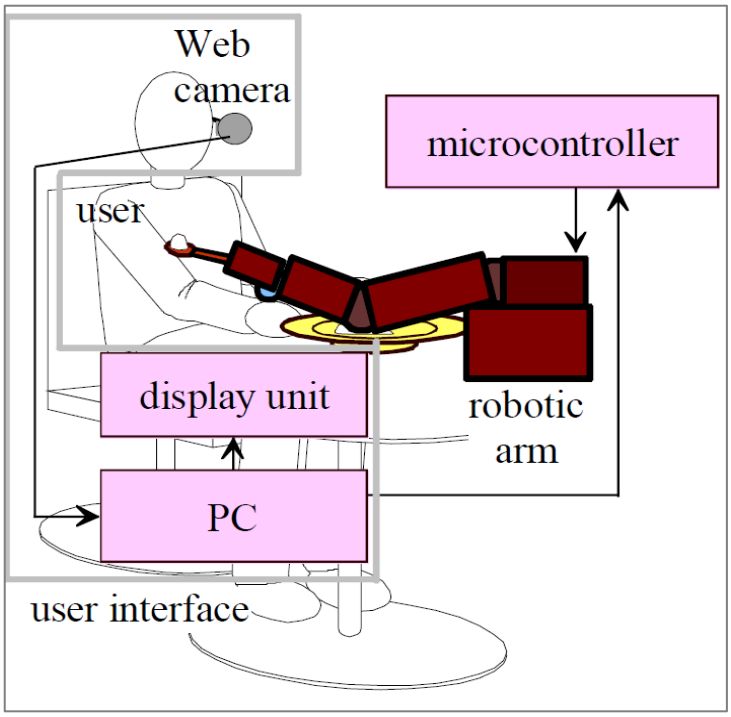


Figure 10: System configuration of mobile robotic arm and proposed user interface surrounded by a grey border, a robotic arm system with the proposed user interface [36]

5) Mats system comprises a robotic system [9] or arm that is fixed on the table or anywhere using the dock system and it serves to assist disabled people in a wheelchair or who are sitting on a chair in front of this arm. This system can be controlled by the user [9]. There are also examples of assistive robots that have been used as references to our work [37]. Assistive robotic arms increasingly enable users with extreme disabilities to carry out activities by themselves [37] the increased ability and dexterity of arms make them more challenging to control with simple, low-dimensional interfaces like joysticks and snap-and-puff interfaces. [37] However, in their interviews with everyday users of the JACO arm of Kinova, mode switches have been identified as a problem for both time and cognitive loads. It has also been objectively confirmed that the switching mode consumes approximately 17.4 percent of execution time, even for skilled users who use JACO [37].

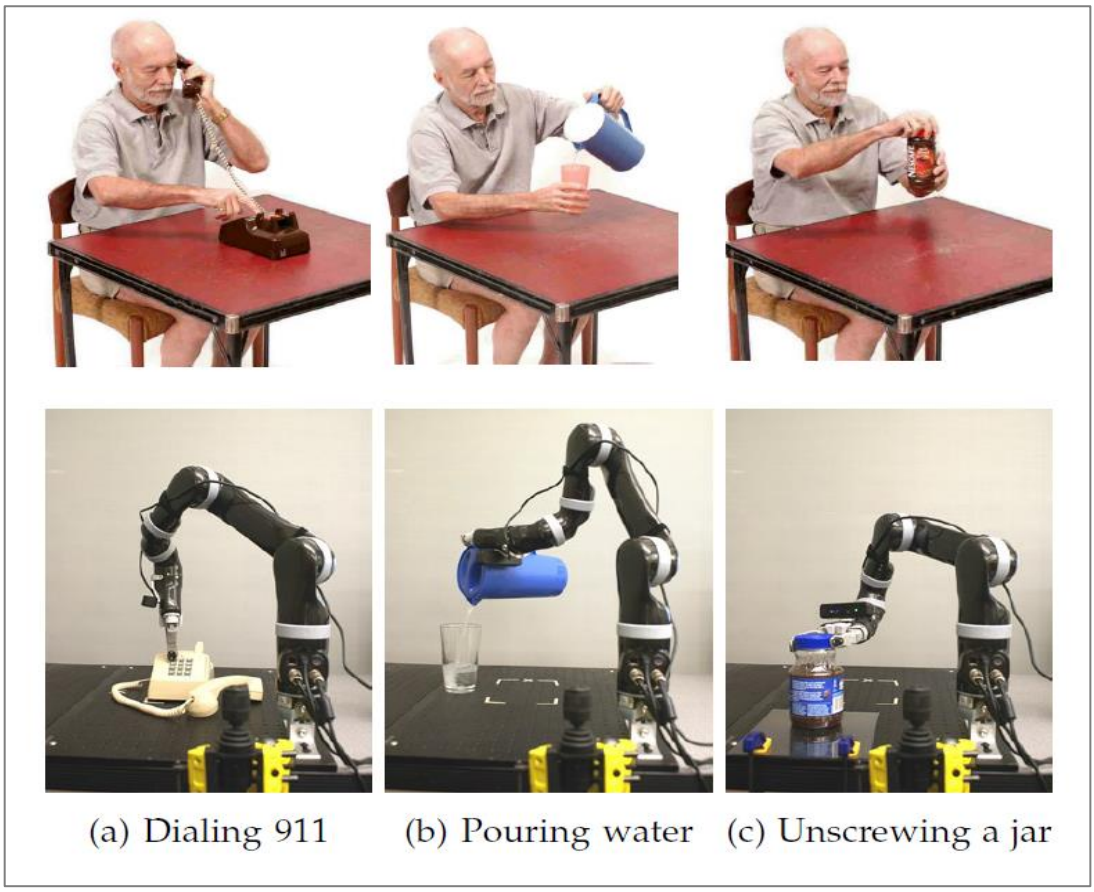


Figure 11: Three modified tasks from the Chedoke Arm and Hand Activity Inventory that able-bodied users perform through teleoperating the MICO robot [37].

iCRAFT [38] has two main parts: the computer with the eye-tracking interface, as well as the interface for the robotic arm. [38] The computer has a USB camera and an external monitor with the microcontroller and the bowls and spoon on the robot arm. [38] The process starts with the eye-tracking system interactions, in which the eye-tracking system calculates where the eye can track, and a servo controller connects the computer to the robot by servo connection. Through a servo controller wire, the servo control unit connects the robot arm, eye trackers send a signal to a robot's arm that moves to either one of the three bowls or the rest position, and the arm carries out an automatic action to collect food on a spoon, which then reaches the robot's arm [38].

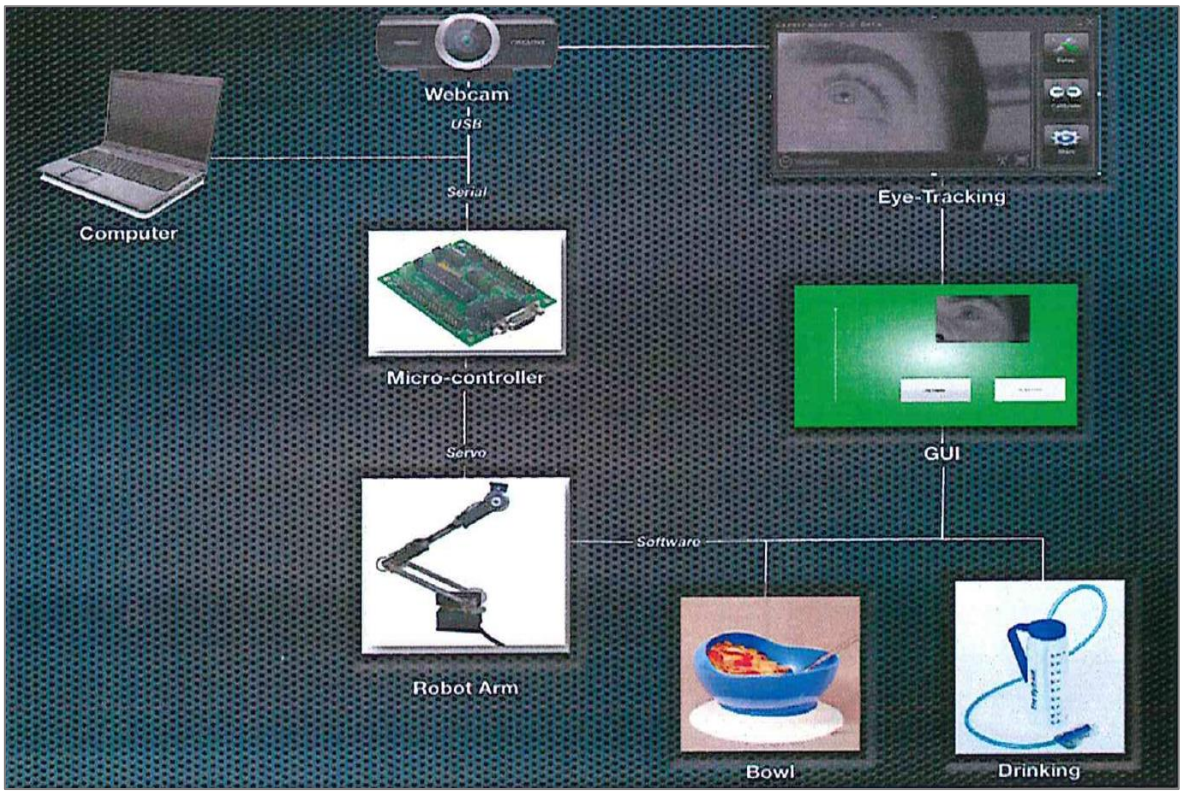


Figure 12: Design overview [38]

B. Simulated human robot interaction [9]:

The new development and the emergence of simulation programs have influenced the assistive robotic design, improving the implementation process through reducing prototype costs, and offering the user more flexibility for testing the system (hardware and software). Given the important of safety for disabled people, with this simulation we can avoid the risk of contact between the hardware components and the human. This software also supports open-source middleware, which encourages users to implement such massive assistive robot projects with low prices compared with the hardware. This project [39] combines the simulator type and the MATS projects type, this project concerned about how to do a specific task in an environment such as living room, kitchen. [39] One of the functions of this system is that it can be fixed to different places and can move along the home environment, whereby this arm can be fixed to the wheelchair, allowing flexibility through using the docking system. [39] Moreover, using simulation software offer the ability to provide more technical and functional specifications to provide the conceptual phase. All of these simulated elements are based on real dimensions and specifications [39].

The implementation process will be classified into the following three steps [39]:

- Building the environment: the working area such as the kitchen, building an architectural structure that is suitable for fixing the robotic arm, as well as specifying the communication system, power distribution and control units [39].
- Implementing the robotic arm: in order to design the robotic arm fitted to the docking system, arm kinematics such as end-effector movement (IK) should provide the interface control system [39].
- Designing the wheelchair: this wheelchair will move independently and be controlled by the user, while the communication control system should also be implement the position for fixing the arm [39].

Figure 13: Simulation of a MAT robot in the different tasks in a different environment [39].

The development [40] in the simulation software is an attractive factor for users due to the reduced execution time and lower effort compared with providing sophisticated software and hardware, reflecting one of the challenges that robots have to contend with when they enter the real world. [40] HRI simulation applications have used a MORSE simulator as an

excellent example for implementing the human-robot interaction. Some distinct purposes for the use of this simulation can be discerned in the HRI literature [40]. As described in terms of how HRI simulations have dealt with concurrent constraints arising from robotic simulation and digital simulation while remaining easy-to-use devices, MORSE is now already being used as a forum in several institutions, effectively facilitating human-robot interaction applications [40].

Figure 14: Simulation of a human model with an assistive mobile robot in MORSE [40].

[41] The simulation software represents the faster and easiest way to develop the robotic system, using the human-robot interaction option to integrate humans in the simulation loop. [41] Robotics can test and verify their research with simulators in a sandbox, which restricts risks within their chosen level of abstraction. A preliminary test involving real consumers in a pick-place-carry job for which positive outcomes will be collected shows the proposition. Accordingly, high-level programming initiatives can use a simulation to describe low levels (e.g. navigation, image processing, localization) to prevent dealing with related problems when testing the device [41].

Furthermore, the simulation setup can also help to test the element design of the onboard robotic system before attempting costly integration it has focused on human-robot interactions such as MORSE and how the human being can be controlled directly by existing codes or a human operator like any other device [41].

Figure 15: Simulation of a human walking and then picking up an object through the MORSE simulator [41].

In our project, several researchers have worked with handicapped robot, such as [42] They have developed a design prototype methodology (task-oriented design), whereby they used the simulator for designing a robotic arm especially for handicapped people, They studied the arm orientation through picking-up process, in which the robotic arm has more freedom through moving due to the rotation corresponding during the fixing station. Their experimental results confirmed that the developed arm accomplished the required task [42].

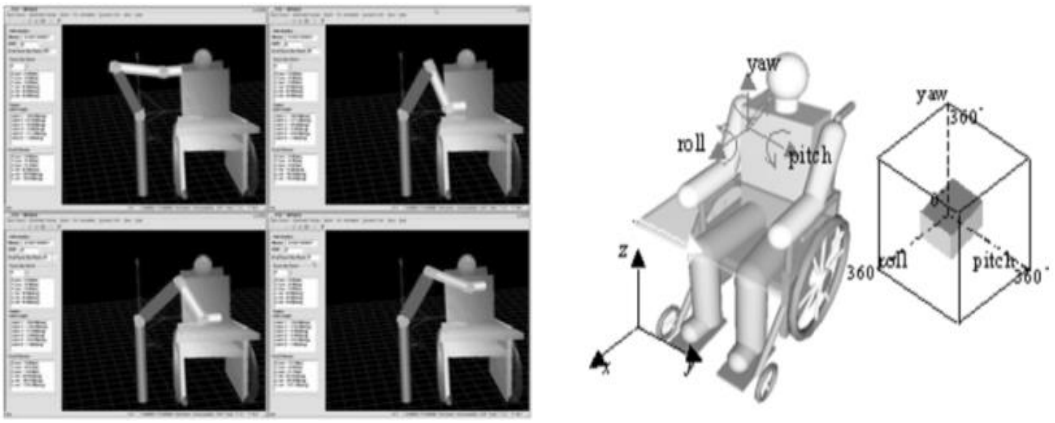


Figure 16: Simulation of robotic arm motion mechanism with the wheelchair [42].

2.4 Sound Recognition System Background

In the project, one of the important objectives was to achieve cooperation between the different systems especially the sound system, because this system cannot connect through message instructions using ROS topic as a carrier. However, after searching, we implemented this system in ROS, as will be explained in chapter five. [43] The main focus is on connecting speech and language with non-linguistic knowledge (perception of objects currently in the visual field), and abilities (e.g. object grasping or speech synthesis) presented as a natural language using a spoken dialog system on the Stanford AI Robot (STAIR). They wrote the ROS node to call Sphinx3, and modified the live program Sphinx3 to accept input from the ROS wrapper (which subscribes to the audio location topic to have audio clips)

[43]. [44] Advances in audio recognition have made it possible for a wide range of interactive voice systems to be successful in the realm over the past two decades. Indeed, the same techniques have proved promising in recognizing audio space non-speech events in recent decades [44]. Their paper presents a new software library – the ROS Open-Source Audio Recognizer (ROAR) – which offers a full set of end-to-end tools for online supervised learning of a new audio events, function extraction, automatic one-classes support vector machine model tuning and real-time audio event model tuning [44]. Implementation of the MetraLabs Scitos G5 humanoid indoor service robot with a natural interface based on language using technologies such as speech reconnaissance and speech synthesis allows the robot to accept and respond to simple voice commands [45]. The implementation of the human robot interface was modularly-structured and partitioned into packages, with a voice input comprising a vocal recognizer, some pre-process, a voice interpreter performing semantic input analysis, generating output and communicating with other software components, such as head control and voice output [45].

3 System implementation

The design process is the major phase of the simulation process, whereby we present the whole environmental design process, in which the first task involves the design of the environment and objects. The goals to be implemented are as follows according to this simulation Program:

1 It should have the capacity to create a reasonable sensor and input information that can be utilized by the control programming, which should correspond with real time.

2 It should give an identical ROS control interface as in real robots. This interface essentially comprises various ROS topics that can be utilized to send control massages to the different robot parts.

3 It should offer a graphical user interface that permits imagining the movements of the robot joints (revolute or translate), while making the crashing and gripping process between the robot part and some environment objects easy to implement.

3.1 Requirements and Software

A PC with an Intel i5 or higher and 4GB RAM as a minimal requirement. Linux (X 86 or X 86-64) as an operating system, and Python 3.4.2. qt creator C++ are the minimal requirements for installing the simulation and control programs, which are listed as follows:

3.1.1 Morse

MORSE is [46] a generic simulator for academic robotics focused on the realistic 3D simulation of small to large environments in- or outdoors, MORSE can be controlled completely through command line, and simulation scenes can be generated by simple Python scripts, and are largely written in Python, MORSE is an application that enables making simple and quick changes to source code, except in computing-intensive processes (such as 3D rendering or physics simulations) MORSE is designed to be modular, with new sensors, the new actuator, post-processing (such as the application of the noise function), adding new services, or even complete communication, MORSE has supported many middlewares such as ROS, yarp and socket. As a different environment in MORSE, the human-robot interaction is shown in the right-hand side of Figure (17)[46].

Figure 17: Different virtual environments in MORSE.[46]

3.1.2 ROS

One of the most difficult challenges in this research is how to create a robot control program and it is in the same centrepiece of other sub-systems, which means that too many types of robots are developed each day. However, these are controlled by a special program that uses one mediumware, and with ROS is able to control the different types of robots. As understood from [47], the Robotic Operating System (ROS) is used to interact with nodes via messages, whereby the nodes are calculation processes. ROS is very modular, and one robot control system usually contains several different nodes, such as voice recognition of one node, navigation control of the other node, and the master nodes uses the ROS Master to find, exchange messages or invoke the server vice [47]. If a node finds another node, the Master functions as a DNS server and they are connected directly to each other [47].

The communication between nodes would be by publishing messages to topics, the datastructure for those messages contains typed fields [47]. Those messages supported Standard primitive types (integer, floating point, boolean, etc.) as are arrays of primitive types [47]. The topics are messages processed through topics, nodes that are capable of publishing (sending) messages and nodes that can subscribe to messages to them [47]. Multiple editors and subscribers can have topics, and a node can post and/or subscribe to various topics [47].

The subject uses a publish/subscribe model and normally carries the bulk of the data into one system (e.g. motor speed measurements or laser scanner distance measurements) [47].

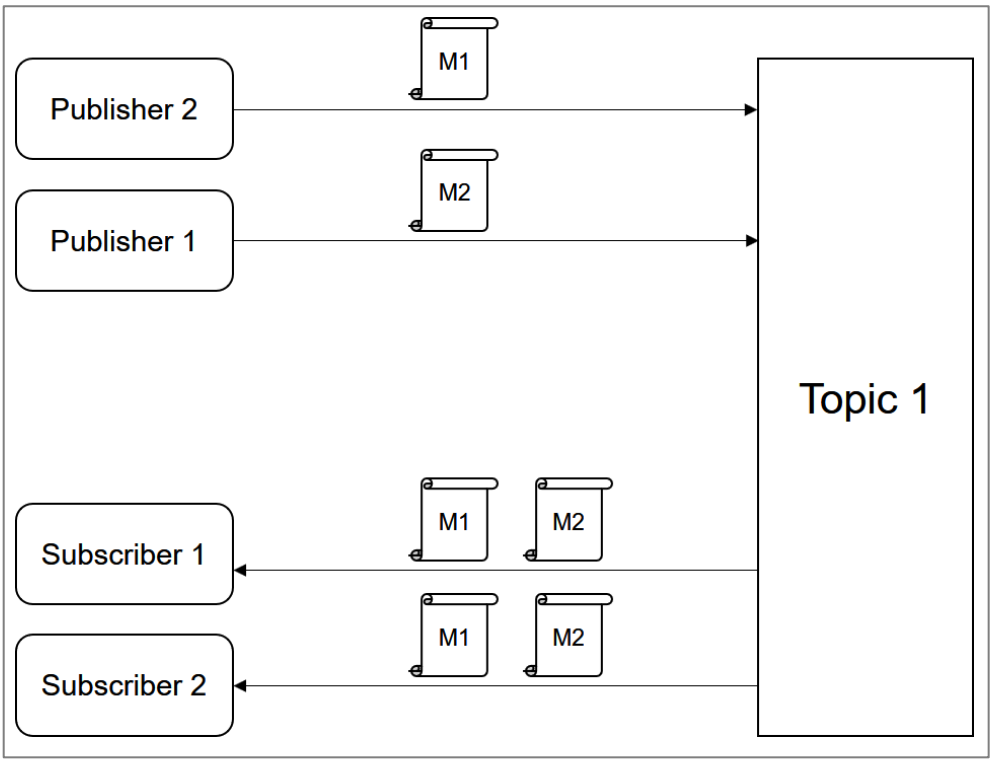


Figure 18: Illustration of a ROS topic shows the broadcast of two messages. Note that there is no guarantee that messages sent from different publishers arrive in the same order at all subscribers, despite it being the case here [47].

3.1.3 Blender

Blender [48] is a 3D open-source creation suite, whereby advanced users use blender's API for the Python ascription to customize the application and write special tools. Blender supports all 3D pipeline modelling, rigging, animation, simulation, rendering, composting, and motion tracker, even video editing, and creating games, minimum (basic use) hardware with 2Ghz dual-core support for SSE2, 2 GB RAM, 24-bit 1280 daily display, a mouse or track-pad, open GL 2.1 graphics with 512 MB RAM compatibility [48]. A number of features are important during the implementation of this project and are also represented the main purpose for using this kind of software:

A. Photorealistic rendering [48]

- Blender provides a real-time imagination using cycle rending type, which allows us to implement our environment (kitchen, furniture, dishes, etc.).

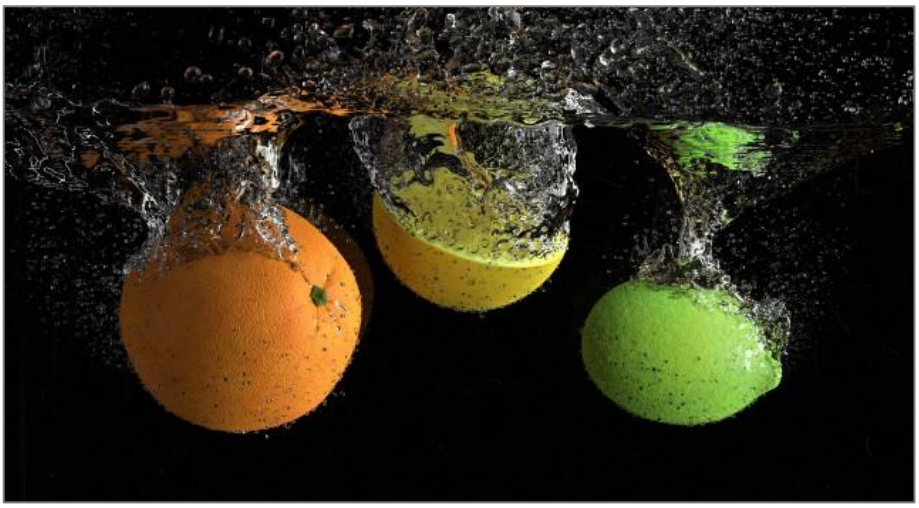


Figure 19: Photo realistic Rendering [48].

B. Fast rigging [48]

- Blender provides a kind of effective armature or structure, whereby the bones are represented the actuators of the robotic arm, and force is distributed using the weight painting method. The skinning of the arm gives us more flexibility over the arm motion.

Figure 20: Human model in Blender [48]

C. Animation toolset [48]

- Blender allows us to implement inverse and forward kinematics specially during the arm pose.

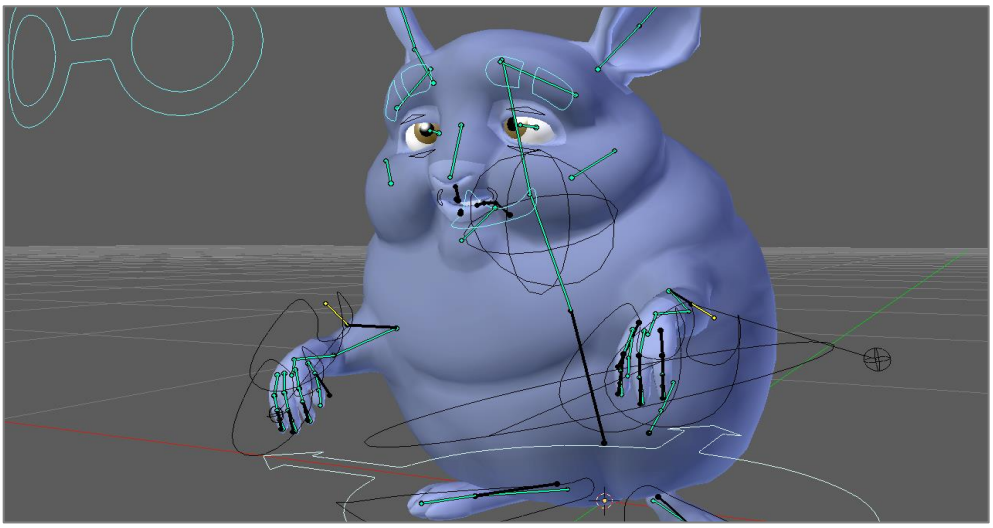


Figure 21: Mechanism of the motion for a rabbit model [48]

In this section, we have described the important features for using the Blender program. In our work, we used some of these features, including the physical properties and the lighting features in the environment drawing. Furthermore, the features related to the skinning, inverse kinematic, weight painting and distribution have been used. These features are mainly needed to implement the robotic arm into Blender. All of these feature will be explained in detail in relation to our work in the design chapter.

3.2 The schematic for the project

The simulation environment – as shown in Figure (22) – includes the human model, dishes, cup, the robotic arm and the semantic camera. Additionally, there is a sound package represented in the external microphone, which connects to the PC. The design and control process of the environment components will be explained with all of the details in chapters three and four, respectively.

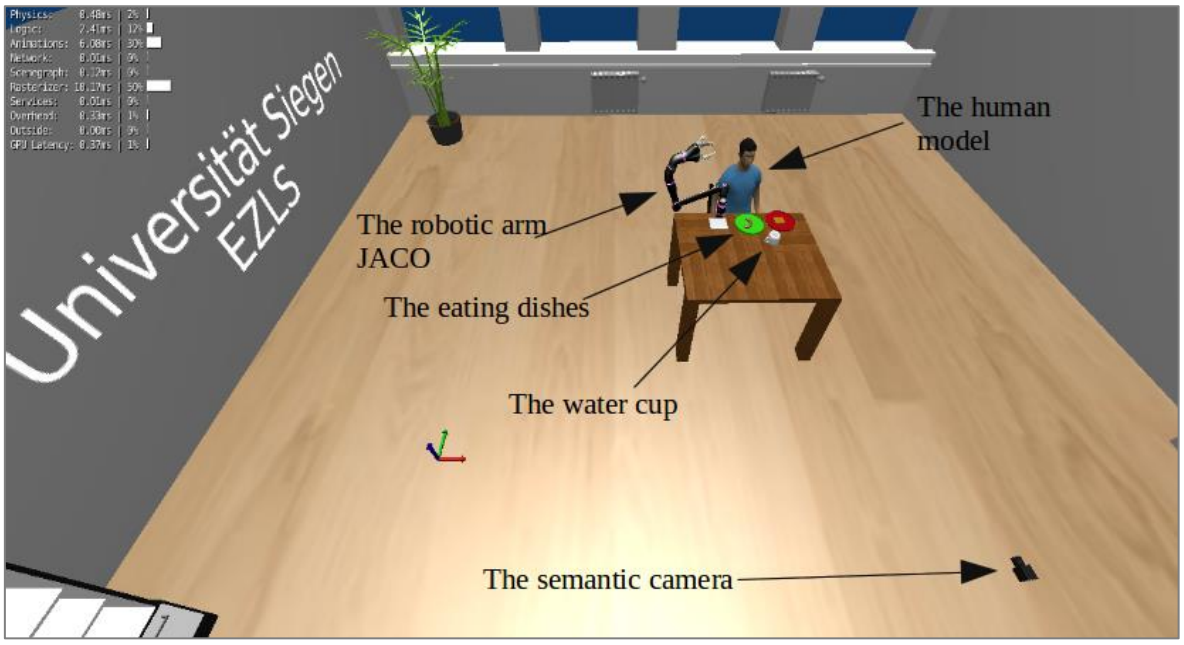


Figure 22: The simulation project.

In order to explain the whole project and collect ideas, the general schematic of the whole system and the data flow between the different systems are shown in Figure (23). After drawing and designing the arm, three Python scripts that should be written and then sorted as follows:

- The general or main Python script, which will be explained in the following sections.
- The second Python script, which will be used for:

1. Defining the actuators and sensors of the arm (because it is a new model and the design has to be implemented into the simulation program). These actuators, sensors and the designing process will be explained in detail in chapter three.
2. Defining the ROS topics used in the control process, for the arm, inverse kinematic or trajectory controller action, as well as for the orientation motion, i.e. this script is responsible for the arm motion strategies.
3. The third Python script for defining the function of the gripper actuator.

These latter two scripts should be stored in the Morse library as the definition scripts for the new arm model in the simulation program, before being called by the general or the main one.

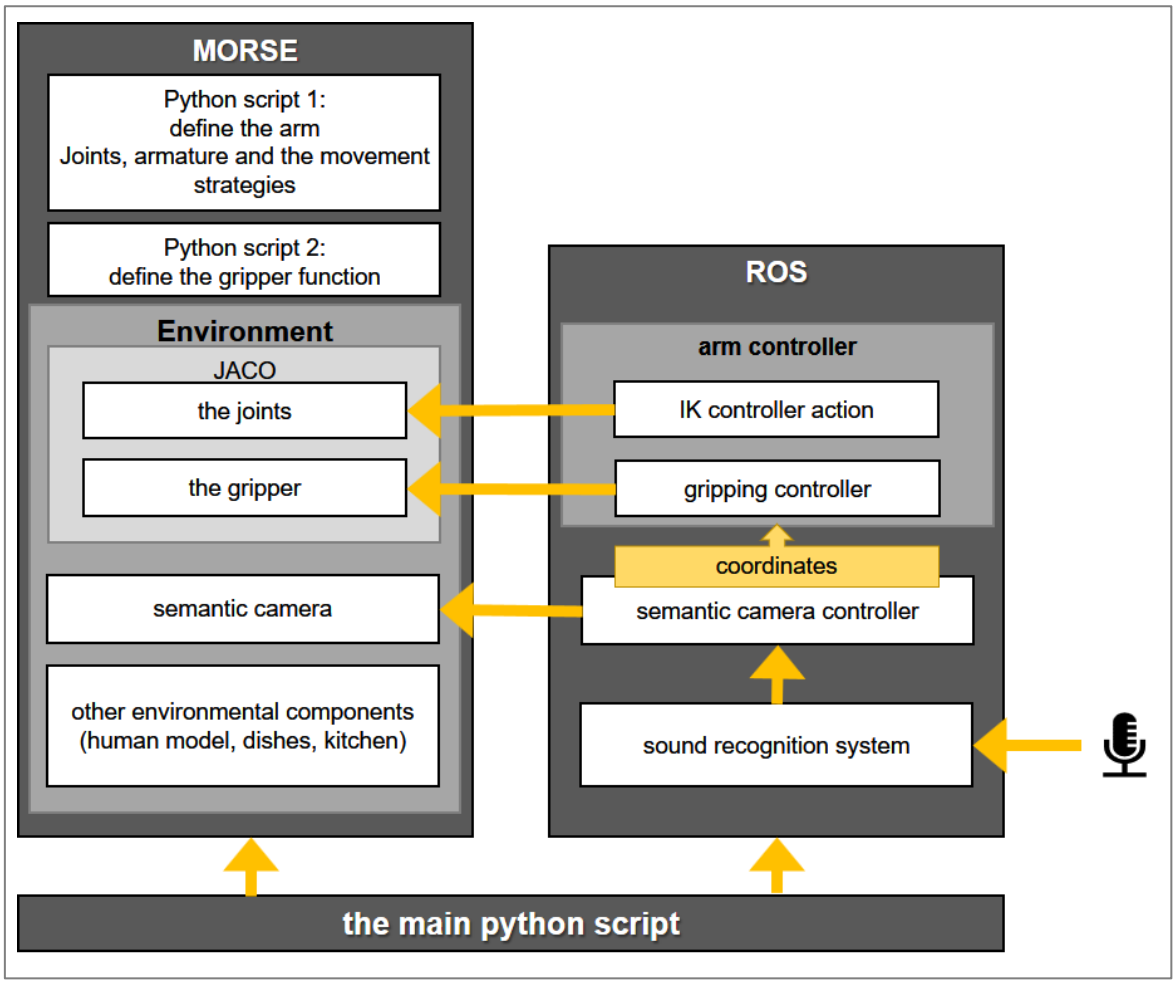


Figure 23: Whole schematic for the project.

As will be explained in chapter three, the ROS topics and control messages were programmed within the ROS node, which contains the semantic camera system, sound system and the other movements arm commands, whereby these messages will be either published to the control system node or subscribed from the system or the subscriber ROS node. The execution of the program will be as follows:

1 Run the main Python script.

2 Run the ROS nodes.

3 The sound comes from the microphone, and then is detected and the specific required signal is sent (i.e. the food type that the person wants).

4 The semantic camera controller sends a signal to the camera and then receives the food type details (coordinates, types, and label).

5 The coordinates are sent to the arm controller, comprising:

6 Inverse kinematic controller action, which it sends the target's coordinates to the arm.

7 Gripping controller, which sends the command to grip the object.

3.3 Design of the environment

The simulation of the environment represents the target of this chapter, which should be the same as the natural living environment. As previously mentioned, the simulation was chosen as it solved many problems concerning human safety, power and time waste, as well as prototype design costs.

Figure 24: Simulated objects in the environment.

Figure (24) shows the objects that have been drawn and then modulated, whereby (1) refers to the dishes, the cup is represented in (2) and segment (3) shows the simulated sausage and waffle. Finally, segment (4) shows the human model and table. After the drawing step using the Blender drawing toolbox, the material should be applied to the required texture. As an example, the material for the water cup (white colour)that would be without any other details is shown in Figure (25).

The other example is the material of the waffle, whereby there are two stages to undertake the required design: first, the material colour is specified as a combination of the yellow and nutty colour with many options for distribution, whereby in Figure (26) the colour has been chosen to accomplish the real design.

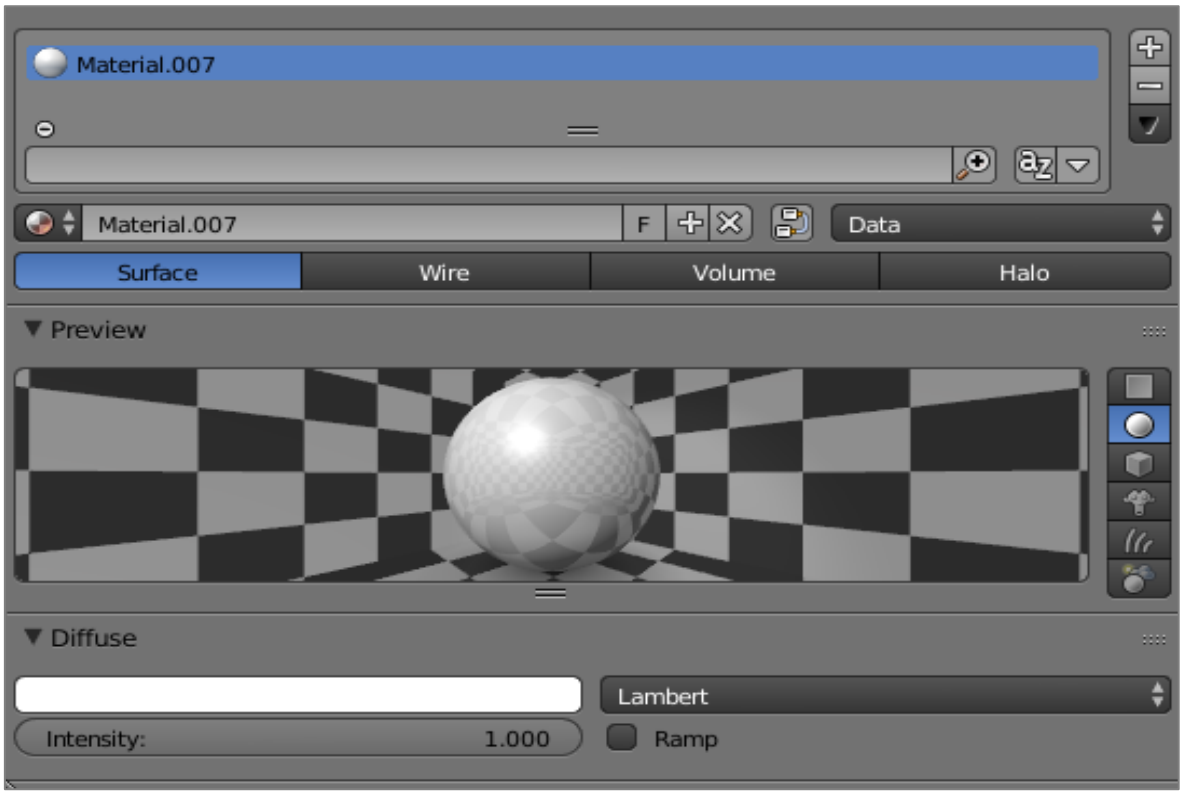


Figure 25: Material of the cup.

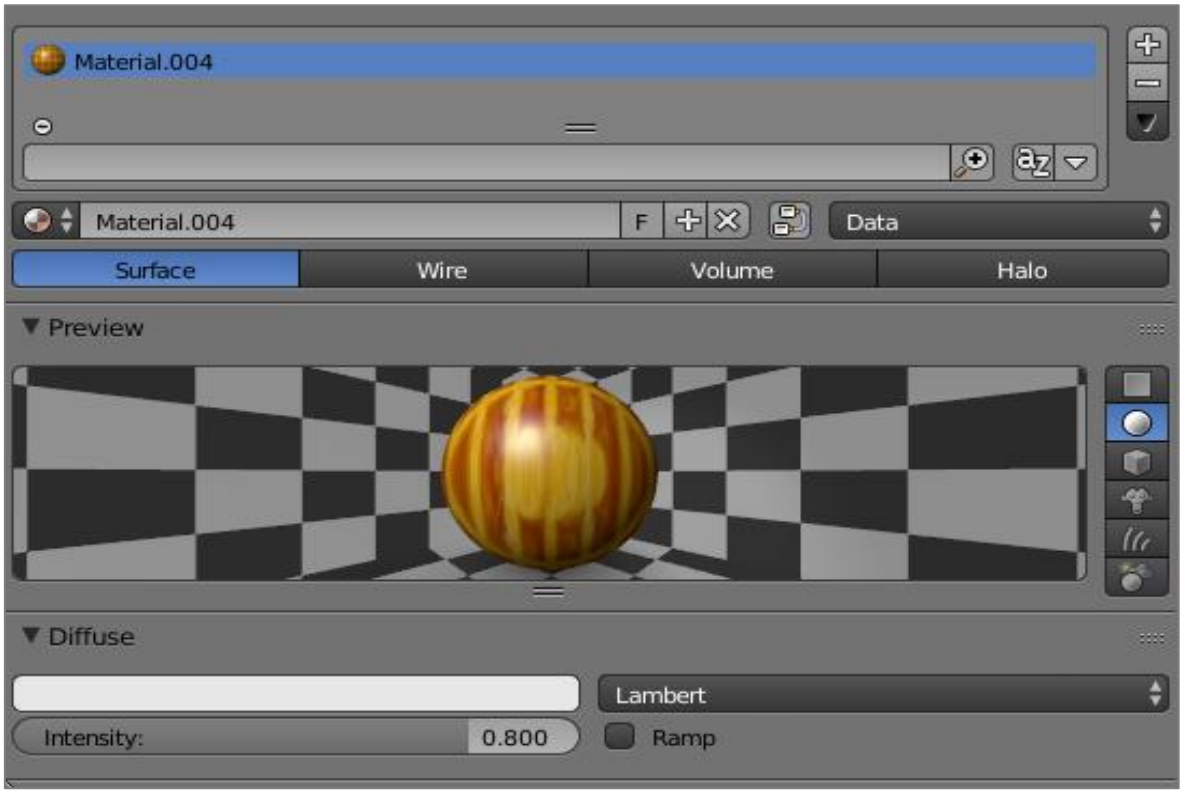


Figure 26: Material of the waffle.

The second step as shown in Figure (27) adds the image that should be applied to the material object (waffle) and then unwraps the image on the material, whereby for this process there are also many options for applying the texture, like smart unwrap, manual and projectiles.

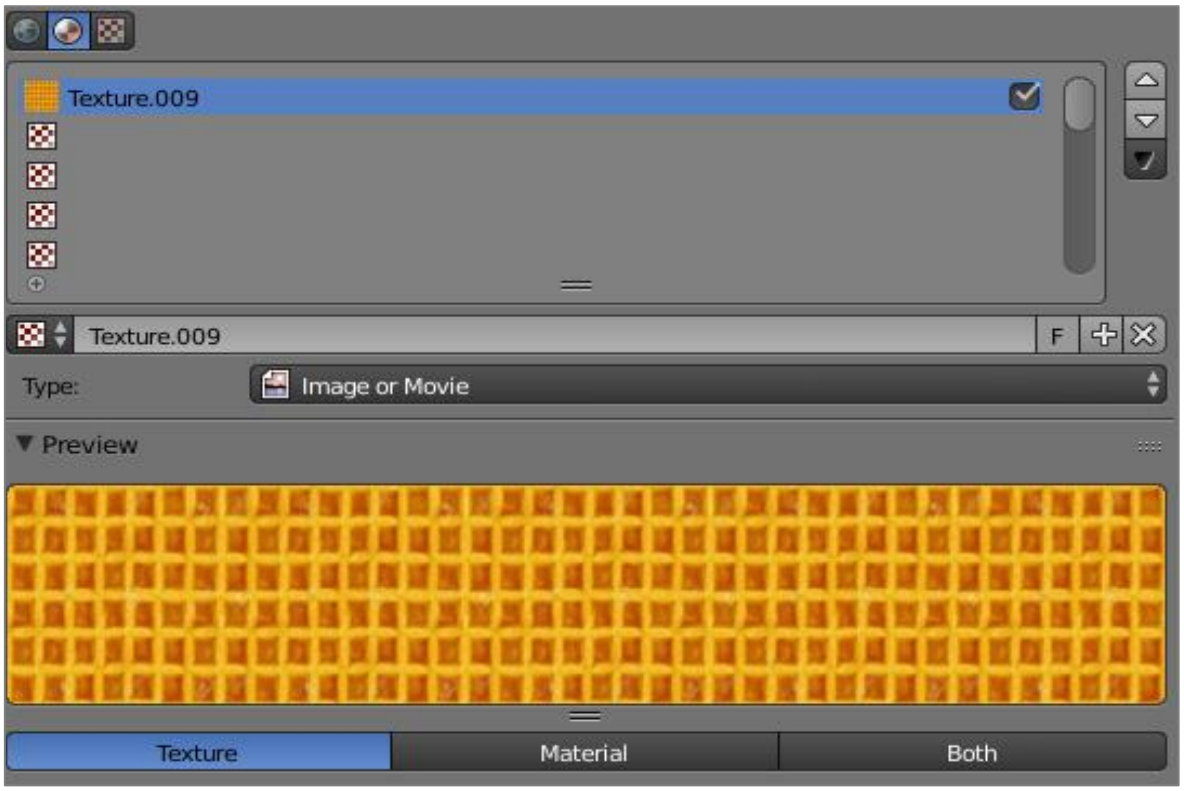


Figure 27: Texture of the waffle.

The previous steps represented the design process of the objects' surfaces dealing with the material colour, the shadow of the lights when reflecting from the object and the real object texture. The main objects of the whole environment comprising a kitchen room are shown in Figure (28).

This environment is represented by the kitchen room, which is a human model (the human who needs to eat or drink), whereby he sits in front of the table and over the table there are different colour dishes, with each colour referring to a varied food type, as well as the water cup and some kitchen furniture as shown in Figure (29).

After drawing the environment in Blender, it should be implanted in Morse. This involves using an object drawing toolbox in Blender, whereby the texture of the environment objects is changed to simulate nature, while the robotic arm JACO has also been covered with the required texture and material design through the design and drawing step.

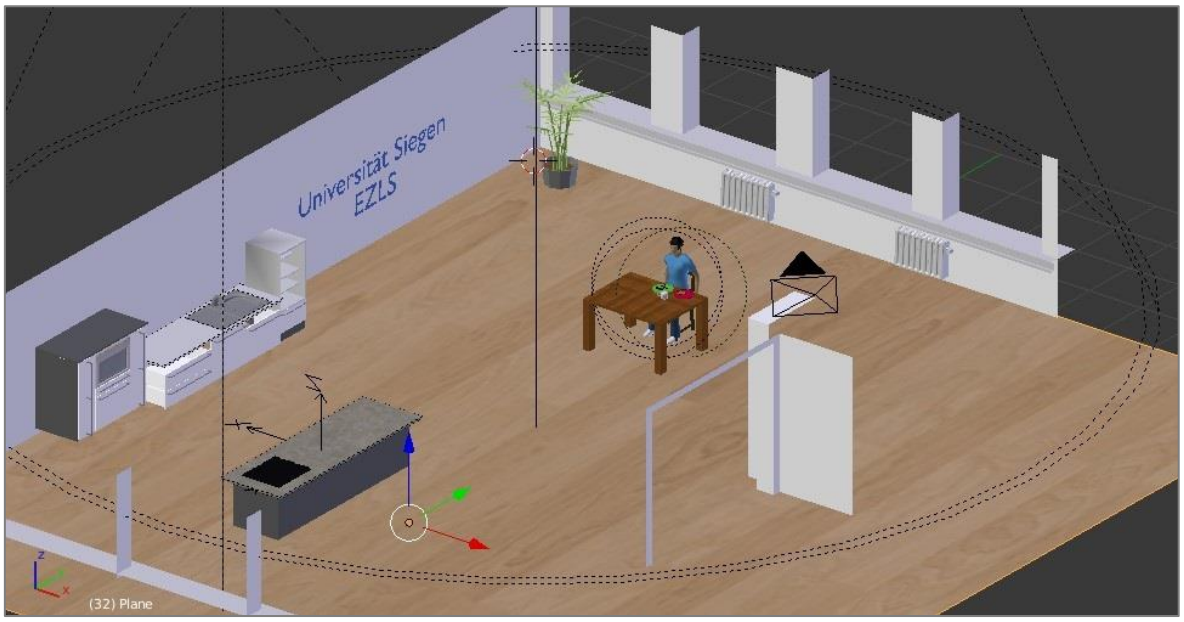


Figure 28: Whole simulated environment.

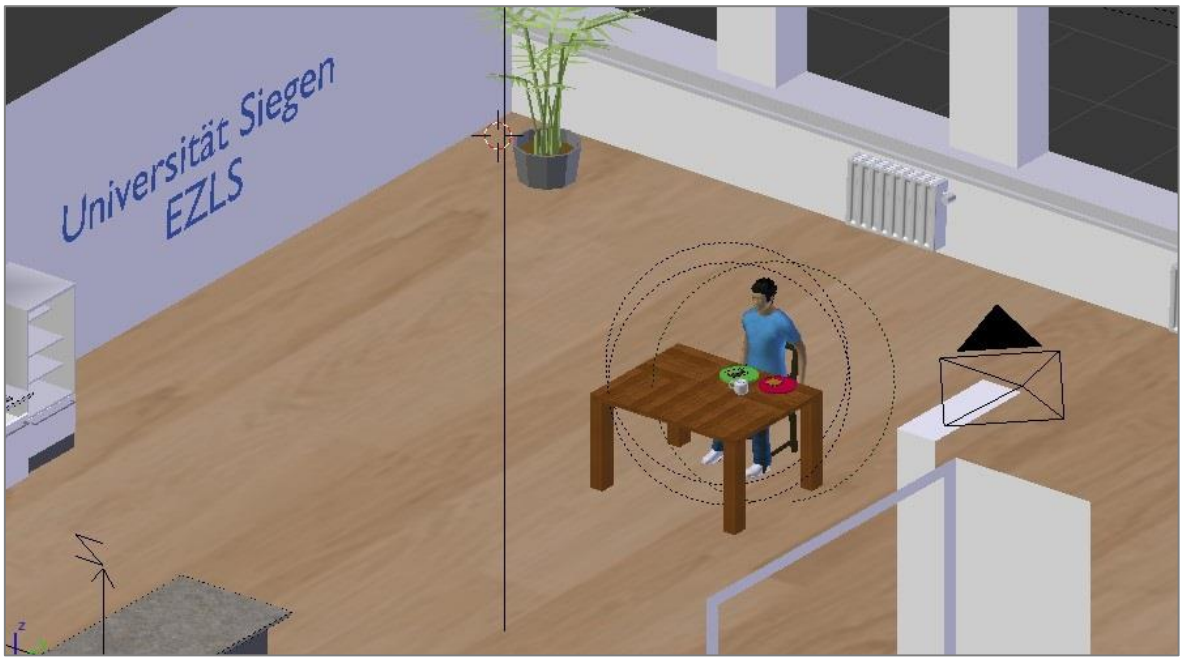


Figure 29: Simulated zoom environment.

3.4 General structure of the system

The whole system as a sub-individual system has already been described in chapters three and four, whereby the sequence of the data cycle will be as follows:

1 The handicapped person states their required food type

- ♦ RED: represents the food in the red dish, namely a waffle.
- ♦ GREEN: represents the food in the green dish, namely a sausage.
- ♦ WATER: represents the water drink.

2 The required word will be processed and recognized by the sound recognition system.

3 The camera detects the required object coordinates and stores them in the system.

4. The coordinates should be sent to the arm and then the arm moves to the required coordinates.

5. The arm picks up the required object and then moves it to the mouth.

6. The arm moves back to put the rest of the food in its position.

Each of the food types will be represented as a single cycle. When the handicapped person chooses the food in the red dish, the red cycle will begin until the end. This ensures that the eating process is carried out with some kind of flexibility, which means that any noise from an external source such as human sound or mistake through speech will be ignored. Another important factor affecting the design is the human model, involving a static model with no ability to move one's mouth or eat. In this situation, it is necessary to know whether the person has finished eating yet. In order to solve this problem, we considered a time for each process regarding each of the food types or drink cycle.

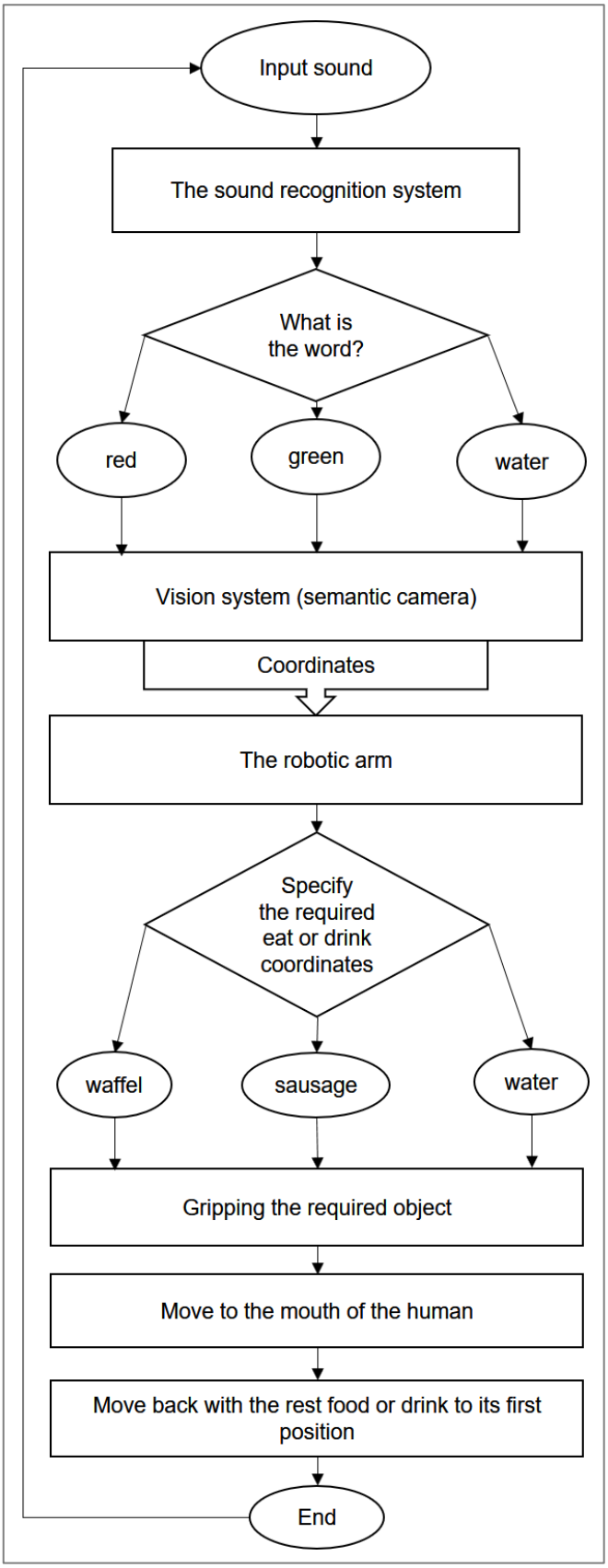


Figure 30: Flow chart of the program.

The time devoted to each process has been specified and could be extended or reduced according to the required design. The sound recognition system can detect the required word in time (msec). The specified time for moving the arm to the required target was considered

as 3 seconds for gripping the food and eating. As explained before, all of this can be changed by the user. The general flow chart is shown in Figure (30).

According to the flow chart, the program should be written to accomplish the required functions as follows. The program comprises two parts: the first part in Morse using a Python script and the other in ROS using C++. The main challenge was how to add a new robotic arm to the simulation program with the same specifications and properties for the real arm, as explained in chapter three. The programming steps for the whole project are divided into two parts: the Python script for Morse representing the output, i.e. the environment for displaying the whole simulated objects, the required sensors and actuators of the robotic arm and the middleware that controls the systems; and the C++ program managing the ROS control messages and the communications between the different systems to undertake the variety of functions such as the gripping process, the sound recognition process, the movement processes and the semantic camera system process.

Python script

1 Call the Blender file of the robotic arm (the mechanical design that has been drawn).

2 Define the armature of the bones as actuators for the robotic arm.

3 Specify the method that can control the armature:

- ♦ using a trajectory action server.
- ♦ using an inverse kinematic.

In our project, we used the inverse kinematics (IK) method.

4 Specify the end-effector, which presents the IK target when the arm is moving towards the goal.

5 Add the gripper and the orientation motion actuators.

6 Add the semantic camera and place it in the correct position.

7 Add the stream middleware, which will control the sensors and armature actuator, in our project using ROS.

8 Add the environment (the table, dishes, the kitchen room with all of the physical specifications).

3.5 Connection control between the systems

In this section, the data flows into the ROS system through the different ROS nodes, whereby the system will begin from the sound and then the semantic camera system until the end of the execution process.

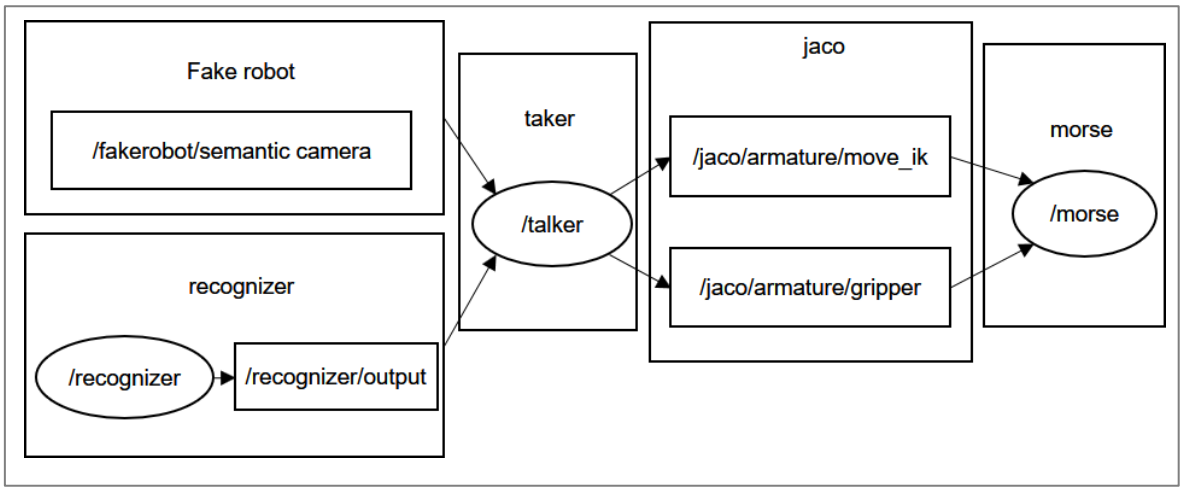


Figure 31: ROS nodes data flow for the whole project.

In Figure (31), the overall schematic ROS nodes and topics are presented, showing the data flow when a word comes from the system of the sound recognition and is then processed into the recognizer ROS node. On the other side, the semantic camera node will detect the object with its coordinates. The data is collected from two system nodes and then crossed to the JACO ROS node through the talker node, which presents the connected chain between the main system (JACO arm) and the other systems (the sound and semantic camera system). Subsequently, the data flows to Morse (the whole environment) to give the output of the system simulation.

4 Robotics system

In this chapter, we will describe the robotic system from the starting stage (the animation design) to the control and implementation, which represented the final stage. During this stage, we will display the problems that occurred with details (i.e. the algorithms used and the improvement of these algorithms, including the new contributions that have been implemented and how).

4.1.1 JACO arm specification

'The JACO robot arm – developed by Kinova at its state-of-the-art department – is a revolutionary device designed for multiple professional applications. [28] The arm is shown in Figure (32), and it is a leading product in a new generation of lightweight portable robotic tools that enables users to interact with their environment with complete safety, freedom, and effectiveness. JACO moves smoothly and silently around 6 degrees of freedom with unlimited rotation on each axis, whereby the axes are aluminium compact actuator discs (CADs) of a unique design, each JACO robot arm comprises two distinct sets of three identical, interchangeable, and easy-to-replace CADs linked together by a ZIF (zero insertion force) cable [28]. and its main structure – entirely made of carbon fiber – delivers optimal robustness and durability as well as a cutting-edge look-and-feel [28]. The arm [28][49] is mounted on a standard aluminium extruded support structure that can be affixed to almost any surface, and the gripper comprises three under-actuated fingers that can be individually controlled, whereby their unique bi-injected plastic structure (patent pending) endows them with great flexibility and unrivalled grip (the gripper two types are shown in Figure (33)). JACO technology allows the fingers to adjust to any object whatever its shape: as a result, they can gently pick up an egg or firmly grasp a jar. Some of the important arm properties are the total weight of 4.4 kg, maximum load of 1.5kg, reach of 90cm [49][28].

Figure 32: JACO manipulator robotic arm.[28]

Figure 33: Gripper for JACO arm (two fingers, three fingers).[28]

4.1.2 The DH parameters frame position

By definition, DH parameters describe “the position and orientation of the links and joints that make up the robotic arm” [50].

As modified from [51][52], Kinova provides DH parameters to the end user. As a movement of the end-effector to a certain known position, the kinematics enables us ignoring strength, torque and inertia, and simply concentrating on the manipulator position in space, regardless of load of the manipulated object and arm inertia while moving. [51][52] The DH parameters for the arm kinematic chain are applied when the joints are connected and driven by the actuators, whereby each joint connection establishes a degree of freedom (DOF) with

a 6-DOF arm. Six joints are found, and the base (link 0) shall be excluded as a degree of freedom because the links maintain a fixed connection between joints. [51][52] Reference frames shall be used to position the manipulator at a desired position by applying DH parameters and coordinating transformations (see Figure (35)). The DH link coordinates and joint parameters are shown in Figure (34) and described as follows [51][52]:

For the link structure [51][52]:

- a_i (length of the link)
- α_i (twist of the link)

For the relative position between links [51][52]:

- d_i (distance between the two x-z normal)
- θ_i (angle between the two-x normal).

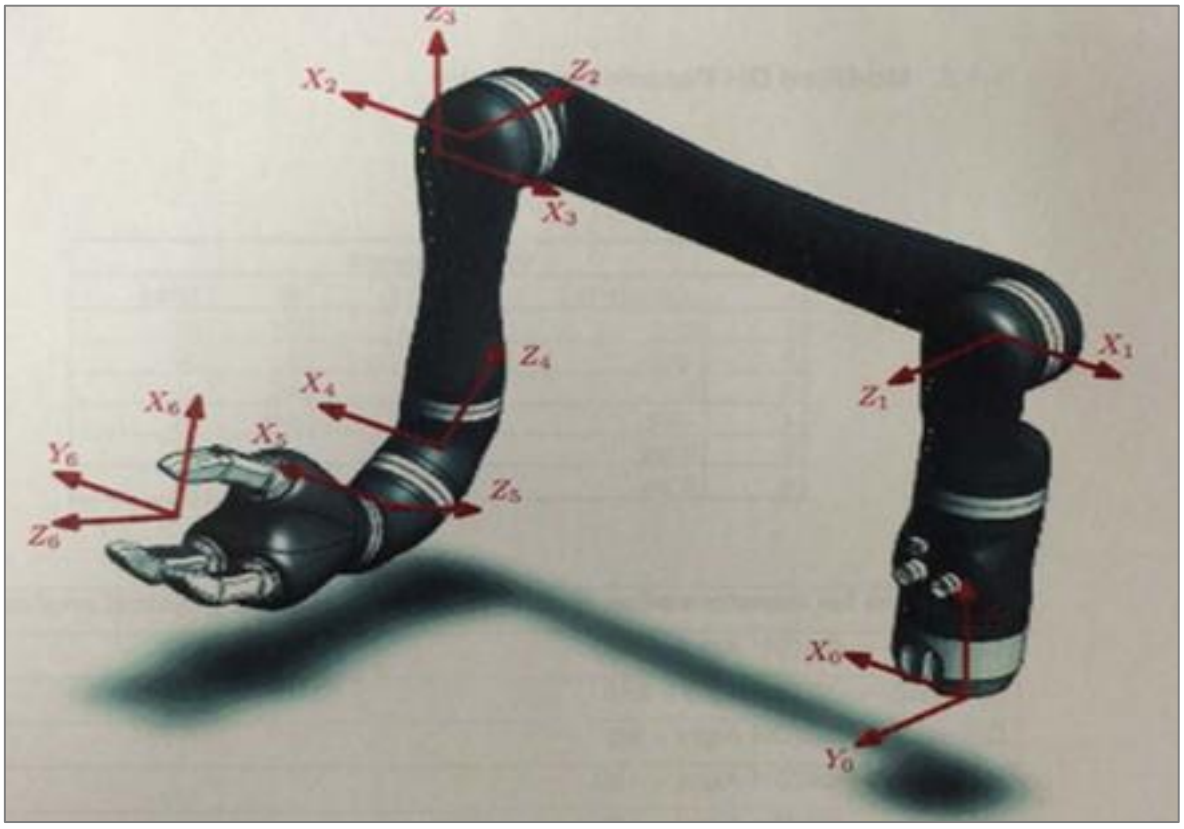


Figure 34: Classic DH parameters frame position.[51][53]

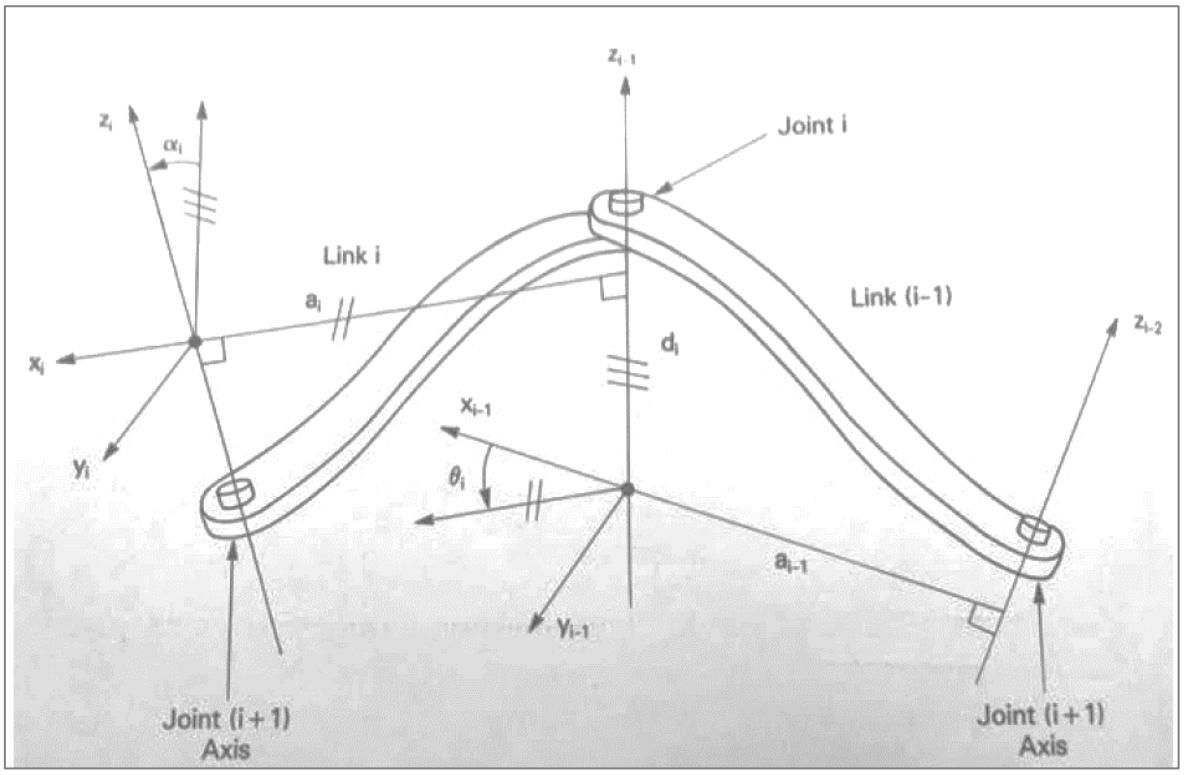


Figure 35: Link coordinate frame and joint parameters [51][52].

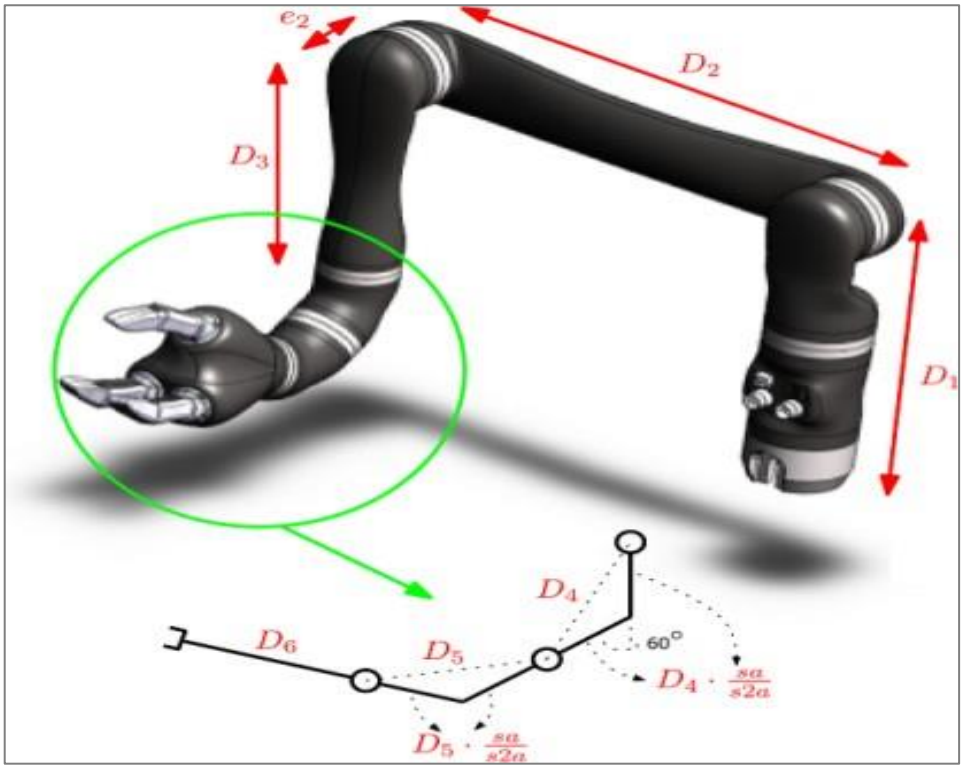


Figure 36: JACO arm link lengths [54].

As modified from the original Equations in [51] and [52], the DH parameters are used to associate each connection with neighbouring relationships and identify each link-joint pair

after the reference frames are fixed. In a transformation matrix, these parameters are used to connect the following frames to create a frame of the link coordinate [51][52]:

- The first rule is the z_{i-1} axis lies along the axis of motion of the Ith joint [51][52]
- The second rule is the xi axis is normal to the z_{i-1} axis directed toward the z_i axis [51][52]
- The third is that the y_i axis is defined by the cross product of z_i and x_i then the three axes form a right-handed system[51][52].

Once the link parameters and reference frames have been determined, each individual can correlate successive frames (i-1) with I via translations and rotations with the two reference frames. For example, see Equation 1 [51][52]

$$A_{(i-1),i} = Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i}$$

$$= \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

$$= \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

"The matrix A is the transform from one joint to another via [51][52]

- A rotation around z,
- A translation along d,
- Another translation along a
- And finally, a rotation around x. In the matrix, ai and αi are constant, whereas d_i and θ_i are joint variables. The sub matrix in blue depicts rotation, the sub matrix in orange depicts translation, the perspective transforms in red and the scaling factor is in green" [51][52].

As understood and modified from [54], one would ask how the transformation matrices of each connection can be obtained analytically. In order to derive these, it is necessary to understand how the various reference frames are physically connected and ascertain this with the DH parameters. The classical manipulator DH parameters and the length values of each link are shown. Table 1 provides the arm dimensions, as also shown in Figure (36) [54].

Robot length values (in meters)			Auxiliary variables	
D1	0.2755	Base to elbow	aa	$\frac{\pi}{6x}$
D2	0.4100	Arm length	ca	cos(aa)
e2	0.0098	Joint 3-4 lateral o set	sa	sin(aa)
D3	0.2073	Front arm length	c2a	cos(2aa)
D4	0.0741	First wrist length	s2a	sin(2aa)
D5	0.0741	Second wrist length	d4b	D3 + D5($\frac{sa}{s2a}$)
D6	0.1600	Wrist to the centre of the hand	d5b	D4($\frac{sa}{s2a}$)+ D5($\frac{sa}{s2a}$)
			d6b	D5($\frac{sa}{s2a}$) + D6

Table 1: Arm lengths and auxiliary parameters [54]

Once the coordinate frames have been established and the values for the DH matrices are known, the position of the JACO arm end effector with the base can be determined via the following transform: [52][51]

$$\mathrm{T} = \mathrm{A_{01}A_{12}A_{23}} \ldots \mathrm{A_{(n-1)n}} \quad (2)$$

As the Kinova JACO arm is a 6-DOF manipulator, it gives the expression [51]:

$$\mathrm{T} = \mathrm{A_{01}A_{12}A_{23}A_{34}A_{45}A_{56}} \quad (3)$$

The T matrix works on the first vector position – as shown in Equation 4 - the end effector final position can be found by applying the DH on the initial end effector position [55]:

$$\mathrm{FinalPosition} = \mathrm{T[P_xP_yP_z1]^T} \quad (4)$$

When the forward cinematic solution – as shown in Equation 3 is established – the end-effector can be located in space given the values of "θ" and "α," since "d" and "a". [55]

Besides, classical DH parameters are shown in Table 2. The variables in that table are listed in Table 2. The transformation matrices for each connection – according to the DH convention – are the same as each link: [54]

$$T_n^{n-1} = \begin{bmatrix} \cos(\theta_i) & -\cos(\alpha_i)\sin(\theta_i) & \sin(\alpha_i)\sin(\theta_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\alpha_i)\cos(\theta_i) & -\sin(\alpha_i)\cos(\theta_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

i	α_{i-1}	a_{i-1}	d_i	θ_i	DH Algorithm Angle	Physical Angle
1	$\frac{\pi}{2}$	0	D1	q_1	q_1	$-q1_{Robot}$
2	π	D2	0	q_2	q_2	$q2_{Robot} - 90°$
3	$\frac{\pi}{2}$	0	- e2	q_3	q_3	$q3_{Robot} + 90°$
4	2·aa	0	-d4b	q_4	q_4	$q4_{Robot}$
5	2·aa	0	-d5b	q_5	q_5	$q5_{Robot} - 180°$
6	π	0	-d6b	q_6	q_6	$q6_{Robot} + 90°$

Table 2: The DH diagram and angle transformation.[54]

The JACO2 physical angles must also be converted to the angles of the Denavit-Harteberg algorithm, as shown in Table 2. The DH parameters are obtained after determining the various coordinate frames for each link shown in Figure (34)[54]. Since the DH parameters are known, by applying Equation 5 the forward kinematics model can be calculated using Equation 2, and it is now interesting to introduce the difference in the kinematics model [54].

4.1.3 Robotic arm model

As understood and modified from[50][54], suppose that an arbitrary point P in space has to be shown regarding the base frame O_0. If its position is only known from the reference frame O_1, one way to achieve this is by taking into account the position from that reference frame, adjusting the point orientation to match that of the base frame and adding the distance between the origin of both reference frames. [54]

The conceptual mapping form for the general transformation of a vector from its description in O_1 to a description in O_0 would be:

$$P^0 = R_1^0 P^1 + O_1^0 \quad (6)$$

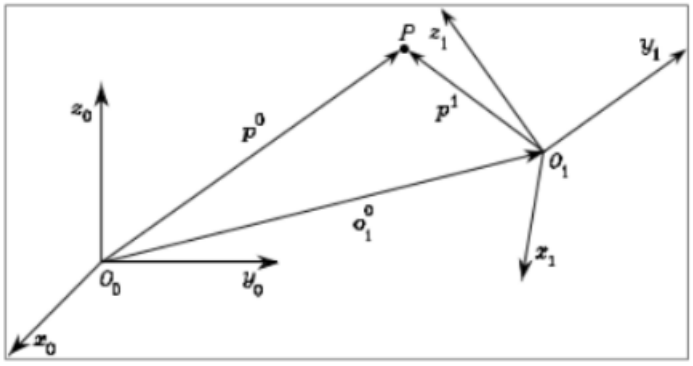


Figure 37: The coordinates system for the point P [54][50]

the coordinates system for the point P, where $O^0{}_1 \in R^3$ is the distance between the reference frames and $R^0{}_1 \in$ SO (3) [54]

According to this Equation, the general transformation matrix represented by T and It would be [54]

$$T_1^0 = \begin{bmatrix} R_1^0 & O_1^0 \\ O^T & 1 \end{bmatrix} \quad (7)$$

as a general form, the homogeneous transformation matrix would be as follows:

$$T = \begin{bmatrix} 1 & 0 & 0 & M_x \\ 0 & 1 & 0 & M_y \\ 0 & 0 & 1 & M_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

The M translation vector is represented by: $\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix}$ (9)

The rotation matrix is written according to the axis of the rotation:

$$R^3x^D(w) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(w) & -\sin(w) \\ 0 & \sin(w) & \cos(w) \end{bmatrix}$$

$$R^3y^D(\varphi) = \begin{bmatrix} \cos(o) & 0 & \sin(o) \\ 0 & 1 & 0 \\ -\sin(o) & 0 & \cos(o) \end{bmatrix} \quad (10)$$

$$R^3z^D(K) = \begin{bmatrix} \cos(K) & -\sin(K) & 0 \\ \sin(K) & \cos(K) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

T_0^1 is usually [54] referred to as a homogeneous matrix for transformations. [54] This captures information about the distance between the origins and the direction of the frames by means of a rotation matrix. [54] These four-dimensional matrices are essential for drawing on the movie model. The transformation matrix of the base to the TCP of the JACO arm is shown in Equation 3 [55][54]

4.1.4 Differential Kinematic of the JACO arm model

As modified from the original Equations in [54], the end effector velocity comprises linear and angular velocity. Let us suppose that the end effector linear velocity is V_n^0 and its angular velocity is W_n^0.

This relationship is contained in the matrix called Jacobian, which is important for analysing singularity and inverse kinematics, but it is also useful for calculating torques on

the joints when a certain force is applied on the end-effector [54]. The joint velocity is $\dot{q}$. [54]

$$V_n^0 = \mathrm{Ja_v}(q)\,\dot{q} \quad (11)$$

$$W_n^0 = \mathrm{Ja_w}(q)\,\dot{q} \quad (12)$$

Where $\mathrm{Ja_v}(q)$ is a (3 X n) matrix for the linear end-effector V_n^0 velocity with the joint velocity $\dot{q}$ and $\mathrm{Ja_w}$ is also a (3 X n) matrix detailing the relation from the angular end-effector velocity to the joint velocity [54]. In combination, these sub-matrices form the Jacobian matrix Ja. [54]

The total end-effector velocity (for linear) = $V_n^o = \sum_{i=1}^{n} \frac{\partial ja_v}{\partial q_i} q_i = \sum_{i=1}^{n} Ja_{vi} q_i$ (13)

The total end-effector velocity (for angular) = $W_n^0 = \sum_{i=1}^{n} W_{i-1,i} = \sum_{i=1}^{n} Ja_{wi} q_i$ (14)

If the joints are prismatic or revolute, the joint velocities qi are expressed differently. In this case, the JACO2 has six joints with frequent hem revolutions, whereby the angular and linear velocity for a revolving joint is indicated as follows, modified from [54]:

- For the revolute joints

When the i^{th} is a revolute joint, then the rotation axis is Z_{i-1}. When $W_{i-1,j}^{i-1}$ represents the angular velocity of the link I, the frame i – 1. Then, we have: [54]

$$W_{i-1,j}^{i-1} = q_i Z_{i-1}^{i-1} \quad (15)$$

- For the prismatic joints:

when the joint is prismatic, the frame I will be translated to the frame i-1, then [54]:

$$W_{i-1,j}^{i-1} = 0 \quad (16)$$

In this case, the JACO has six joints, all of which are revolute, so the angular and linear velocity for a revolute joint for subsequent links is represented as: [54]

$$V_i = V_{i-1} + W_i \times r_{i-1,i} \quad (17)$$

$$W_i = W_{i-1} \times \emptyset \cdot Z_{i-1} \quad (18)$$

Where [54] Z_{i-1} is often the vector of the unit, the joint i axis and the r_{i-1}, i corresponds to the total distance from origin when the frame i is coordinated regarding the origin of the frame i–1. [54] This means that the speed in connection i is the same as in connection i–1 with the increase of a component associated entirely changing due to the rotation of link i. Subsequently, if the calculation of the linear speed of each link is made regarding the frame, the equality presented in Equations (13)(14) is reordered as [54]:

$$\begin{cases} Ja_{vi}q_i = w_{i-1,i} \times r_{i-1,e} = \emptyset_i z_{i-1} \times (p_e - p_{i-1}) \\ Ja_{wi}q_i = \emptyset_i z_{i-1} \end{cases} \leftrightarrow \begin{cases} Ja_{vi} = Z_{i-1} \times (P_e - P_{i-1}) \\ Ja_{wi} = Z_{i-1} \end{cases} \quad (19)$$

Where P_e is the distance from the origin of the end effector coordinate framework totals the basis frame and p_{i-1} the analogue distance from link i-1. As previously mentioned, the Jacobian for the angular and linear velocity have been derived [54].

These two Equations are logical equivalence to the two following Equations: [54]

$$\begin{bmatrix} Ja_{vi} \\ Ja_{wi} \end{bmatrix} = \begin{bmatrix} z_{i-1} \times (p_e - p_{i-1}) \\ z_{i-1} \end{bmatrix} \quad (20)$$

As previous noted, [54] a recursive methodology can be used to calculate the Jacobian matrix, where Z_{i-1} and P_{i-1} are cyclically calculated. It is therefore clear that the Jacobian depends on the manipulator's duration. [54] The reason for this is that the variables described in Equation (20) are extracted from the homogeneous changes derived from Equation (3), which rely entirely on articular variables. That being said, P_e corresponds to the first three elements of the last column T_n^0 and p_{i-1} is equivalent to the unit vectors of joint I and z_{-1}, which are available in a third column of rotation matrix from Equations 2 and 3 [54].

4.2 Armature

The armature plays a significant objective in the simulation of the arm in Morse, which will be the main core for the movement of the skeleton. For implementing the process of the motion for the robotic arm, the following steps should be known, as modified from [56]:

4.2.1 Create the armature

The armature in Morse could be defined as how the kinematics links or chains have been simulated that are involved in either revolute or prismatic joints, the number of bones collected together to make an armature. [56]

4.2.2 Actuator type of the armature

The armature is represented in Morse [56] as actuators that can be controlled using ROS, such as the trajectory action controller, joint space controller and path mechanism controller. The actuators can move by two methods using an armature chain: first, by giving the target position to the end effector of the chain (inverse kinematics solution); and second, by setting values for each arm joint using data streams that represent trajectory controllers [56]

4.2.3 Armature pose sensor

This sensor transmits the joint state of a specific parent armature as a child of this armature – which is specified depending on the armature – to the data structure of the export data stream from the armature sensor and on the pair dictionary (joint name, rotation angle value), whereby the combined value is the radius value (for revolute connects). [56]

4.3 JACO arm simulation

This section presents how to implement the JACO robotic arm in the simulation program. First of all, the arm has been drawn using a normal toolbox in Blender and then it should have the same real arm texture and colour, which could be achieved by using the same texture and colours as the real one, as shown in Figure (38).

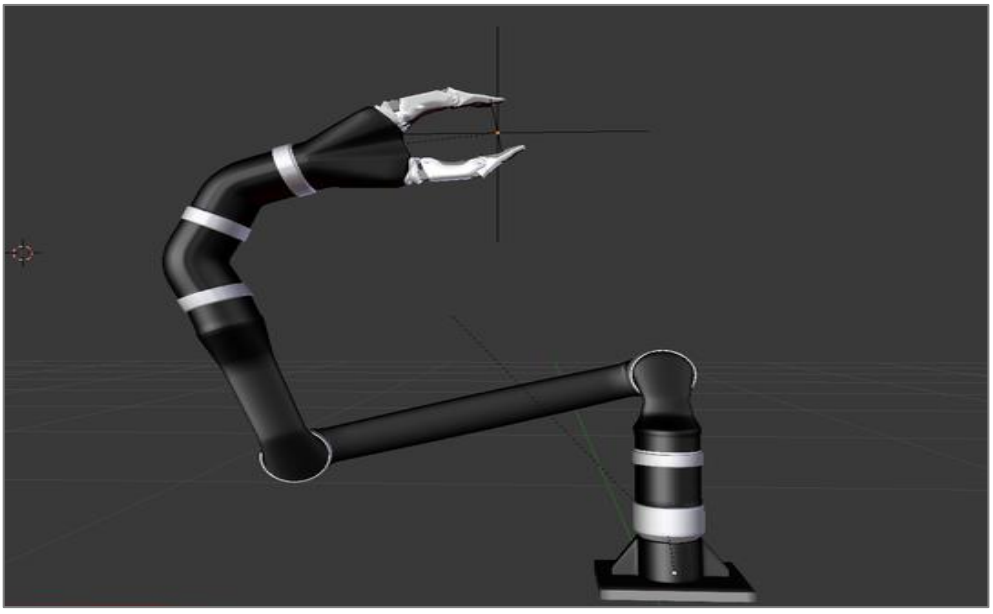


Figure 38: The simulated JACO arm.

The JACO robot arm should be drawn in the simulation as the real design of this arm (the number of joints, links and type of the material), whereby the arm comprises the number of links and the joints of each link are controlled by one bone. When the bones rotate on the required axis, this motion will be reflected on the body (skin) of the robotic manipulator arm and this will undertake the required motion.

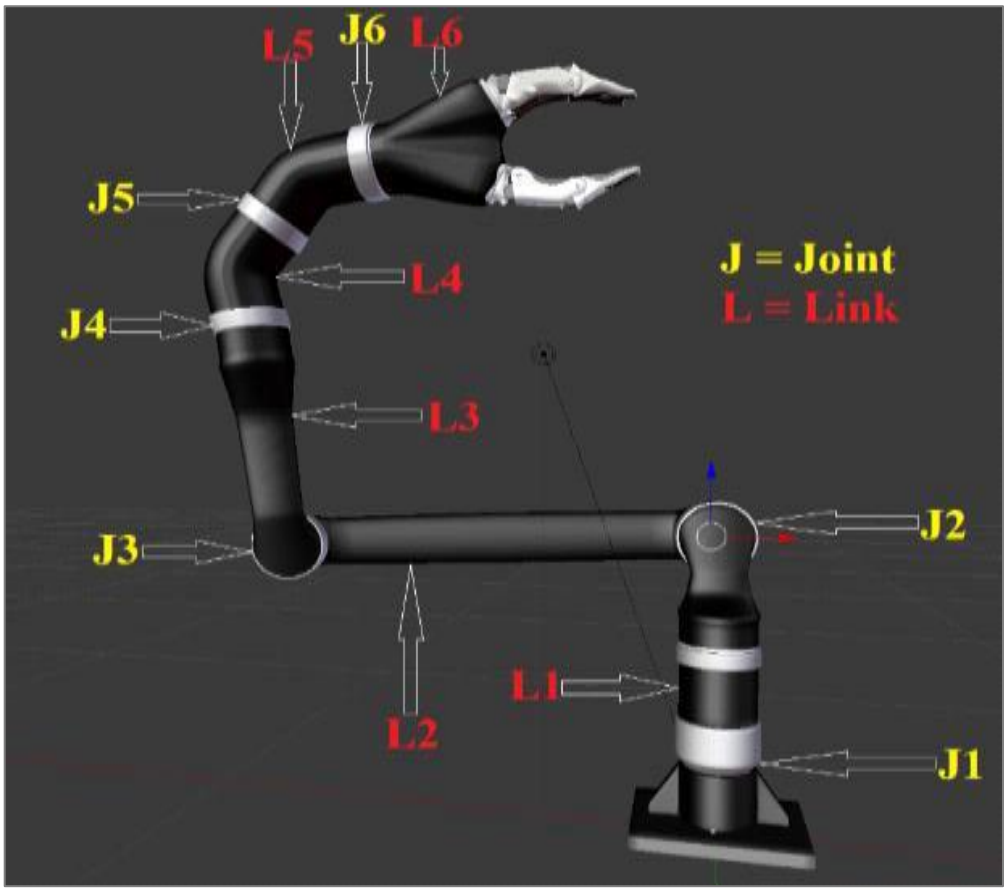


Figure 39: Links and joints for the robotic arm

The simulated robotic arm joints and links are clearly shown in Figures (39) and (40), whereby the first figure represents the whole links and joints for the arm and the latter represents them for the gripper, whereby each link is described with red colour and the joints are described with yellow colour.

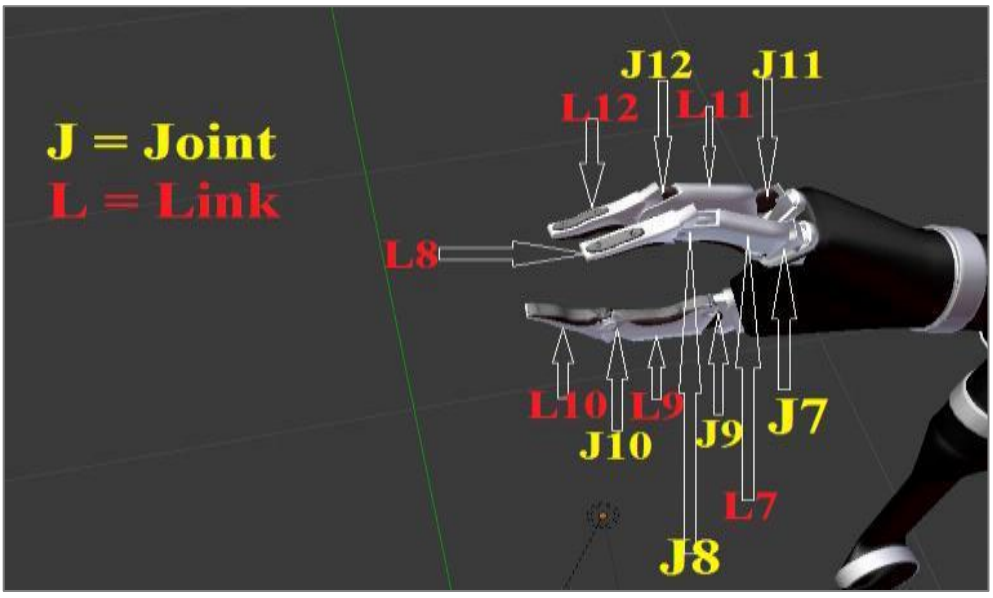


Figure 40: Joints and links for the gripper of JACO.

After drawing the arm in Blender, the bones (actuators) distributed along each link are used for moving the simulated robotic arm. These bones move according to the joint motion. The bones have the ability to move vertically and horizontally (rotating about its axis), as shown in Figure (41).

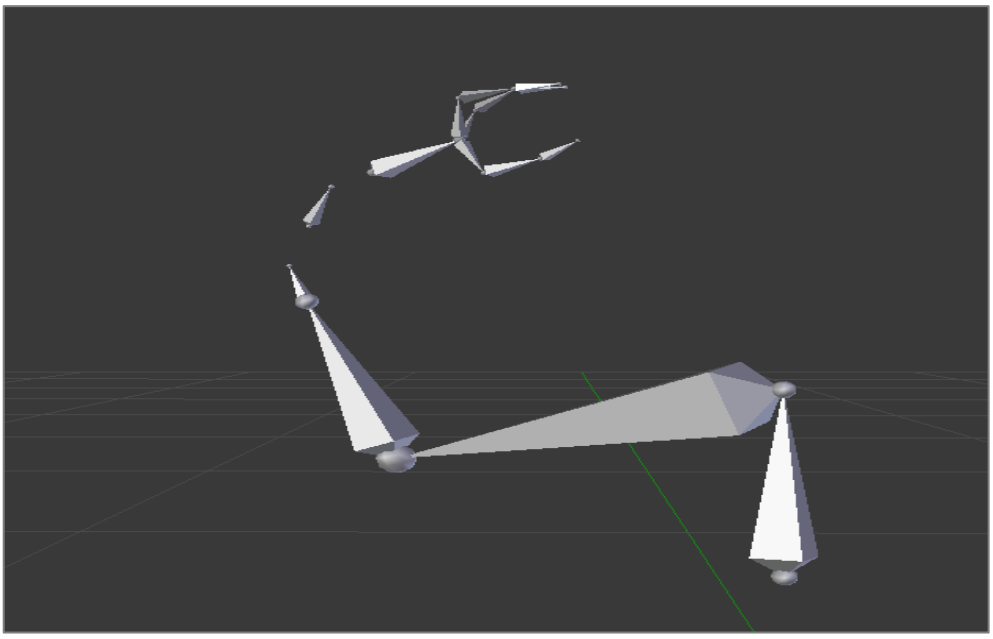


Figure 41: Skeleton used for moving the arm.

The skeleton of the robot arm comprises a group of the connected bones, whereby the bones should be added to the skin of the arm as shown in Figure (42). Each bone should represent the link that will move according to the bone rotation. The axis of the rotation for each one of the armature bones should be specified, and each joint must specify its type (rotation, translation), while the value of its maximum movement should be specified, i.e. the maximum angle of the rotation in rotational joint and the maximum placement in the translation joint. The relationships between the bones should also be changed according to the required movement, whereby the parent-child relationship will be implemented to the bones and the mesh.

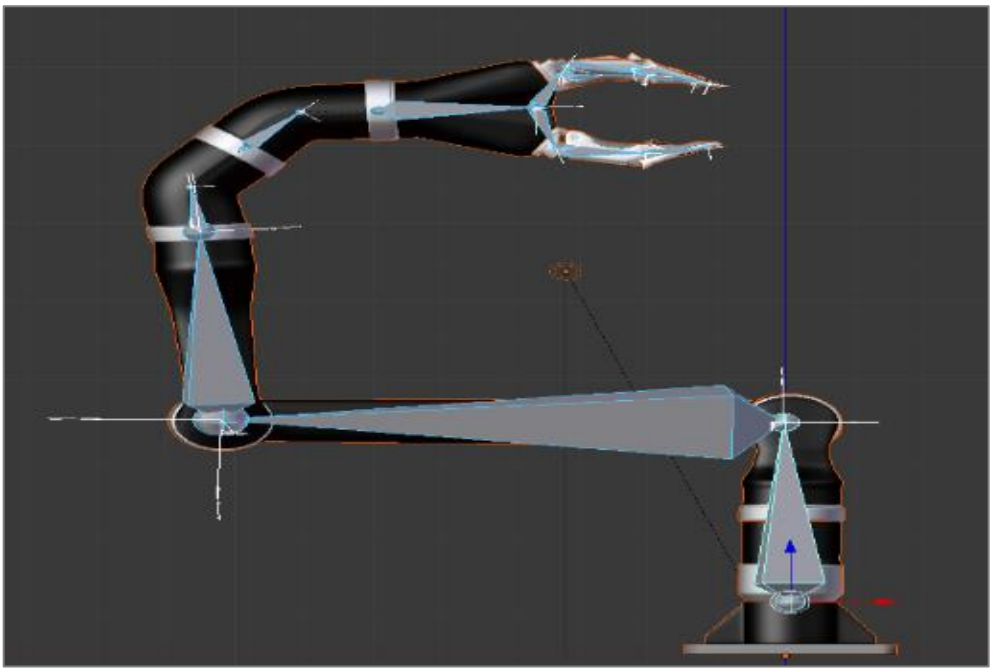


Figure 42: Bones added to the skin of JACO arm.

The number of bones that should be added to the arm depends on three factors:

- ♦ Number of joints and links used the bones for direct motion
- ♦ Number of axis rotation for each of the joints
- ♦ Kind of joint motion (prismatic or rotation).

After adding the armature to the skin of the arm, each of the bones is applied to the suitable weight (the weight represented the bone force, which will reflect the motion to the skin) according to its movement on the joint and which link should be moved. This can be specified using the weight paint tool, as shown in Figure (43). The first step should be to select the bone that will move, then select the arm part by part (link by link) and check the link in terms of whether it needs to move with the bone. As an example, with bone number (2) and link number (3), the link will not move while the selected bone is moving. There are many options to make the weight paint process easily controllable.

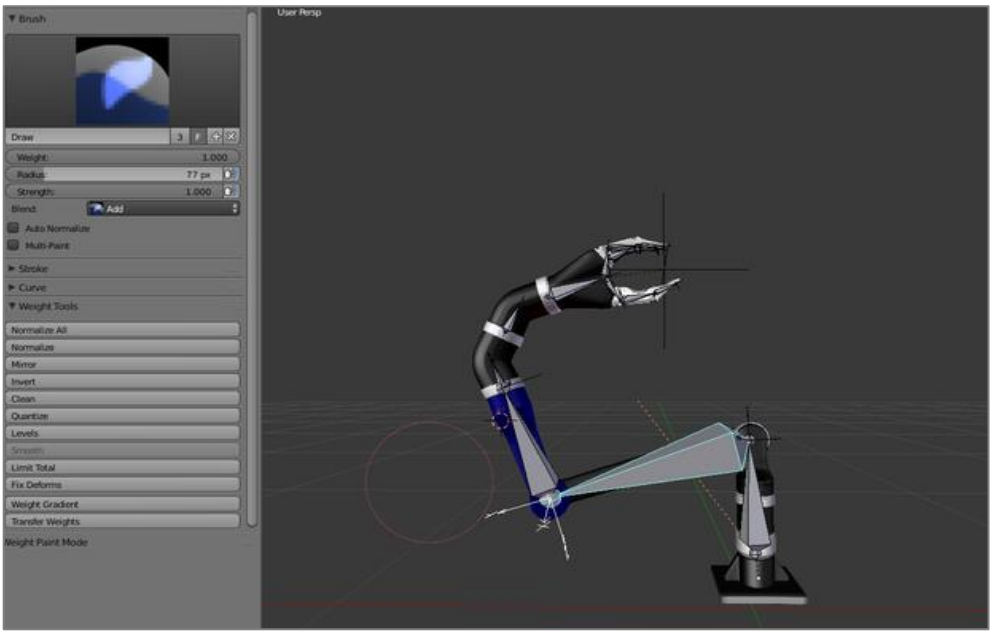


Figure 43: Link number three through the weight paint stage.

The diameter of the selector can be changed as shown in Figure (43). This allows selecting the vertex group precisely, and it can be selected in any position, even the small places of vertex. The weight can be added using the colour intensity: as shown in Figure (44), for the colours every weight has a special spectrum [57]. This could be using the add option, whereby many options refer to special processes like subtract when removing the load, blur normally smothers without any basics, among other options.

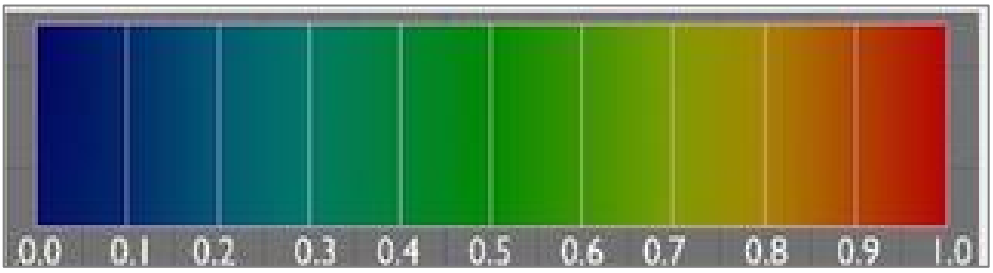


Figure 44: Colour spectrum and respective weights.[57]

4.4 Skinning

Skinning as a general definition relates to how to overcome and control the deformation resulting from bone movement, i.e. when we want to move the skin according to the bone movement, there is some mismatch between the two movements, caused by deformation. The work deals with the robot arm as the human arm and distributes the bones on the arm joints, whereby the arms with the bones are like the complete moving system and one completes the other in motion. Accordingly, the bones should move accurately and exactly like real human joints when moved with the skin of the arm. Therefore, we used Blender as

a program for drawing the arm and adding the bones. There are many algorithms for the skinning, the most famous of which is linear skinning. All details about this algorithm are briefly explained in this work, whereby the distribution of the weight in blender follows this algorithm. Here, we study the effect when applying automatic skinning (automatic weight distributed on the bones) and the exact weight distribution according to the arm motion effect, whereby the weight distribution effect represents the colour percentage and thus the colour ratio from dark blue to dark red (from the zero-weight effect to the maximum weight).

Enveloping (or skinning) [58] in computer graphics is a common and fundamental task. Whenever an animator controls a character via a skeleton, enveloping is used to map these controls to deform a mesh surface. [58] There is no standard way to envelope: an artist may run a physical simulation to model muscle deformations or tune complex systems of shape blending for more direct control. For interactive applications, linear blend skinning enjoys the most popularity, it is easy to use and can be accelerated on graphics hardware [58]. The basic and most popular algorithm for direct skeletal deformation is the linear blend skinning (also known as a sub-spatial skeleton deformation, (single-weight) enveloped or matrix-palette skinning [59].

"The traditional interactive skinning model goes by many names" [60], "including skeleton sub-space deformation (SSD). Maya calls it smooth skinning and we call it linear blend skinning' '[61][60].

The challenge in this work is implementing the movement of the manipulator robot arm (JACO), which is an example. The arm has many joints, each of which must rotate about a specific axis after drawing the arm with all of the material specifications and vertex to be like the real arm in reality.

The bones have been added to the skin of the robotic arm, whereby the bones here are the actuators, which will move the skin accordingly. The basic idea is to transfer the arm parts softly from one place to another without deformation in the arm joints and links. The first step in blender skinning is the weight distribution: to move the skin, we need to distribute the effect of each bone on the right skin part, with each bone being responsible for moving a specific arm part.

4.4.1 Skinning for rigid skinning

The general Equations are as in [62], which each vertex for the object is applied to exactly one bone, as can be seen in the Figure (45). The vertex on the upper link side is subjected to a different bone than the skin on the lower side, whereby the position of each vertex on the skin would be as follows in the space world: [62]

Let us suppose – as modified from [62] – the position of the vertex as P_v in its space coordinates, so the vertex position of the vertex according to the world coordinates should be: [62]

$$P_{vi}\ \text{world} = T_i\ P_{vi} \quad (21)$$

T is the transformation matrix representing the motion of the bone from its space to the world space. [62]

[62] Remove the deformation through the motion of the arm by applying the linear blend to the vertex near the joint, and which has been done by subjecting each arm skin vertex to all affected bones. [62] This leads to $P_{i,j}$ for a different bones coordinates system, not just one. Here, the weight is the main objective, especially for the differently-oriented skin vertex. Each vertex J that belongs to the bone i is assigned to a weight $W_{i,j}$. Accordingly, the vertex position in the world coordinates is obtained from Equation: [62]

$$P_{vi}\ \text{world} = \sum_j \quad W_{i,j} T_i P_{vj}^i \quad (22)$$

as modified from [62]. Besides the border of each object, to prevent collisions we can also assign the normal and tangent vectors: [62]

$$N'_{rj} = \sum_i W_{ij} T_j^{-T} n_j^i \quad (23)$$

$$T'_j = \sum_i W_{ij} T_i t_j^i \quad (24)$$

The applied weights are independent of the motion of the arm: if the angle of a joint is changed, the skin should not become deformed. This is called the spare weight, which will be quickly integrated and should be smooth to avoid dislocations through the motion.

There is one limitation for this method that forced us to improve and add some variables to this motion, which is when the joint angles are large or when a bone undergoes a twisting motion. So that the axis of the rotation for each link should be subjected as in:

$$W_{i,j} = 1if\ \frac{1}{5} \leq \frac{(X_{pi} - T_j) * (X_{ci} - T_i)}{|X_{ci} - T_i|} \leq \frac{4}{5} \quad (25)$$

Otherwise, $W_{i,j}$= 1 / R, where R is the number of the JACO joints.

For the JACO simulated arm where X_{pi} and X_{ci} represented respectively by the parent and child link positions which are located near the vertex P_v, R is the number of the rotation joints.

This makes the vertex that is far away from the bone end point influenced by its bone, although the closer would gradually influence. The deformation will be smother, especially the dislocations through the motion [62].

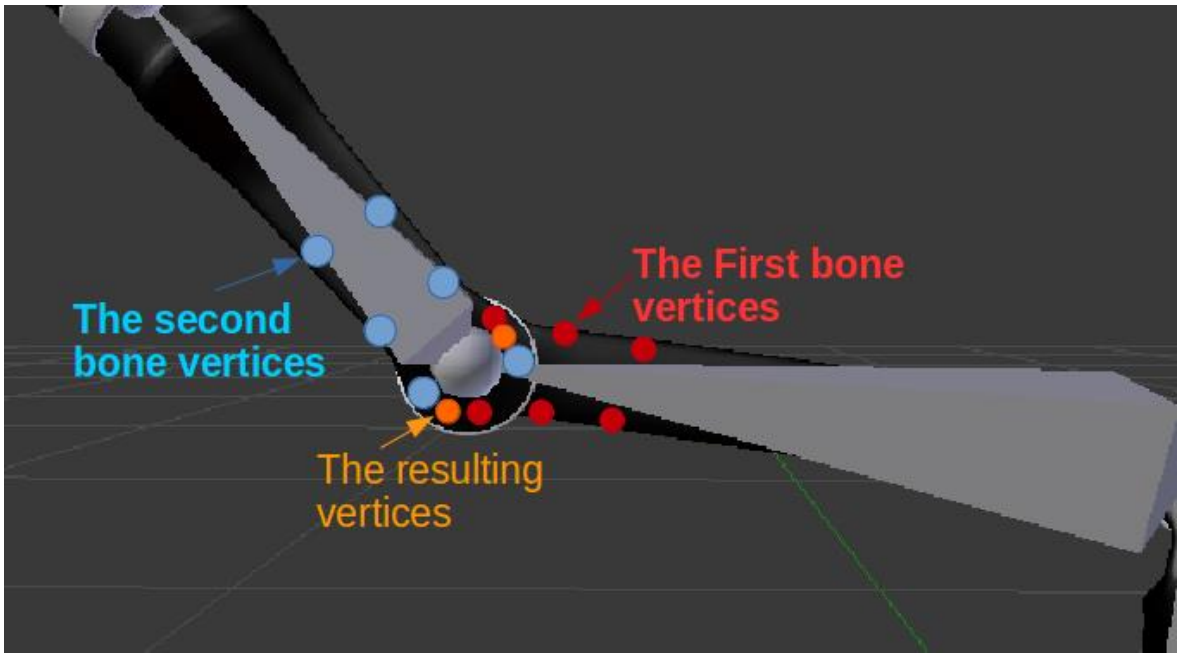


Figure 45: Vertex between the second and the third bone of JACO

4.4.2 Weight distribution

The main step of skinning is how to apply the suitable weight to the vertex, for which there are two options in Blender:

1 Applying a constant value of weight to the whole vertex and then editing the vertex position when the deformation occurs.

2 Applying the suitable and specific weight to each vertex, which allows the vertex to move flexibly according to the required applied weight.

The second way is more flexible and reliable because the arm has many joints and vertexes, and each group of vertexes moves in a specific axis. By contrast, the first way is suitable for an animation that does not feature many vertex groups, while in the second way the deformation may happen or not due to the good weight distribution, as shown in Figure (46). Before entering the weight and how it has been distributed, the bone will connect two respective joints, whereby the motion of the bone will affect the last joint if the motion is rotated about horizontally or along the bone's vertical axis.

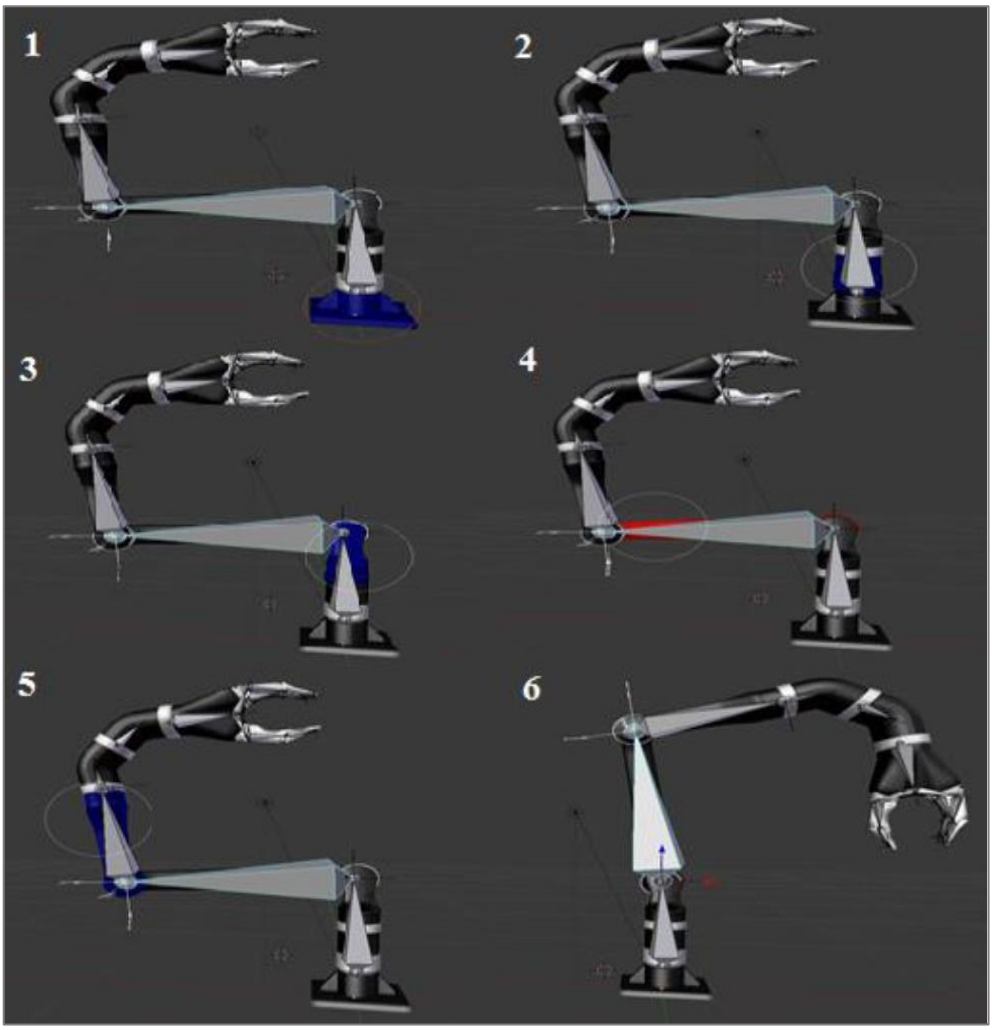


Figure 46: Applying the second way in weight distribution.

We will take the rotation of one link as an example and apply this link to these two methods for weight distribution. In the second method, each one of the vertex groups will be applied to the specific weight that is exactly or close to the required value. As shown here with the motion of bone number two as an example, obviously bone number two has an effect on the base vertex group in photo (1), link two vertex group in photo (2), link three vertex group in photo (3), link number four vertex group in photo (4) and link five vertex group, whereby we can see that link number four moves very softly and without any deformation. This is shown in photo (6).

Applying the first way, when the deformation happens in one vertex group moving in a specified axis, we need to edit the vertex position in this axis to overcome the deformation, although this has a side effect on the other vertex groups and other deformation will appear in the other vertex groups. We can see the effect of bone number two on each of the vertex group links in Figure (47), although here when applying automatic weight distribution and the deformation will clearly happen due to the non-effective distribution of the weight along the vertexes in Figure (47) the effect of bone number two on the base vertex group in photo

(1), link two vertex group in photo (2), link three vertex group in photo (3) and link four vertex group in photo (4) and link five vertex group.

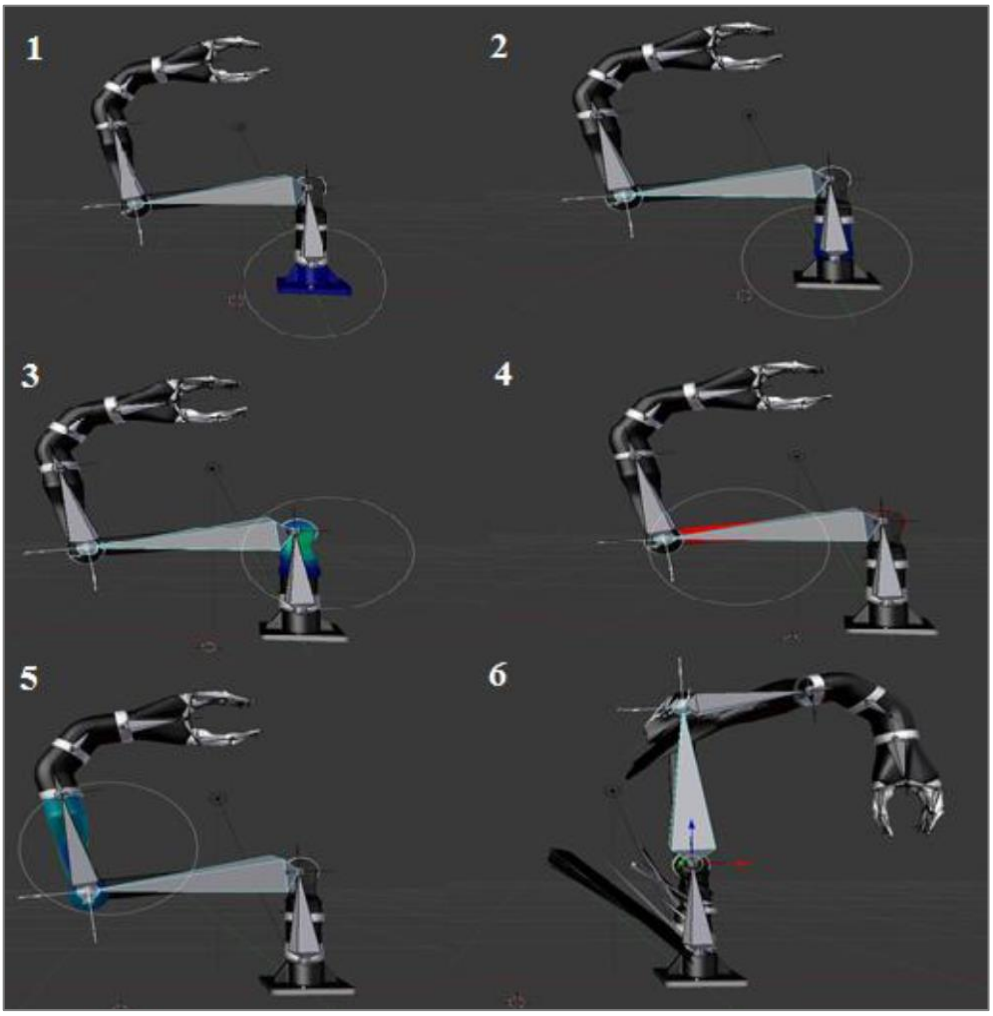


Figure 47: Applying the first way in the weight distribution.

4.4.3 Rotational axis

The axis of the bone rotation must be specified, and an incorrect bones axis can cause many deformations in the skin of the robot arm. Accordingly, the rotation axis must be known for each joint to make the right decision when choosing the rotational axis, as shown in Figure (48). In Blender, many rotational axes can be shown, but the problem is how to choose the suitable axis for the rotation to take place.

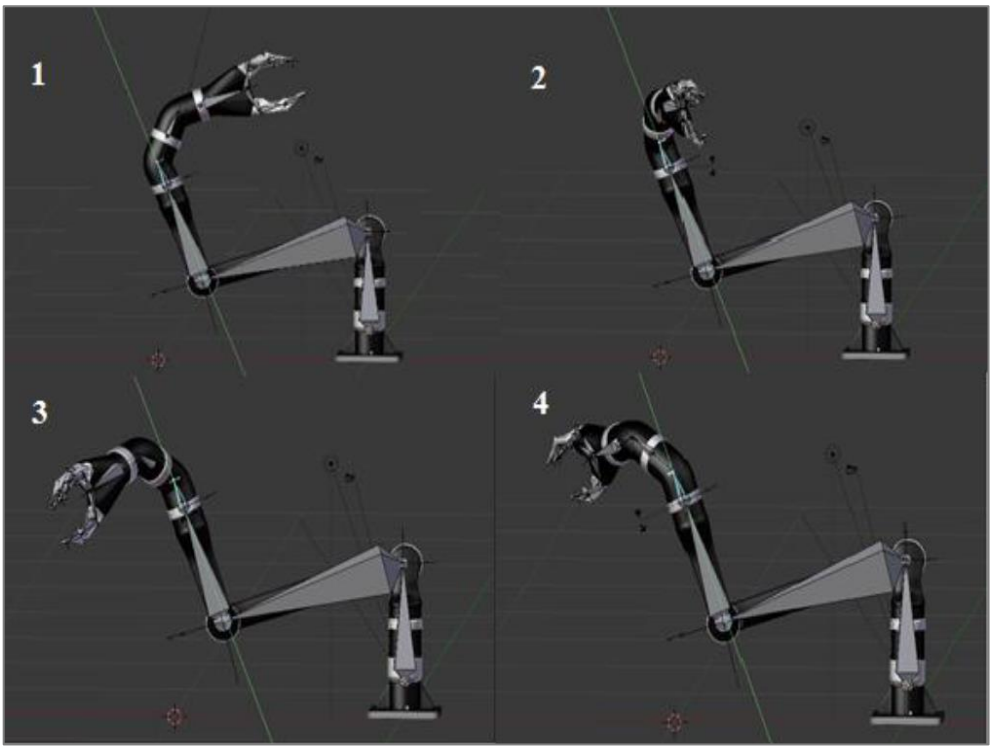


Figure 48: Arm rotates softly with the right specified axis.

We can see here an example of one joint that is surrounded by the red circle on the left side and we can see the zoom joint and the deformation, which clearly happened in Figure (49). The reasons for this are discussed in detail in [63]**.** By splitting a rigid transformation into a rotation and translation pair, we are committing to a specific pivot point (centre of rotation), around which the rotations will be interpolated [63]. By default, this centre of rotation corresponds to the origin of material-space coordinates, which are typically located near the object's centre of mass, thus explaining the unusual result [63].

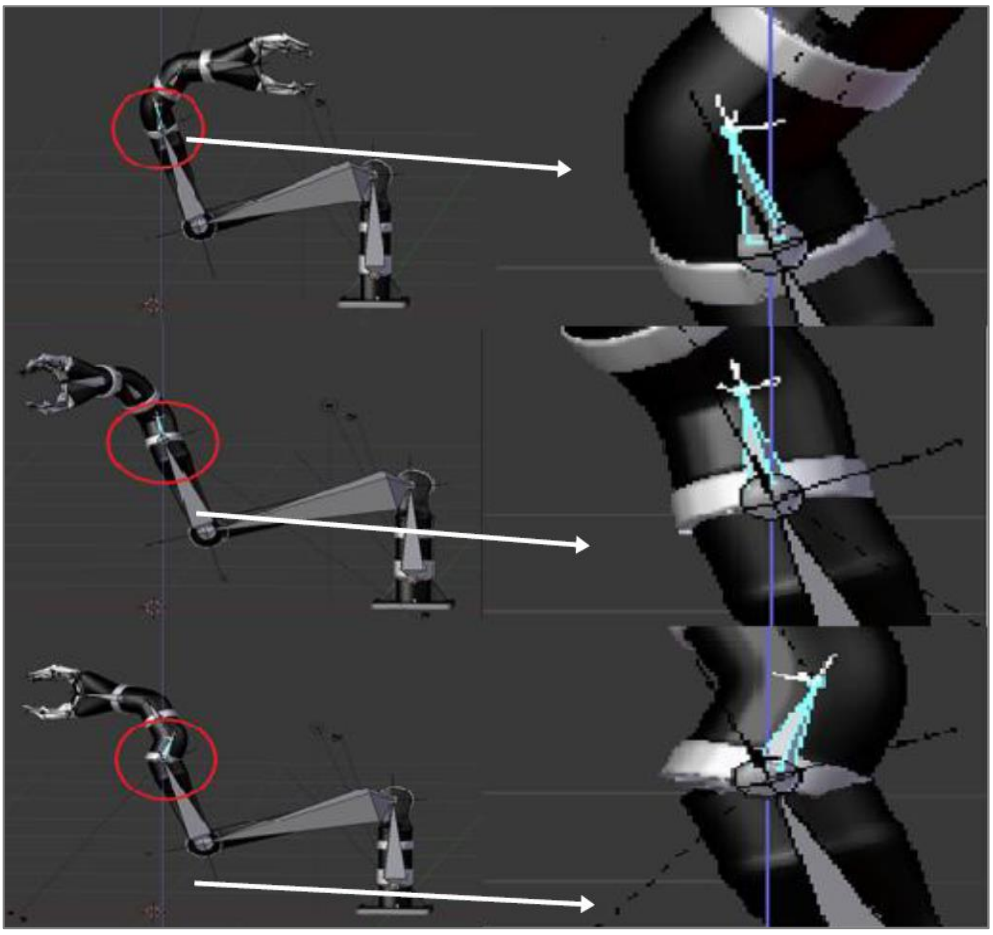


Figure 49: Deformation when moving the joint in the incorrect axis.

The arm has been modelled using Python script and then implemented in MORSE, whereby this should also be defined physically to apply the natural arm movement. After this, the sensors and actuators should be added to the arm. In our work, the components that should be added are as follows:

- Armature actuator
- Semantic camera sensor
- Armature pose sensor.

In this case, it will not be allowable to use this sensor and actuators as stand-alone elements ([64]) in the simulation, although if the user wants to implement these to work together, he should add a virtual-fake robot. The fake robot as in [64] will be like the parent of the sensors/actuators. There is no visual representation of this robot and it comprised a single empty blender, whereby its only purpose is the base to connect the sensors [64].

4.5 Controlling the arm

The mechanism to control the chain of the bones in MORSE can employ two methods, either using the joint-space trajectory method or inverse kinematic by putting the end-effector to the target's coordinates and then moving towards it. Each method has special steps to implement it.

The robot [65] control interprets the robot's application program and generates a series of joint values and joint velocities (and sometimes accelerations) in an appropriate way for the feedback control. [65] At present, mainly an independent joint control approach is taken. However, increasingly sophisticated control approaches like adaptive control, non-linear control, etc. are involved in robotics, whereby the programmer uses the teach box of the robot control and drives the robot to the desired positions and stores them and other values like travel speeds, process parameters, etc. For the programming period – which can take a significant time for complex tasks – the production is idle, and the robot mechanics will transform the joint torques applied by the servo drives into an appropriate motion [65].

In our work the bones will be the actuators for the arm joints. Based on this information the motion mechanism would be classified into two groups as follow and as modified from [65]:

- PTP (Point to Point) motion

The TCP follows the desired (Cartesian) path, whereby the motion between these points specifies the configuration of the robot at the beginning and the endpoint, and it is mainly relevant to pick and place tasks where the position and orientation on the road holds no importance since this movement is complicated to predict for the Cartesian space [65].

- CP (Continuous Path) motion

The motion type that defines the entire TCP position and orientation path (the position of the manipulator by providing a description or the TCP frame (TCP = Tool Centre Point) attached to the end-effector relative to the base frame attached to the manipulator (non-moving base this means that the motions in each of the axes are mutually mutable [65].

4.6 Inverse kinematic architecture simulation

It was necessary to determine the trajectory path and inverse kinematic and how we implemented them in the Morse simulation, as well as which type was supported in the simulation (i.e. how we can move the arm, by which mechanism, trajectory joints).

The inverse kinematic (IK) is defined in [66] as the problem of determining a set of appropriate joint configurations in which the end-effectors move to the desired positions as

smoothly, quickly and exactly as possible. In recent decades, several advanced or algorithmic techniques and methodologies have been proposed to offer fast and realistic solutions to the problem in inverse kinematics (IK), although many of the methods currently available are subject to high computational costs and unrealistic poses compared with popular inverse kinematic (IK) reliability.

We can explain the idea of IK in a simple way: first, we have an object and a robot arm, whereby the robot arm should follow the object wherever it goes, which is achieved by calculating the coordinates of the object relative to each one of the robotic arm joints, as shown in Figure (50),

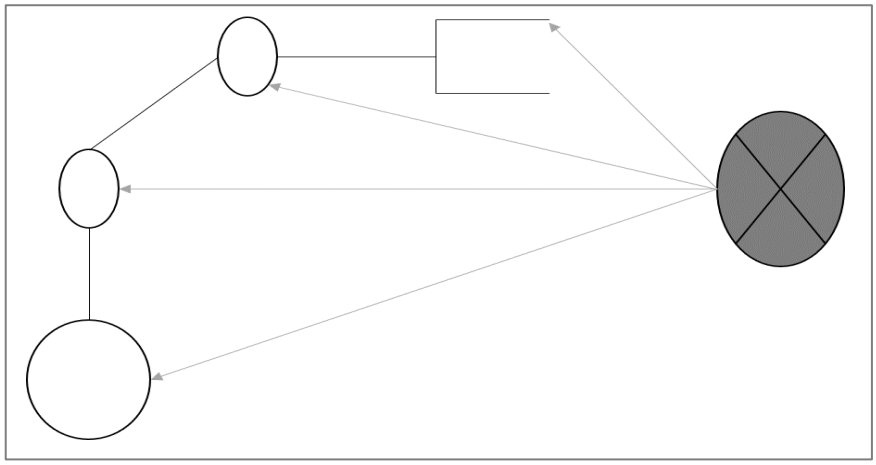


Figure 50: Simple idea of inverse kinematics.

Every robot arm has an end-effector. This end-effector is at the end of the robot arm and it is changeable according to the specific applications; for example, for gripping, welding and others. IK uses the Equations that give the joint parameters at each desired position for each end-effector**.** The robot arm must know the coordinates of the ball relatively and then calculate the rotation angle required for each robot joint to reach the ball, although in the simulation the object will be created from the same plane of the robot arm axis, whereby the implementation of IK mainly needs to study the axis compatibility, bones relationships, IK options and use of each one of these options. This will be much easier than with classic algorithms.

In order to implement IK, the first step starts with the mesh, while a cube in the centre of the armature origin has been added as shown in Figure (51) This cube represents the target according to whose movement the armature moves, and the mesh (cube) should be in the same plane of axis with the armature bone, which will be fixed and will work as a reference for the other bones, as shown in Figure (51).

We can see the execution of IK when the cube moves in the direction as the parent, so the armature should follow the cube as the child, which happens when the whole system is in pose mode. We can observe this step to ensure that the motion becomes soft without the deformation in the arm links. Figure (52) illustrates the results of this implementation, namely that the bones will certainly move into “pose mode.” Figure (53) also displays the axis of the rotation object with the gripper.

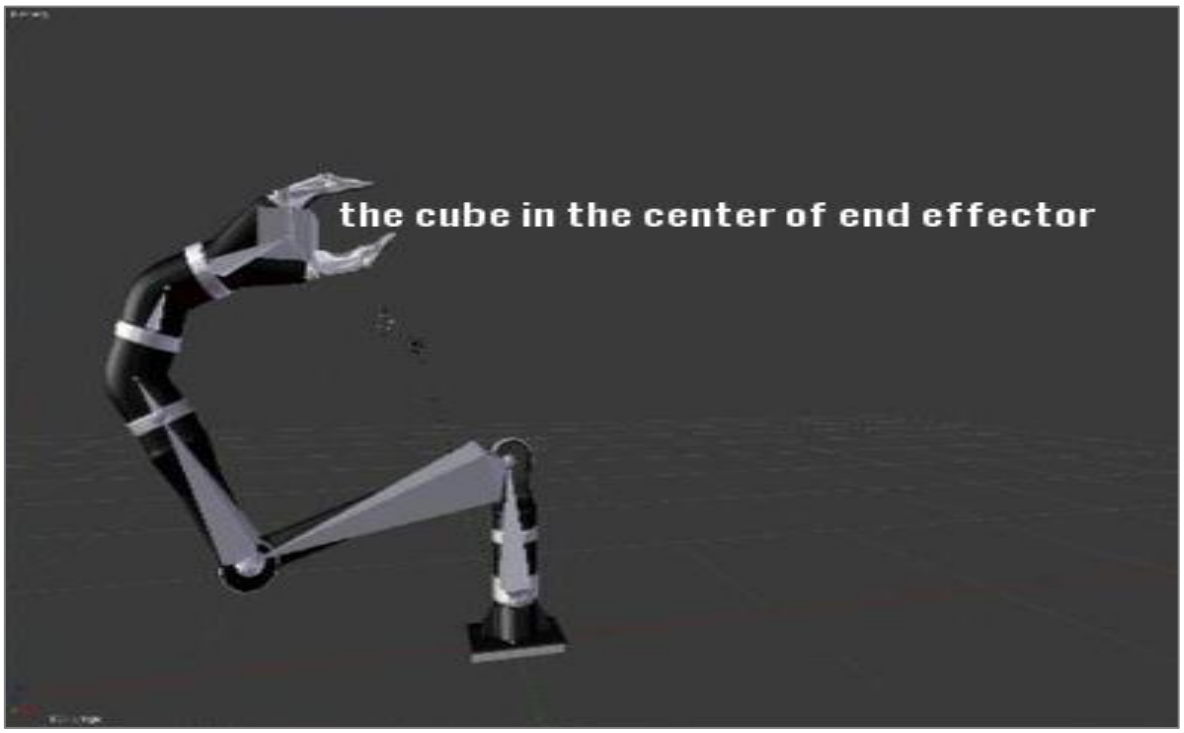


Figure 51: Cube in the origin axis of the armature.

The IK technique has been programmed in ROS node (JACO), as shown in Figure (54). This node is responsible for two main functions:

The gripper function: the ROS topic on the branch/Jaco/armature/gripper, which will receive two messages, namely TRUE for gripping the object and FALSE for releasing the object.

The IK function: the ROS topic on the branch/Jaco/armature/move_IK, whereby the received message will be classified into options:

1 Name of the IK target (this will be a string referring to the name of the target)

2 Translation vector (x, y, z), which represents the translate position of the target

3 Rotation vector (rx, ry, rz) representing the rotational position for the target, the angles in radian

4 Relative: specifies the translation and the rotation relative to the actual target movement (the default option is true)

5 Linear speed (m/s): the linear speed for the inverse kinematic target

6 Rotational speed (rad/s): the rotational speed for the inverse kinematic target

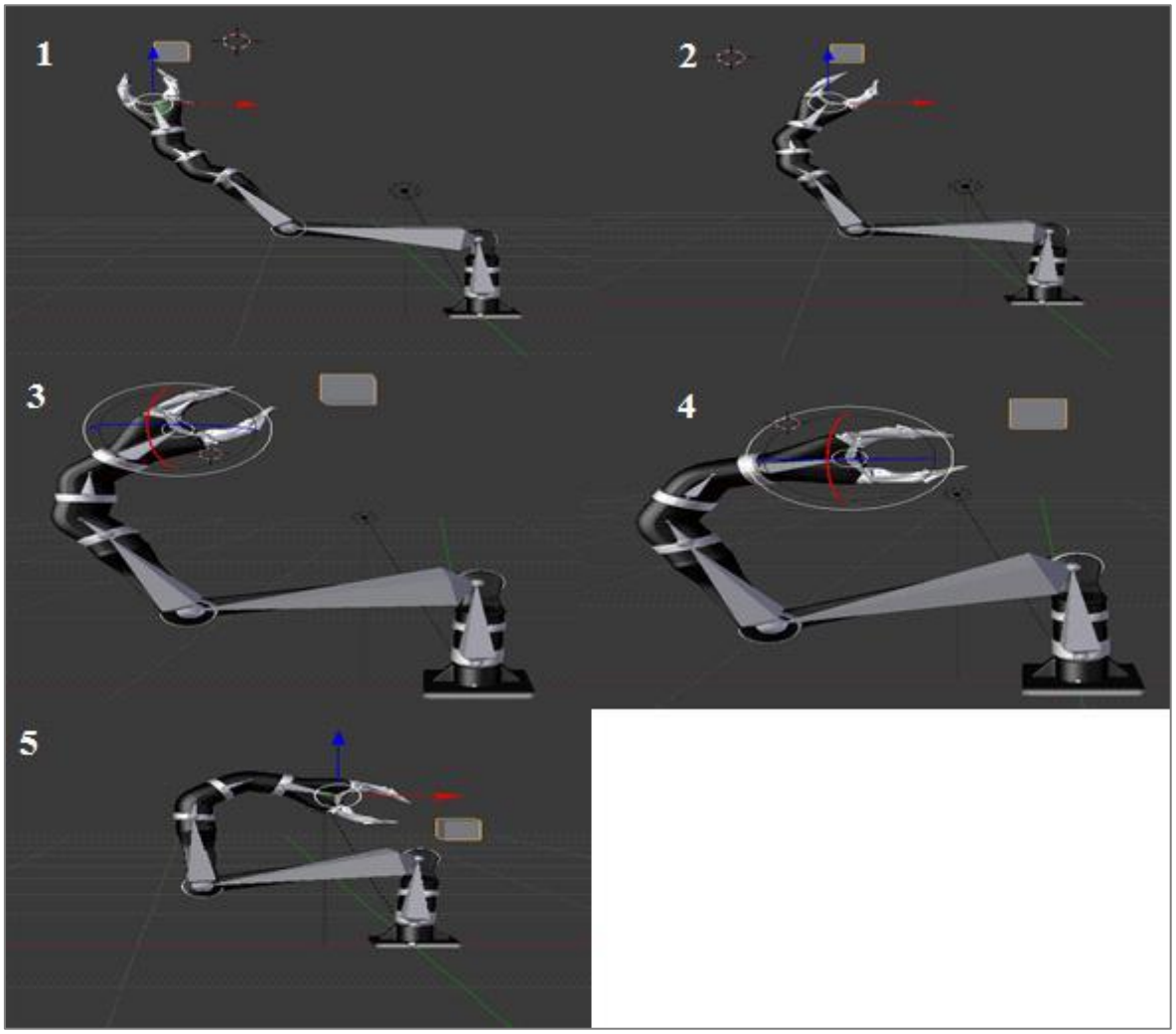


Figure 52: Plane axis of the cube compatible with the axis of the base bone.

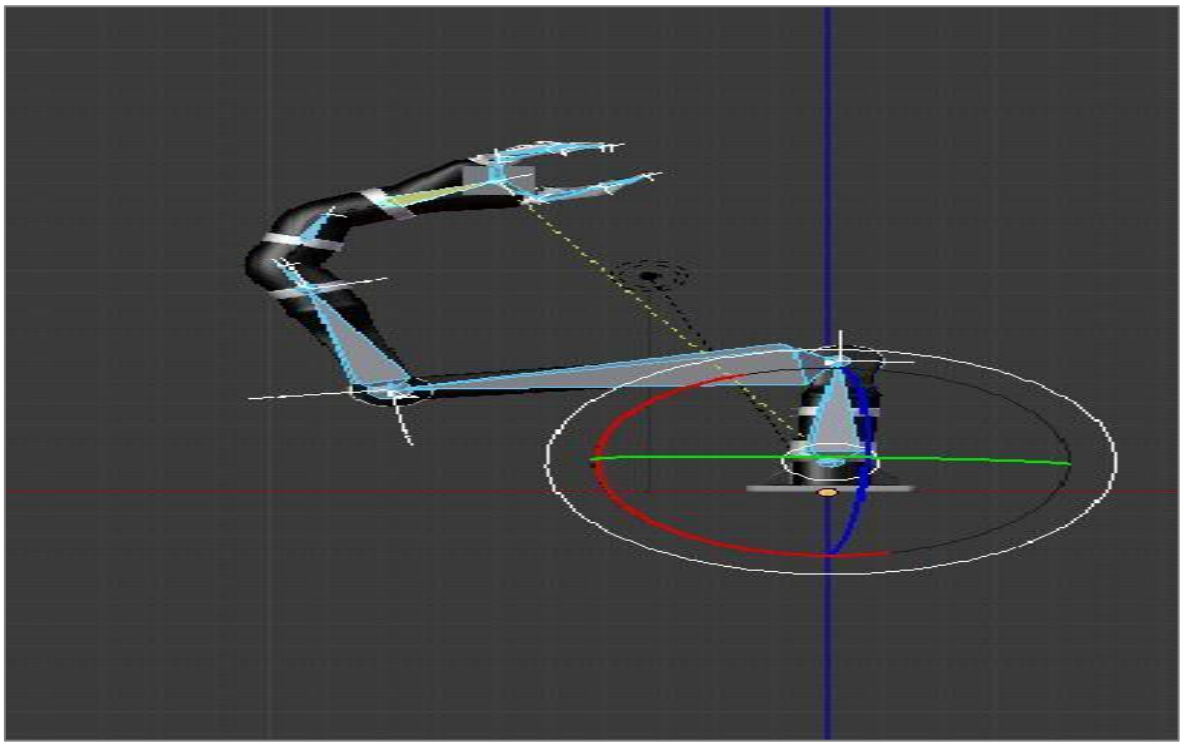


Figure 53: Arm follows the created object.

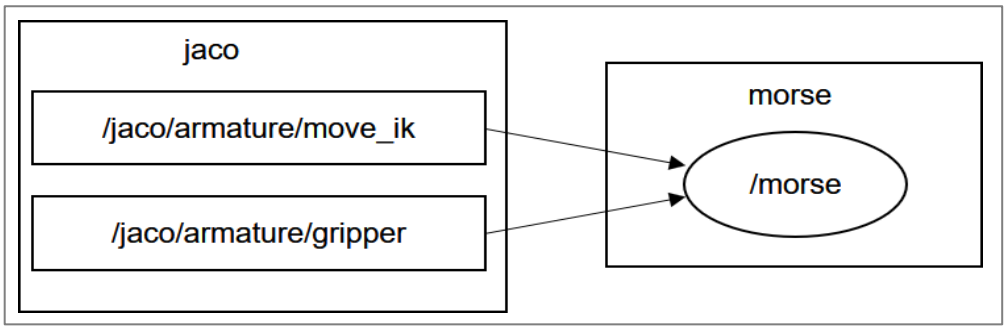


Figure 54: IK ROS node schematic.

4.7 Trajectory action control

The trajectory is a set of points that should be reached at a specific time in ROS. These points are represented as way points, each of which is expressed by its position as a main selection. Moreover, velocity and acceleration are also selected, albeit as unessential options. A simple schematic for the joint trajectory action node is shown in Figure (55), in which the trajectory controller sends the massages (joints angles) to the arm joints immediately after the trajectory target coordinates have been received through the joint trajectory action. These messages are recognized according to the joint names, whereby every bone added to the arm represents the joint name. The joint trajectory node gives us feedback after the operation is successfully completed.

As modified from [67], the joint trajectory action controller performs joint-space paths on a set of joints. It takes as a command a message specifying the desired position and velocity of each joint at every point in time and executes this command [67]. Furthermore, the mechanism [67] allows the controller template to work with multiple trajectory representations by default. [67] A spline interpolator is provided, although it is possible to support other representations. Depending on the way point specification, the spline interpolator uses the following interpolation strategies, as in [67]:

- Linear: only the position is specified, guarantees continuity at the position level, although it is discouraged because it yields trajectories with discontinuous velocities at the way points [67].
- Cubic: position and velocity are specified, guarantees continuity at the velocity level [67].
- Quintic: position, velocity and acceleration are specified, guarantees continuity at the acceleration level [67].

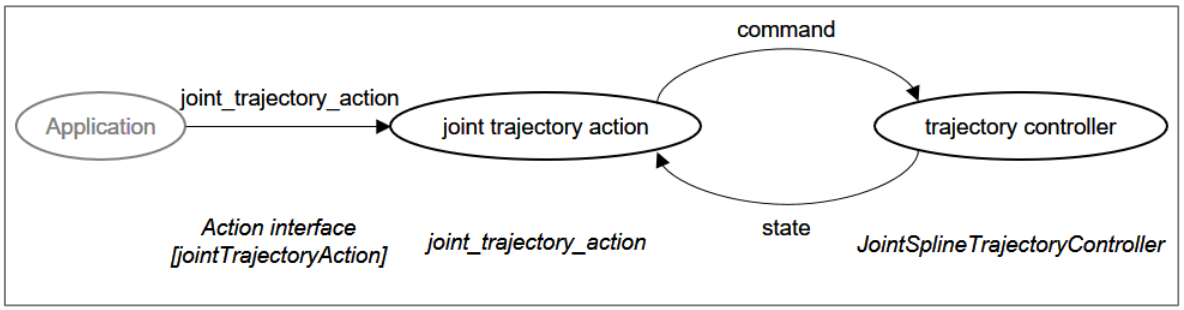


Figure 55: Joint trajectory action schematic [68].

The principles of the controller node for JACO have some differences compared with the other modulated arms: because it does not modulate in Morse, the arm pose will receive the actions from the trajectory and orientation and send them to the arm directly. These actions have been defined in MORSE, whereas JACO needs to correspond these actions with the modulated joint movement, i.e. the bones' rotation must follow the action directions precisely.

The control of the JACO arm is conducted through the MORSE-ROS interface, after simulating the arm and adding the actuators to it. Every joint can move according to the message that has been published. As described in the theoretical algorithm, the JACO arm comprises many joints with different rotation axes according to which the bones will rotate either about themselves or about the meeting point, which will be with the other bones.

All of this needs to be defined in Morse, as shown in Figure (56). The joint trajectory node is responsible for moving every joint to a specified published coordinate (joint angles) with the option of velocity specification, including the time of goal reach. The orientation node has to select the bones that need to rotate about themselves with a specified axis and angle magnitude. This actuator reads the angle values of rotation around three axes and applies them to the associated robot [69]. The function of the JACO gripper still picks up the object when the command comes from the controller unless another message comes for releasing the object, which is what the gripper node does.

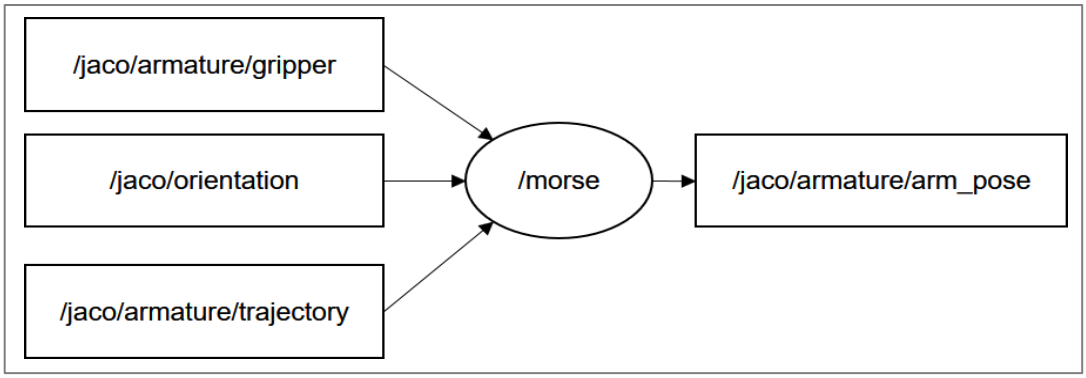


Figure 56: Whole schematic of the trajectory ROS node.

In Figure (57) above, we can classify the ROS topic and their messages:

1 ROS topic/JACO/armature/gripper.

2 ROS topic/JACO/orientation responsible for moving the bone around the vertical axis or the horizontal axis depending on the bone axis, the message when the bone needs the orientation and about the specific axis. Moreover, it specifies the limited orientation values.

3 ROS topic/JACO/armature/trajectory responsible for giving each one of the bones the acceleration, position, velocity, and timing in seconds.

Start ROS node publisher: in this C++ program, it is necessary to enter the message types and magnitude, which will be published to the specific node for moving the arm.

Send trajectory coordinates to arm joint: in this case, we took bones number 1 and number 2 as examples. In Figure (57), we can see bone number 1, which represents the second link of the JACO arm when moved with published topic position (2). This shows us the magnitude of the position (joint angles), whereby we can also specify the time and velocity to reach the target.

Bone number	Bone name	Position
1	JACO 2	2

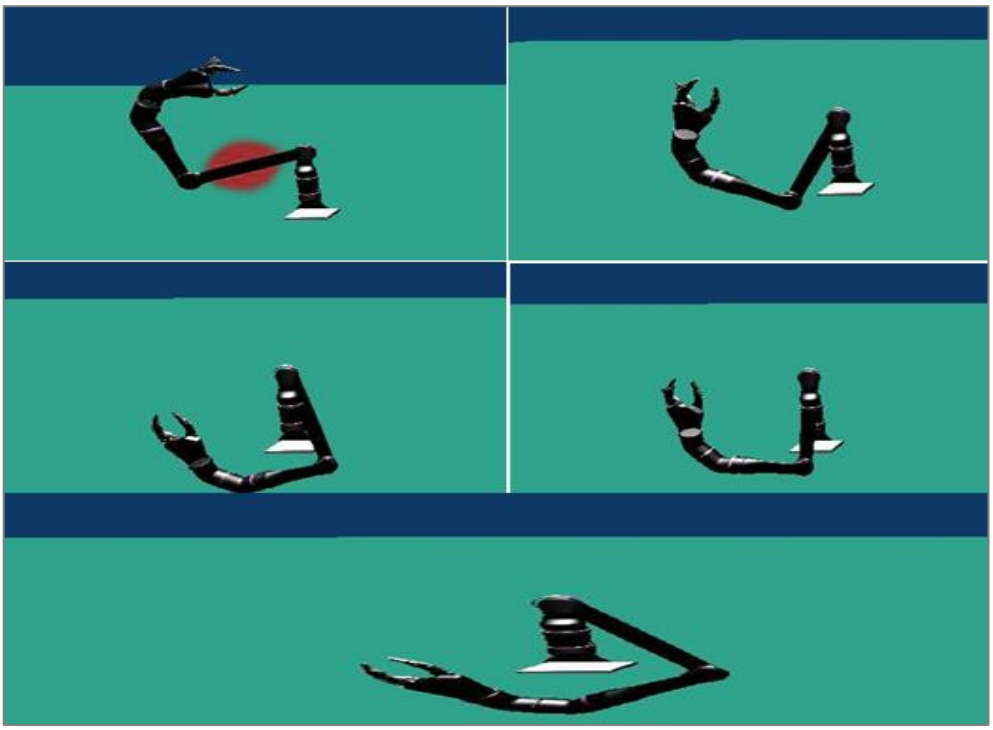


Figure 57: JACO arm link number two moved.

Send an angle to rotate the bones around itself: Sometimes the JACO arm joints have to rotate about along the rotate axis like joint number one as shown in Figure (58) when the rotation angle has been published on JACO link number one along the Z axis.

The gripping operation in MORSE simulation program is like putting glue between the arm gripper and the object that we want to catch it, Figure (59) shows us when a ROS topic published to the gripper to keep catching the red object, and the Figure (60) shows the releasing object process.

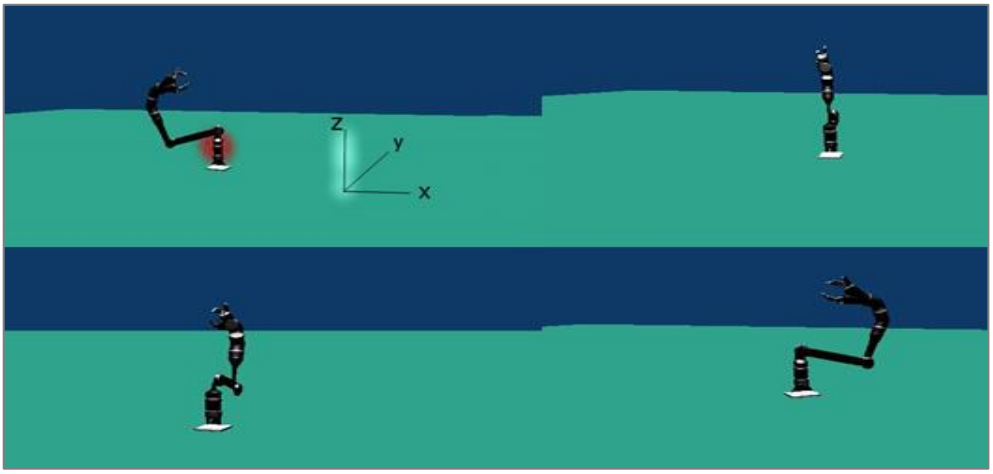


Figure 58: JACO link number one rotated about the Z axis.

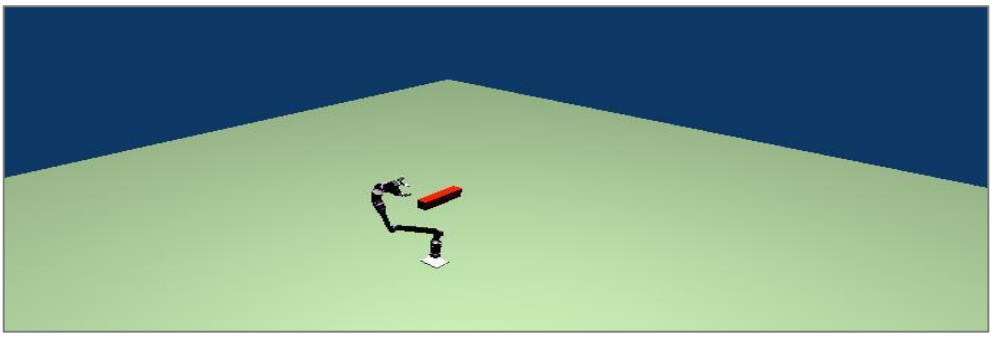


Figure 59: JACO gripper hold the red object as a picking process.

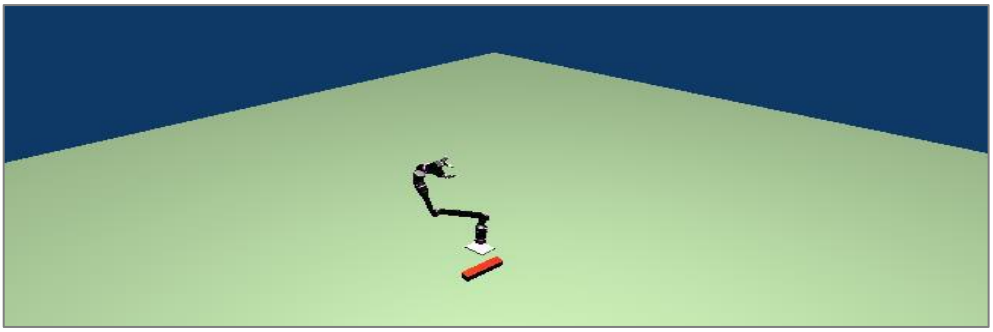


Figure 60: Releasing process when the gripper releases the object to fall away.

This actuator doesn't simulate the interaction between the gripper fingers and the objects that it takes [70].

The gripper in Morse could be programmed using Python script in the same time, whereby some proprieties should be defined in the Blender file: for the graspable object, there are options for defining the border of the object, namely the collision properties. For the gripper, another kind of sensor should be added, called the radar. This sensor is responsible for the gripper physical properties as follows: [70]

- Angle: the amount of angle within which the arm can recognize the object [70].
- Distance: the distance between the gripper and the object to pick up the object [70].
- Axis: the axis of the right pick for the graspable object [70].

5 The semantic camera system

In this chapter we will display the semantic camera system. The system design consists of many steps, from the way objects are defined to the way objects coordinates can be discovered. This system should have the same real controlling system as the real MORSE semantic camera which will be designed with the ROS control nodes, as explained below.

5.1 Overview of the semantic MORSE semantic camera methodology

The working MORSE semantic camera could be described as follows. The MORSE semantic camera is a high-level sensor which consists of three units. The MORSE semantic camera decides which objects should be detected in the simulated environments (This could be done using the Logical properties i.e. define if the object is passive or active), each object should be defined in the environments of the Blender as in the Section (5.2), Figure (61) can describe the input and output to the MORSE semantic camera through the processing. This MORSE semantic camera is used to detect the environment simulated object quickly and this will be a suitable and effective solution as when using the image processing methods for the same task.

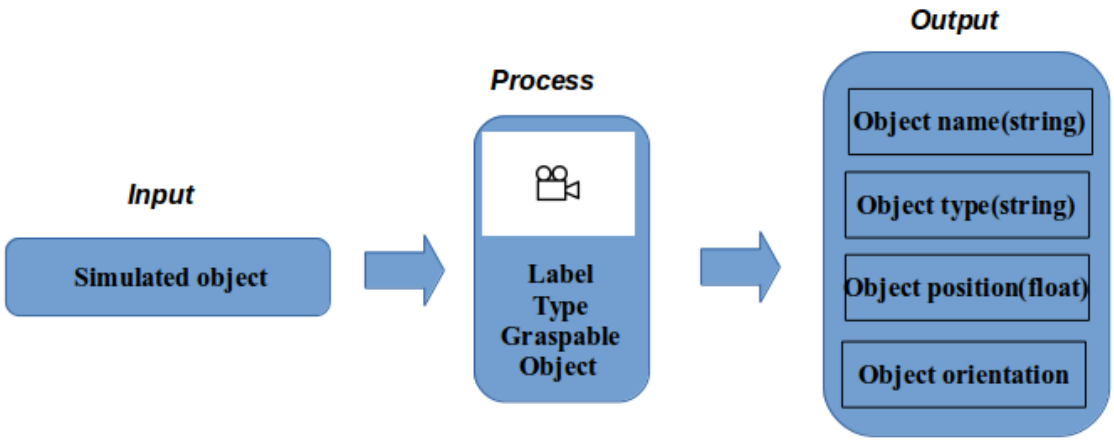


Figure 61: The general work description of the MORSE semantic camera.

i.e the semantic camera will publish the objects information from 3D blender simulation into the Morse simulation environment. The input would be internal data of blender environments for the simulated objects i.e. every object has several details such as label, type, etc. The object's information has been stored in a Morse data bank with its label, type and other

specifications, such as Graspable and object visibility. The MORSE semantic camera will give an output according to the input type and would be as follows:

1 The object name: the name that has been given and pre-defined into the blender environments. The type of the data is string, such as cup, red dish.

2 The object type: the type of each object which also is also pre-defined, the type of the data is string.

3 The object position: the global 3D position of the object, the type of the data is vec3(float).

4 The object orientation: the object orientation will be clarified in the section of Quaternion.

The MORSE semantic camera always triggers. If the object found has been defined in the database using blender, the MORSE semantic camera will display all the pre-defined information related to the object, in addition, a 6D object pose will be sent. The 6D pose is combined of the object position in the environment and the orientation of the object. The output coordinates represent the centre of the object relative to the centre of the MORSE semantic camera (the view region). Actually, the MORSE semantic camera detects the object through its border which has actor physical properties. The object border is represented by the physical simulation properties (bounding box). To demonstrate the process of converting the real data image and using it through the semantic MORSE semantic camera we had to use a point cloud MORSE semantic camera which gives us 3D scene data.

To convert 3D scene data into blender it has to be ply format which allows us to make a mesh image like in blender with vertexes and the normal simulated colours. Before that, the file should be in pts format as shown in the Figure (62).

```
20234
3.6578 -3.45677 6.20938  167  212  254  221
3.6290 -3.71893 5.98388  219  143  213  195
```

Figure 62: The image as .pts file

The first number represents the header of the image, the second line consists of seven numbers. Then for each point we got X, Y, Z and then the scanner intensity, which represented as a grey level image inside the scanner, the last three numbers represented the values of the RGB.

Under the header the definition of the parameters which involved the .ply file. It started from the format of the file, then the x, y, z and then in the normal x, y, z after that the colours and then the intensity of the scanner until the element face (the faces represented polygons). These faces have been read after reading the vertices, i.e. the number of the face gives us an indication of how many polygons there are, we can then summarise the process from reading a 3D scene data until covert into blender with the Figure (63).

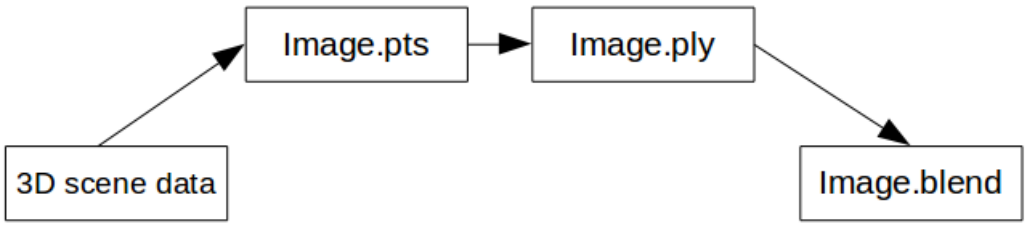


Figure 63: The sequence of transferring the image to blender.

The general software process into the camera would be as follow:

- Import the necessary libraries (objects type [active or passive], 3D transformation, objects properties, colours, sensors).
- Read the image through real time triggering.
- Compare each detecting the object with the reference's objects.
- Detecting the object bounding border and determine the centre of the object.
- Send the name, type, label, positions as output to the semantic camera ROS node.

As clarified, each object border has to be defined by the Blender; this option could be done using a 3D bounding box. The semantic camera will subscribe to the centre of the detected object. The boundary box of the object will surround the whole faces and vertices of the object, the object would be in the centre of the boundary. The object acts like a blob and it should be detected, another algorithm should be used to connect the object pixels.

5.1.1 The connectivity

After creating the bounding box, the challenge will be to determine how to recognize the positions of the object's faces. The connectivity is a method where the pixel in 2 or 3 D is

connected to the neighbour's pixels with the similar amount of grey level, i.e. each two pixels can be connected if they are adjacent in the same sense. This algorithm will take the object pixels inside the boundary box and check every pixel to see if it belongs to the object or not until the whole pixel checking is finished and then the object will be detected. As a simple concept let us suppose pixel P with the coordinated (x, y) and P has a neighbour in the horizontal (x ± 1, y) and in the vertical (x, y ± 1) as shown in the Figure (64).

	(x, y-1)	
(x-1, y)	P (x, y)	(x+1, y)
	(x, y+1)	

Figure 64: 4-pixel connectivity.

for the 3-Dsystem (x, y, z) this P has a relationship with its neighbour (x ±1, y, z) , (x, y ±1, z),(x, y ,z ±1), this relationship in 2-D and 3-D is effected by the grey level values, there are two types of connectivity according to the type of the dimensions 4 Pixel ,8-pixel, 18 pixel ,26 pixel connected.

5.1.2 The objects coordinate

There is a different form for the bounding box. This box has different form options such as rectangular, circle, sphere or free-hand sketch. These forms are according to the vertices of the mesh; the orientation of the 3D bounding box is usually relative to the global world coordinates system.

- Bounding box: the rectangle which contains the object as shown in Figure (65), this rectangle will be around the object in the three dimensions (X, Y, Z).

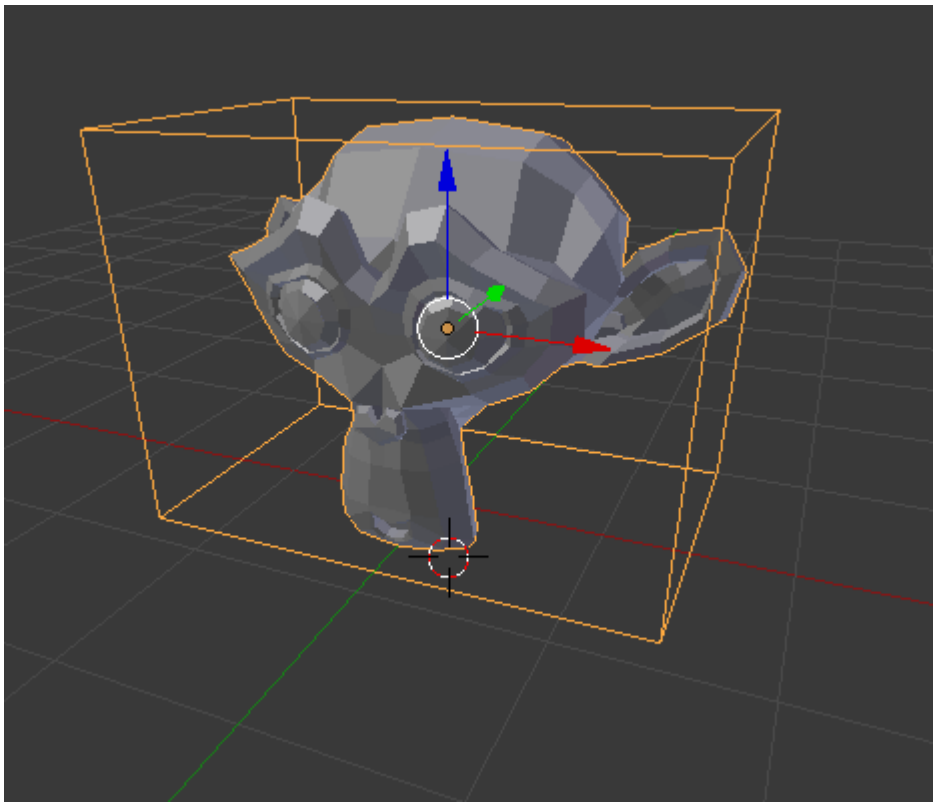


Figure 65: The object surrounded by the box boundary [89].

- Bounding sphere: the minimum sphere which contains the object as shown in Figure (66), the object will be at the centre of the sphere.

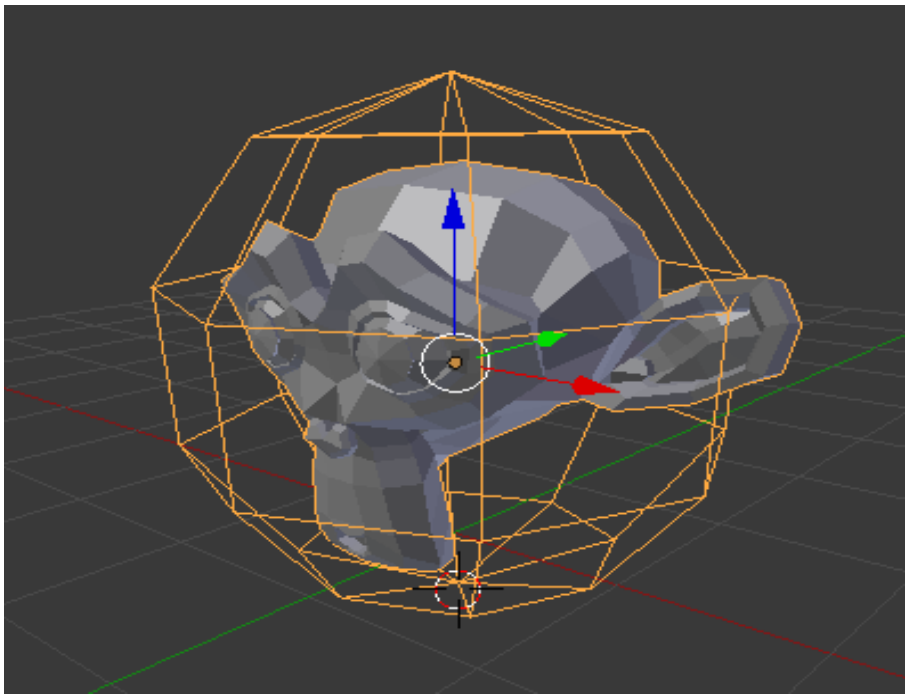


Figure 66: The object surrounded by the sphere boundary [89].

- Bounding cylinder: the minimum cylinder which contains the object as shown in Figure (67).

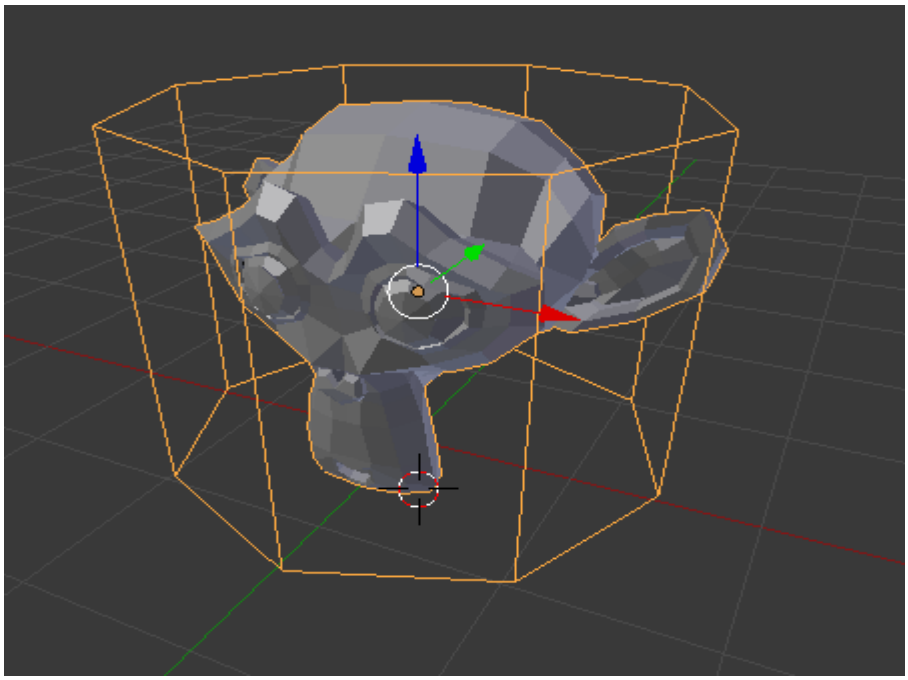


Figure 67: The object surrounded by the cylinder bounder [89].

The centre of the mass for the object inside the boundary box has completed for the 2-D dimension as [71] and this is modified from [71] for our object. It should be noticed that it is a 3-D dimension object, normally the average x, y, z position represents the centroid of the object, this can be calculated using this formula:

$$X_c = \frac{1}{N}\sum_{i=0}^{N} X_i \quad (26)$$

$$Y_c = \frac{1}{N}\sum_{i=0}^{N} Y_i \quad (27)$$

$$Z_c = \frac{1}{N}\sum_{i=0}^{N} Z_i \quad (28)$$

Where N represented the number of the pixels, x_i y_i z_i the coordinates of the N pixel, that means the x value is the summation of x coordinated for the whole pixels in the detected object inside the boundary box. Also, another alternative method to find the centre of the mass is approximately the same to the centre of the boundary box for the 2-D as [71], but for our objects 3-D the x boundary centre, y boundary centre, z boundary centre.

$$X_{bb} = \frac{X_{min} + X_{max}}{2} \quad (29)$$

$$Y_{bb} = \frac{Y_{min} + Y_{max}}{2} \quad (30)$$

$$Z_{bb} = \frac{Z_{min} + Z_{max}}{2} \quad (31)$$

This can be done by scanning the whole object and find the summation of the pixels inside.

5.2 Implementation of the semantic objects in the environment

The semantic map is a new requirement in the robot field using the simulation program (Morse), a simulator completely programmed by Python that is used to implement 3-D environments. Robots comprise many possible actuators and sensors and this simulator will execute in Blender, as this program is based on Blender. The communication between these will be with ROS controlling the robot motion and displaying information, ROS representing the communication ring between various programs and making it possible to share information between these programs. Normally there are many middle wares for communicating between Morse and Blender, like socket, yarb and ROS. With ROS there are many facility options and while the control is more effective and reliable than with others, multi-robot types can be chosen and objects can be defined and then placed in the environment.

The semantic camera is programmed in Python to give us the position of the object related to its orientation in the (x, y, z) axis, as well as its position in the environment. This is modified and understood from [72] the sensor and emulates a high-level camera which outputs the names of the objects which are to be placed in the viewing region. The sensor [72] detects the objects (objects which are defined with a proprietary logic as in Figure (69)), and a single visibility test is carried out for casting a ray from the centre of the camera to the centre of the object.

After drawing the objects and adding them to the environment, the objects need to be defined in the semantic camera database in order to be recognizable later. The passive objects in [73] are modelled on the simulation and will act as static objects moving without any force, with the animation offering us the chance to control their movement with keyframes. When simulating a ball falling on a table, the ball is an active object, as the gravity affects

the ball (the table is passive because its position is fixed). As shown in Figure (22), the dishes and the food above it, the cup and the human model are also already in the simulation. All these objects have to be defined in the semantic database.

The objects classified into two groups: one which is picked up by the arm and another which is not picked up. The first set of coordinates are stored in the system and then sent to the arm for the picking-up process. In Figure (68), the cake is already selected and the arm should pick up it. In the options window, there is an object checkbox: this defines every object in the simulation as an object visible to all other objects. The other option is graspable: this check box will be true when the selected object should be picked up. The other properties are optional, through which the object will gain more detail and be precisely described. We can see in Figure (69) that there are more options for the object, where here the red dish has been selected. The red dish should not be picked up by the robotic arm. This option executes the object behaviour; then the object will act as if it is in a real environment, in addition to the previous options (Object, Graspable).

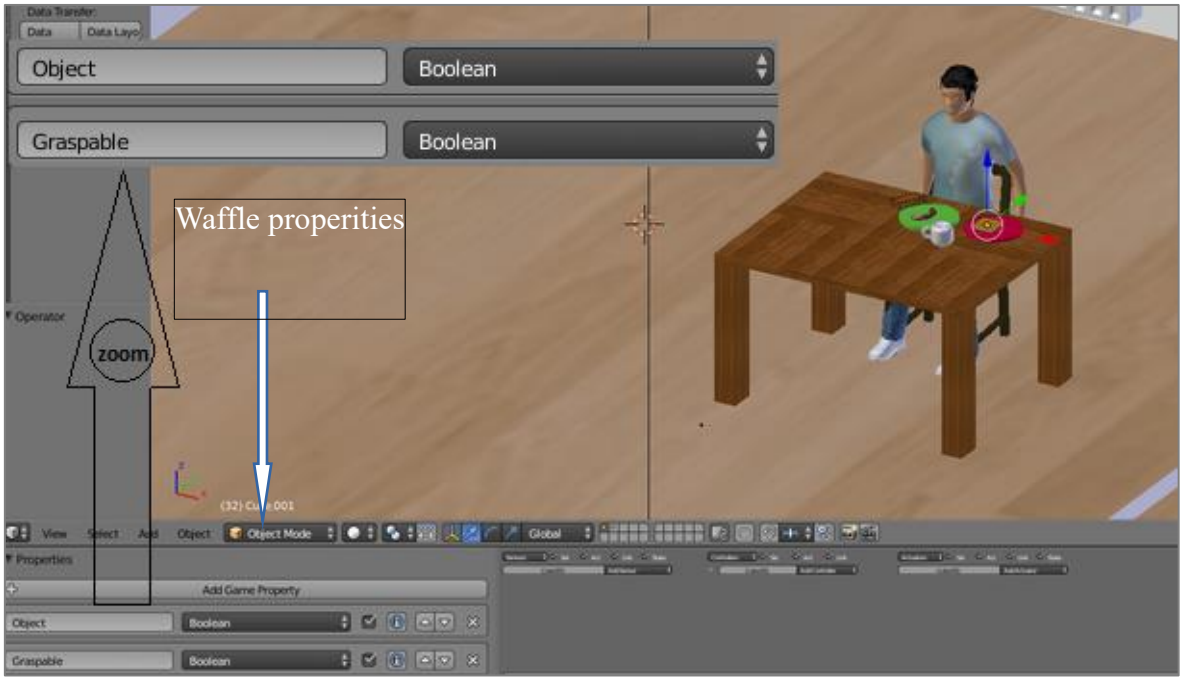


Figure 68: The environment with the cake (waffle) selected and the options for picking up process.

The objects as in [72] that will be defined in three main categories:

- Type: usually the type of the objects that will be recognized
- Label: simple label or name to define the object
- Description: more objects details can be added here

In our work every passive object has been defined in the data base we can see that obviously in the Table (3)

the object	type	label	description
red dish	dish	red	red dish
green dish	dish2	green	green dish
cup	water cup	cup	water cup
sausage	food	sausage	meat
waffle	food	waffle	cake
mouth	skin	mouth	human mouth

Table 3: The objects with its type, label, and description.

Subsequently, the main issue in the physical object definition is to implement a bounding box, which allows specifying the border of the object. The object after that would be comparable with Morse physics.

In Figure (70), we can see the semantic camera fixed in Front of the human model who sits on the chair. It is fixed to discover and detect the whole objects defined in the database.

Figure 69: The environment with the selected red dish and options

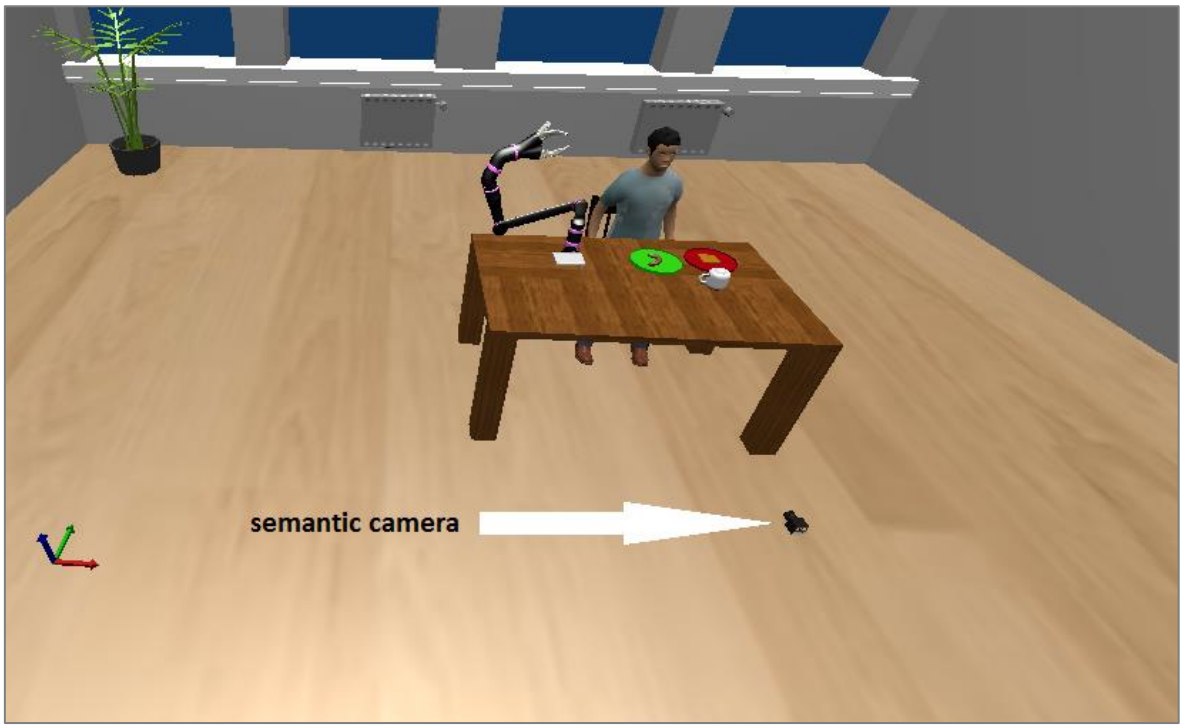


Figure 70: The semantic camera in the environment.

After implementing the objects in blender, as explained in Section 5.1, the input would be the internal data of blender, i.e. every object has a number of details such as label, type, etc. The object's information has stored in blender with its label, type and the other specifications of the object, such as graspable. The semantic camera acts as a sensor when it receives the data and will give output (string) according to the input type (the object image with its type, label); the semantic camera always triggers. If the object found has been defined in the database using blender, the semantic camera will display all the pre-defined information related to the object, and besides a 6D object pose will be sent. The 6D pose combined with the object position in the environment furthers the orientation of the object. The output coordinates represent the centre of the object relative to the centre of the camera (the view region).

5.3 The Objects orientation

The semantic camera gives the position and the orientation of the 3D modelling objects concerning the environment. The homogeneous transformation provides orientation and translation; these transformations are commonly used in the robotic field. To represent the orientation, there are two algorithms, either using Euler angles or Quaternions.

5.3.1 Euler angles

Leonhard Euler discovered these angles. Euler angles are three angles which used for describing the rigid body rotation. These angles are commonly used in aircraft production or 3-D computer graphics. As a simple explanation for these angles, we will take the 3-D rocket in Figure (71) as an example.

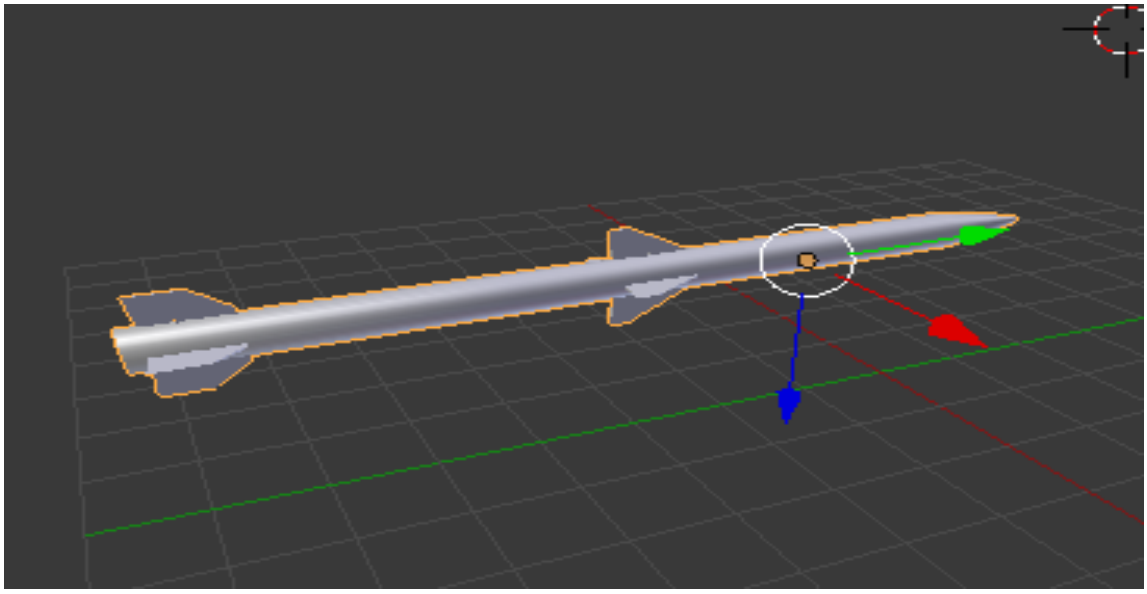


Figure 71: 3D rocket

There are three angles for rotating this rocket:

- pitch: the pitch angle knows the rotation of the red X axis, as in Figure (72), the X axis rotation has to be aligned to the world. In this way, the rocket goes up and down around the X axis.

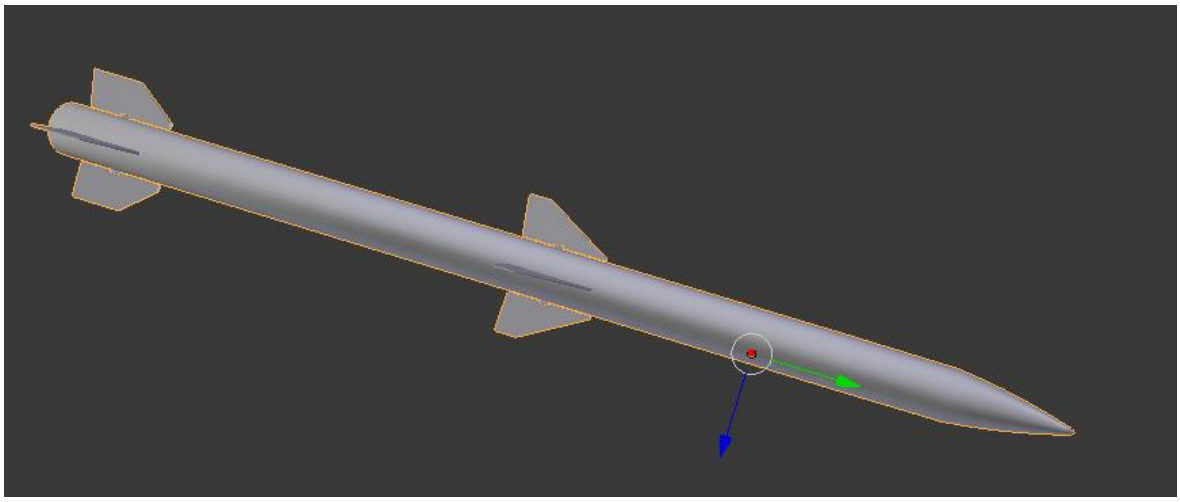


Figure 72: The Pitch rotation angle.

- yaw/head: the angle as the head of the rocket rotates about it as shown in the Figure (73) when the rocket rotates around the Blue Y axis.

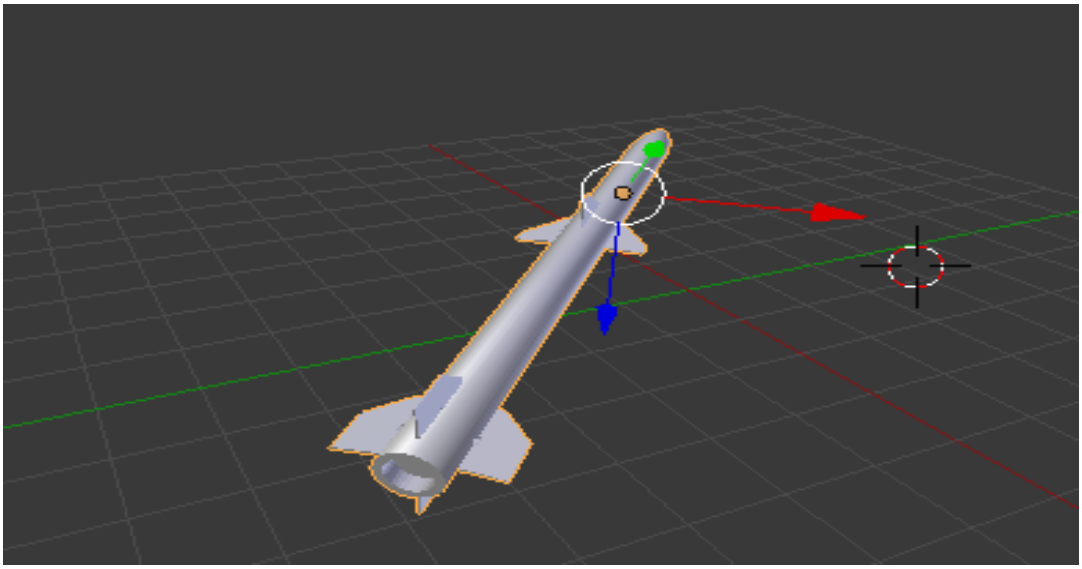


Figure 73: The Yaw rotation angle.

- When the value of the blue Y axis and of the X axis change, the rocket will not rotate about the Y world axis, but around the plane defined by X rotation. The Y rotation is applied with respect to the reference frame of the rocket, not the world as shown in Figure (74).

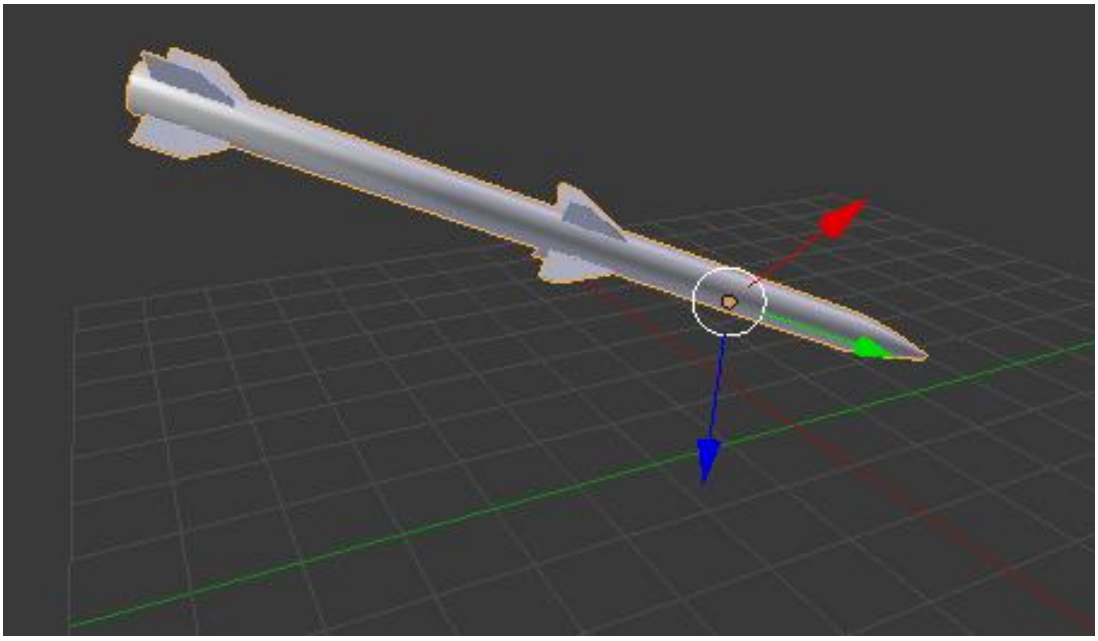


Figure 74: The rotation around two axises.

- Roll: this angle represents the amount of rotation around the rocket itself, as shown in Figure (75), the roll rotation (around green Z) also changes with respect to the rocket's orientation, all the rotations are applied with respect to the rocket, not the world.

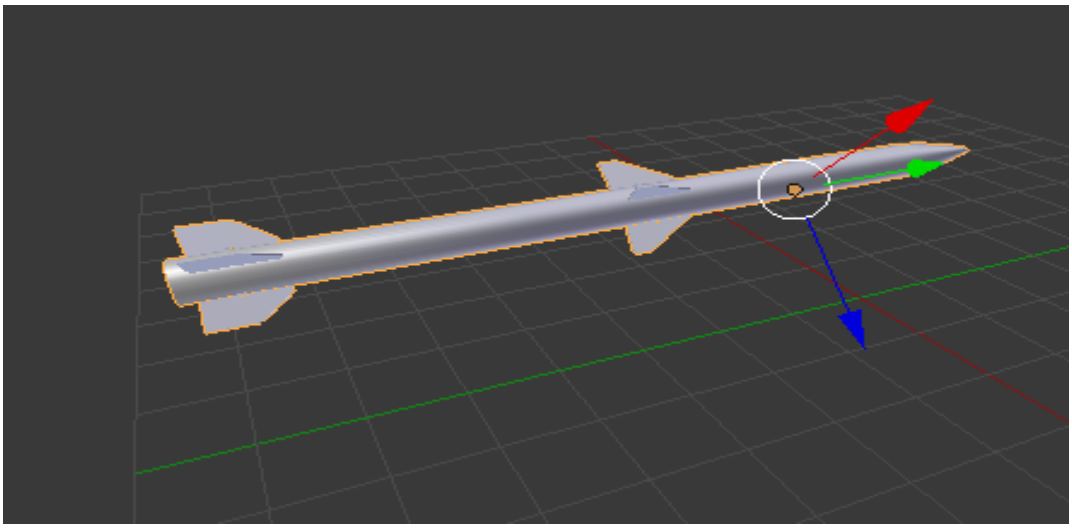


Figure 75: The roll rotation angle.

Euler angles are useful for mobile, robotically filed and flight applications. They are commonly used to control the orientation of the wrist joint in robotic arms. Euler angles also have limitations, such as the Gimbal Lock problem. If we rotate the rocket 90 degrees around the Y-axis (as shown in Figure (76) and try to rotate the rocket around the X red axis or Z green axis, the rocket has the same rotational effect. This means that we have lost a degree of freedom, as shown in Figure (77) and Figure (78).

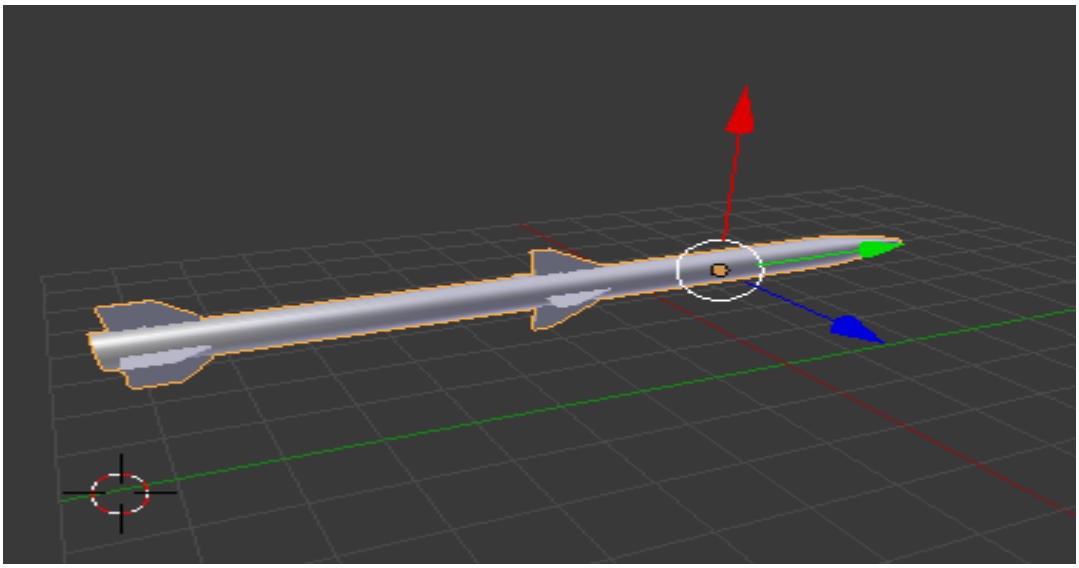


Figure 76: Set 90 degree to the Y-green axis.

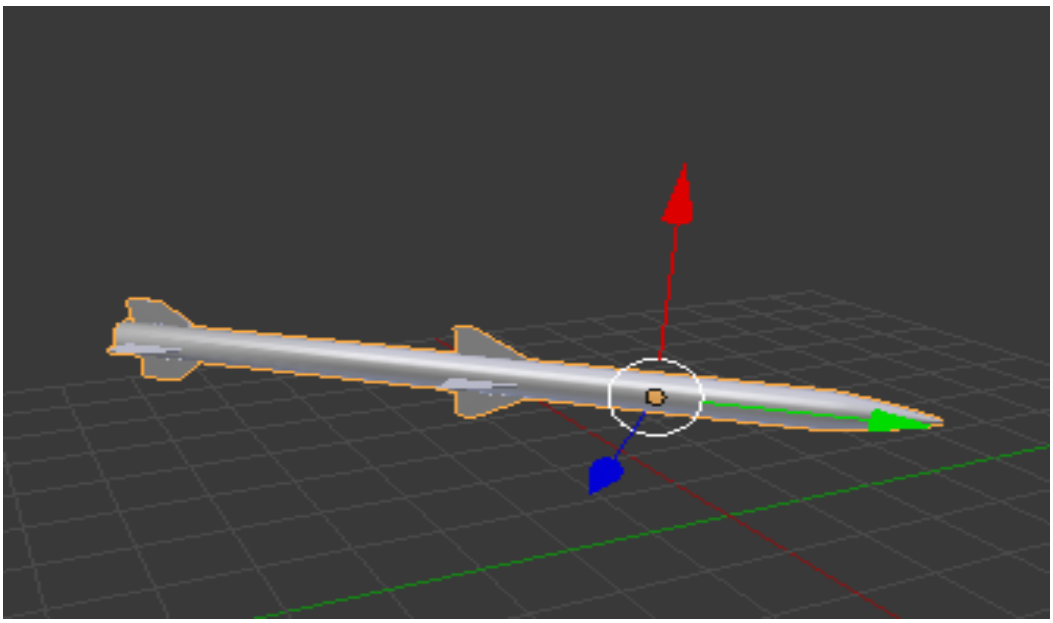


Figure 77: Rocket rotates around the X axis.

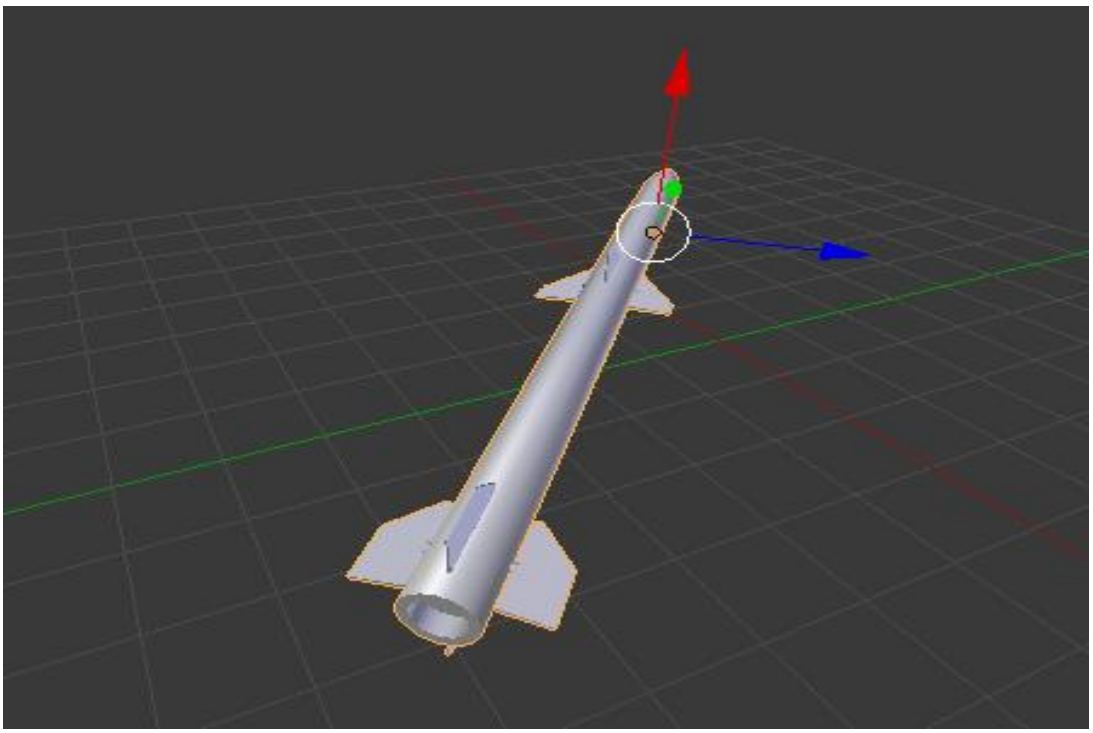


Figure 78: Rocket rotates around the Z axis.

In summary, Euler angles are straightforward and easy to analyse and regulate. However, they do have one limitation problem which is the Gimbal Lock, this problem can influence on the measurement of the orientation and preventing the right measurement.

5.3.2 Quaternion

A discussion of how to rotate a vector in three-dimensions by multiplying three quaternions together follows.

Quaternions as modified from [74] consists of 4-tuples of real numbers: q = (q0, q1, q2, q3). Equivalently, quaternions can be written in the form

$$q = q_0 + q_1 i + q_2 j + q_3 k \quad (32)$$

Where

$$\begin{aligned} i &= (0,1,0,0) \\ j &= (0,0,1,0) \\ k &= (0,0,0,1) \end{aligned} \quad (33)$$

In R^3, a vector is defined as $\mathbf{v} = \upsilon_x \mathbf{i} + \upsilon_y \mathbf{j} + \upsilon_z \mathbf{k}$ where **i**, **j**, and **k** are vectors of the form

$$\begin{aligned} i &= (1,0,0) \\ j &= (0,1,0) \\ k &= (0,0,1). \end{aligned} \quad (34)$$

Any vector in R^3 can map to R^4 by $(\upsilon_x, \upsilon_y, \upsilon_z) \rightarrow (0, \upsilon_x, \upsilon_y, \upsilon_z)$ This mapping can represent by $\mathbf{v} = 0 + \mathbf{v} = \upsilon$. By convention, the i, j, and k components of a quaternion. Thus, any quaternion q may be written as [74]

$$q = q_0 + q = q_0 + q_1 i + q_2 j + q_3 k \quad (35)$$

Where

$$i^2 = j^2 = k^2 = ijk = -1. \quad (36)$$

q_0 represented the scalar part of the quaternion and $q = q_1 i + q_2 j + q_3 k$ represented the vector part of the quaternion [74].

5.3.3 Equality

Two quaternions, $p = p_0 + p_1 \mathbf{i} + p_2 \mathbf{j} + p_3 \mathbf{k}$ and $q = q_0 + q_1 \mathbf{i} + q_2 \mathbf{j} + q_3 \mathbf{k}$ will be equal if and only if their corresponding scalar components are equal [74]:

$$\begin{aligned} p_0 &= q_0 \\ p_1 &= q_1 \\ p_2 &= q_2 \\ p_3 &= q_3 \end{aligned} \quad (37)$$

5.3.4 Addition

Two quaternions, $p = p_0 + p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k}$ and $q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$, can be added together by combining their corresponding scalar and vector components [74]:

$$\begin{aligned} p + q &= (p_0 + p_1\boldsymbol{i} + p_2\boldsymbol{j} + p_3\boldsymbol{k}) + (q_0 + q_1\boldsymbol{i} + q_2\boldsymbol{j} + q_3\boldsymbol{k}) \\ &= (p_0 + q_0) + (p_1 + q_1)\boldsymbol{i} + (p_2 + q_2)\boldsymbol{j} + (p_3 + q_3)\boldsymbol{k} \end{aligned} \quad (38)$$

5.3.5 Multiplication

There are two kinds of multiplication: either scalar multiplication or multiplication of two quaternions. The multiplication of two quaternions as follow: $p = p_0 + p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k}$ and $q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$. [74].

5.3.5.1 Scalar Multiplication

Multiplication of a quaternion $q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$ by a scalar s [74]

$$Sq = Sq_0 + Sq = Sq_0 + Sq_1\boldsymbol{i} + Sq_2\boldsymbol{j} + Sq_3\boldsymbol{k}. \quad (39)$$

5.3.5.2 Multiplication of Two Quaternions

The multiplication of two quaternions, $p = p_0 + \mathbf{p}$ and $q = q_0 + \mathbf{q}$, is defined as [74]

$$pq = (p_0 + \mathbf{p})(q_0 + \mathbf{q}) = p_o q_0 + p_o\mathbf{q} + q_o\mathbf{p} + \mathbf{pq}. \quad (40)$$

Continuation of multiplication pq is given in chapter 5.3.5.5.

5.3.5.3 Associative Under Multiplication

Quaternions [74] are associative under multiplication, for any three quaternions, $p = p_0 + p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k}$, $q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$, and $r = r_0 + r_1\mathbf{i} + r_2\mathbf{j} + r_3\mathbf{k}$, the order in which the quaternions are gathered together [74]:

$$(pq)r = (p_0q_0 + p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{pq})(r_0 + \mathbf{r})$$
$$= p_0q_0r_0 + p_0q_0\mathbf{r} + p_0r_0\mathbf{q} + p_0\mathbf{qr} + q_0r_0\mathbf{0p} + q_0\mathbf{pr} + r_0\mathbf{pq} + \mathbf{pqr}.$$

Factoring p_0 out of the first four terms and **p** out of the last four terms [74],

$$(pq)r = p_0(q_0r_0 + q_0\mathbf{r} + r_0\mathbf{q} + \mathbf{qr}) + \mathbf{p}(q_0r_0 + q_0\boldsymbol{r} + r_0\mathbf{q} + \mathbf{q}\boldsymbol{r})$$
$$= (p_0 + \mathbf{p})(q_0r_0 + q_0\mathbf{r} + r_0\mathbf{q} + \mathbf{qr})$$

By definition, $qr = q_0r_0 + q_0\mathbf{r} + r_0\mathbf{q} + \mathbf{qr}$. [74]

$$(pq)r = p(qr). \quad (41)$$

5.3.5.4 Implications of $\mathbf{i^2 = j^2 = k^2 = ijk = -1}$

The relationships $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$ must hold; some important relationships that follow from these statements will drive [74].

Since $\mathbf{i}^2 = \mathbf{ii} = \mathbf{ijk} = -1$, this implies that

$$\mathbf{i} = \mathbf{jk}. \quad (42)$$

equivalently, since $\mathbf{k}^2 = \mathbf{kk} = \mathbf{ijk} = -1$,[74]

$$\mathbf{k} = \mathbf{ij}. \quad (43)$$

Using Equation (43) then multiplying by **j** on the right,[74]

$$\mathbf{k}(\mathbf{j}) = \mathbf{ij}(\mathbf{j})$$
$$\mathbf{kj} = \mathbf{i}(\mathbf{jj}) = \mathbf{i}(\boldsymbol{j}^2) = \mathbf{i}(-1) = -\mathbf{i}$$

so

$$\mathbf{i} = -\mathbf{kj}. \quad (44)$$

From Equations (42) and (44), the following relationships can find for **i**:[74]

$$\mathbf{i} = \mathbf{jk} = -\mathbf{kj}.$$

Using Equation (42) and multiplying on the left by **j**,[74]

$$(\mathbf{j})\mathbf{i} = (\mathbf{j})\mathbf{jk}$$
$$\mathbf{ji} = (\mathbf{jj})\,\mathbf{k} = (\mathbf{j}^2)\,\mathbf{k} = (-1)\,\mathbf{k} = -\mathbf{k}$$

so

$$\mathbf{k} = -\mathbf{ji}. \tag{45}$$

From Equation (43) and (45), the following [74] relationships are found for **k**:

$$\mathbf{k} = \mathbf{ij} = -\mathbf{ji}.$$

Using Equation (45) and multiplying on the right by **i** [74].

$$(\mathbf{k})\mathbf{i} = -\mathbf{ji}\ (\mathbf{i})$$
$$\mathbf{ki} = -\mathbf{j}\ (\mathbf{ii}) = -\mathbf{j}\ (\mathbf{i}^2) = -\mathbf{j}\ (-1) = \mathbf{j}.$$

Thus,

$$\mathbf{j} = \mathbf{ki}. \tag{46}$$

Using Equation (43) and multiplying on the left by **i** [74].

$$(\mathbf{i})\ \mathbf{k} = (\mathbf{i})\mathbf{ij}$$
$$\mathbf{ik} = (\mathbf{ii})\ \mathbf{j} = (\mathbf{i}^2)\ \mathbf{j} = (-1)\ \mathbf{j} = -\mathbf{j}.$$

Hence,

$$\mathbf{j} = -\mathbf{ik}. \tag{47}$$

From Equations (46) and (47), the following [74] relationships are found for j:

$\mathbf{j} = \mathbf{ki} = -\mathbf{ik}.$

To summarize, when multiplying two quaternions together, the relationships $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$ must always hold. These relationships imply the following [74]:

$$\begin{aligned} \mathbf{ij} &= -\mathbf{ji} = \mathbf{k} \\ \mathbf{jk} &= -\mathbf{kj} = \mathbf{i} \\ \mathbf{ki} &= -\mathbf{ik} = \mathbf{j} \end{aligned} \tag{48}$$

5.3.5.5 Multiplication of Two Quaternions Revisited

The multiplication [74] of two quaternions, $p = p_0 + \mathbf{p}$ and $q = q_0 + \mathbf{q}$, defined as

$$pq = (p_0 + \mathbf{p})(q_0 + \mathbf{q}) = p_o q_0 + p_o\mathbf{q} + q_o\mathbf{p} + \mathbf{pq}.$$

The last term, **pq**, is the product of two vectors. This term can be expanded [74]

$$\begin{aligned} \mathbf{pq} &= (p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k})(q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}) \\ &= p_1\mathbf{i}\,(q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}) + p_2\mathbf{j}\,(q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}) + p_3\mathbf{k}(qq_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}) \end{aligned}$$

$$= p_1q_1\mathbf{ii} + p_1q_2\mathbf{ij} + p_1q_3\mathbf{ik} + p_2q_1\mathbf{ji} + p_2q_2\mathbf{jj} + p_2q_3\mathbf{jk} + p_3q_1\mathbf{ki} + p_3q_2\mathbf{kj} + p_3q_3\mathbf{kk}.$$

The relationships developed in the previous subsection (48), and the fact that $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1$, the above Equation can be further simplified as [74]

$$\begin{aligned}\mathbf{pq} = {} & p_1q_1(-1) + p_1q_2\mathbf{k} + p_1q_3(-\mathbf{j}) + p_2q_1(-\mathbf{k}) + p_2q_2(-1) \\ & + p_2q_3\mathbf{i} + p_3q_1\mathbf{j} + p_3q_2(-\mathbf{i}) + p_3q_3(-1) \\ = {} & -(p_1q_1 + p_2q_2 + p_3q_3) \\ & + [(p_2q_3 - p_3q_2)\mathbf{i} + (p_3q_1 - p_1q_3)\mathbf{j} + (p_1q_2 - p_2q_1)\mathbf{k}].\end{aligned}$$

The term $p_1q_1 + p_2q_2 + p_3q_3$ is the familiar dot product of two vectors. The term $[(p_2q_3 - p_3q_2)\mathbf{i} + (p_3q_1 - p_1q_3)\mathbf{j} + (p_1q_2 - p_2q_1)\mathbf{k}]$ is the cross product of two vectors [74].

$$\mathbf{pq} = -(\mathbf{p}\cdot\mathbf{q}) + \mathbf{p}\times\mathbf{q}. \qquad (49)$$

The above Equation indicates that the product of two vectors in R^4 results in a scalar (the dot product) and another vector (the cross product) [74].

Plugging $\mathbf{pq} = -(\mathbf{p}\cdot\mathbf{q}) + \mathbf{p}\times\mathbf{q}$ into Equation (40), the multiplication of two quaternions, $p = p_0 + \mathbf{p}$ and $q = q_0 + \mathbf{q}$, [74] is

$$\begin{aligned}pq &= p_0q_0 + p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{pq} \\ &= p_0q_0 + p_0\mathbf{q} + q_0\mathbf{p} + [-(\mathbf{p}\cdot\mathbf{q}) + \mathbf{p}\times\mathbf{q}].\end{aligned}$$

The terms p_0q_0 and $-(\mathbf{p}\cdot\mathbf{q})$ are scalars. The other three terms are all vectors. Gathering scalar terms together and vector terms together, the Equation for the multiplication of two quaternions becomes [74]

$$pq = p_0q_0 - (\mathbf{p}\cdot\mathbf{q}) + p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p}\times\mathbf{q}. \qquad (50)$$

5.3.5.6 Closed Under Multiplication

The multiplication [74] of any two quaternions, $p = p_0 + \mathbf{p}$ and $q = q_0 + \mathbf{q}$, will result in a third quaternion, r: $r_0 + \mathbf{r}$:

$$\begin{aligned}pq &= p_0q_0 - (\mathbf{p}\cdot\mathbf{q}) + p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p}\times\mathbf{q} \\ &= (p_0q_0 - p_1q_1 - p_2q_2 - p_3q_3) + (p_0q_1 + p_1q_0 + p_2q_3 - p_3q_2)\mathbf{i} \\ &\quad + (p_0q_2 - p_1q_3 + p_2q_0 + p_3q_1)\mathbf{j} + (p_0q_3 + p_1q_2 - p_2q_1 + p_3q_0)\mathbf{k} \\ &= r_0 + r_1\mathbf{i} + r_2\mathbf{j} + r_3\mathbf{k} \\ &= r.\end{aligned}$$

5.3.5.7 Multiplicative Identity

The product [74] of a quaternion $q = q_0 + \mathbf{q}$ and the multiplicative identity should return the quaternion q. For quaternions, the multiplicative identity is the quaternion with a scalar part of 1 and zero vector part [74]:

$$\begin{aligned} q\,(1) &= (q_0 + \mathbf{q})(1 + \mathbf{0}) \\ &= q_0(1) - (\mathbf{q} \cdot \mathbf{0}) + q_0(0) + (1)\mathbf{q} + \mathbf{q} \times \mathbf{0} \\ &= q_0 + \mathbf{q} \\ &= q. \end{aligned}$$

If the order of multiplication is reversed [74]:

$$\begin{aligned} (1)\,q &= (1 + \mathbf{0})(q_0 + \mathbf{q}) \\ &= (1)q_0 - (0 \cdot \mathbf{q}) + (1)\mathbf{q} + q_0(\mathbf{0}) + \mathbf{0} \times \mathbf{q} \\ &= q_0 + \mathbf{q} \\ &= q. \end{aligned}$$

Since $q(1) = (1)q, 1$ is the multiplicative identity for R^4 [74].

5.3.5.8 Conjugate

The complex conjugate [74] of a quaternion $q = q_0 + \mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$ defined as $q^* = q_0 - \mathbf{q} = q_0 - q_1\mathbf{i} - q_2\mathbf{j} - q_3\mathbf{k}$. The result of the multiplication of a quaternion and its complex conjugate would always be in scalar product. Multiplying q by q^*,

$$\begin{aligned} qq^* &= (q_0 + \boldsymbol{q})\,(q_0 - \boldsymbol{q}) \\ &= q_0^2 - (\boldsymbol{q} \cdot -\boldsymbol{q}) + q_0\boldsymbol{q} - q_0\boldsymbol{q} + \boldsymbol{q} \times (-\boldsymbol{q}). \end{aligned} \qquad (51)$$

The cross product of $\mathbf{q}$ and $-\mathbf{q}$ can be done by pulling the negative out: $\mathbf{q} \times -\mathbf{q} = -(\mathbf{q} \times \mathbf{q})$. The cross product of any vector with itself always be zero ($\mathbf{q} \times \mathbf{q} = \mathbf{0}$), the above Equation will be as follows [74]:

$$\begin{aligned} qq^* &= (q_0^2 + \boldsymbol{q} \cdot \boldsymbol{q}) \\ &= q_0^2 + q_1^2 + q_2^2 + q_3^2 \end{aligned} \qquad (52)$$

$$\begin{aligned} q^*q &= (q_0 - \boldsymbol{q})\,(q_0 + \boldsymbol{q}) \\ &= q_0^2 - (-\boldsymbol{q} \cdot \boldsymbol{q}) + q_0\boldsymbol{q} - q_0\boldsymbol{q} + (-\boldsymbol{q}) \times \boldsymbol{q}. \end{aligned} \qquad (53)$$

The negative sign pulled out if the cross product $(-q) \times q = -(q \times q) = 0$. The above Equation will be in a simple form as follows [74]:

$$\begin{aligned} q^*q &= q_0^2 + \boldsymbol{q}\cdot\boldsymbol{q} \\ &= q_0^2 + q_1^2 + q_2^2 + q_3^2 \end{aligned} \quad (54)$$

The product of a quaternion and its complex conjugate is equal to [74]:

$$qq^* = q^*q = q_0^2 + q_1^2 + q_2^2 + q_3^2. \quad (55)$$

5.3.5.9 Norm

The norm [74] of a quaternion $q = q_0 + \mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$ defined

$$\begin{aligned} N(q) &= \sqrt{q^*q} \\ &= \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \\ &= \|q\|. \end{aligned} \quad (56)$$

5.3.6 Rotations in Three-Space

As an example [74], for the 3-D rotation the vector $\mathbf{v} = v_x\mathbf{i} + v_y\mathbf{j} + v_x\mathbf{k}$ rotates through an angle 2θ about vector **u**. The vector **u** is a unit vector and both **v** and **u** are pass through the origin point. The vector $\mathbf{w} = w_x\mathbf{i} + w_y\mathbf{j} + w_z\mathbf{k}$ represents the image of **v** after rotation.
The vectors **v** and **w** are in R^3 should be mapped to R^4

$$\begin{aligned} \boldsymbol{v} &= 0 + \boldsymbol{v} \\ \boldsymbol{w} &= 0 + \boldsymbol{w}. \end{aligned} \quad (57)$$

The quaternions $\upsilon = 0 + \mathbf{v}$ and $\omega = 0 + \mathbf{w}$ usually represented by the vectors **v** and **w**.

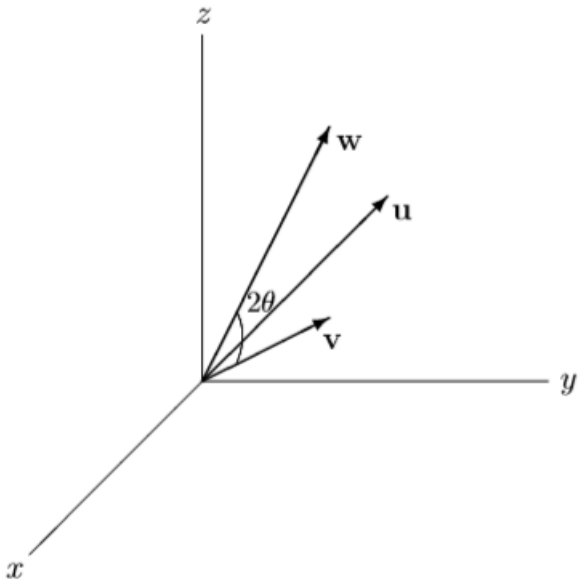


Figure 79: The vector **v** will be rotated clockwise about u through an angle of 2θ [74].

By multiplying the vector **v** by a quaternion of the form $q = \cos\theta + \mathbf{u}\sin\theta$ then multiplying by a quaternion form $q* = \cos\theta - \mathbf{u}\sin\theta$, the vector **w** which represented an image of rotation the vector **v** about **u** through angle 2θ [74]

$$\begin{aligned} \boldsymbol{W} &= q\mathbf{v}q^{*} \\ &= (cos\theta + \mathbf{u}\,sin\,\theta)\,\mathbf{v}\,(cos\,\theta - \mathbf{u}\,sin\,\theta). \end{aligned} \quad (58)$$

The Equation (60) shows a vector **w** that is the image of **v** rotated about **u** through an angle 2θ.

To rotate [74] a vector $\mathbf{v} = \upsilon_x\mathbf{i} + \upsilon_y\mathbf{j} + \upsilon_z\mathbf{k}$ about an axis $\mathbf{u} = \mathbf{u}_x\mathbf{i} + \mathbf{u}_y\mathbf{j} + \mathbf{u}_z\mathbf{k}$ through an angle 2θ where **v** and **u** pass through the origin point, the following steps should be done:

1 The vector $\mathbf{v} = \upsilon_x\mathbf{i} + \upsilon_y\mathbf{j} + \upsilon_z\mathbf{k}$ must map from R^3 to R^4 [74]:

$$\boldsymbol{\nu} = = 0 + \mathbf{v}. \quad (59)$$

2 The axis of rotation must have length 1. otherwise the desired axis of rotation, $\mathbf{m} = m_x\mathbf{i} + m_y\mathbf{j} + m_z\mathbf{k}$, does not have length 1, then **m** would be [74]:

$$m = \frac{m_x\boldsymbol{i} + m_y\boldsymbol{j} + m_z\boldsymbol{k}}{\sqrt{m_x^2 + m_y^2 + m_z^2}}. \quad (60)$$

3 Plug **v**, **u**, and 2θ into $\mathbf{w} = \mathbf{v}\cos 2\theta + (\mathbf{u} \times \mathbf{v})\sin 2\theta + 2(\mathbf{u} \cdot \mathbf{v})\,\mathbf{u}\sin 2\theta$ [74]:

$$\mathbf{w} = \mathbf{v}\cos 2\theta + (\mathbf{u}\times\mathbf{v})\sin 2\theta + 2(\mathbf{u}\cdot\mathbf{v})\,\mathbf{u}\sin^2\theta. \quad (61)$$

4 The quaternion ω = 0 + **w** must have a zero-scalar. The vector **w** which represented the rotation of **v** around **u** through an angle of 2θ [74].

Another example [74] for rotating the vector **v** = 3**i**−5**j** + 2**k** 90 ° about the axis (**i** + **j**). The first step is to normalize the rotation axis:

$$\begin{aligned}\boldsymbol{u} &= \frac{1}{\sqrt{1^2+1^2+0^2}}\boldsymbol{i} + \frac{1}{\sqrt{1^2+1^2+0^2}}\boldsymbol{i} + \frac{0}{\sqrt{1^2+1^2+0^2}}\boldsymbol{k} \\ &= \frac{1}{\sqrt{2}}\boldsymbol{i} + \frac{1}{\sqrt{2}}\boldsymbol{j}\end{aligned} \quad (62)$$

Now, find the rotating vector, **w** as in Equation (61) [74]

$$\boldsymbol{w} = \mathbf{v}\cos 2\theta + (\mathbf{u}\,\mathrm{x}\,\mathbf{v})\sin 2\theta + 2(\mathbf{u}\cdot\mathbf{v})\,\mathbf{u}\sin^2\theta \quad (63)$$

$$\begin{aligned}\cos 2\theta &= \cos 90^\circ = 0 \\ \sin 2\theta &= \sin 90^\circ = 1 \\ \sin 2\theta &= \sin^2 45^\circ = \left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2}\end{aligned} \quad (64)$$

[74] Plugging the above trig identities in,

$$\begin{aligned}\boldsymbol{w} &= \mathbf{v}\cos 90^\circ + (\boldsymbol{u}\times\boldsymbol{v})\sin 90^\circ + 2(\mathbf{u}\cdot\mathbf{v})\,\mathbf{u}\sin^2 45^\circ \\ &= \mathbf{v}(0) + (\mathbf{u}\times\mathbf{v})(1) + 2(\mathbf{u}\cdot\mathbf{v})\,\mathbf{u}\frac{1}{2} \\ &= (\mathbf{u}\times\mathbf{v}) + (\mathbf{u}\cdot\mathbf{v})\mathbf{u}.\end{aligned} \quad (65)$$

To calculate the dot and cross products of $\mathbf{u} = \frac{1}{\sqrt{2}}\boldsymbol{i} + \frac{1}{\sqrt{2}}\boldsymbol{j}$ and v = 3**i** − 5**j** + 2**k** separately and then plug them to find **w**, the dot product of **u** and **v** is [74]

$$\begin{aligned}\mathbf{u}\cdot\mathbf{v} &= \left(\frac{1}{\sqrt{2}}\boldsymbol{i} + \frac{1}{\sqrt{2}}\boldsymbol{j}\right)\cdot(3\boldsymbol{i} - 5\boldsymbol{j} + 2\boldsymbol{k}) \\ &= \frac{1}{\sqrt{2}}(3) + \frac{1}{\sqrt{2}}(-5) + 0(2) \\ &= \frac{3-5}{\sqrt{2}} \\ &= \frac{-2}{\sqrt{2}} \\ &= -\sqrt{2}.\end{aligned} \quad (66)$$

[74] And the cross product of **u** and **v** is

$$\begin{aligned}\mathbf{u}\times\mathbf{v} &= \left(\frac{1}{\sqrt{2}}\boldsymbol{i}+\frac{1}{\sqrt{2}}\boldsymbol{j}\right)\times(3\boldsymbol{i}-5\boldsymbol{j}+2\boldsymbol{k})\\ &= \left[\frac{1}{\sqrt{2}}(2)-0(5)\right]\boldsymbol{i}+\left[0(3)-\frac{1}{\sqrt{2}}(2)\right]\boldsymbol{j}+\left[\frac{1}{\sqrt{2}}(-5)-\frac{1}{\sqrt{2}}(3)\right]\boldsymbol{k}\\ &= \frac{2}{\sqrt{2}}\boldsymbol{i}-\frac{\mathbf{2}}{\sqrt{2}}\boldsymbol{j}+\frac{-5-3}{\sqrt{2}}\boldsymbol{k}\\ &= \sqrt{2}\boldsymbol{i}-\sqrt{2}\boldsymbol{j}-4\sqrt{2}\boldsymbol{k}.\end{aligned} \quad (67)$$

The dot and cross product can be plugged in Equation (65) to solve for **w**,[74]

$$\begin{aligned}\mathbf{w} &= (\mathbf{u}\times\mathbf{v})+(\mathbf{u}\cdot\mathbf{v})\mathbf{u}\\ &= \left(\sqrt{2}\boldsymbol{i}-\sqrt{2}\boldsymbol{j}-4\sqrt{2}\boldsymbol{k}\right)+\left(-\sqrt{2}\right)\left(\frac{1}{\sqrt{2}}\boldsymbol{i}+\frac{1}{\sqrt{2}}\boldsymbol{j}\right)\\ &= \left(\sqrt{2}\boldsymbol{i}-\sqrt{2}\boldsymbol{j}-4\sqrt{2}\boldsymbol{k}\right)+(-\boldsymbol{i}-\boldsymbol{j})\\ &= \left(-1+\sqrt{2}\right)\boldsymbol{i}-\left(1+\sqrt{2}\right)\boldsymbol{j}-4\sqrt{2}\boldsymbol{k}.\end{aligned} \quad (68)$$

As shown in Figure (80), the vector **v** = 3**i** − 5**j** + 2**k** has been rotated 90° about the axis **i** + **j** into the vector **w** = (−1 + √2) **i** − (1 + √2) **j** − 4√2 **k**.

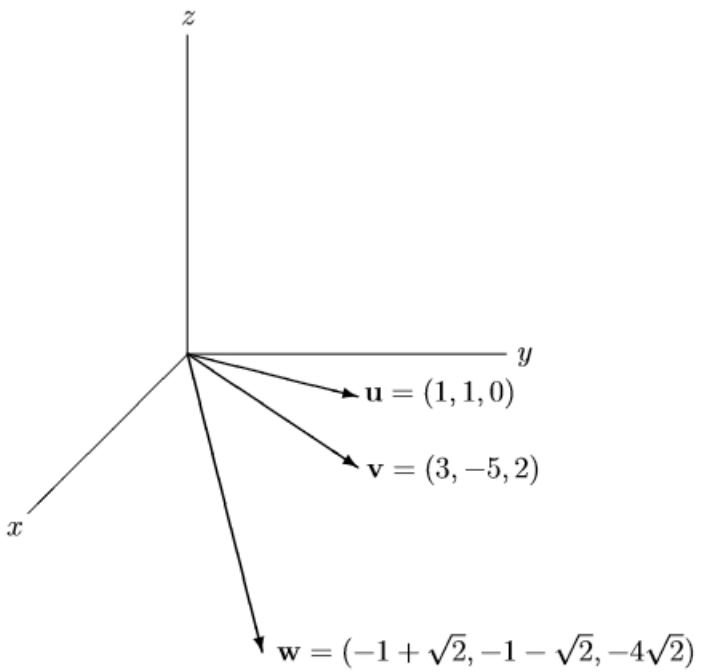


Figure 80: The vector **v** = 3i − 5j + 2k is rotated 90° clockwise about the axis (**i** + **j**) into **w** = (−1 +√2)**i** − (1 + √2) j − 4√2**k. [**74].

5.4 Control the semantic camera

After the definition of each object in the environments (as shown in section above) is complete, the discovery of the coordinates, type, label and state of the object follows. The camera looks for the objects that have been defined before. The ROS topic for the semantic camera gives the status of this topic, representing the information stored, which is related to many objects in the simulated environment. This Q quaternion which is an extension of a non-switching operation of complex numbers provides a suitable and easy solution for calculating the orientation or rotation of the 3D objects, it can also represent the orientation or 3D rotation by these four values. The quaternion represents two things, first (x, y, z) components which represent the vector and W represents the scalar value of the rotation around that vector, as modified from [76].

This quaternion call will always be the normal of quaternion q when it rotates interactively in the quaternion – mode [76]:

$$||q|| = \sqrt{X^2 + Y^2 + Z^2 + W^2} \quad (69)$$

The following formulation applies: W = cos(a/2) [76] to find the actual rotation at the defined angle, where a is really the angle of rotation we're looking for, in the Figure (81) is shown obviously that the objects with its coordinates and types.


```
data: [{"type": "water Cup", "position": [0.01475217193365097, -0.6894313097000122, 0.9663282632827759], "description":
 "water cup", "orientation": {"y": 0.0, "x": 0.0, "z": -0.9876964092254639, "w": 0.15638384222984314}, "name": "CUP"},
{"type": "", "position": [0.23577862977981567, -0.37367385625839233, 0.9209336042404175], "description": "HUMAN Mouth",
 "orientation": {"y": 0.0, "x": 0.0, "z": 0.0, "w": 1.0}, "name": "MOUTH"}, {"type": "Dish", "position": [-0.2998799681
663513, -0.3997405767440796, 0.8795033097267151], "description": "green color", "orientation": {"y": 0.0, "x": 0.0, "z"
: 0.0, "w": 1.0}, "name": "GREEN DISH "}, {"type": "Dish 2", "position": [0.06851635873317719, -0.35229092836380005, 0.
8711251616477966], "description": "RED COLOR", "orientation": {"y": 0.0, "x": 0.0, "z": 0.0, "w": 1.0}, "name": "RED di
sh "}]
---
data: [{"type": "water Cup", "position": [0.01475217193365097, -0.6894313097000122, 0.9663282632827759], "description":
 "water cup", "orientation": {"y": 0.0, "x": 0.0, "z": -0.9876964092254639, "w": 0.15638384222984314}, "name": "CUP"},
{"type": "", "position": [0.23577862977981567, -0.37367385625839233, 0.9209336042404175], "description": "HUMAN Mouth",
 "orientation": {"y": 0.0, "x": 0.0, "z": 0.0, "w": 1.0}, "name": "MOUTH"}, {"type": "Dish", "position": [-0.2998799681
663513, -0.3997405767440796, 0.8795033097267151], "description": "green color", "orientation": {"y": 0.0, "x": 0.0, "z"
: 0.0, "w": 1.0}, "name": "GREEN DISH "}, {"type": "Dish 2", "position": [0.06851635873317719, -0.35229092836380005, 0.
8711251616477966], "description": "RED COLOR", "orientation": {"y": 0.0, "x": 0.0, "z": 0.0, "w": 1.0}, "name": "RED di
sh "}]
```


Figure 81: The output of the semantic camera (each object with its details).

These coordinates are presented to the camera system. Let us assume that the general coordinates matrix will be B, the matrix from the camera is B where i is the number of objects. We have the base coordinates matrix which is represented by Bi where I represent the robot, camera or human, etc. Every object in the environment has a coordinates matrix according to the base of the environment.

$$B_{base}^{camera} X B_{robot}^{base} = B_{robot}^{camera} \quad (70)$$

then according to the results which have been extracted from the camera system

$$B_{Camera}^{Object} X\ B_{robot}^{Camera} = B_{robot}^{Object} \quad (71)$$

So, the robot will go directly to the object coordinates, each object has a specific transformation matrix.

As shown the objects classified to:

1 Red dish: represented the first type of the food. (eat)

2 Green dish: represented the second type of the food. (eat)

3 Water cup: represented the drinkable liquid. (drink)

In Figure (82), the ROS schematic for the semantic camera control node is shown. This node is programmed to take the coordinates and the other object details that have been discovered by the camera, this information is then stored for later use in the system. The ROS message carried on the ROS topic semantic camera represents the environment object's details. These details will enter the main node called Morse and then be processed. The ROS topic /fake robot/semantic camera will receive the message about each object's details and as explained before, be used later in the arm coordinates. In Table (4), all of the objects will need to be defined and stored for later use.

	Position			**Orientation**			
The object	X	Y	Z	X	Y	Z	W
Cup	0.01475	-0.68943	0.96633	0	0	-0.98770	0.15638
Red dish	0.06852	-0.35229	0.87113	0	0	0	1
Green dish	-0.29988	-0.39974	0.87950	0	0	0	1
Mouth	0.23578	-0.37367	0.92093	0	0	0	1

Table 4: Every object with its coordinates

Moreover, we can see that the camera node sends the specific messages to the n_taker, which represents the connected ring between the node and the whole environment. The semantic camera sensor is applied to the fake robot as a child relationship with it.

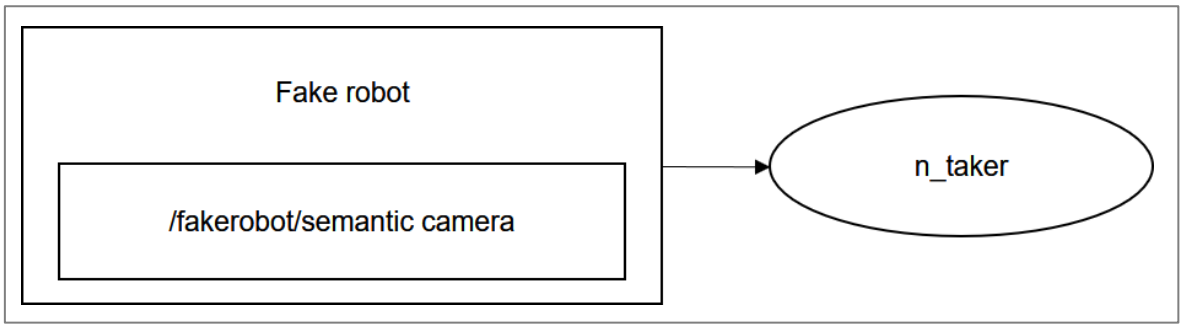


Figure 82: The semantic camera system Node.

It is not permitted in the simulation to simultaneously implement many sensors or actuators, but if the user wants to implement these to work together, the user should add a virtual fake robot. These sensor and actuators are to act as stand-alone elements ([75]). This fake robot is similar to the parent of the sensors because the robot does not show any imagery and has one blender empty, its sole aim is to provide the base for connecting sensors with the parent of as many sensors/actuators as necessary can be connected to a fake robot.

6 The sound recognizer system

The sound recognition system is one of the essential steps, the exact word required is most accurate. The algorithms used in this project are based on [77] and has improved using our new algorithm, this algorithm used with some editing according to the project requirements, the reader can also find more details and examples. One of the difficulties is how to communicate the system with other systems, so we found the best way to program ROS nodes that are responsible for recognizing words, but this technique bumped into several problems.

6.1 The system scripts

The sound system will consist of three main subdivided script:

1 The main script: will run the node to publish the messages on the specific topic also will be like the administer which arranges the function. Also, responsible for launching the node of the recognition system

2 The dictionary script (.doc): in this script contains the words as real pronounced and as follow:

WATER W AO T ER

RED R EH D

GREEN G R IY N

3 The static language model script (. lm): this is an essential part of our sound system; it is a type of the language models (explained in 6.5). As shown below:

-2.8116 GREEN -0.3004

-2.8116 RED -0.2990

-2.8116 WATER -0.2647

6.2 The language model

The main part of the sound recognition system, it contains the decoder of the words that could be recognized, there are different types of the model language like keywords list, grammars and static language model which used in our system [77].

6.3 Keywords list

It is the type of language template, the principle in this model is set as a threshold of every word. It can be easily detected with the speech; the threshold is different from the long key phrase and the short key phrase; the keywords list has also supported the pocket sphinx [77].

6.4 Grammars

This model would be easy to control, and there are not many options for the word sequences, i.e. the word only has one or two possible options that can be detected [77]. This method must precisely and carefully specify the input data. If the user makes unintended mistakes and misspells some words, the recognition process fails and the words are not used. [77] The grammars language model can be created with the Java Speech Grammar Format JSGF and have a file extension .gram or .jsgf. Users should avoid using complex grammar and sentences with complex rules which will take a long time to recognize, as such a process could be subject to failure [77].

6.5 Static language model

In this model, the design contains many words or a combination of words and it is quite possible to edit these terms easily. Therefore, it is highly recommended to use the sound system design, so the user can simply treat this model by saying everything in everyday language and then defining, saving and programming these words with a simple engineering effort [77].

For [77] the new generation of sound recognition interfaces, the static language model uses the above properties because, when depending on natural language, it becomes more effective to use this model to escape the control language of the old versions or generations of commands. There are several methods which are used to create the static language model [77]:

- Small amount of data: using an on-line quick web service [77].
- Large amount of data: using a CMU language modelling toolkit [77].
- Sometimes needing to build a favourite toolkit: using ARPA [77].

"The static language model can be saved into three formats: [77]

- Text ARPA: it will take up a lot of storage space, but be very easy and to edit. The extension will be. Lm [77].

- Binary BIN: this takes less space and is faster to load but is difficult to edit. The extension will be. lm.bin [77].
- Binary DMP: this is not recommended for use, because it is very complicated [77].

[77] Changing between formats and changing from one to another is permissible, building the static language and preparing the suitable text or words, and then training the ARPA. This [77] can be done using toolkits like SRILM, CMUCLMTK, IRSLM and MITLM, and is recommended if the user wants an efficient and straightforward system without complicated control commands. If the language is English, then a file.tx should be created that includes the necessary project words (red, green, water), then it would need to be loaded into the project. This page is then able to create a.dic and .lm file and download it once a launch script has been programmed to be used as a guide between the two scripts for the information cycle. The program is then written and programmed to create the sound recognition node inside the ROS[77].

In order to illustrate the numbers in the .lm script, the language model based on the probably distribution for the words. In our work each one of the words (green, red, water) has a threshold which has been specified by the probably distribution. As modified from [88] the sequence of the N word represented by the term n-gram, the n-grams has many types according to the word sequence such as unigram for one word sequence as we have in our work , bigram for two words sequence, n-gram for N words sequence. The sequence of the N words will be represented by (w_1, w_2, ..., w_n).

The value of joint probabilities for each word in a sequence would be represented by [88]:

$$\mathrm{P}\ (w_1, w_2, ..., w_n) \tag{72}$$

this can compute in the easiest way by using the chain rule of the probability[88]:

$$\begin{aligned} \mathrm{P}(X_1 \dots X_n) &= P(X_1)P(X_2|X_1)P(X_3|X_1^2) \dots P(X_n|X_1^{n-1}) \\ &= \prod_{k=1}^{n} P\left(X_k|X_1^{k-1}\right) \end{aligned} \tag{73}$$

after applying this to words we get [88]:

$$\begin{aligned} \mathrm{P}(w_1^n) &= P(w_1)P(w_2|w_1)P(w_3|w_1^2) \dots P(w_n|w_1^{n-1}) \\ &= \prod_{k=1}^{n} P\left(w_k|w_1^{k-1}\right) \end{aligned} \tag{74}$$

[88] The chain rule is the link between computing the joint probability of the word sequence and computing the conditional probability of a word given previous words. The suggested method for estimating the joint probability in Equation (73) does not tend to be the optimum solution due to the creativity in the language, and any context might have never occurred before [88] so that there are many algorithms such as Markov Models, Maximum-Likelihood Estimates for computing the practical value (threshold) of the probability [88].

6.6 The control design

The ROS pocket defines everything concerning, the microphone and how to connect to the program. It contains the data line, how to get the sound and how to handle it. This pocket usually requires a model and dictionary file for the sound sensor access, which is programmed to recognize specific words, as shown in Figure (83). The ROS node recognizer consists of two parts for processing and one for output connection between the output of the ROS node and the output environment.

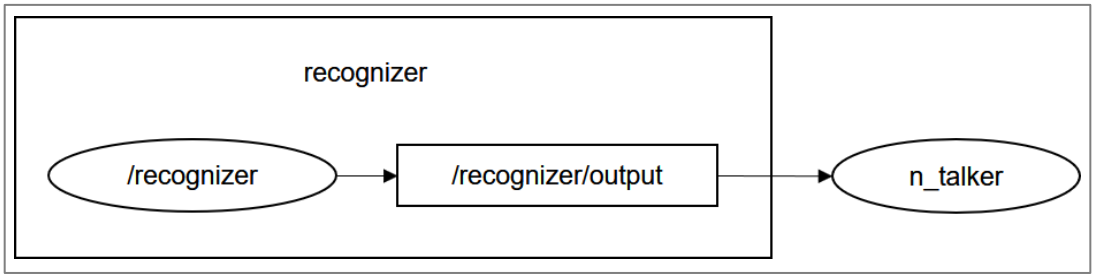


Figure 83: The ROS topic recognizer schematic.

In Table (5), we test and evaluate the sound recognition system. In the test, we repeated the word several times and proved whether the recognizer recognized the word, which involved a different time period because the word was repeated many times. In our table, we took only five random attempts. The repeat of each word will first be alone; for example, the word red in the first attempt is repeated many times without another word, just red to see the efficiency of the system. When repeating the same words many times and testing the recognition time in terms of whether it would still be constant or the time would be in a closed period. The second attempt is to repeat the word after another word, for example red-green-red, the third would be like red-green-green-red, the fourth red-green-water and the last one could be a random sequence such as red-green-water-water-green-green-red-water-red-green.

The words	No. of repeated	Time to recognize (m sec)				
Red	5	0.543	0.789	0.876	0.854	0.986
green	5	0.456	0.675	0.789	0.921	0.965
water	5	0.523	0.603	0.890	0.876	1.02

Table 5: The response of the sound recognition system with time

From this information, the average time for recognizing the red word is about 0.8096 msec; therefore, the response of the system will be very good. For green and water, it is 0.7612 and 0.7824 msec, respectively.

One of the improvements that have been improved the system for recognizing the words precisely and quicker than the old system this has been done in the following steps:

1 Edit the threshold value for the word

2 Establish the static language

3 Prepare the delectable text or words

4 Learn ARPA

5 Create the files.lm and.dic

6 Import it into the pocketsphinx

This improvement shows new results as in Figure (84) which represented a comparison of the detection time of the word red between the old and the new method, a comparison of green - word detection time between old and new method shown in the Figure (85), the last figure shows the comparison of the white – word between the old and the new approach. In those figures the right side represents the detection time for the sound system with the old language model probably threshold, and the left side the represents the new detection time with the new value of the language model threshold.

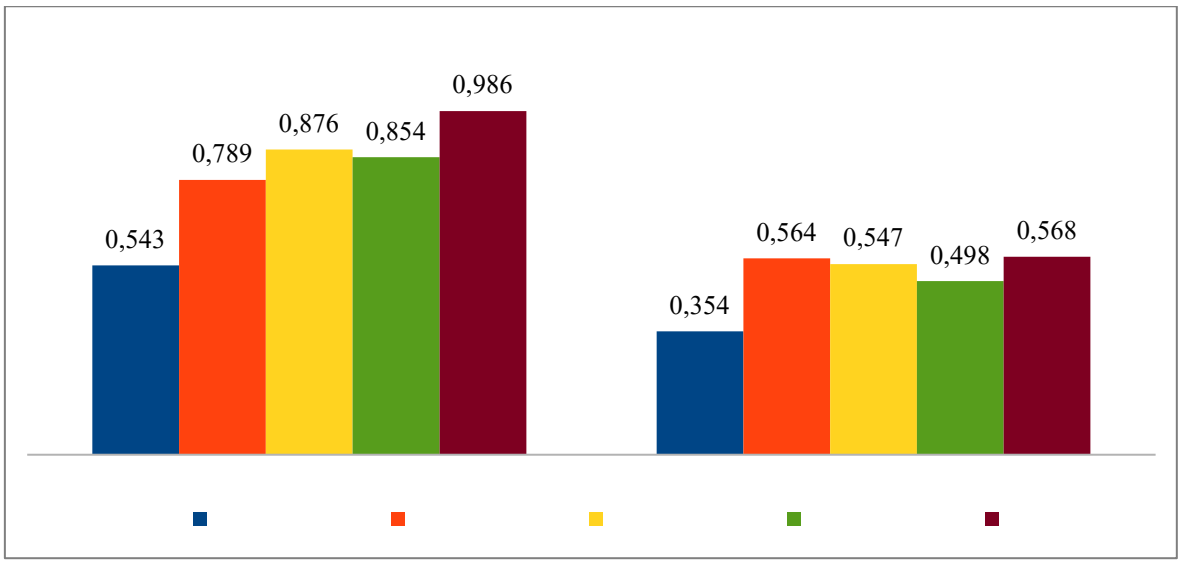


Figure 84: The detection time for the red word

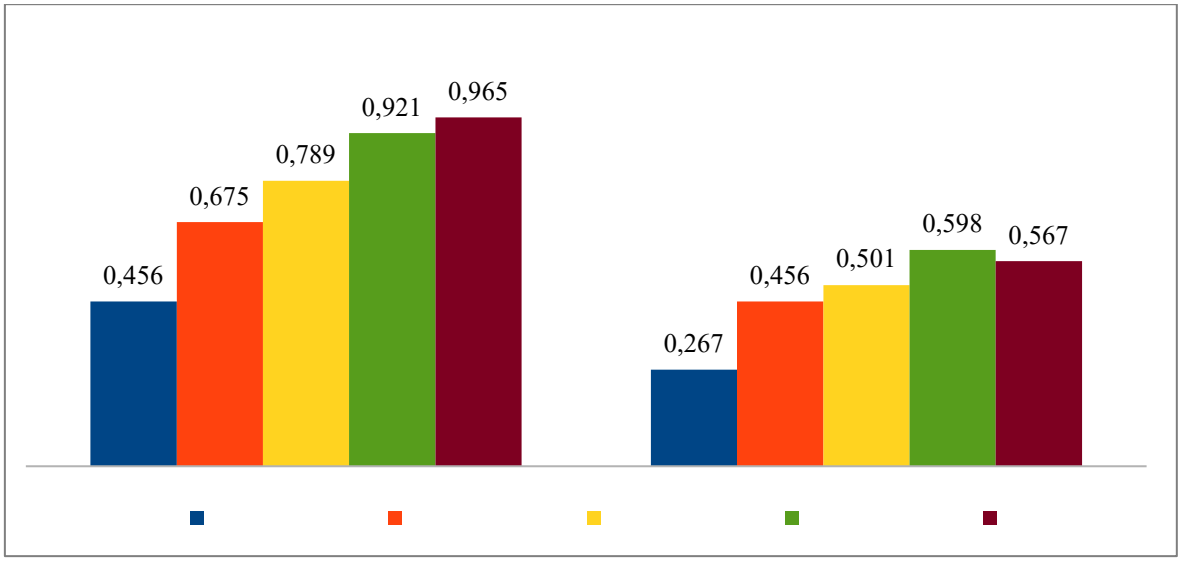


Figure 85: The detection time for the green word

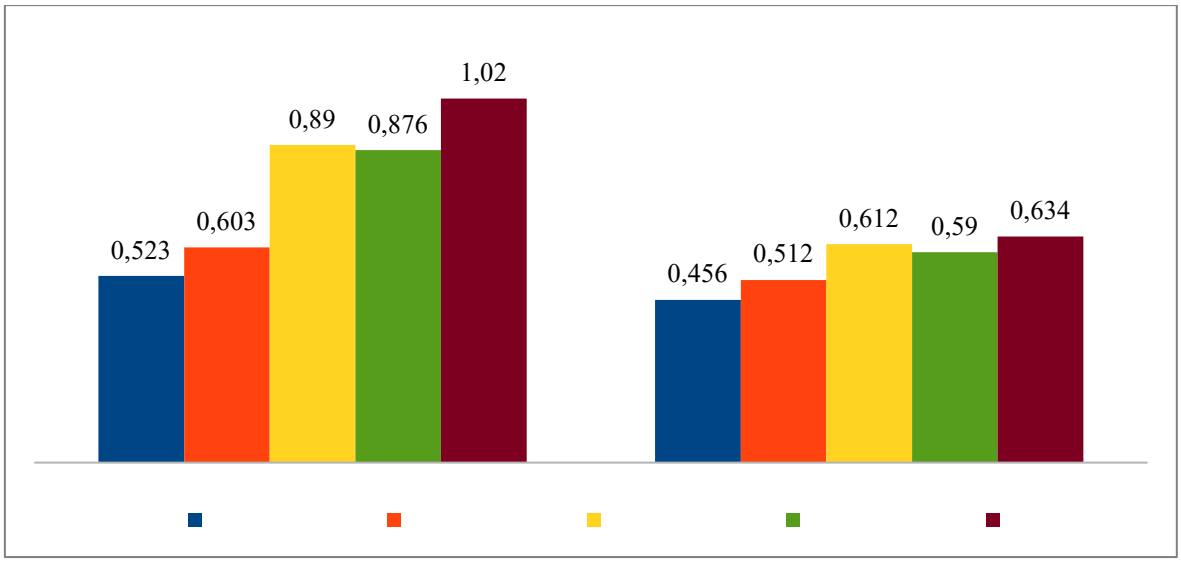


Figure 86: The detection time for the white word.

7 End effector task space position optimization using the PID controller

The PID controller is widely used in different control systems. It is a technique used today in industrial fields to achieve a stable system, as it improves and increases stability. This occurs by reducing the error with regard to physical parameters such as position and temperature. The PID working principles are based on the gain of the PID types, i.e. the PID controller is based on proportional, integral and derivative gains. These gains tune their values in different ways, as is clarified in the following sections.

The first theory of the PID controller was founded in 1922 when the Russian American engineer Nicolas Minorsky refined an automatic steering system for the U.S. Navy. This system is based on the steerman's observations of the rating change between the actual error and past error, this ratio keeps the ship on the correct path [78].

7.1 Related work

As an overview for our suggested algorithm, the position of the end effector consists of x, y, z and each one has an error ratio due to the simulated environment; so we suggest using the PID controller to reduce the sole error ratio problem. [79] This proposed approach aims to find an auto-tuning method for PID controllers of robotic arm manipulators. [79] This method for the PID controller which is used for finding the absolute value to achieve the best trajectory path uses particle swarm optimization for the task and deals with the maximum joint torque and maximum position error, and the integral of the error used for an optimization algorithm. In this field, there are two ways to optimize the arm motion. First is to use joint space to control the joint rotation angle as [80] this controls the joint angle position using the PID controller. [80] They then adjusted the PID gain value, which oscillated between different joint angles, the PID controller tuned every instant to avoid the high peak overshoot through the motion to the stable joint angle value. They then medalled and designed two links robotic arm using the simulation and implemented the 2- DOF Robot Forward and Inverse Kinematics with Denavit-Hartenberg parameters (DH). The second way to control the arm motion is by using the end-effector position. [81] The suggested (PID) [proportional, integral and derivative] and friction compensator controller increased the motion stability and reduced the position error into the trajectory tracking and point control. [81] Due to the significant error in the positioning system, they then presented the PID position controller to control the position of the two linked robot arm joints, the modelling

of the vertical movement represented the kinematics and dynamics of the two linked robotic arms.

7.2 PID Theory

The main purpose of the PID is to reduce the error in the system parameters, which then increases the system stability. The PID consists of three controllers; proportional, integral and derivative. The control block, as shown in Figure (87), consists of the input and the PID blocks, which are connected to the plant or process and then to the output.

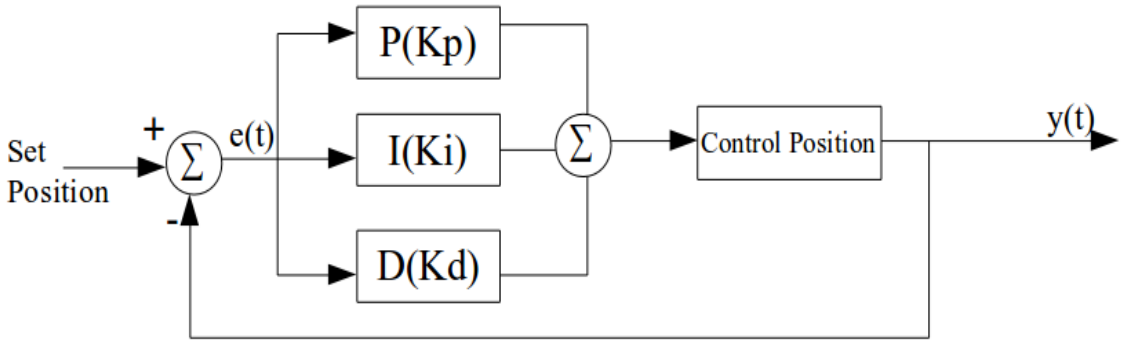


Figure 87: The control block diagram

The equation for the PID algorithm would be as follow:

$$u(t) = K\,(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt} \qquad (75)$$

Where y is the measured process variable, u is the control signal and e is the control error [85].

There some parameters should be clarified with regards to the response Figure (88) which is modified from [82]:

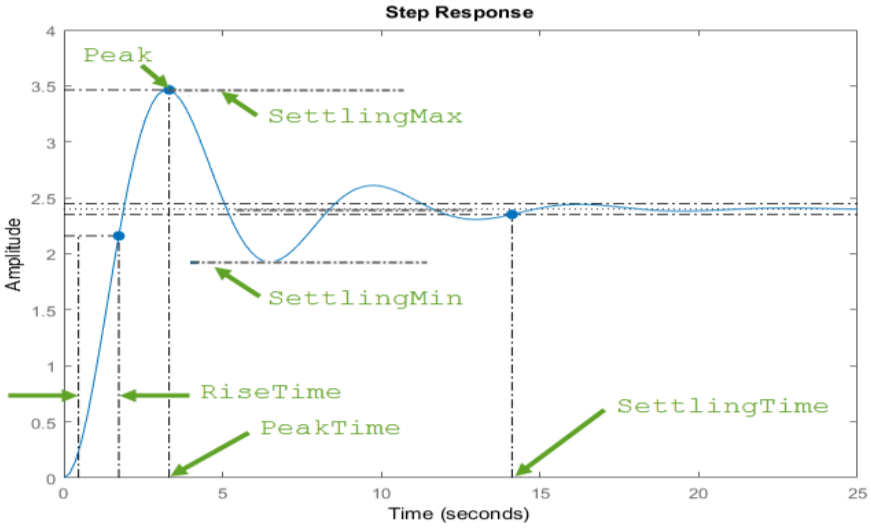


Figure 88: Step response with its parameters [82]

- Rise Time: The required response time from 10 % to 90% of the steady state.
- Settling Time: The required time to stay with the given error limits.
- Overshoot: The maximum Peak of the response.
- Steady state error: The difference between the input signal and the output signal regarding to the time in steady state.

For studying the effect of the proportional, integral, derivative gains on the response, the Figure (89) represented a normal step response as an example.

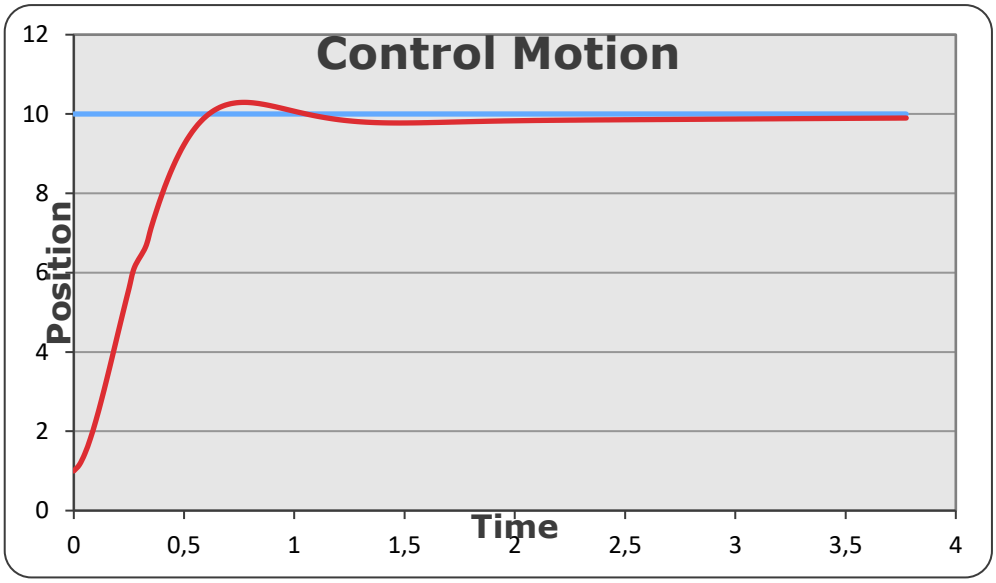


Figure 89: Step response for the control motion

The PID controller is commonly used in industrial fields; the principles of the PID controller are based on the comparison between the present and the input and the output of the whole

PID controller block. The feedback control has two categories, positive and negative feedback in the feedback the output sensed. [83] The signals in the positive feedback tend to boost their values and become larger; the output value can be added to the input when they are at the same phase. In the negative feedback, the system tends to be more stable which decreases the input value [83]. The PID controller consists of [84]: -

- ♦ PC proportional controller
- ♦ IC integral controller
- ♦ DC derivative controller

7.2.1 The Proportional Controller (PC)

The proportional Controller [83] will applied to control the system parameters such as the position (end effector position of the robot arm) which will be proportional to the difference between the desired value and the actual value. The proportional response is achieved by multiplying the error by a constant Kp, the Kp is called the proportional gain which is given by:

$$P = K_p.error(t) \quad (76)$$

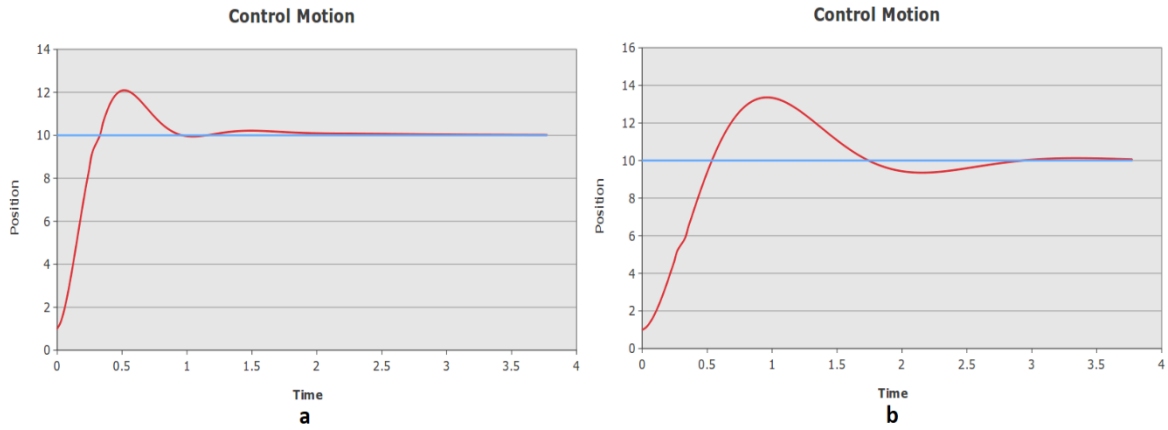


Figure 90: (a) the step response when kp=5 (b) the step response when Kp =2

The system [83] will be unstable when the proportional gain is a high value; the high proportional gain value would affect the output and increase it for a given change in the error. On the other hand, if it is very small, the control action will not respond correctly to the system disturbance. The proportional gain would affect the control parameters and will decrease the values of rise time, settling time, steady-state error and the expectation of the overshoot will increase [83].

7.2.2 The integral controller (IC)

This control represents [83] the sum of the rapid error over time, Ki is called the integral gain which has an effect on the control parameters as follows. The Ki will decrease the values of rise time, steady-state error and the expectation of the overshoot and settling time will increase. The integral controller is given by the Equation:

$$I = K_I \int_0^t error(t)dt \qquad (77)$$

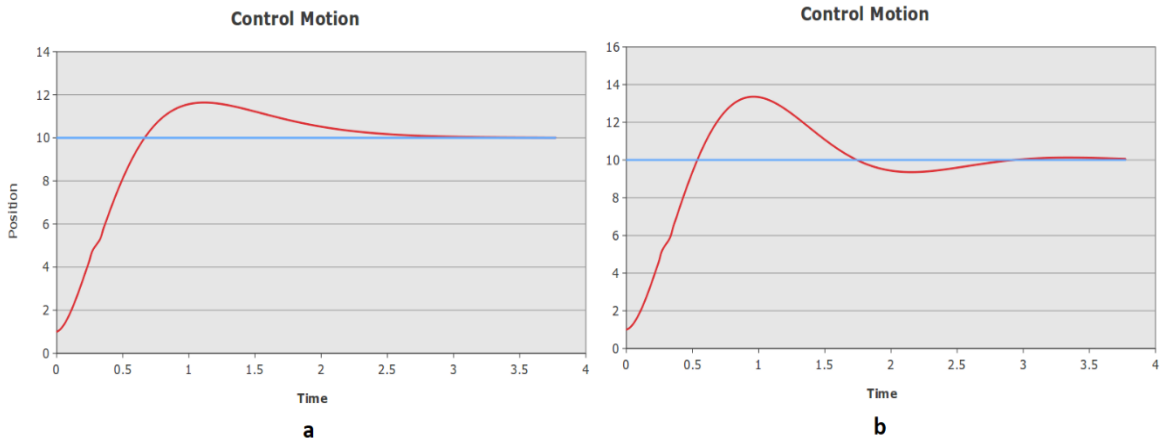


Figure 91: (a) the step response when ki=2 (b) the step response when Ki =4

The output of the Integral Controller (IC) [83] is proportional to both the magnitude of the error and the duration of the error. The integral in a PID controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been previously corrected. Steady state accuracy is usually given if there are an integral part, independent of the value of Ki as long as the system is stable. [83].

7.2.3 Derivative Controller (DC)

The derivative controller [83] could be represented by the slope of the error over time multiplied by the derivative gain, Kd is called the derivative gain. The derivative Equation will be as follows:

$$D = K_D \cdot \frac{derror(t)}{dt} \qquad (78)$$

The derivative controller will reduce the values of rise time, the overshoot and settling time which improves the stability of the system [83].

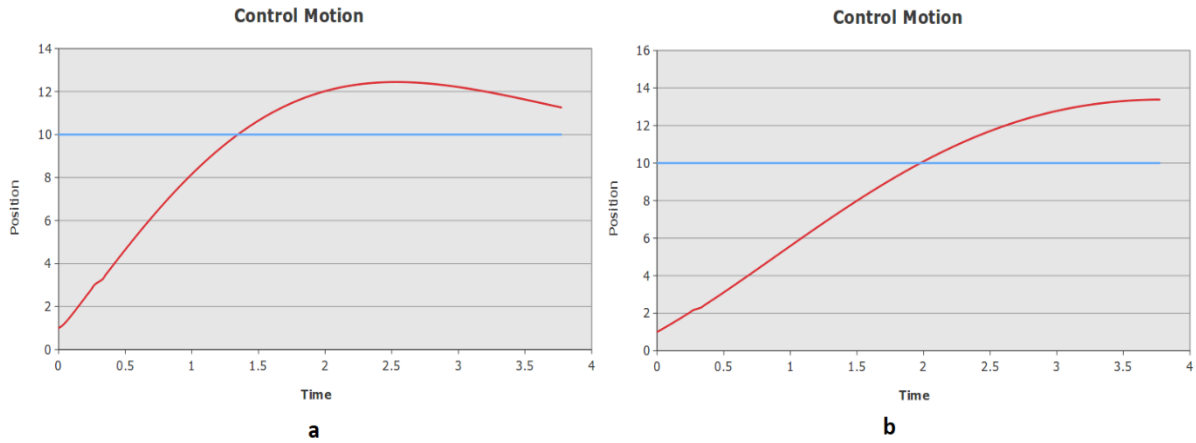


Figure 92: (a) the step response when kd=2 (b) the step response when Kd =4

7.3 Filtering

The noise has a significant influence on differentiation; this influence can be seen in the transfer function G(s)=s of a differentiator which goes to infinity for large s. [85]

$$y(t) = sin\ t + n(t) = sint + a_n\ sin\ \omega_n\ t \qquad (79)$$

The derivative of the signal will be (80) assuming the noise is sinusoidal noise with frequency ω. [85].

$$\frac{dy(t)}{dt} = cos\ t + \dot{n}(t) = cos\ t + a_n\ \omega\ cos\ \omega_n\ t \qquad (80)$$

The signal to noise ratio for the original signal is $1/a_n$ but the signal to noise ratio of the differentiated signal is ω/a_n [85]. Therefore, in a functional controller with derivative action, it is necessary to limit the derivative term high-frequency gain. This can be achieved using the derivative definition as [85].

$$D = -\frac{sKT_d}{1 + sT_d/N}\ Y \qquad (81)$$

instead of $D = sT_dY$. The approximation given by (81) can be interpreted as the ideal derivative sT_d filtered by a first-order system with the time constant T_d/N. The approximation acts as a derivative for low-frequency signal components. The gain, however, is limited to KN. This means that high-frequency measurement noise is amplified at most by a factor KN. Typical values of N are 8 to 20 [85].

Here we can see some drawback of the high frequencies gain, the transfer function from measurement y to controller output u of a PID controller with the derivative is [85]

$$C(s) = -K(1 + \frac{1}{sT_i} + \frac{sT_d}{1 + sT_d/N}) \qquad (82)$$

This controller has constant gain [85]

$$\lim_{s\to\infty} C(s) = -K(1+N) \quad (83)$$

at high frequencies. This can be achieved by additional low pass filtering of the control signal by [85].

$$F(s) = \frac{1}{(1+sT_f)^n} \quad (84)$$

Here T_f is the filter time constant and n is the order of the filter. The choice of T_f is a compromise between filtering capacity and performance. The value of T_f can be coupled to the controller time constants in the same way as for the derivative filter above. If the derivative time used, $T_f = T_d/N$ is a suitable choice. If the controller is only PI, $T_f = T_i/N$ may be suitable [85].

The controller can also be implemented as

$$C(s) = -K\left(1 + \frac{1}{sT_i} + sT_d\right)\frac{1}{(1 + sT_d/N)^2} \quad (85)$$

This model has the advantage of designing model methods for an ideal PID controller and using an iterative design process [85].

7.4 Set Point Weighting

Any step changes in the reference signal will result in an impulse of the control signal; this will be unwanted issue so that the derivative action is usually not applied to the reference signal, one of the solutions for this problem is filtering the reference signal before feeding it to the controller, a second suggestion solution would by applying the proportional part only to the reference signal which is called set point weighting [85].

$$u(t) = K(br(t) - y(t)) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\left(c\frac{dr(t)}{dt} - \frac{dy(t)}{dt}\right)) \quad (86)$$

b and c represented the additional parameters. The integral controller part would be based on the feedback error to ensure the required steady state. The controller is given by (86) represented by a structure with two degrees of freedom due to that the signal path from y to u is different from that from r to u. so that [85] the transfer function from r to u is:

$$\frac{U(s)}{R(s)} = C_r(s) = K\left(b + \frac{1}{sT_i} + csT_d\right) \quad (87)$$

and the transfer function from y to u is [85]

$$\frac{U(s)}{Y(s)} = C_y(s) = K\left(1 + \frac{1}{sT_i} + sT_d\right) \quad (88)$$

Set point weighting is thus a special case of controllers having two degrees of freedom. The system obtained with the controller (86) respond to load disturbances and measurement noise in the same way as the controller (75) [85].

7.5 Different Parameterizations

The PID algorithm given by Equation (75) can be represented by the transfer function [85].

$$G(s) = K\left(1 + \frac{1}{sT_i} + sT_d\right) \quad (89)$$

A slightly different version is most common in commercial controllers. This controller is described by [85]

$$G'(s) = K'\left(1 + \frac{1}{sT_i'}\right)(1 + sT_d') = K'(1 + \frac{T_d'}{T_i'} + \frac{1}{sT_i'} + sT_d') \quad (90)$$

The controller given by Equation (89) called non-interacting, and the one given by Equation (90) interacting. The interacting controller Equation (90) represented as a non-interacting controller whose coefficients are given by [85].

$$\begin{aligned} K &= K'\frac{T_i' + T_d'}{T_i'} \\ T_i &= T_i' + T_d' \\ T_d &= \frac{T_i' T_d'}{T_i' + T_d'} \end{aligned} \quad (91)$$

The non interacting controller corresponds the interacting controller of the Equation (90) under the following condition [85]

$$T_i \geq \mathbf{4}\, T_d \quad (92)$$

The parameters [85] are given by

$$K' = \frac{K}{2}\left(1 + \sqrt{1 - \frac{4T_d}{T_i}}\right)$$
$$T_i' = \frac{T_i}{2}\left(1 + \sqrt{1 - 4T_d/T_i}\right) \quad (93)$$
$$T_d' = \frac{T_i}{2}\left(1 - \sqrt{1 - 4T_d/T_i}\right)$$

The non-interacting [85] controller given by Equation (89) is more general, and we will use that in the future. However, it is [85] sometimes claimed that manually tuning the interacting controller is easier [85]. Remember that different controllers may have different structures when operating with PID controllers, if another controller type replaces a controller, the controller parameters may need to be modified, interacting and non-interacting modes varied only when controller parts I and D used, if we only use the controller as P, PI or PD controller, all types are equivalent. Yet another representation of the PID algorithm is given by when both the I and the D parts of the controller are used [85].

$$G''(s) = k + \frac{k_i}{s} + sk_d \quad (94)$$

The parameters are related to the parameters of standard form through [85]

$$k = K \qquad k_i = \frac{K}{T_i} \qquad k_d = KT_d \quad (95)$$

The representation Equation (94) equals the standard form, but the parameter values are quite different. This [85] can cause significant difficulties for anyone unaware of the differences, particularly if parameter $1/K_i$ called integral time and K_d derivative time, it's even more confusing to call K_i integration time, Equation (94) form is often useful in analytical calculations as the parameters appear linear. The representation also has the advantage of obtaining pure additive, integral, or derivative operation by finite parameter values [85].

7.6 The PID tuning (Ziegler-Nichols)

There are many methods for tuning the Kp, Ki and Kd for the PID controller. One of the famous methods is by Ziegler-Nichols. This method is generally used when the function system is hard to find or when it is a very complex Equation. It is also called the on-line tuning method. In terms of our project, we used the PID controller for controlling the end-effector position, these positions had been received from the semantic camera and

represented the food coordinates. For adjusting the PID gain values, there are many methods on for tuning the PID controller such as Ziegler-Nichols, in the Figure (93) we can see the output signal response.

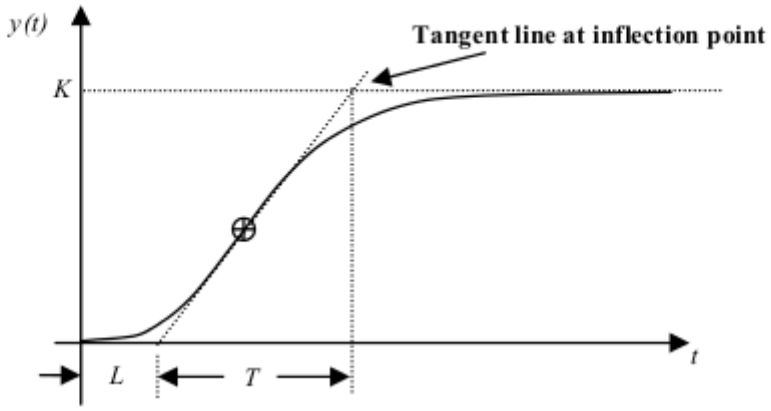


Figure 93: An example for time response signal [86]

In this method there are a number of parameters which are used for getting the PID gains as shown in the Figure (93). These parameters such as K, L and T with the Ziegler-Nichols formula as shown in the table (6) which is used for getting the PID gains. The table (6) shows the proportional, integral and derivative gains that have been specified by the Ziegler-Nichols method. In many cases the PI or P would be enough and gives the required results and minimizes the error.

PID Type	K_p	$T_i=K_p/K_i$	$T_d=K_d/K_p$
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

Table 6: Ziegler-Nichols Recipe [86]

For each object in the simulated environment, there are specific coordinates, as shown in Table 4. As an example, for the cup, we have x, y, z coordinates as position and the other as quaternion to describe the orientation of the object. In the PID design, we take each X output from the semantic camera and try to reduce its difference with the X, which is being sent to

the robotic arm. To reduce the difference between the x or y or z, a PID controller with a specific gain (K_d, K_i, K_p) should be designed.

Let us take as example the z coordinates for the cup after approximation 0.9667, according to our specifications the time to reach the position is 3 sec. The drawing of the Ziegler-Nichols sketch is shown in the Figure (94)

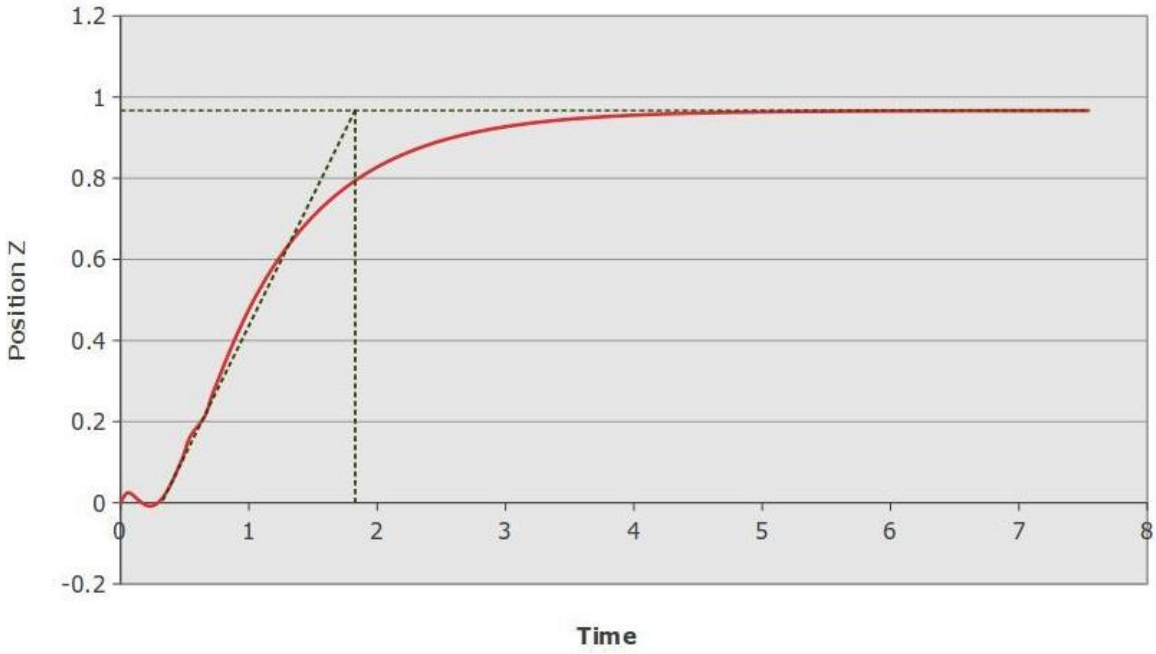


Figure 94: Time response of the position Z for the cup in the simulator.

According to the values from the figure below and when substituted with the parameters table, the gain values will be tuned for designing the required PID controller; this will be classified into three categories as follows:

- only proportional (P): $K_p = 1.8/0.3 = 6$
- Proportional and integral (PI): $K_p = 0.9 * 1.8/0.3 = 5,4$, $K_i = 0.3/0.3 = 1$
- Proportional and integral and derivative (PID): $K_p = 1.2*1.8/0,3 = 7,2$
  $K_i = 2*0.3 = 0,6$, $K_d = 0.5 * 0.3 = 0,15$

7.7 PID In ROS

This algorithm has been modified from the [87]. The PID controller in ROS represented by package of a Proportional-Integral-Derivative controller, it would be useful in case of having a straightforward control problem which need to be solved using PID loop [87]. There are many features for this package such as: Low-pass filter in the error derivative with a parameterized cut-off frequency provides smoother derivative term and a Ziegler-Nichols

auto tuner [87]. There are several parameters can be influenced in the PID designing method using ROS as follows:

- Set point: represented the desired value for the controlling process in the system. It will be the desired position of end effector for the JACO robotic arm [87].
- Plant status: represents the actual value that has been gotten from the controlling process. In our project it would be the actual value of the end effector position [87].
- Control effort: how much force does the controller take to make the plant status equal to the set point. In this case it is the bone rotational angle [87].

The PID controller Node schematic shown in Figure (95)

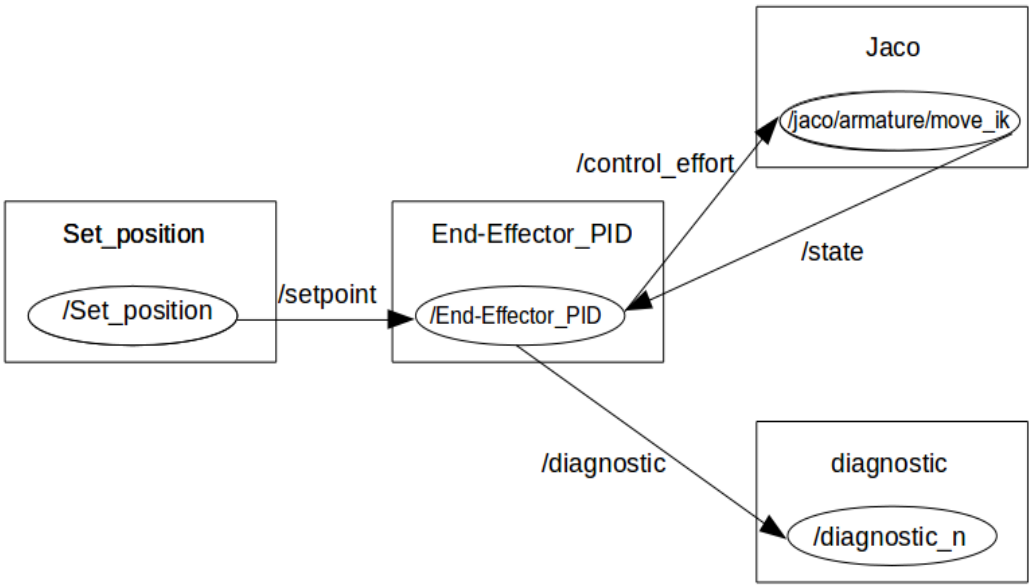


Figure 95: The general schematic ROS nodes with a PID controller.

The controller node represents the main controller node in the PID ROS package, this controller will be responsible for making the set point equal to the plant values equal to the set point then reducing the error and the difference between the actual and desired values. To illustrate Figure (96) /setpoint, /state, and /control effort represented the topic names used by the PID controller through the subscribing and publishing processes. The set position_node publishes the real-time value to the PID controller on the end effector_PID node; the corrections values would be applied to the IK_JACO node via / control_effort topic. The IK_JACO node publishes the actual end-effector position of the simulated JACO arm to the /state topic which the PID controller subscribes to its control effort. One of the

important challenges when designing the software PID is how to tune the gains, as shown in Figure (96).

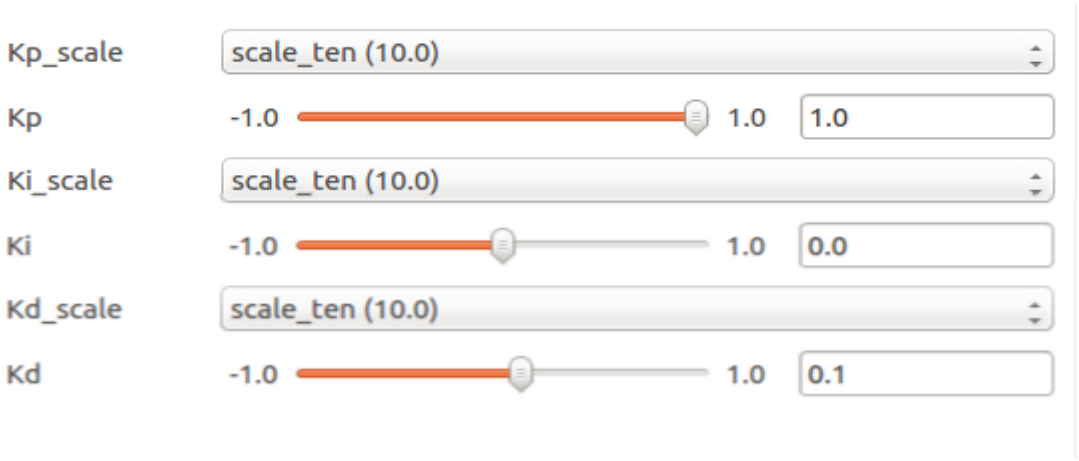


Figure 96: The configuration of the PID gains.[87]

One of the important reasons for using the automatic PID gain configuration is to achieve a smooth, repeatable movement which gives an experiment indication about the right gain values. This could be another method for setting those values and then importing those values in the launch file after specifying them. Every gain has some characteristics which will affect the output response, as explained in Section 7.2. The important challenge will be to determine how to find a compatibility between the different gains values with the working range as quickly as possible with a high efficiency to action the required task and transfer the system in the stability zone.

7.8 The results

After implementing the PID controller, there are some facts should be clarified:

- The tuning of the K_p, K_i, K_d has been done using automatic Ziegler-Nichols in ROS.
- Each x, y, z is applied to the PID controller system to reduce the error difference between the actual and the desired coordinates values.
- The system has been working with efficiency 99%

Here in the Figures we can see the time response between the coordinates from the camera and the coordinates which have been sent to the arm. Those results can be classified into four groups:

1 The Mouth coordinates: The MX, MY, MZ represented the coordinates of the mouth as received from the semantic camera as shown in the Figure (97) which represented the

time response without using PID controller the vertical axis represented the position in (mm) and the horizontal axis represented the time in (sec).

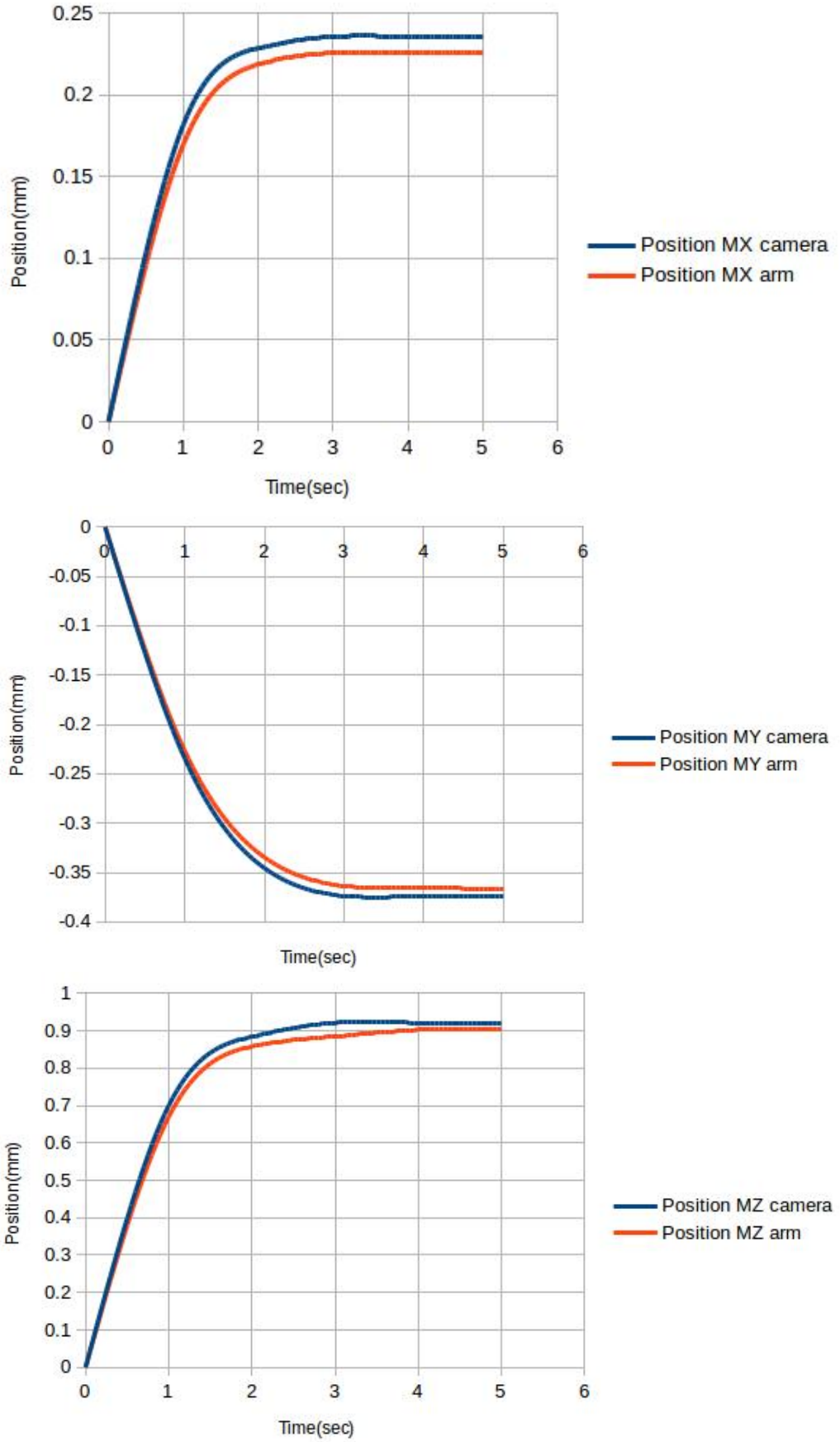


Figure 97: The time response of the Mouth coordinates without using a PID controller.

Next, Figure (98) represented the time response while using the PID controller. And how the output coordinates either in X or Y or Z changed to be optimum with the desired values.

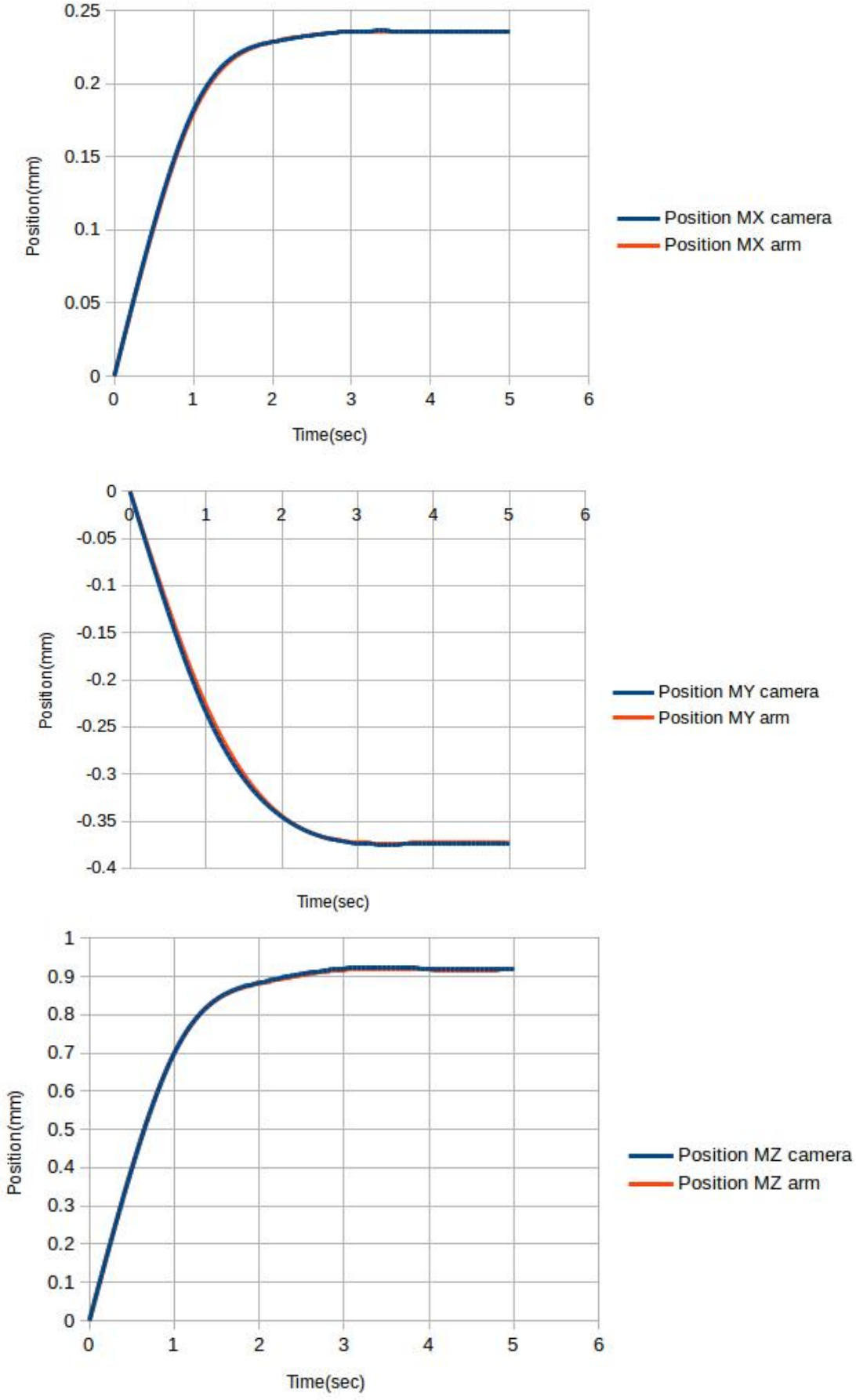


Figure 98: The Time response of the mouth coordinates using a PID controller.

2 The Cup coordinates: The CX, CY, CZ represented the coordinates of the water cup as received from the semantic camera as shown in the Figure (99) which represented the time response without using PID controller.

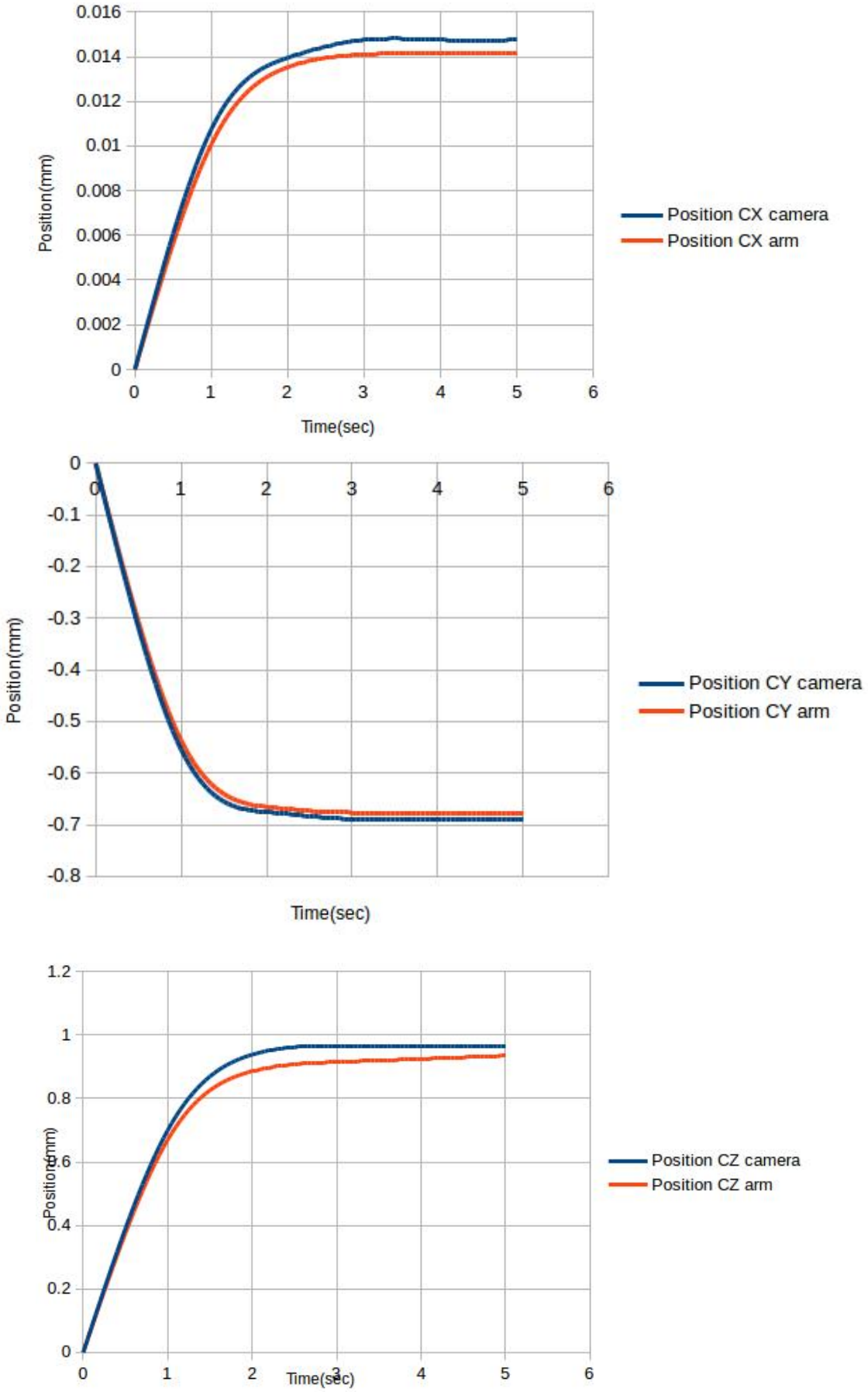


Figure 99: The time response of the cup coordinates without using a PID controller.

The next level would be using the PID controller to reduce the error difference between the desired value and the actual value of the end effector position as shown in the Figure (100).

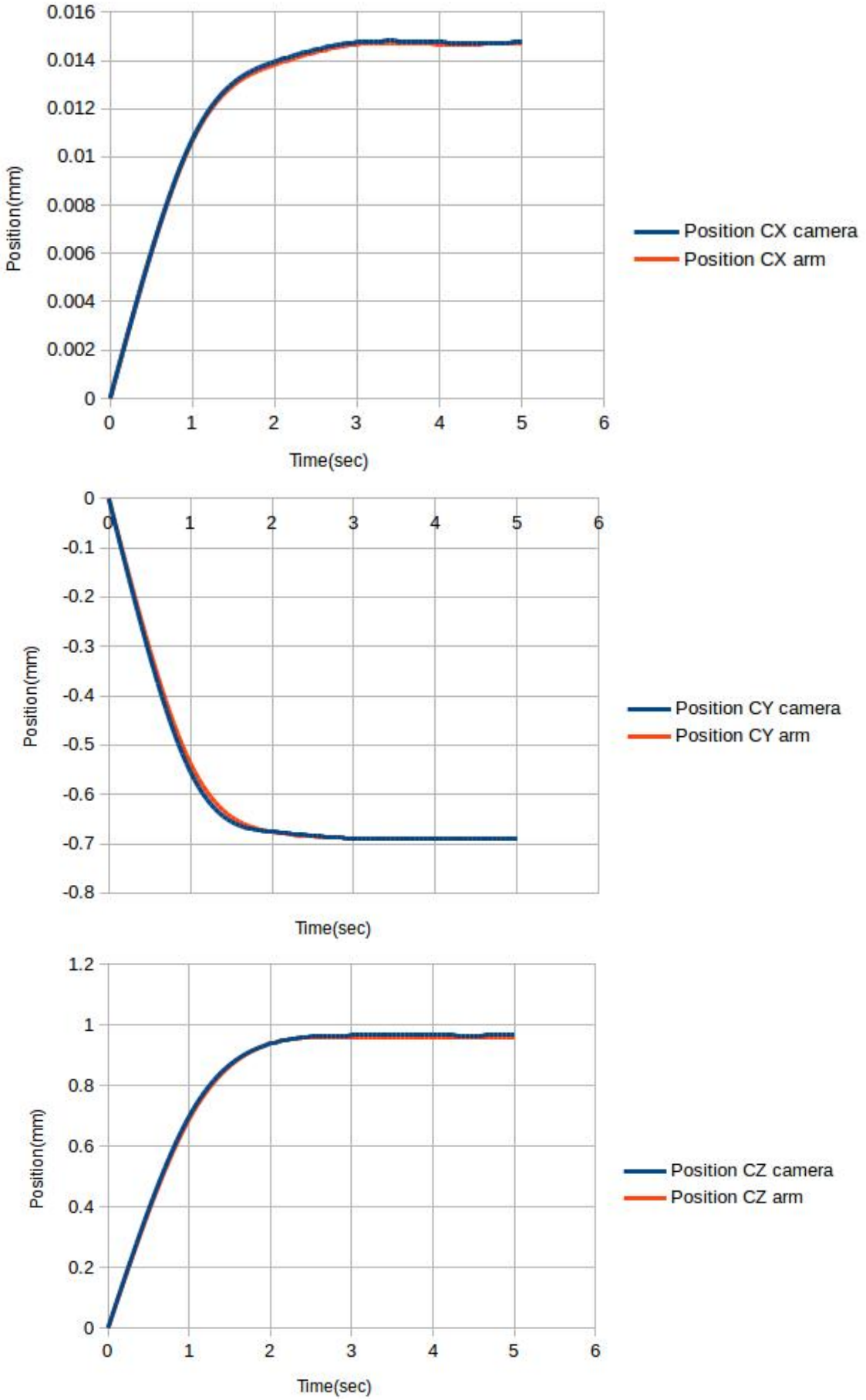


Figure 100: The Time response fort the CUP coordinates without PID

3 The red dish coordinates: The RX, RY, RZ represented the coordinates of the red dish as received from the semantic camera as shown in the Figure (101) which represented the Time response without using PID controller.

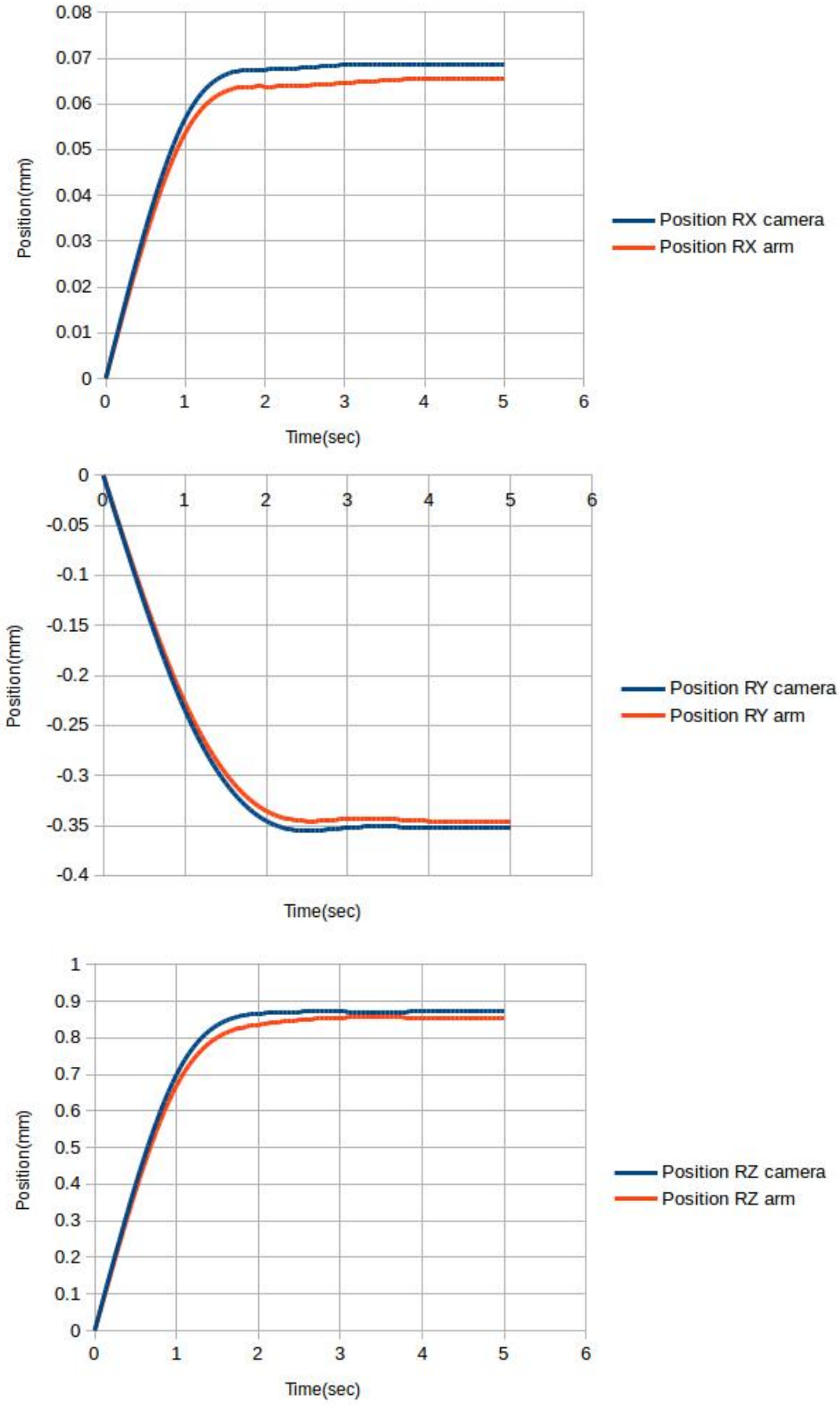


Figure 101: The time response for the red dish coordinates without using a PID controller.

After this result which represented the time response for the red dish coordinates without using the PID controller, the next level would be using the PID controller to reduce the error difference between the desired value and the actual value as shown in the Figure (102)

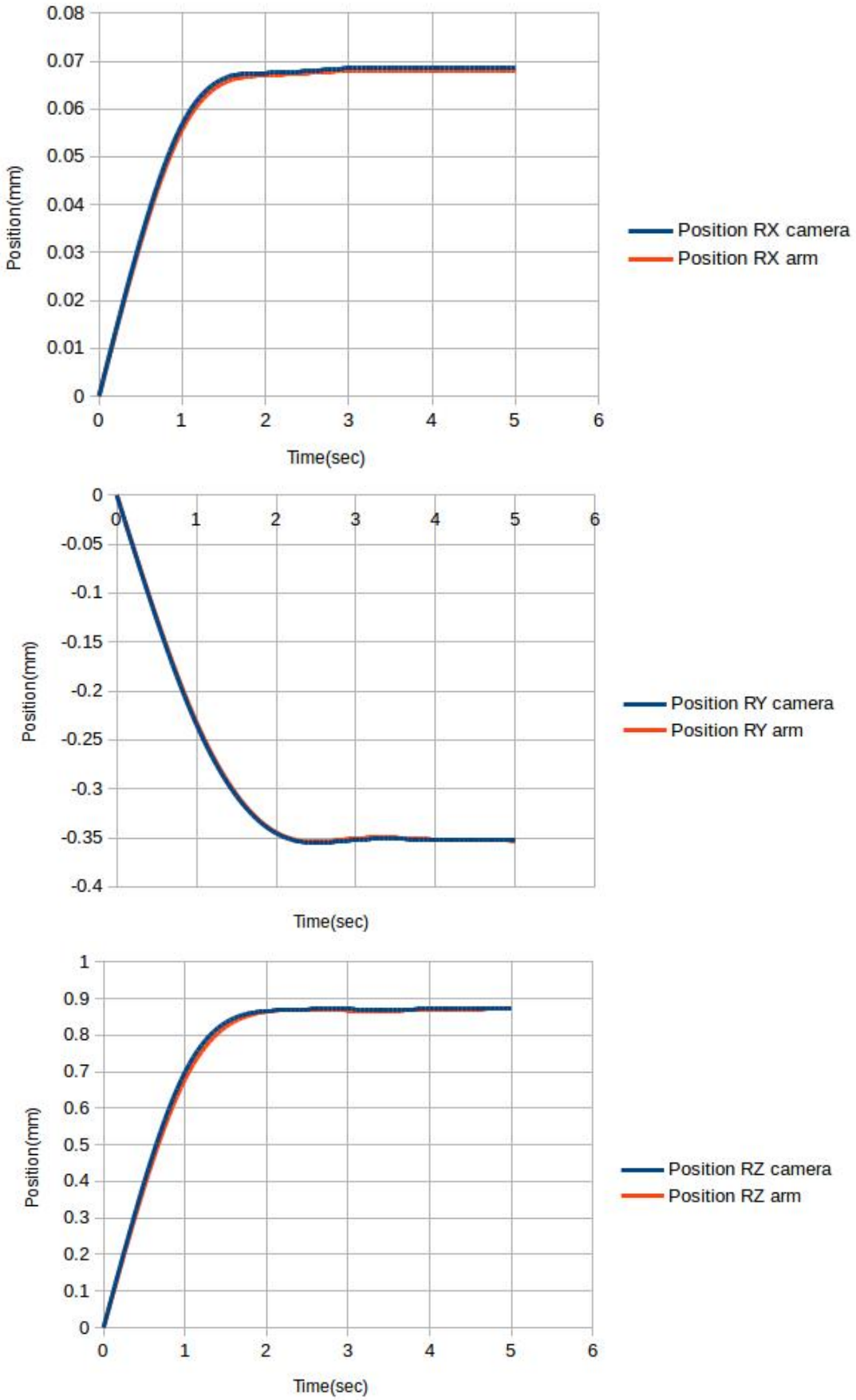


Figure 102: The time response for the desired and actual green dish coordinates.

4 The green dish coordinates: The GX, GY, GZ represented the coordinates of the green dish as received from the semantic camera as shown in the Figure (103) which represented the time response without using PID controller

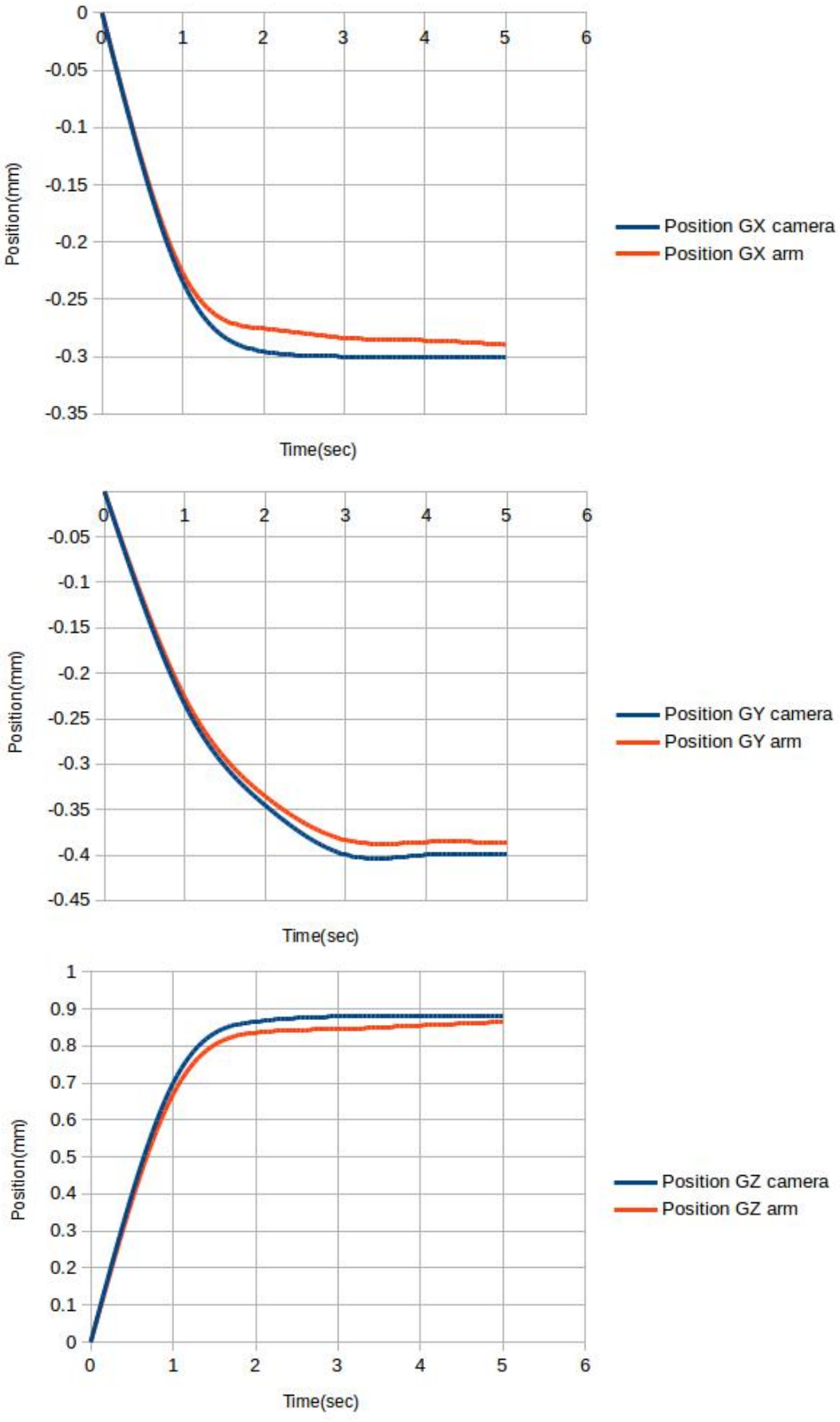


Figure 103: The time response of the green dish coordinates without using a PID controller.

With using of the PID controller to make the actual and the desired values equal to get more stability that can be shown in the Figure (104)

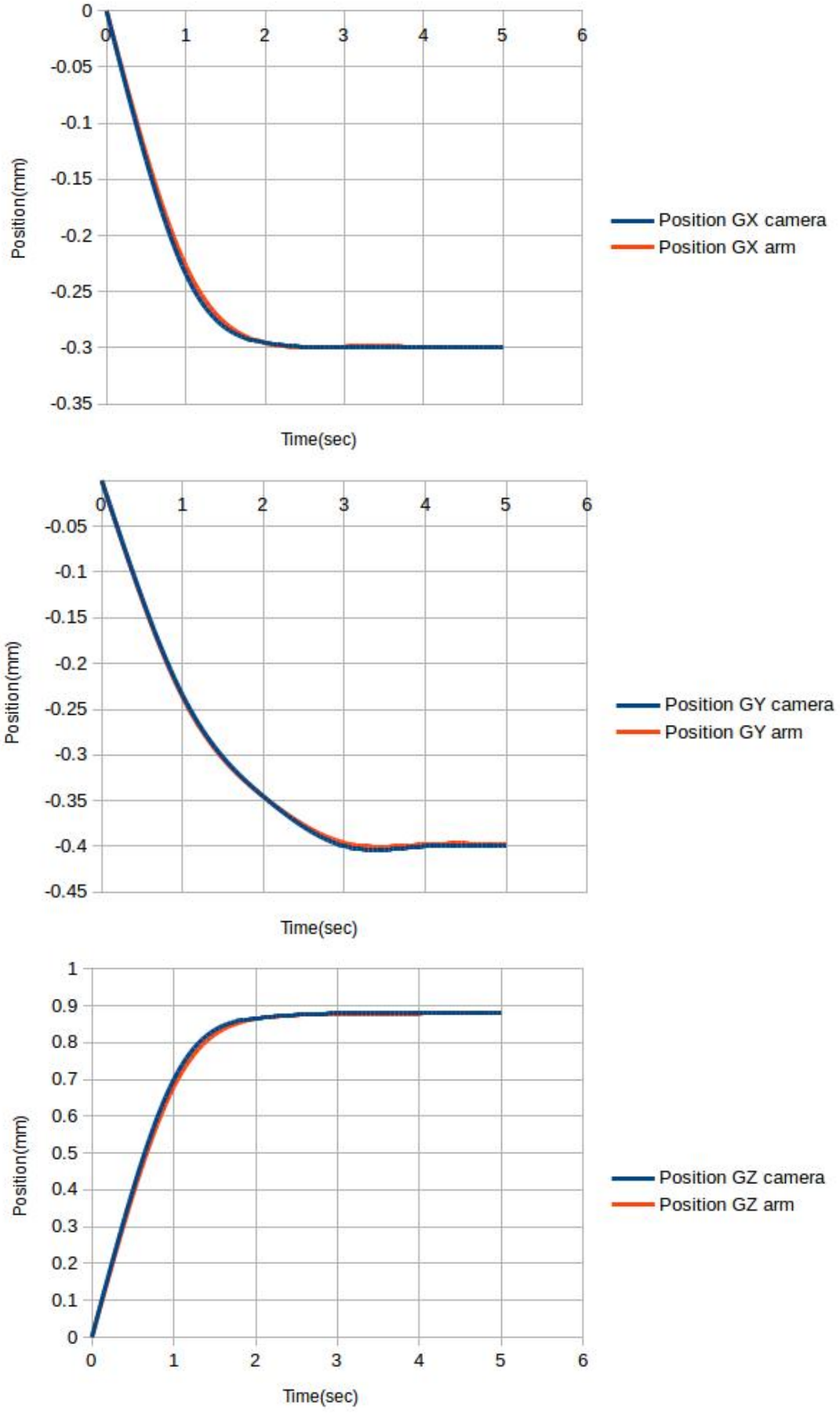


Figure 104: The time response for the green dish coordinates with the PID controller.

8 Experimental results

The main purpose of this research is how to build a fully-simulated controlled intelligent robotic system, which deals with handicapped people who cannot eat or drink by themselves. In this chapter, the results will be displayed in figures.

8.1 The simulation results

The results were divided into the following four categories:

8.1.1 Case 1: Waffle

If the word red comes from the microphone:

- The semantic camera discovers the red dish coordinates and stores them in ROS.
- Send these coordinates to the arm using the ROS topic IK and the required messages.
- Send the ROS topic that is responsible for gripping the required food (waffle).
- After the gripping process, the arm moves to the mouth.
- The arm moves back with the rest of the food to the red dish position.

As shown in Figure (105), for the ROS node output when the (red) word appears, the response for the microphone words and sound recognition would be very fast in msec. The specified time for the eating or drinking, gripping movement has been estimated and could be changed, whereby this cycle will be for the red dish or waffle cycle. The time has taken at different periods, and it has specified as 3 sec after the beginning of the process, either eating or drinking.

```
 go to the red dish
2.99981
 go to the red dish
3.00983
 grip the food
0.0098969
 grip the food
0.0199043
 grip the food
0.0299422
 grip the food
0.0399032
 grip the food
0.0499237
 grip the food
0.0599284
 grip the food
0.0699212
 grip the food
0.0799294
```

Figure 105: The red cycle begins (go then grip).

The arm moves to the red dish, which will take time 3 sec, as specified before. Subsequently, the gripping message is sent to the gripper to pick up the waffle, after which in Figure (106) the cycle engages when the person starts to eat the waffle.

```
 grip the food
2.99993
 grip the food
3.00993
 eat the food
0.00997616
 eat the food
0.0199572
 eat the food
0.0299777
 eat the food
0.0399677
 eat the food
0.049966
 eat the food
0.0599709
 eat the food
0.0699838
 eat the food
0.080002
```

Figure 106: The red cycle continues (grip then eat).

At the end of this cycle, the rest of the food will return to the original position as shown in Figure (107).

```
 eat the food
2.98992
 eat the food
2.99995
 eat the food
3.00995
 back to the red dish
0.00996414
 back to the red dish
0.0199804
 back to the red dish
0.0299632
 back to the red dish
0.0399452
 back to the red dish
0.0499903
```

Figure 107: The red cycle last step (eat then return to the origin coordinates of red food).

The first case study is when the handicapped person wants a waffle, i.e. he wants to eat from the red dish. When the word red comes from the microphone, this recognized through the sound recognition system, and thus the red cycle should start, as shown in Figure (108).


Universität Siegen
EZLS


Universität Siegen
EZLS


EZLS

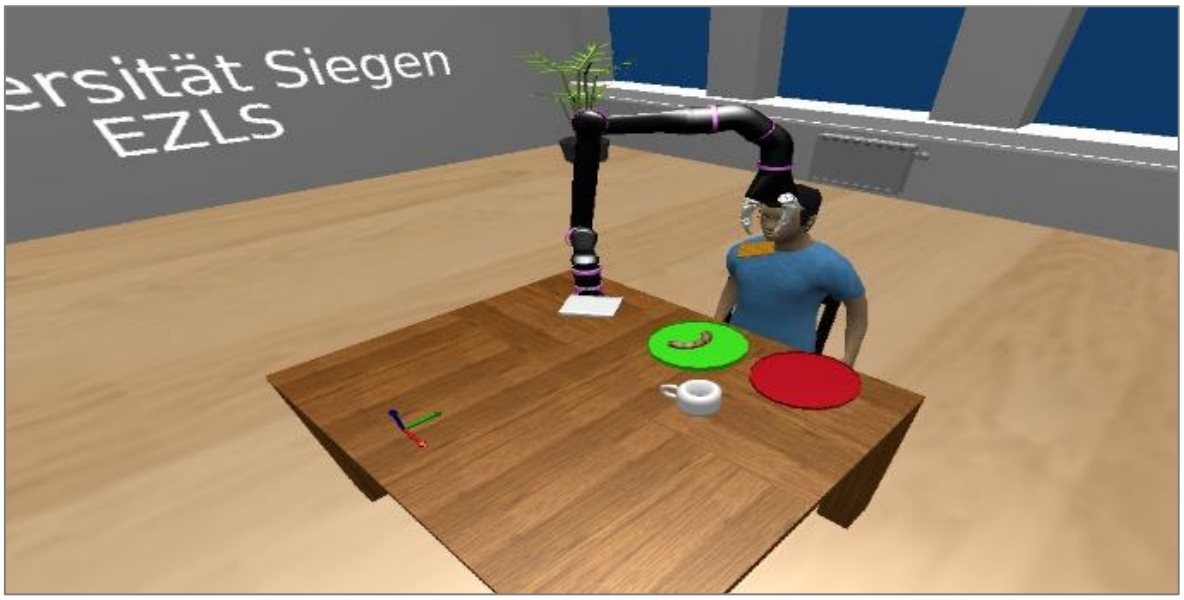


Figure 108: The arm cycle from moving to the food in the red dish to the end.

8.1.2 Case 2: Sausage

If the word green comes from the microphone:

- The semantic camera discovers the green dish coordinates and stores them in ROS.
- These coordinates are sent to the arm using the ROS topic IK and the required message.
- Send the ROS topic that is responsible for gripping the required food (sausage).
- After the gripping process, the arm moves to the mouth.
- The arm moves back with the rest of the food to the green dish position.

As shown in Figure (109), the cycle arm moves to the green dish then grips the sausage. For the ROS node output when the (green) word appears, the response for the microphone words and sound recognition would be very fast in msec. The specified time for the eating or drinking, gripping movement has been estimated and could be changed, whereby this cycle will be for the green dish or sausage cycle. The time has taken at different periods, and it has specified as 3 sec after the beginning of the process, either eating or drinking

```
 go to the green dish
2.98986
 go to the green dish
2.99991
 go to the green dish
3.0099
 grip the food
0.00989776
 grip the food
0.0198719
```

Figure 109: The green cycle begins (go + grip).

The arm moves to the green dish, which will take time 3 sec, as specified before. Subsequently, the gripping message is sent to the gripper to pick up the sausage, after which in Figure (110) the cycle engages when the person starts to eat the sausage.

Figure 110: The green cycle continues (grip then eat).

The last cycle will return the rest of the sausage to its first position, as shown in Figure (111).

Figure 111: The green cycle last step (eat then return to the origin coordinates of green food).

The second case study is when the handicapped person wants to eat sausage, i.e. he wants to eat from the green dish. When the word green comes from the microphone, it recognized through the sound recognition system, and thus the green cycle should start, as shown in Figure (112).

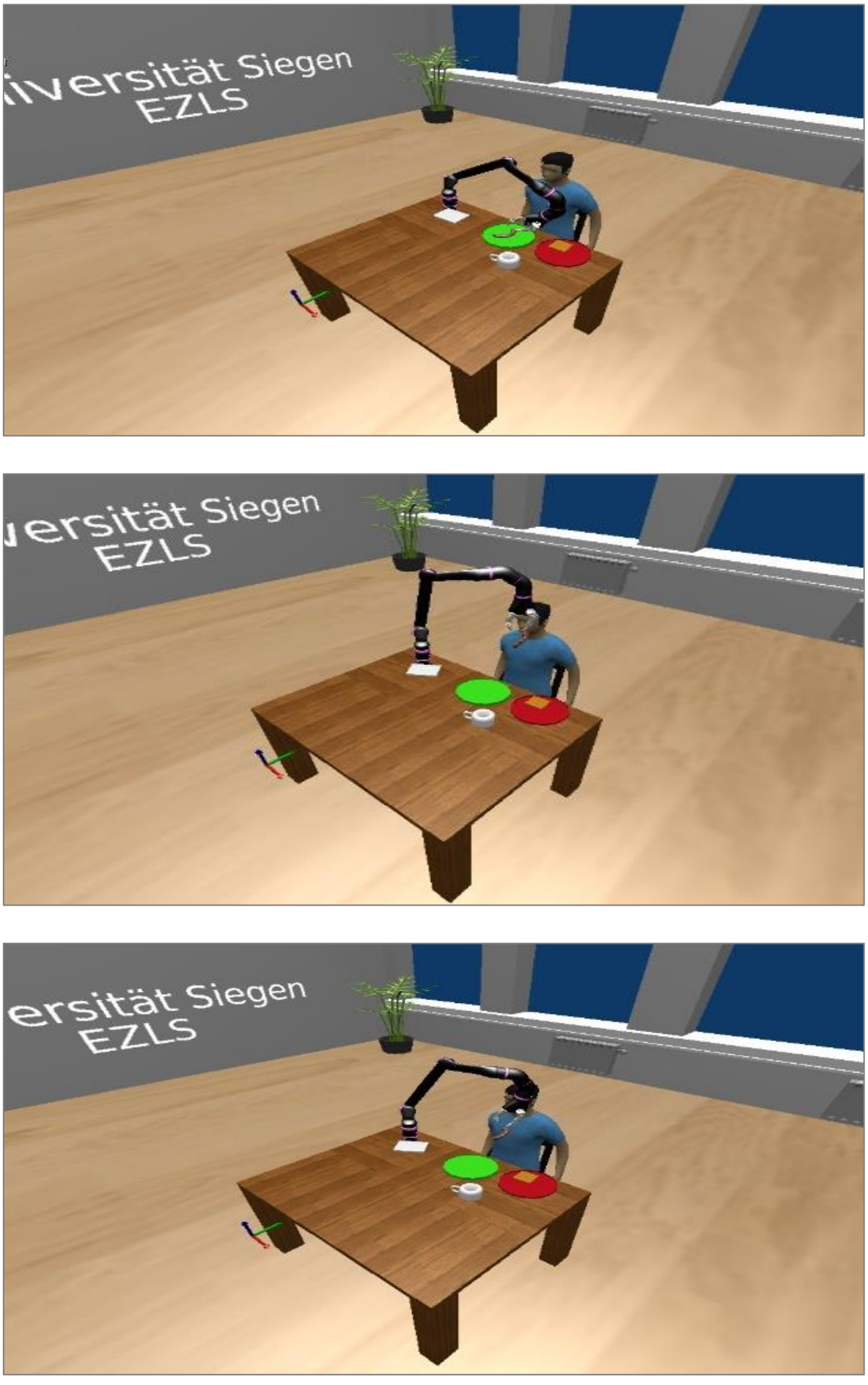


Figure 112: The arm cycle from moving to the food in the sausage to the end.

8.1.3 Case 3: Water

If the word water comes from the microphone:

- The semantic camera discovers the water white cup coordinates and stores them in ROS.
- These coordinates are sent to the arm using the ROS topic IK and the required message.
- Send the ROS topic that is responsible for gripping the cup.
- After the gripping process, the arm moves to the mouth.
- The arm moves back with the cup to its position.

The cycle of the moving arm to the cup, then gripping is shown in Figure (113).

```
 go to the cup
2.97989
 go to the cup
2.98984
 go to the cup
2.99988
 go to the cup
3.00988
 grip the cup
0.00992901
 grip the cup
0.0198654
 grip the cup
0.0299164
 grip the cup
```

Figure 113: The cup cycle begins (go then grip)

Subsequently, it Figure (114) the period when the handicapped person drinks the water is shown.

```
 grip the cup
2.97996
 grip the cup
2.98996
 grip the cup
2.99995
 grip the cup
3.00993
 drink the water
0.0100173
 drink the water
0.0201154
 drink the water
0.0300536
```

Figure 114: The cup cycle continues (grip then eat)

The last sequence is to return the cup to its position, as shown in Figure (115).

```
 drink the water
2.97991
 drink the water
2.9899
 drink the water
2.99989
 drink the water
3.00991
 back to cup position
0.00996356
 back to cup position
0.0199502
 back to cup position
0.029952
```

Figure 115: The red cycle last step (eat then return to the origin coordinates of red food)

The third case study is when the handicapped person wants to drink water, i.e. wants to grip the white cup. When the word water comes from the microphone, this recognized through the sound recognition system, and thus the water cup cycle should start, as shown in Figure (116).

8.1.4 Case 4

If there is no specific word or the word has not been defined to detect, the output message would be:

- ♦ nothing is done

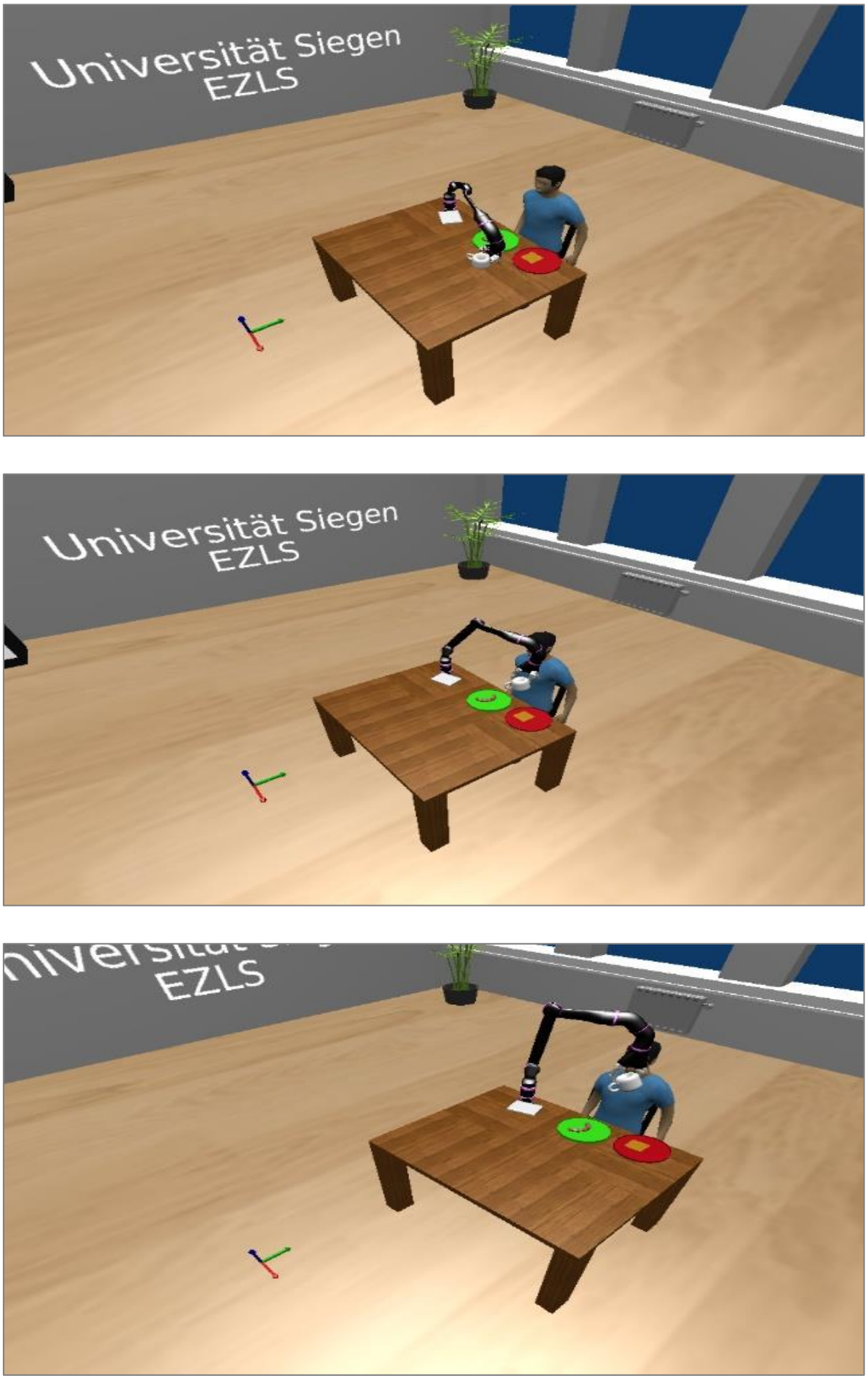


Figure 116: The arm cycle from moving the water cup.

8.2 Safety:

There has always been considerable attention paid to human-robotic interaction, as robots should not harm natural beings. With a technological growth, the entirely separate robotic procedure provides further collaboration. It is also a complicated manufacturing process, owing to issues of contact and operating safety among beings and machines, and job terms in a cooperative exchange. To ensure that JACO robot operates efficiently and safely in dynamic uncertain environments, we have built a safety command. Our intelligent system deals directly with the human body so that any errors in the programming procedures can be affected by the human. In spite of the JACO arm having made with suitable material for humans (which would not harm the body), the high velocity of acceleration could be very dangerous. The best solution for this is to build a new command for stopping the arm if something happens.

The command which has programmed to implement the safety command is "STOP" word and the sound system consists of three main subdivided script:

1 The main script: publish messages on a specific topic will also run as an administrator arranging the function. Also responsible for launching the recognition node

2 The dictionary script (.doc): includes the word as in nature pronounced

STOP ST AA P

3 3. The static language model script (.lm): consists of the type of language models (explained in 4.3.1). As shown below:

-3.1126 STOP -0.2647

This has programmed into the ROS node which is responsible for the sound system, this would be as follow:

- ♦ Establish the static language
- ♦ Prepare the delectable text or words
- ♦ Learn ARPA
- ♦ Crate the files.lm and .dic
- ♦ Import it into the pocketsphinx

In Figure (117) The ROS will start the main node, which connected with the sound and semantic camera system. Someone will then say green, red or water: which represented the word of the required food. When something wrong happens, they should say stop. After this, the arm and the whole system will stop immediately. This command will break the ROS

communication nodes and stop the sequence of the main program. This stop command acts like the emergency button in industrial applications.

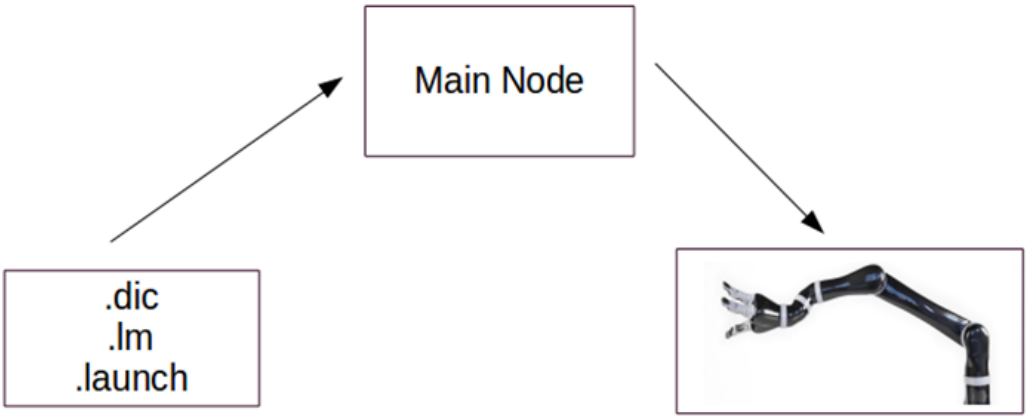


Figure 117: The sequence of the sound commands to the robotic arm.

In our case study, we made some mistake in the coordinates system for the arm; the arm has been moved to the wrong position as you see in the Figure (118), we used offset in the coordinates of each location for example:

The correct coordinates for the sausage (X1, Y1, Z1) + Offset (X2, Y2, Z2) {this has been added by us to make a new wrong coordinate through the moving of the arm}

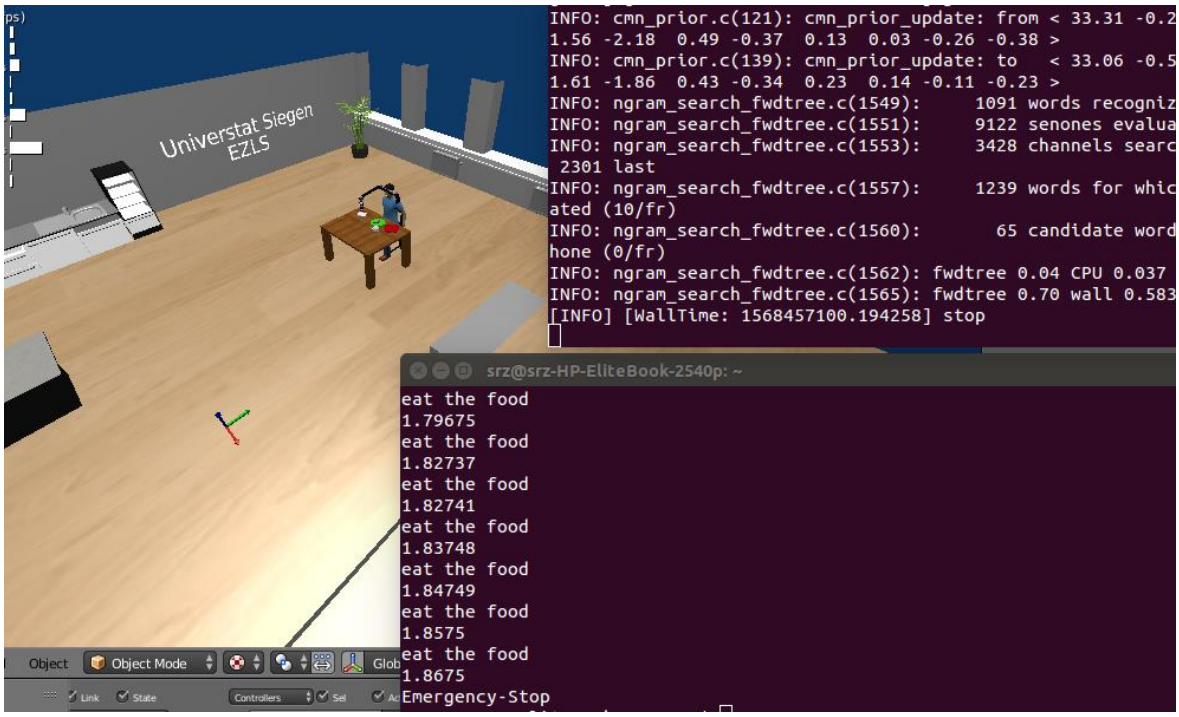


Figure 118: The arm moves to a crash position; the person stops the whole system.

In Figure (119) we can see another example for the emergency stop when the arm moves to grip the food in the green dish.

Figure 119: The arm stops through gripping the sausage.

As a third emergency stop example in Figure (120) the JACO stops when the arm goes to move the food to the mouth.

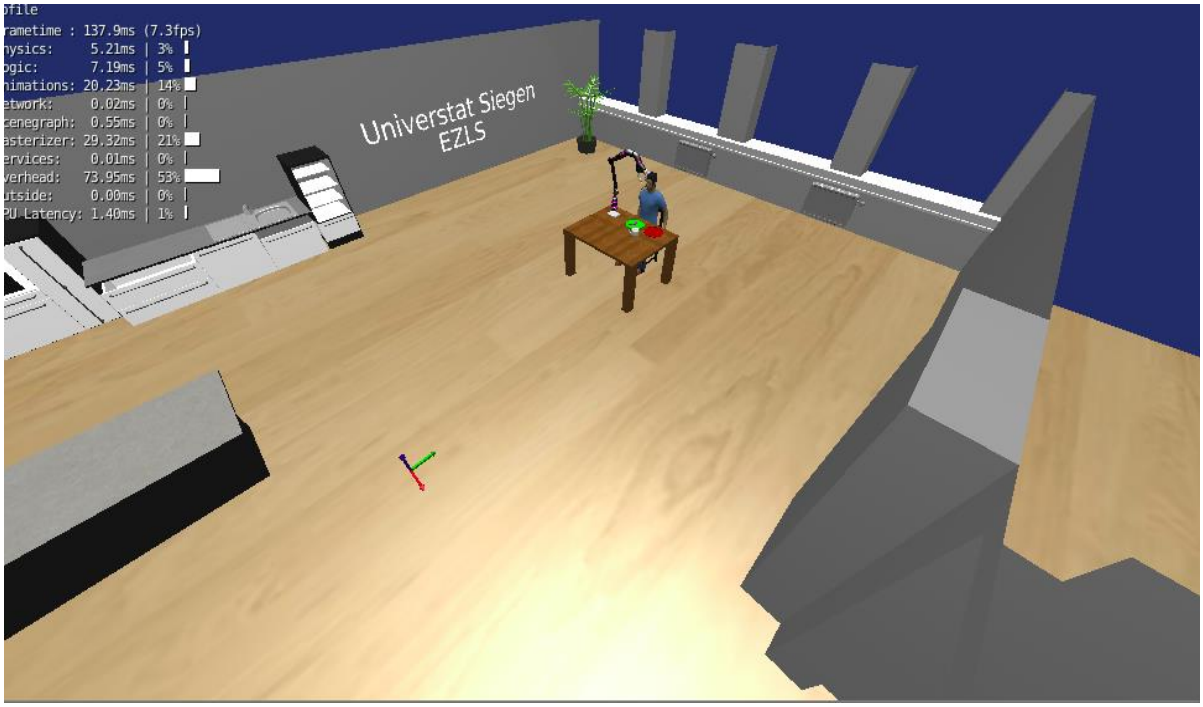


Figure 120: The arm stops through the eating process.

9 Conclusions and future work

In this research, there are many objectives, the main one being how to build an automated robotic intelligent system for handicapped people. This target comprises several sub-objectives. After having achieved our goal, these objectives could present as follows:

1 The environment (human model, dishes, kitchen furniture, table) has been drawn then accurately implemented in the program to follow the reality design.

2 The arm model:

- ♦ The arm has been simulated close to the original design; for example, it has the same joint rotation angles and links length.
- ♦ The arm could be fully controlled using the ROS trajectory action or IK action to give the perfect arm motion.
- ♦ The gripper is also controlled using ROS commands.

3 The sound recognition system: this system has been built then programmed to give us a high level of accuracy and response. It is a beneficial system and could recognize the words precisely.

4 The semantic camera system: this system is a very accurate system, which is noticed clearly after the building and programming of this system. It gives the exact coordinates and details of each object.

5 The overall system worked in an efficient, accurate and successful way and undertook the required tasks intelligibly.

In this thesis, we have presented a new design for the assistive robot. Several developments give many new features in this project compared with others, detailed as follows:

1 Using a new robot model JACO instead of the other robotic arms: this arm has been designed for helping disabled people with eating, drinking and other life activities. The challenge was how to build this arm and model it into the simulation then work with this arm through controlling the process for creating a new automated assistive model. This gives the project more realistic.

2 Using a semantic camera instead of the normal camera: this camera represents a new generation technique for the smart environment. With this camera, the environment objects could be defined relative to their definition in the environment, type and nature such as the innate intelligence. This process based on the internal data of each object into environment which has been implemented such as type, label and getting this orientation and position related to the blender environment.

3 Using a sound system in addition to other systems: this ensures flexible treatment between the disabled and the system, which gives the disabled person more options to say what he wants without any effort, such as pressing a button or moving a joystick.

For implementing this system in real world, we need the following hardware components:

1 A vision system: represented by two cameras for detecting the environment objects and discover their coordinates, we can use for example point cloud camera.

2 A robotic system: for execution the pick and place the food and drink.

3 A sound recognition system: which is already has been implanted in our project by a microphone and a programmed sound package into ROS.

4 Other components for getting more safety such as external emergency stop, laser range sensor to specify the arm work space.

5 Metal frame to fix the human face and gives more flexibility through the mouth detection process.

As software requirements will be ROS (robot operating system) and OpenCV.

9.1 The research Objectives answer

1 A new intelligent robotic library has created in MORSE, which can be used for the handicapped people; this system has been worked effectively.

2 The JACO robot arm has been implemented with the original arm specifications, the joints rotation angles and the length of the links also has been planning. The inverse kinematic mechanism motion has been calculated and provided.

3 The software for the project was very complicated. The combination of the three systems was a hard challenge, specially the Morse should be programmed in Python but ROS nodes

4 The software can implement the intelligent sensors which have worked in the real-time with an adequate response.

5 The graphical user Interface gives us the real environment with an option for providing the physical properties (I.e. the user has the opportunity to implement the crash, the gravity...etc.). The software also can display the motion of the arm motion and the object precisely.

6 The semantic camera system worked and discovered the coordinates based on the world (environment) coordination system, also can recognize the object-type.

7 The response of the sound recognition system was perfect especially after our improvement in the threshold which reduced the response time

9.2 The future work

For future work there are a number of the suggestions as follows:

- Using two JACO robotic arms instead of one arm to give more options in functions such as one of the two arms for eating and the other for drinking.
- Using a point cloud camera instead of the semantic camera to give more options, whereby many objects can be defined specifically in different environments. These objects could be stored in the database then using the neural network to detect the different objects and their coordinates. Moreover, it could help with the mouth detection relative to the human face.
- Using a KUKA robotic arm instead of the JACO arm, while the environment should also be changed to be the industrial environment to undertake industry tasks.
- There is a new project version for handicapped people using brain signals to move the arm or grip the required objects, which could also be a useful suggestion for future work.

References

[1] Serena Ivaldi, Vincent Padois, Francesco Nori, Tools for dynamics simulation of robots: a survey based on user feedback, 14 IEEE-RAS international conference on humanoid robots (humanoids), Spain, 2014.

[2] Nathan Koenig, Andrew Howard, Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator, IEEE/RJS international conference on intelligent robots and systems, Japan, 2004.

[3] http://all-about-linguistics.group.shef.ac.uk/branches-of-linguistics/semantics/what-does-semantics-study/, all-about-linguistics [has been accessed on 12 April 2017].

[4] Chaitanya Mitash, Kostas E. Bekris and Abdeslam Boularias ; Physics-aware Simulation for Object Detection and Pose Estimation , Department of Computer Science, Rutgers University 2017.

[5] Josip Josifovski, Matthias Kerzel, Christoph Pregizer, Lukas Posniak, Stefan Wermter; Object Detection and Pose Estimation based on Convolutional Neural Networks Trained with Synthetic Data , IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Madrid, Spain, October 1-5, 2018.

[6] Matthew R. Walter, Sachithra Hemachandra, Bianca Homberg, Stefanie Tellex, Seth Teller, Learning Semantic Maps from Natural Language Descriptions, Proceedings of the 2013 Robotics: Science and Systems IX Conference, June 24-28, 2013, Berlin, Germany.

[7] Marcin Marszałek, Cordelia Schmid, Semantic Hierarchies for Visual Object Recognition, Computer Vision and Pattern Recognition, CVPR '07. IEEE Conference on Minneapolis, MN, USA, 2007.

[8] Jörg Stückler, Nenad Biresev, Sven Behnke, Semantic Mapping Using Object-Class Segmentation of RGB-D Images, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal, October 2012.

[9] Claire Dune, Christophe Leroux, Eric Marchand, Intuitive human interaction with an arm robot for severely handicapped people A One Click Approach, Proceedings of the IEEE 10th International Conference on Rehabilitation Robotics, Netherlands, June 12-15, 2007.

[10] H F M Van der Loos. Vastanford, rehabilitation robotics research and development program: Lessons learned in the application of robotics technology to the field of rehabilitation. IEEE Trans. on Neural Systems and Rehabilitation Engineering, 3:46–55, March 1995.

[11] H F M Van der Loos, J J Wagner, N Smaby, K Chang, O Madrigal, L J Leifer, and O Khatib. Provar assistive robot system architecture. In IEEE Int. Conf. on Robotics and Automation, Detroit, May 2000.

[12] T. Jones. Raid, Toward greater independence in the office& home environment. In IEEE Int. Conf. on Rehabilitation Robotics, pages 201–206, 1999

[13] M. Busnel, R. Gelin, and Lesigne B. Evaluation of a robotized master/raid workstation at home : protocol and first results. In IEEE Int. Conf. on Rehabilitation Robotics, 2001

[14] M. Topping, H. Heck, G. Bolmsjo, and D. Weightman. The develop- ment of r.a.i.l. In TIDE, pages 23–25, 1998

[15] J. Topping, M.and Smith. The developpement of handy 1 , a reha- bilitation robotic system to assist the severely disabled. In Industrial robot, pages 316–320. 1998.

[16] S. Ishii, S. Tanaka, and F. Hiramatsu. Meal assistance robot for severely handicapped people. In IEEE Int, Conf on Rehabilitation and Automation, pages 1308–1313, San Francisco, USA, 1995.

[17] R. Soyama, S. Ishii, and A. Fukase. The development of meal- assistance robot 'myspoon'. In IEEE Int. Conf. on Rehabilitation Robotics, pages 88–91, 2003.

[18] A Casals, R Villa, and D Casals. A soft assistance arm for tetraplegics. In 1st TIDE Cong., pages 103–107, April 1993.

[19] K. Kawamura and M. Iskarous. Trends in service robots for the disabled and the elderly. Special session on service robots for the disabled and elderly people, 1994.

[20] Won-Kyung Song; Jongbae Kim; Kwang-Ok An; In-Ho Lee; Won-Jin Song; BumSuk Lee; Sung-Il Hwang; Mi-Ok Son; Eun-Chang Lee, Design of Novel Feeding Robot for Korean Food. Aging Friendly Technology for Health and Independence. ICOST Lecture Notes in Computer Science, Vol. 6159. Springer, Berlin, Heidelberg, 2010.

[21] H. Eftring and K. Boschian. Technical results from manus user trials. In IEEE Int. Conf. on Rehabilitation Robotics, pages 136–141, 99.

[22] Duimel Kwee H H, J J, SMits, A A J.J, Tuinhofde Moed, and J A van Woerden. The manus wheelchair-borne manipulator : System review and first results. In IARP, 2nd Workshop Medical and Healthcare Robotics, pages 385–395, 1989.

[23] I Volosyak, Ivlev O., and Graser A. rehabilitation robot friend ii - the general concept and current implementation. In IEEE Int. Conf. on Rehabilitation Robotics, pages 540–544, Chicago, IL, USA, June 2005.

[24] C Leroux, G Chalubert, O Tahri, S Schmutz, N Biard, I Lafont, Dsert J-F, and R Alexandre, J Mand Gelin. Interface intelligente pour la saisie d'objets robotise, handicap 2006. In National Conf. Handicap, paris, june 2006

[25] C Leroux, M Guerrand, C. Leroy, Y Masson, and B. Boukarri. Magritte: a graphic supervisor for remote handling interventions. In ESA Workshop on Advanced Space Technologies for Robotics and Automation, 'ASTRA 2004, Noordwijk, The Netherlands, November 2004.

[26] F. Bley, M. Rous, U. Canzler, and K. Karl-Friedrich. Supervised navigation and manipulation for impaired wheelchair users. IEEE Trans.on Machine Man and Cybernetics, pages 152–165, 2004.

[27] R. Mahoney. The raptor wheelchair robot system. In IEEE Int. Conf. on Rehabilitation Robotics, pages 135–141, Evry, France, 2001

[28] http://www.robotnik.eu/robotics-arms/kinova-jaco-arm [has been accessed on 28 August 2017].

[29] H. Neveryd and G. Bolmsj. Walky, an ultrasonic navigating mobile robot for the disabled. In TIDE, pages 366–370, Paris, France, 1995.

[30] P Dario, E Guglielmelli, C Laschi, and G Teti. Movaid: A mobile robotic system residential care to disabled and elderly people. The First MobiNet Symposium, 1997.

[31] P. Hoppenot and E. Colle. Localization and control of a rehabilitation mobile robot by close human - machine cooperation. IEEE Trans. on Neural Systems and Rehabilitation Engineering, 9:1534–1724, 2001.

[32] R. Bischoff. Design concept and realization of the humanoid service robot hermes. In A. Zelinsky, editor, In Field and Service Robotics, pages 485–492. London, springer edition, 1998

[33] Z. Bien, J.S. Kim, M.J. Chung, Kwon D.S., and Chang P.H. Devel- oppement of a wheelchair-based rehabilitation robotic system (kares ii) with various human robot interaction interfaces for the disabled. In Int. Conf. on Advanced Intelligent Mechatronics, volume 20. IEEE/ASME, 2003

[34] B. Graf, M Hans, and R D Schraft. Care-o-bot ii:development of a next generation robotic home assistant. Autonomous robots, 2004.

[35] Fei Gao, Hiroki Higa, Hideyuki Uehara, and Takashi Soken, A Mobile Robotic Arm for People with Severe Disabilities: Trial Development of a Vision-Based User Interface , Journal of Advanced Control, Automation and Robotics (JACAR), 1 (1): 25-30, 2015 ISSN 2186-9154

[36] F. Gao, H. Higa, H. Uehara, and T. Soken, "A vision-based user interface of a mobile robotic arm for people with severe disabilities," Proc. Int'l conf. Intelligent Informatics and Biomedical Sciences, pp. 172–175, 2015.

[37] Laura V. Herlant; Rachel M. Holladay; Siddhartha S. Srinivasa, Assistive Teleoperation of Robot Arms via Automatic Time-Optimal Mode Switching, Human-Robot Interaction (HRI)11th ACM/IEEE International Conference on Christchurch, New Zealand, 2016.

[38] Pedro Lopes; Ryan Lavoie; Rishi Faldu; Nick Aquino; Jason Barron; Mohamed Kante; Basel Magfory; Waleed Meleis (Advisor), 2012 Available on http://www.ece.neu.edu/personal/meleis/icraft.pdf.

[39] Gunnar Bolmsjö ; Magnus Olsson and Ulf Lorentzon, Development of a general purpose robot arm for use by disabled and elderly at home , International Symposium on Robotics, ISR2002 - Stockholm

[40] Severin Lemaignan ; Marc Hanheide; Michael Karg; Harmish Khambhaita; Lars Kunze; Florian Lier; Ingo Lutkebohle and Gregoire Milliez , Simulation and HRI Recent Perspectives with the MORSE Simulator, 4th International Conference, SIMPAR 2014, Bergamo, Italy, October 20-23, 2014.

[41] Gregoire Milliez, Emmanuel Ferreira, Michelangelo Fiore, Rachid Alami, and Fabrice Lefevre, Simulating human-robot interactions for dialogue strategy learning , 4th International Conference, SIMPAR 2014, Bergamo, Italy, October 20-23, 2014.

[42] Pyung Hun Chang; Hyung-Soon Park, Development of a Robotic Arm for Handicapped People: A Task-Oriented Design Approach, Autonomous Robots 15, 81–92, 2003 c 2003 Kluwer Academic Publishers. Manufactured in The Netherlands.

[43] Adam Vogel; Karthik Raghunathan; Stefan Krawczyk, A Situated, Embodied Spoken Language System for Household Robotics, Department of Computer Science Stanford University California, 2006.

[44] Joseph M. Romano; Jordan P. Brindza; Katherine J. Kuchenbecker, ROS open-source audio recognizer: ROAR environmental sound detection tools for robot programming, 2013. Available on the following website: https://pdfs.semanticscholar.org/0b1e/ec864bb1dec63f0d1500a730f4427354d891.pdf.

[45] Mihael Simoniˇc, Bachelor Thesis: A Voice User Interface for Human-Robot Interaction on a Service Robot UNIVERSITY OF TÜBINGEN, Institute for Informatics the Chair of Cognitive Systems, 2015.

[46] https://www.openrobots.org/morse/doc/latest/what_is_morse.html [has been accessed on 12 June 2017].

[47] The Robot Operating System, "Ros basic concepts." http://ros.org/images/wiki/ROS_basic_concepts.png. Accessed May 19, 2017.

[48] https://www.blender.org/features/ [has been accessed on 24. August 2017].

[49] http://www.roboticmagazine.com/robot-review/jaco-robot-arm-2, admin on June 1, 2011.

[50] Craig, John J., Introduction to Robotics Mechanics and Control, 3rd ed. Upper Saddle River: Pearson Prentice Hall, 2005.

[51] Ronald H. Palacios, robotic arm manipulation laboratory with a six degree of freedom JACO arm, submitted in partial fulfillment of the requirements for the degree of Master of Science in applied physics, Monterey, California: Naval Postgraduate School 2015.

[52] R. D. Klafter; T. A. Chmielewski; M. Negin, Robotic Engineering: An Integrated Approach, 1st ed. Englewood Cliffs, NJ: Prentice Hall, 1989.

[53] JACO DH Parameters of JACO R&D, V1.1.5, Kinova, Boisbriand, Canada, 2013.

[54] Miguel Pereira Mendes, Computed torque-control of the Kinova JACO Arm, Coimbra, Sep. 2017

[55] A. Jacinto, Unmanned systems: A lab-based robotic arm for grasping, M.S. thesis, Dept. Physics, Naval Postgraduate School, Monterey, CA, 2015.

[56] http://www.openrobots.org/morse/doc/1.2/user/actuators.html [has been accessed on 27. November 2017].

[57] https://wiki.blender.org/index.php/Doc:2.4/Manual/Modeling/ Meshes/Weight Paint [has been accessed on 24. August 2017].

[58] Robert Y.Wang; Kari Pulli; Jovan Popovic, Real-time enveloping with rotational regression, ACM Transactions on Graphics (TOG)-Proceedings of ACM SIGGRAPH, Volume 26 Issue 3, July 2007.

[59] Ladislav Kavan, Skinning: Real-time Shape Deformation Part I: Direct Skinning Methods and Deformation Primitives, SIGGRAPH Course 2014.

[60] Alex Mohr; Michael Gleicher, Building Efficient, Accurate Character Skins from Examples, ACM Transactions on Graphics (TOG) Proceedings of ACM SIGGRAPH,Volume 22 Issue 3, July 2003.

[61] J. P. Lewis; Matt Cordner; Nickson Fong, Pose space deformations: A unified approach to shape interpolation and skeleton-driven deformation, proceedings of the 27th annual conference on computer graphics and interactive techniques, pages 165-172, 2000.

[62] https://web.stanford.edu/class/cs248/pdf/class_13_skinning.pdf. [has been accessed on 24. September 2017].

[63] Ladislav Kavan; Steven Collins; Jirı Zara; Carol O'Sullivan1, Geometric skinning with approximate dual quaternion blending, Journal ACM Transactions on Graphics (TOG), Volume 27 Issue 4, 2008.

[64] http://www.openrobots.org/morse/doc/1.2/user/robots/fakerobot.html [has been accessed on 10. September 2017].

[65] http://www.easy-rob.com/uploads/media/LectureRobotics.pdf, FH Darmstadt, summer term 2000 [has been accessed on 15. October 2017].

[66] Andreas Aristidou; Joan Lasenby, Technical Report Inverse Kinematics: a review of existing techniques and introduction of a new fast iterative solver, September 2009, Available on: http://www.researchgate.net/publication/273166356.

[67] http://wiki.ros.org/joint_trajectory_controller [has been accessed on 27. November 2017].

[68] http://wiki.ros.org/joint_trajectory_action [has been accessed on 27. November 2017].

[69] http://www.openrobots.org/morse/doc/1.2/user/actuators/orientation.html [has been accessed on 27. November 2017].

[70] https://www.openrobots.org/morse/doc/stable/user/actuators/gripper.html [has been accessed on 30. November 2017].

[71] Thomas B. Moeslund, Introduction to Video and Image Processing,Springer-Verlag London Limited 2012

[72] https://www.openrobots.org/morse/doc/stable/user/sensors/semantic_camera.html [has been accessed on 17. October 2017].

[73] https://blender.stackexchange.com/questions/1552/what-is-the-difference-between-an-active-animated-rigid-body-and-a-passive-rigid [has been accessed on 15. October 2017].

[74] Jeanette Schofield, Quaternions and 3D Rotations , April 7, 2011

[75] http://www.openrobots.org/morse/doc/1.2/user/robots/fakerobot.html [has been accessed on 10. September 2017].

[76] https://de.wikipedia.org/wiki/Quaternion [has been accessed on 20. October 2017].

[77] https://cmusphinx.github.io/wiki/tutoriallm/, CMUSphinx [has been accessed on 15. November 2017].

[78] Anthony K. Ho, Fundamental of PID Control, PDHonline Course E331 (3 PDH), P.E.2014, PDH Centre

[79] Zidan, Ahmed & Kotlarski, Jens & Ortmaier, Tobias. (2017). A Practical Approach for the Auto-tuning of PD Controllers for Robotic Manipulators using Particle Swarm Optimization. 34-40. 10.5220/0006419700340040.

[80] Nasr M. Ghaleb and Ayman A. Aly, Modeling and Control of 2-DOF Robot Arm, International Journal of Emerging Engineering Research and Technology, Volume 6, Issue 11, 2018, PP 24-31

[81] Kaung Khant Ko Ko Han, Aung Myo Thant Sin, Theingi, Kinematic and Dynamic Analysis of Two Link Robot Arm using PID with Friction Compensator, International Journal of Scientific & Engineering Research 2017,ISSN 2229-5518

[82] https://au.mathworks.com/help/control/ref/stepinfo.html.

[83] Kambiz Arab Tehrani and Augustin Mpanda, PID Control Theory, Introduction to PID Controllers Theory, Tuning and Application to Frontier Areas, Prof. Rames C. Panda (Ed.), 2012, ISBN: 978-953-307-927-1, InTech, Available from: http://www.intechopen.com/books/introduction-to-pid-controllers-theory-tuning-and-application-to-frontier-areas/theory-of-pid-and-fractional-order-pid-fopid-controller

[84] K. Astrom, K. and T.Hagglund, PID Controllers: Theory, Design, and Tuning, Instrument Society of America, ISBN 1-55617-516-7, 1995

[85] Karl Johan Åström, control system design, lecture notes for ME 155 A, department of mechanical and environmental, university of california 2002.

[86] Brian R Copeland, The Design of PID Controllers using Ziegler Nichols Tuning,application of computers in processes control, Purdue university, , March 2008.

[87] http://wiki.ros.org/pid

[88] Daniel Jurafsky, James H. Martin; Speech and Language Processing (3rd ed. Draft), 2019 Available on https://web.stanford.edu/~jurafsky/slp3/ .

[89] https://docs.blender.org/manual/en/2.79/editors/3dview/object/editing/transform/control/pivot_point/bounding_box_center.html