3D visualization of autonomous underwater robots

Christian Heimdal Sunde

Marine Technology
Submission date: June 2014
Supervisor: Ingrid Schjølberg, IMT

Norwegian University of Science and Technology
Department of Marine Technology

Project description

In most marine operations today there is a low level of autonomy (meaning automatic control without human interaction). Introducing autonomy creates a number of challenges related to human-machine interaction and information gathering. Visualisation and simulation are playing an important role in remote operation of subsea and ocean floor equipment both as support for operators, but also as back-up in case of reduced vision. This work will focus on 3D visualization of underwater robot manipulator motions. The objective of the project is to develop and demonstrate a real-time 3D visualization of a robot manipulator, and prepare it for connection to a real manipulator system. The following points should be conducted

- Develop a 3D model of the manipulator on the ROV SF 30K
- Identify the kinematic model of the arm
- Develop system specification and requirements
- Design and set-up system architecture
- Prepare for connection to a real robot manipulator
- Carry out simulations to verify the modelling
- Document the findings in a report

The report shall be written in English and edited as a research report including literature survey, description of mathematical models, description of control algorithms, simulation results, model test results, discussion and a conclusion including a proposal for further work. Source code should be provided in the attachments with code listing enclosed in appendix. It is supposed that Department of Marine Technology, NTNU, can use the results freely in its research work, unless otherwise agreed upon, by referring to the student's work. The thesis should be submitted within June 10th, 2014.

Supervisor: Professor II Ingrid Schjølberg, IMT, AMOS, NTNU

Sammendrag

Denne oppgaven har som mål å utvikle en fungerende sanntidsvisualisering av en undervannsrobot, samt å presentere relevant teori og steg i utviklingen av et slikt system. Deretter skal programvaren klargjøres for tilkobling til en fysisk robot. Et slikt system kan være et viktig bidrag for økt sikkerhet ved undervannsoperasjoner fordi den kan bedre situasjonsforståelsen til operatørene. Roboten som skal modelleres og visualiseres er en 'Raptor – Force Feedback Manipulator' produsert av Kraft TeleRobotics.

For utvikling av et slikt system er det viktig å velge en programvare som oppfyller alle krav til funksjonalitet. Feil og begrensinger i programmet kan være kritisk i en virkelig situasjon hvor operatøren stoler på informasjonen som programmet gir. Konsekvensene av et uhell under en undervannsoperasjon kan være katastrofale både for sikkerheten til personalet og for miljøet.

En ferdig utviklet 3D modell av manipulatoren ble tilsendt fra produsenten. Ved hjelp av denne var det mulig å ta nødvendige målinger til utviklingen av armens kinematikk. Programvaren som ble valgt til visualiseringen var ROS (Robot Operating System). Dette programmet har en rekke muligheter for utvikling av robotsystemer, inkludert overvåkning og visualisering, og det ble konkludert med at det var et godt egnet program for denne oppgaven. Det er i skrivende stund ikke mye brukt i undervannsvisualisering, men under arbeidet med oppgaven ble det ikke avdekket noen grunner til ikke å bruke det til slike oppgaver.

Ved å bruke grunnleggende teori om kinematikk og Denavit-Hartenberg konvensjonen, ble det utviklet en kinematisk modell for manipulatorarmen. Teorien bak denne modellen var viktig å kjenne til når den skulle implementeres i programvaren. Målinger gjort av 3D modellen ble så inkludert i programmet ved å bruke URDF-filer (Unified Robot Description Format), som er filformatet ROS bruker for å beskrive en robots utseende og egenskaper. Deretter ble programvaren tilpasset til å være enkel og intuitiv under bruk, og en beskrivelse av hvordan det brukes og testes ble laget. En fremstilling av det fysiske manipulator-systemet og hvordan programvaren kan kobles til denne ble også utviklet.

For å verifisere at programmet fungerte ble simulerte hendelser gjort isolert på hver del av modellen. Disse testene bekreftet at modellen og programmet fungerte som det skulle og at det er klart for tilkobling til den fysiske armen. Når denne er klar til bruk, vil dette programmet være enkelt å implementere.

Abstract

This thesis has an objective aimed to develop a functional real-time visualization of an underwater robot manipulator, and present the relevant theory and steps related to this system. Then the system should be prepared for connection to a real robot manipulator. Such a system is important for increased safety in underwater operations and to improve the operator's situation awareness. The manipulator that was modeled and tested in this thesis was an 'Raptor - Force Feedback Manipulator' produced by Kraft TeleRobotics.

To develop the system it was important to choose the right software for use, and verify that the end product was working properly. Errors in the system can be critical in a real situation where an operator trusts the information that the software provides. Also the consequences of an accident during operation can be fatal for human safety and the environment.

With a 3D model of the manipulator received from the manufacturer, it was possible to analyze and take measurements of the manipulator for use in the visualization software. The software that was chosen was the Robot Operating System (ROS) framework. This framework has a huge amount of opportunities including monitoring and visualization of robotic systems, which makes it a good choice both for this thesis and in terms of further development. It has not been used much in underwater robotics, but arguments for not using it hasn't been revealed during the work with this thesis.

By using basic kinematic theory and the Denavit-Hartenberg representation the forward kinematics for the arm was derived and explained in detail. It is important to know this theory when implementing the model into the software. Measurements from the analysis of the model and CAD-files (Computer-aided design) was applied and described in URDF-files (Unified Robot Description Format) so that it could be imported into the software. The software solution was configured and customized to be easy and intuitive in use, and an explanation of how to run and test the system was included. In addition, the hardware set-up and procedures for how to connect the software to a real manipulator was produced and documented.

To verify that the system was working properly, simulated events was conducted for each part of the model. These simulations confirmed that the system was acting as wished. With the real manipulator up and running, this software should be ready for use with just a little effort put into the signal input connection.

Acknowledgements

It is a pleasure for me to acknowledge the help and support that I have received from my supervisor, Professor II Ingrid Schjølberg. She has been very helpful and engaging throughout the whole semester, and her insight and knowledge in the robotic field has been valuable for the progression and the end result of this thesis.

I will also like to thank Ph.D. candidate Lars Tingelstad at the Department of Production and Quality Engineering for recommending and introducing me to the software that was used in this dissertation. The work with this software has been very educational for me. A thanks also goes to Ph.D. candidate Anastasios Lekkas for setting up the manipulator lab at the Marine Technology Centre. It has been very inspiring to see the arm move in real life, and it has given this thesis a very practical approach. Thanks also to Kraft TeleRobotics for providing me the manipulator CAD files.

During the last weeks of the semester, a jury consisting of Prof. Svein Sævik, Prof. Ingrid Bouwer Utne and Prof. Bernt Johan Leira decided to give me the award for *Best Master Thesis Poster of the year* at the yearly Master Thesis Poster Exhibition Awards. This is a great honour, and it is very motivating to be selected among so many good candidates.

I also would like to thank all of my fellow students in the class of 2014, and especially my office colleagues. They have all given me five great years both through academic and social events.

[illegible]

Christian Heimdal Sunde
Trondheim, June 9, 2014

Contents

List of Figures

List of Tables

Nomenclature

Abbreviation	**Description**
3D	3-Dimensional
AUV	Autonomous Underwater Vehicle
CAD	Computer-Aided-Design
D-H	Denavit-Hartenberg
DOF	Degrees of Freedom
FFC	Force Feedback Control
FFM	Force Feedback Manipulator
GPS	Global Positioning System
PDF	Portable Document Format
PLM	Product Lifecycle Management
ROS	Robot Operating System
ROV	Remotely Operated Vehicle
RS-	Recommended Standard
STEP	Standard for the Exchange of Product model data
STL	STereoLithography
URDF	Unified Robot Description Format
XACRO	XML Macros
XML	Extensible Markup Language

Symbol	**Description**
$\mathbf{A}_i$	Transformation matrix
$\mathbf{H}_j^i$	Homogeneous transformation matrix
$\mathbf{O}_j^i$	Origin Coordinate Vector
$\mathbf{P}^i$	Position coordinates
$\mathbf{q}$	Joint angles
$\mathbf{R}_j^i$	Rotation matrix
$\mathbf{T}_j^i$	Transformation matrix

Chapter 1

Introduction

Utilization of the seas resources is becoming more important every day, and also more challenging. Increased demands for fuel, food and energy is forcing the industry to seek more inaccessible resources in ultra deep water and in rougher and more fragile environments. Safe and controlled operations are more important than ever, and the industry is putting huge efforts and money to make sure the operations is conducted as planned with no damages on humans or the environment.

This development requires in many cases new and better technology to keep track of the demands. Accurate and stable control systems is something that has been in focus for many years, and it has evolved even more significantly in modern time with the development of new technology. This technology gives the opportunity to have systems with robust security that automatically reacts to possible threats and insecure situations. These kinds of systems can provide safety to the operator, but it might also give them false safety if the information provided has errors. Therefore it is important not to remove the possibility to observe and handle parts of the operation manually [15].

This thesis aims to develop a 3D visualization of a manipulator system that can provide a close to reality visualization of a manipulator arm built for underwater operations. The manipulator is an 'Raptor FFM' (Force Feedback Manipulator) produced by Kraft TeleRobotics. A tool like this will provide the operator with improved situation awareness and the risk of failures such as collision, damages or other unexpected obstacles will hopefully decrease. Also visual data sources like cameras can be used by the system to determine the environmental status, locate objects and determine the position of the vessel. This important field is in a huge development process since there are many big challenges in the industry that requires these kinds of solutions now and in the future.

In the next chapters the theory behind the Raptor's kinematic properties will be explained. The manipulator will be described and inspected to determine all the necessary data for the visualization. Also each step in the development process of the visualization system will be covered, together with the work done to prepare it for connection to a real robot manipulator.

This first chapter will give an introduction to the motivations for developing such tool, and an introduction to the basic concepts regarding underwater vehicles and underwater software.

1.1 Motivation

Offshore and underwater operations are often much more challenging than operations onshore. The main reasons for this is because the operations will be exposed to bigger and in many cases unforeseen forces that require more planning, more advanced equipment and more robust safety and security systems. A huge part of such systems is the ability to have a good overview of the operation. When working in an underwater environment it might be impossible to have humans presented exactly where the operations are performed due to high water depths, harsh conditions, expenses and safety issues.

During an underwater operation it is important to have full control over the situation, meaning that the operator should have good situation awareness. This can be obtained by having access to some visual data from the operation to achieve overall situation awareness, and to identify errors that the instruments on the vehicle cannot find. These errors can be small cracks in a structure or unwanted objects. Bad visibility, limited space and accessibility are things that can cause difficulties for visual sources. When working with equipment under water there is always a risk for damages on the equipment itself and structures around due to unexpected disturbances such as hydrodynamic forces, errors made by humans or defects in the software. An accident can cost a lot of money in repair, replacement and production downtime, in addition to the risk of harming the environment and humans.

By having a reliable visualization system the risk for such accidents to happen will most likely decrease. It can help providing important information both to the operator and to the system itself. Such system might also form a base for a system that can simulate and visualize a planned operation so that its feasibility can be verified.

1.2 Remotely operated underwater vehicles

Remotely Operated Vehicles (ROV) are widely used in the offshore industry today. They can be used to conduct difficult and detailed tasks on deep-water operations. It is unmanned and is being controlled by a crew located on a vessel at sea surface. The link between the ROV and the control station is by a tether, and if necessary with an umbilical cable attached. This umbilical contains a set of different cables that transports power, video signals and measuring data among others. In Figure 1.1 is a picture of the ROV Minerva that is a research vessel owned by Norwegian University of Science and Technology. It is used in many different types of operations such as biological research and sampling, geological operations and development of new technology both for research and in education [16]. ROV's in general are used in many other types of operations as well. Military operations, science, rescuing, salvage, education and industry is some of them [3].

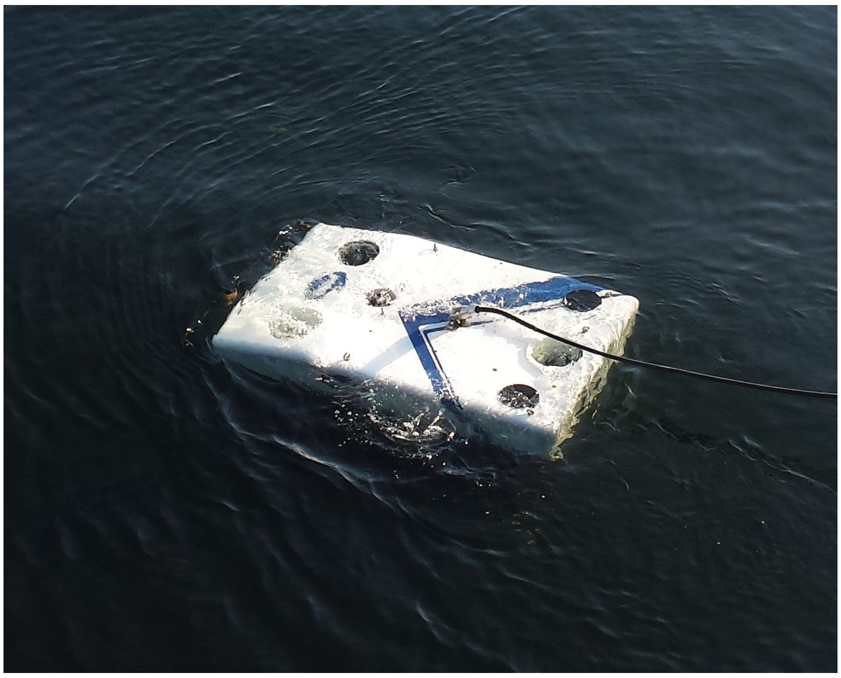


Figure 1.1: ROV Minerva

A classical ROV is constructed with a large buoyancy tank on the top to make the weight under water manageable. It is equipped with thrusters that is strategically located to make it as easy to manoeuvre as possible. Normally there are cameras and lights mounted together on the vehicle with the necessary equipment that the task requires.

1.3 Manipulator systems

Robot manipulator systems are increasingly common in the modern world, and most seen in the manufacturing industry. They are often referred to as robotic arms that can perform tasks quick and precise. This makes them very important in the industry because they can increase production, improve quality and lower the cost. ROVs is often equipped with one or more manipulator arms. They are designed to manipulate or handle materials without being directly in contact with it. It makes the ROV able to perform various tasks that require precise tool handling or gripping [2] [7].

There are many different types of manipulators. They are put together by links and joints that can behave in the way that is necessary to perform the tasks it is supposed to do. All these links and joints together creates a kinematic chain that will define the manipulators properties. These characteristics will vary depending on what tasks the manipulator is intended for, and they will be described more in Chapter 3.

1.4 Software

Software tools is an important part of robotics. A good software should help the operator during the operation and in the planning process so that the situation awareness is good. There are many different types of software solutions. Planning software, controlling software and visualization software is some of them. The visualization software that this thesis will be about has a purpose of providing the operator with all the necessary information needed to know the real-time status of the visualized underwater manipulator. This might be useful if the visibility from visual sources such as a camera is bad. The operator can use this information to plan further actions or determine how safe the operation is.

The software tool chosen in this thesis is the Robot Operating System (ROS) framework [21]. This flexible and versatile tool has many useful features that can be used to solve the software solution problem. More discussion about ROS and why it has been chosen will be discussed in Section 2.3 and 5.1.1.

Chapter 2

3D Visualization

Visualization is described as any tool or technique used for creating a visual picture, object or animation to describe data or information. It is used in many different situations all over the world to convey any abstract or imagined information. In engineering it is used to represent data from calculations such as graphs, diagrams, models and product visualizations. In most of today's engineering problems a computer graphical interface is used. This gives the opportunity to visualize huge and complex models and calculations, and give a clearer and more intuitive understanding of a problem, situation or a result [30].

In scientific visualization engineers presents representation, selection or transformation of data from simulations or experiments. Mostly shown by geometric structures put together into an image that will give a better understanding of the data that is represented.

For this thesis, a 3D visualization tool is to be developed to visualize the Raptor FFM. This can be a first step in the development of a complete visualization application for an ROV with a manipulator arm. To do this, sensor data from the manipulator will be recorded and transformed into coordinates that will be used by the 3D visualization tool. A simple way this can be done is to represent each link as a geometrical figure and orient this figure to fit with the calculated points. This way the visualization will give a realistic image of how the robot manipulator is acting.

2.1 Visualization advantages

There are many advantages in having a good visualization software. A visualization software can, if it provides enough accurate information, be a very useful tool to increase the efficiency, accuracy, and safety of an operation. It is therefore important to include all vital knowledge in such visualization application.

For inspection and maintenance of an underwater structure or system, an ROV is often used with a camera or sensor attached, so that the operator can overview, locate areas of interest and perform actions if necessary. This is an important task both due to environmental issues and to make sure that the production or ongoing operation is optimal and safe. If a damage or problem is detected, it is in most cases necessary to have a complete overview of the situation so that a plan for measures can be developed. Mistakes and missing information can be critical in such tasks, and if something goes wrong the consequences can be severe and expensive. A visualization tool can therefore help by increasing the situation awareness and by this lower the risk for accidents to happen.

During maintenance special equipment is often required depending on the damage type that must be fixed. A manipulator arm is very common because it gives the ability to act and perform tasks almost as if a human arm would have done it. Since a good visualization software can provide important information about the operation, it might be possible to complete an operation due to the good overview. In a situation where there is not a good visualization software, it might be a higher risk for accidents and the operations can be delayed or maybe not be possible to complete. The software can also increase the efficiency of operations and make them more profitable.

2.2 Challenges in underwater visualization

As mentioned earlier underwater operations involves many challenges. Underwater visualization is also affected by these. Reliable sensors, localization and communication is some of the challenges that is involved. A visualization software will be depending on an accurate and reliable information source so that its reproduction of the situation also becomes reliable. Limited bandwidth, distances and unforeseen failures can suddenly make the information that the visualization provides less dependable. Also what methods or techniques that is being used to solve these challenges in underwater visualization can make a big difference on the software reliability [34] [25].

2.2.1 Sensors

Sensors are very important when creating good robotic systems. A controlling system is dependent on sensor data to be able to carry out the necessary actions for completing a task, and a visualization system is dependent on accurate sensor data to be able to give an image on how the reality looks like. Sensors are usually divided into two types, proprioceptive sensors and exteroceptive sensors [20]. Proprioceptive sensors are measuring the internal state of the unit. This is including quantities such as joint position, joint velocity and joint torque. The exteroceptive sensors are providing information about the surrounding environment. They can be force sensors, tactile sensors, proximity sensors, range sensors and vision sensors.

The goals with such types of sensors is to give the system all the necessary information needed to conduct the desired action. If there were no sensor data it would have been impossible to determine the errors in the system or changes in the surroundings. Therefore, it is important to have reliable sensors. Underwater operations can be harsh and demanding, and the quality of the sensors must be adapted for such use.

2.2.2 Localization

Robot localization, also called robot mapping, is a way that a robot can localize itself in space. This is very important because it can be very difficult to manoeuvre an underwater vessel with only visual data. If the sight is bad there are no references in space that can help the operator lead the vehicle to the desired position. GPS-signals are almost impossible to use under water because its radio waves does not propagate well in water. Therefore it is necessary to find other feasible ways to determinate its position [4].

There are many different techniques that can be used to do this. The robot can track its movements and use this to always know its absolute position. However, the risk here is that it can be small errors in the sensors that will create a bigger error over time. Another way to localize is to use acoustic tags. They are sound emitting devices that have quite large detection ranges (up to 1 km in freshwater) and are widely used in industries like fishing to track the fish.

In space technology it is common to use reference points to find a spacecraft's position. By using more than two reference points, such as a selection of stars, it is possible to calculate the position in three dimensions. It is possible to do the same in under water localization. By having several reference signals, i.e. acoustic signal buoys, it is possible to calculate the position under water as well.

2.2.3 Data and communication

When working under water it is important to have a good communication link between the operating station and the underwater vessel. This to make sure that important data like visual images and sensor data among others is transferred to the operating station without losses. On many ROV's it is common to have a tethered link with an umbilical cable between the ROV and a floating vessel. This cable is transferring information such as video signals, sensor data, tasks and electrical power. An umbilical cable can be as the one seen in Figure 1.1.

Wireless solutions can also be used. However, in many cases this will limit the bandwidth of the data transfer, and there will be a higher risk of signal loss. Wireless communication is more common for AUVs (Autonomous underwater vehicles) because they are working with little or no data transfer to the operating station, which significantly decreases or eliminates the data transfer problem.

2.3 Why use ROS for visualization?

Figure 2.1: ROS - Robot Operating System logo [21]

Design of software for robotic solutions can be a complicated task. The field is always in progress and this makes the software requirements more advanced and complex. Different systems requires different resources so that the software must be customized to fit one specific system. To make this progression less challenging for developers, ROS has been created [21] [19]. Although its name says it is an operating system, it is not. It's a framework designed for writing robot software. Almost all robotic fields that is relevant is covered, and it is continuously being developed by users from all over the world. It has not yet been used much for underwater robotics, but this thesis has not discovered any arguments for not using it in this field.

The framework only runs on Linux operating systems as an official release, but it is in an experimental phase for a version that can run on Windows and Mac OS X. It aims to simplify the development process of new and advanced robotic systems. Also it is open source and has a huge worldwide community of developers. ROS is not directly a real-time framework, but it has the possibility to be integrated with real time code. This is beneficial since it opens the possibility to implement many relevant systems such as control systems, guidance systems and monitoring systems.

The goal with ROS is to have an open and worldwide used framework for development in the robotic field. By having such framework available, the numbers of new custom-made frameworks will be lower and the community will become bigger and with broader disciplines. Developers all over the world can benefit from the knowledge in a huge open community. Including user scenarios such as debugging, logging, controlling and development, ROS can also be used for visualization and monitoring. This combination of different opportunities makes this framework a great choice for this thesis because of the possibility to expand its functionality. Also the visualization package for ROS, rviz, is a great tool with many good features [22]. Its intuitive and easy interface and set-up procedure is also a great advantage. More about ROS and how it is used will be discussed and explained in Chapter 5.1.1.

Chapter 3

Modeling

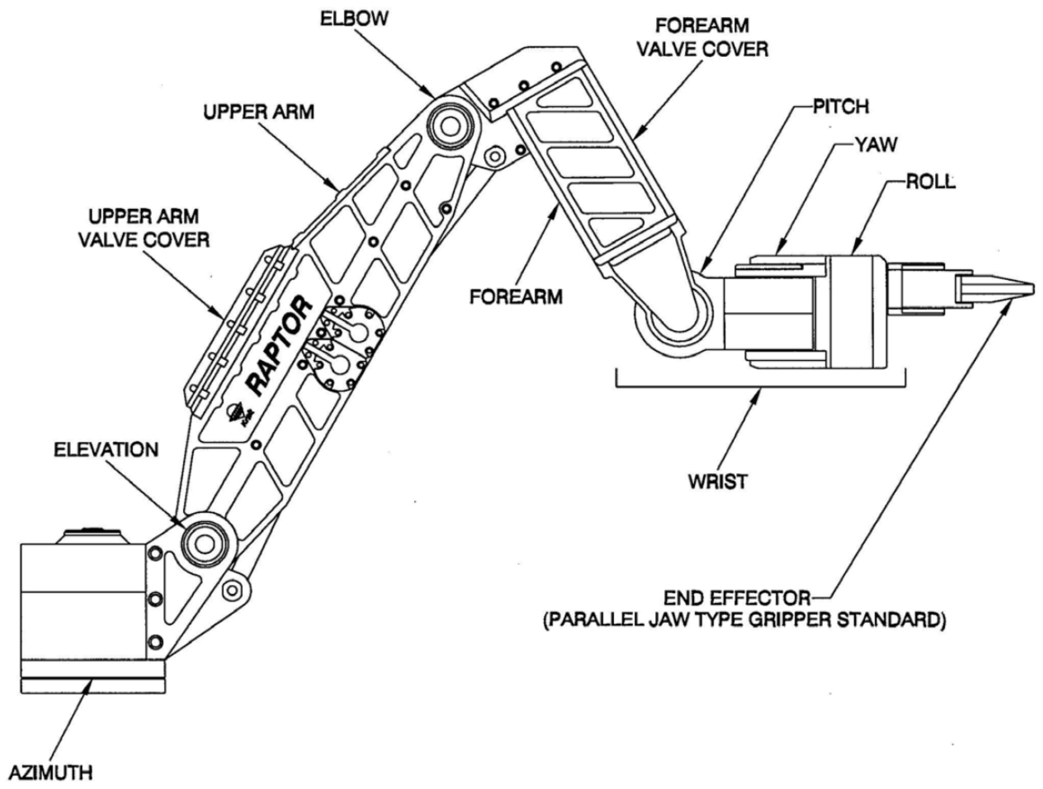


Figure 3.1: Parts of the Raptor manipulator [11]

The manipulator is as mentioned a Raptor - Force Feedback Manipulator produced by Kraft TeleRobotics. This manipulator has documentation available from the product data sheet and the user manual, but the quantity of usable information for a visualization software development was very limited. However, after contacting TeleRobotics directly, they were able to send the CAD-files (Computer-Aided-Design) of the manipulator. By using this, the necessary measurements and inspections could be done. With these files available, the whole thesis can be completed with fewer assumptions, which most likely will make the final product a lot better.

In the following chapter the theoretical manipulator kinematics will be presented. Not all the derivations will be explained in detail, but the necessary knowledge needed for an implementation of this in the visualization software will be described. A few comments on the dynamics of the manipulator will also be outlined to give a wider understanding of the manipulator, and to understand the possibilities for further development. The theory is based on the content found in [26].

3.1 Notation

Robot dynamic models involves many mathematical expressions. Therefore it might be helpful to give a brief introduction to the notation that will be used in the process. Vectors and matrices will always be printed in bold face so that it is easy to see the difference between these and regular variables. Vectors and points are also marked with a superscript to denote the respective reference frame.

A manipulator variable can be represented by q_i, where i refers to the joint number it belongs to. A set of joint variables is then given as $\mathbf{q} = [q_1, q_2, ..., q_n]^T$. Rotation- and Transformation matrices will be written as $\mathbf{R}_j^i$ and $\mathbf{T}_j^i$, where j denotes the frame that will be rotated or transformed, and i is the resulting frame of the rotation or transformation. Note that most of the symbols is explained in the nomenclature on page xv.

3.2 Manipulator theory

As mentioned in Section 1.3 robot manipulators consists of a series of links that is connected together with joints, and all this together forms a kinematic chain. The complexity of this chain depends on what types of joints the system consists of. Some are simpler than others.

The joints can have revolute geometry, which means that it behaves almost like a human elbow. However, usually they can move in a bigger rotational angle than a real human elbow. Joints can also be prismatic, which means that the joint allows a linear relative motion between two links. The combination of these joints is called an open kinematic chain. Based on the structure of the kinematic chain, its geometry, its purpose, power source and how they are controlled, the manipulator can be classified into classes or configurations. Some of the most common classes are cartesian, articulated, spherical and cylindrical [29]. Some different types of joints can be seen in Figure 3.2.

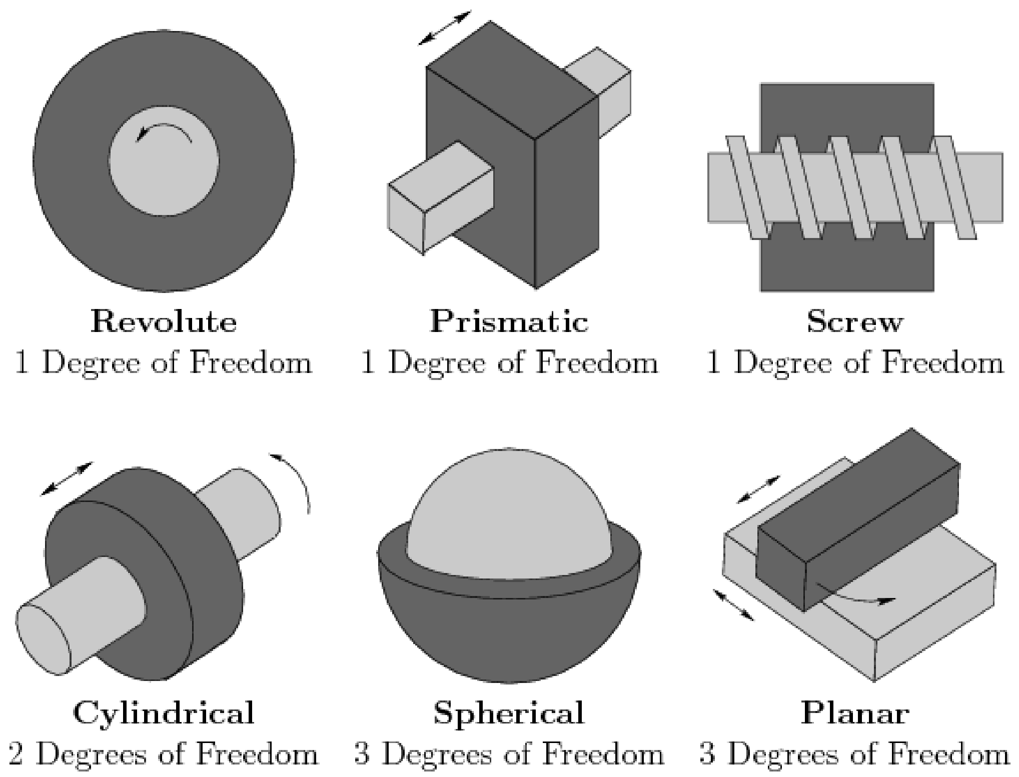


Figure 3.2: Different types of joints [14]

A manipulator where the first three joints are prismatic is a Cartesian or rectangular manipulator. It can operate in any position in a rectangular workspace. This type of manipulators has of course the simplest kinematic description of all the configurations because there are no angular motions involved. Which means that it can be described simply by adding all the measured states together to find the end effector.

The articulated configuration is also called a revolute configuration. Here are all the joints revolute, which gives the manipulator the ability to work in compact spaces and with a relatively large freedom of movements. Compared to the Cartesian manipulator, an articulated manipulator can reach both over and under an object, but the kinematics is also much more complex than for the Cartesian.

If the third joint or the elbow in an articulated configuration is replaced with a prismatic joint, it will be a spherical configuration. Its workspace will be a part of a spherical coordinate system. However, this will also have a complex kinematic model that makes it more difficult to compute and visualize.

In a cylindrical configuration, the first joint is revolute and the second and third is prismatic. This first one will produce rotation around the base so that the whole workspace will have a shape of a cylinder.

In Figure 3.3 the different types of manipulator geometries are illustrated.

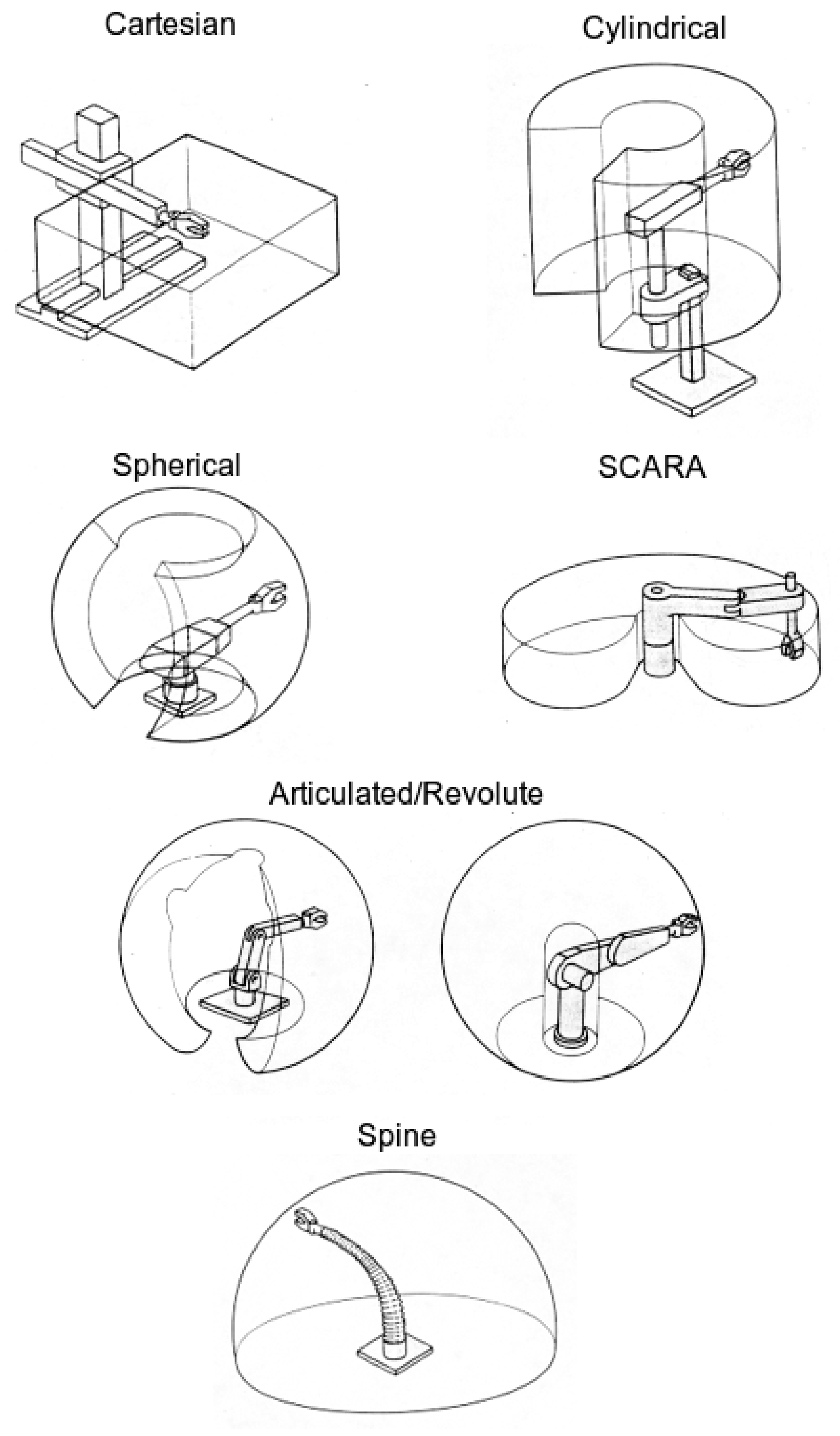


Figure 3.3: Different robot arm geometries [8]

As a result of the total kinematic chain and its limitations, the manipulators workspace can be defined. It is described as the space where the end-effector or the last link can reach. The last link is normally a gripper or a tool. If assuming that the Raptor joints has no limits in its rotations, the workspace would have been a sphere with a radius that is equal to the maximum length of the arm. This is of course not the case. The manipulator has limits in both directions of movement and angular displacement. Figure 3.4 shows the workspace of a spherical manipulator with limitations. Later in this thesis it will also be seen that the Raptor manipulator has a workspace that is similar to Figure 3.4.

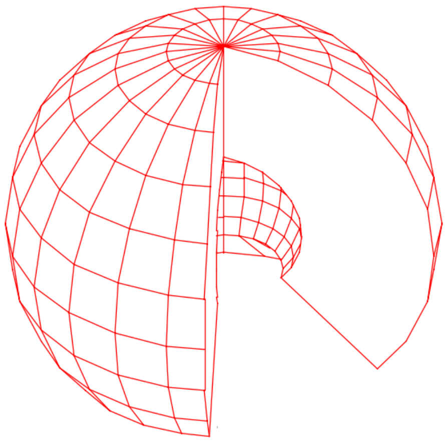


Figure 3.4: Workspace of a spherical geometry [1]

3.3 Transformations

In a kinematic system it is an advantage to use several coordinate systems to represent each systems relative position and orientation to the other coordinate systems. To perform algebraic manipulations with such coordinate systems it is crucial that vectors are defined with the same coordinate system as reference. This requires mathematical operations that transforms and rotates the vectors in a given coordinate system into vectors with respect to the desired coordinate frame. Such transformations are very important in robot kinematics and will be described in detail in the following sub-chapters.

3.3.1 Orientation

If a coordinate system have the same origin as a second one but with a different orientation, one can define the rotation matrix $\mathbf{R}$ that describes its rotation relative to the other coordinate system. In Figure 3.5 the blue coordinate system is rotated an angle ψ about the z-axis. Equivalently, the rotation angle θ is defined as the rotation about the y-axis, and ϕ about the x-axis.

From [6] we have the following notation for transformation between coordinate systems. A 3×1 vector $\mathbf{v}_a^0$ in the first coordinate frame can be rotated from a

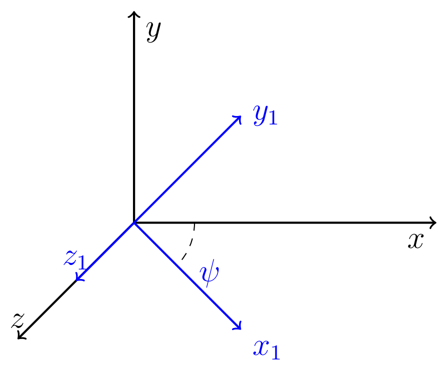


Figure 3.5: Two coordinate systems with same origin, but different orientation

vector $\mathbf{v}_a^1$ into a second one with the following relationship.

$$\mathbf{v}_a^0 = \mathbf{R}_1^0 \mathbf{v}_a^1 \tag{3.1}$$

Where the rotation matrix $\mathbf{R}_1^0$ is given as

$$\begin{aligned}\mathbf{R}_1^0 &= \mathbf{R}_{z,\psi}\mathbf{R}_{y,\theta}\mathbf{R}_{x,\phi} \\ &= \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\psi s\theta s\phi & -c\psi s\phi + s\psi s\phi c\theta \\ -s\theta & c\theta s\phi & c\psi c\phi \end{bmatrix}\end{aligned} \tag{3.2}$$

Here the sine and cosine functions are shortened to s and c for convenience. The rotation matrix is expressed in what we call Euler angles that consists of three principal rotational matrices that rotates respectively about the z-, y- and x-axis. It is always orthogonal which implies the following notation [33].

$$\mathbf{R}^{-1} = \mathbf{R}^T \tag{3.3}$$

$$\mathbf{R}\mathbf{R}^T = \mathbf{R}^T\mathbf{R} = \mathbf{I} \tag{3.4}$$

$$\det(\mathbf{R}) = \mathbf{I} \tag{3.5}$$

3.3.2 Homogeneous Transformations

On manipulators such as the Raptor manipulator, there is not just the orientation that is important. The position of the body fixed reference frames should also be known. When combining these two concepts the homogeneous transformation is defined.

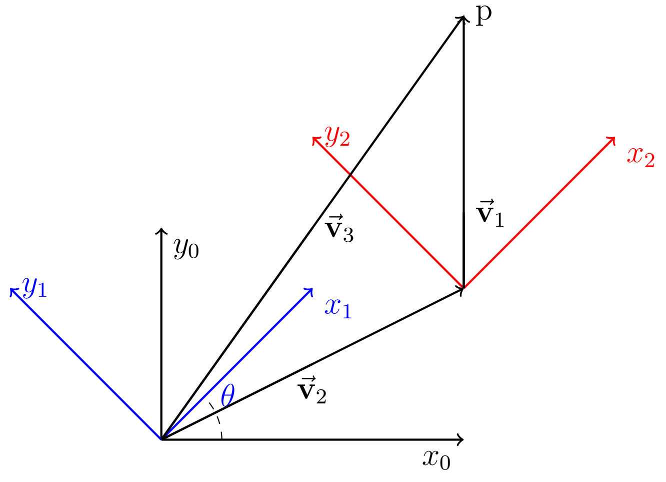


Figure 3.6: Homogeneous transformations in two dimensions

In Figure 3.6 the point p is given with respect to the red frame $o_2x_2y_2$. If the orientation and position of this frame is known, it is possible to calculate the position p with respect to the fixed frame $o_0x_0y_0$. The red frame's orientation is represented as the blue coordinate system $o_1x_1y_1$ with $o_1 = o_0$. From the figure one can see that the point p is displaced by a vector $\vec{\mathbf{v}}_3$ from the fixed origin. This vector is also a sum of the vectors $\vec{\mathbf{v}}_1$ and $\vec{\mathbf{v}}_2$.

$$\vec{\mathbf{v}}_3^0 = \vec{\mathbf{v}}_2^0 + \vec{\mathbf{v}}_1^0 \tag{3.6}$$

The direction and magnitude of $\vec{\mathbf{v}}_1^0$ can be found by rotating the coordinates of $\mathbf{p}^2$ an angle θ, where the superscript 2 describes from what frame the coordinates are described in. In this case the red coordinate system $o_2x_2y_2$.

$$\vec{\mathbf{v}}_1^0 = \mathbf{R}_2^0\mathbf{p}^2 \tag{3.7}$$

Further the vector $\vec{\mathbf{v}}_2^0$ is describing the displacement of the coordinate system $o_2x_2y_2$ relative to the fixed coordinate system, for convenience renamed to $\mathbf{d}_2^0$. By substituting this into the equation (3.6) the following relation is found.

$$\mathbf{p}^0 = \mathbf{R}_2^0\mathbf{p}^2 + \mathbf{d}_2^0 \tag{3.8}$$

And therefore the following relations is also valid.

$$\mathbf{p}^0 = \mathbf{R}_1^0\mathbf{p}^1 + \mathbf{d}_1^0 \tag{3.9}$$
$$\mathbf{p}^1 = \mathbf{R}_2^1\mathbf{p}^2 + \mathbf{d}_2^1 \tag{3.10}$$

These two substituted together will then give

$$\begin{aligned}\mathbf{p}^0 &= \mathbf{R}_1^0\mathbf{p}^1 + \mathbf{d}_1^0 \\ &= \mathbf{R}_1^0\mathbf{R}_2^1\mathbf{p}^2 + \mathbf{R}_1^0\mathbf{d}_2^1 + \mathbf{d}_1^0 \end{aligned} \tag{3.11}$$

The relations are also valid in a three dimensional system. This will naturally result in large complex equations as the number of coordinate systems and the DOF (Degrees of Freedom) increases. And it is therefore convenient to describe this system in matrix form to simplify the notations. To do this a matrix H, that describes the homogeneous transformation is introduced.

$$\mathbf{H}_j^i = \begin{bmatrix} \mathbf{R}_j^i & \mathbf{d}_j^i \\ \mathbf{0} & 1 \end{bmatrix} \tag{3.12}$$

Where $\mathbf{0}$ is an 3×1 vector of zeroes, $\mathbf{R}_j^i$ is the rotational matrix from coordinate system j with respect to the i^{th}. And $\mathbf{d}_j^i$ is the origin position of the j^{th} coordinate system with respect to the i^{th}. Further derivations then gives the following.

$$\mathbf{P}^1 = \begin{bmatrix} \mathbf{p}^1 \\ 1 \end{bmatrix} = \mathbf{H}_2^1 \begin{bmatrix} \mathbf{p}^2 \\ 1 \end{bmatrix} \tag{3.13}$$

$$\begin{aligned}\mathbf{P}^0 = \begin{bmatrix} \mathbf{p}^0 \\ 1 \end{bmatrix} &= \mathbf{H}_1^0\mathbf{P}^1 \qquad (3.14) \\ &= \mathbf{H}_1^0\mathbf{H}_2^1 \begin{bmatrix} \mathbf{p}^2 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}_1^0 & \mathbf{d}_1^0 \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_2^1 & \mathbf{d}_2^1 \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}^2 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}_1^0\mathbf{R}_2^1 & \mathbf{R}_1^0\mathbf{d}_2^1 + \mathbf{d}_1^0 \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}^2 \\ 1 \end{bmatrix}\end{aligned}$$

So that (3.11) can be written on matrix form as

$$\mathbf{P}^0 = \begin{bmatrix} \mathbf{R}_2^0 & \mathbf{R}_1^0\mathbf{d}_2^1 + \mathbf{d}_1^0 \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}^2 \\ 1 \end{bmatrix} \tag{3.15}$$

3.4 Kinematics

Now that the basic transformation equations has been derived, it is time to have a look at the motions of the manipulator body itself. The kinematics describes this without looking at the forces that causes the motions. Normally kinematics is divided into two types; Forward and Inverse kinematics. The forward kinematics is about determine the position and orientation of an object end-effector when the joint variables are known. Since the variables in tasks like this are given, they will only be geometrical problems. With inverse kinematics it is the opposite. Here the joint variables must be determined based on a known end-effector.

Because the visualization of the Raptor manipulator is based on sensor data from the physical manipulator, meaning that the joint variables are known, this task will only consider the forward kinematics problem. In a way this theory is not actually necessary to have when making this visualization application. However, it is a major advantage to have this knowledge when setting up the kinematic chain for use in a visualization software. And that is also the reason for why this theory is so important to include in this thesis.

3.4.1 Forward Kinematics

The forward kinematics problem is as mentioned a pure geometrical problem. All the angles are known, and the only problem is to make sure that the calculations are done correctly when processing the sensor data from the manipulator. These variables are given as angles between the links. To do these calculations as clean and efficiently as possible it is a beneficial to do it systematically [26].

As a start, the joint and links are numbered to describe how the links are connected through the manipulator. The joints can be simple 1-DOF joints, or more complex joints like those seen in Figure 3.2 with more than 1-DOF. A conventional manipulator will have one more link than joints. Mathematically this means that if the manipulator have n joints, it will have $n + 1$ links, including the manipulators base. This also means that link i will be fixed with respect to link $i - 1$. Naturally the joint variable associated with a joint will have the notation q_i and is as mentioned the angle of rotation when the joint is a revolute joint. It can also be in other units such as displacement if the joint is a prismatic joint. For the Raptor manipulator in this thesis all the joints are revolute.

As shown in Figure 3.7 each link has a coordinate frame attached rigidly, noted as $o_i x_i y_i z_i$ where i is the link number. Each frame has a non-constant homogeneous transformation matrix $\mathbf{A}_i(q_i)$ that describes the associated frames position and orientation with respect to the previous link $o_i x_i y_i z_i$. This matrix will only depend on one variable, q_i. By using many homogeneous transformation matrices one can produce a transformation matrix, $\mathbf{T}_j^i$. Where the matrix expresses the position and orientation of $o_j x_j y_j z_j$ with respect to $o_i x_i y_i z_i$. This matrix is given as

$$\mathbf{T}_j^i = \begin{cases} \mathbf{A}_{i+1}\mathbf{A}_{i+2}\cdots\mathbf{A}_{j-1}\mathbf{A}_j, & \text{if } i < j \\ \mathbf{I}, & \text{if } i = j \\ (\mathbf{T}_i^j)^{-1}, & \text{if } i > j \end{cases} \tag{3.16}$$

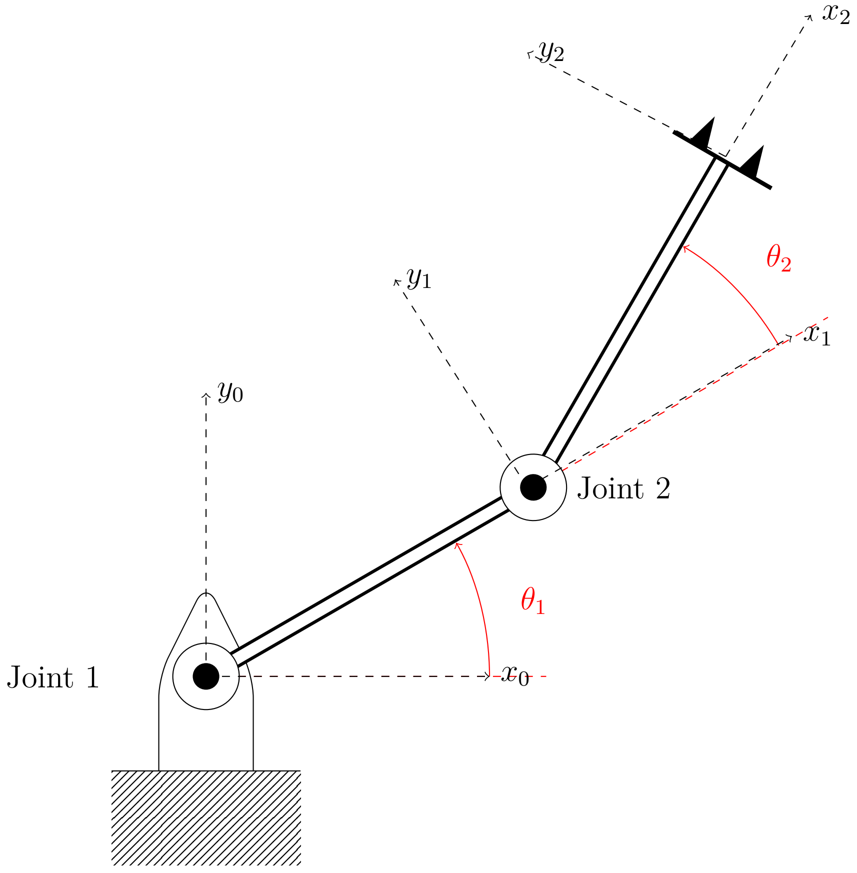


Figure 3.7: Example of reference frame attachment [9]

3.4.2 Denavit-Hartenberg Representation

The transformation matrix $\mathbf{A}_i$ that was explained in Subsection 3.4.1 gives the opportunity to solve the forward kinematics problem. However, it is necessary to choose the right reference frames to fully simplify the analysis. A way to choose these reference frames is the Denavit-Hartenberg (D-H) convention [24]. This convention represents the homogeneous transformation matrix as a product of four basic transformations.

$$\mathbf{A}_i = \mathbf{R}_{z,\theta_i}\mathbf{T}_{z,d_i}\mathbf{T}_{x,a_i}\mathbf{R}_{x,\alpha_i} \tag{3.17}$$

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

$$= \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.18}$$

Where

a_i = distance along x_i from o_i to the intersection of the x_i and z_{i-1} axes.

α_i = the angle between z_{i-1} and z_i measured about x_i.

d_i = distance along z_{i-1} from o_{i-1} to the intersection of x_i and z_{i-1} axes.

θ_i = angle between x_{i-1} and x_i measured about z_{i-1}.

And the different axes are as shown in Figure 3.8.

When comparing this matrix with the transformation matrix derived in Subsection 3.3.2, one can see that the D-H-representation has reduced the number of parameters from six to four. This is valid as long as these two requirements to the reference frame is introduced

1. The axis x_i is perpendicular to the axis z_{i-1}
2. The axis x_i intersects the axis z_{i-1}

Every homogeneous transformation matrix that is satisfying these requirements can be represented in the form given in (3.18). D-H convention can therefore be seen as guidelines that makes sure that these requirements are fulfilled. Note that these guidelines does not lead to a unique set of reference frames, but the final result $\mathbf{T}_n^0$ will be the same.

The coordinate frames must be placed so that the z-axes is positioned in a satisfactory location. To do this the z_i is placed so that it describes the actuation of joint $i+1$. This means that if the joint is a prismatic joint, the z_i axis will be the axis of translation. And if it is revolute it will be the axis of rotation. The base frame can more or less be chosen to be anywhere as long as it follows the right hand rule as described in Figure 3.8, and that the origin is located on z_0.

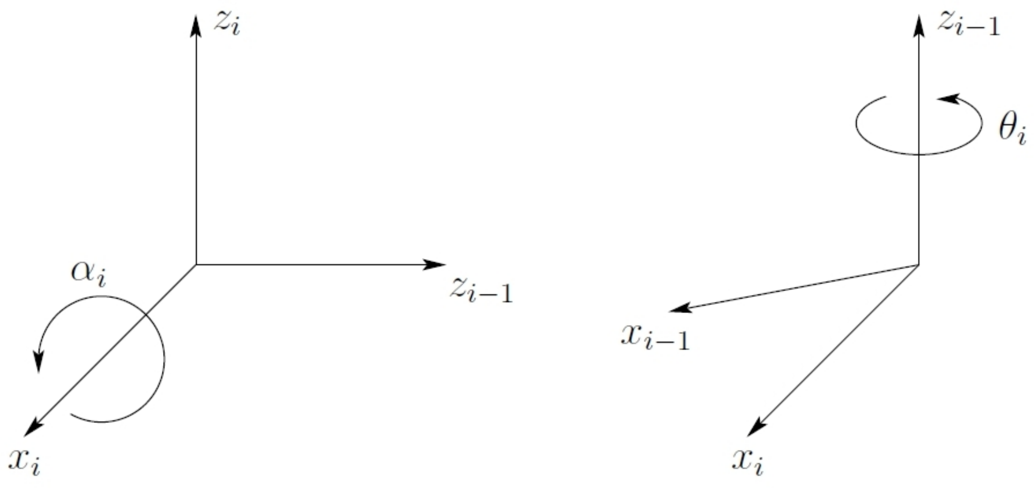


Figure 3.8: Right hand rule - positive rotation directions for α_i and θ_i [26].

Now a process of establishing the remaining reference frames must be conducted. To do this there are three common situations that can occur when performing this iterative process [26]. These three are

1. z_{i-1} **and** z_i **are not coplanar:** A unique line segment exists that is perpendicular to both z_{i-1} and z_i that connects the axes. This line defines x_i, and the point where this line intersects z_i defines the origin o_i. The y axis is then chosen to form a right-hand rule. Using this procedure ensures that both condition 1. and 2. is satisfied.

2. z_{i-1} **is parallel to** z_i**:** The origin o_i can be chosen anywhere along z_i, but it is often chosen to simplify the calculations. The x_i axis is then chosen either toward or away from z_{i-1} along the common normal. If the x_i axis is chosen as the normal that passed through o_{i-1}, both the link offset d_i and the link twist α_i will be equal to zero.

3. z_{i-1} **intersects** z_i**:** x_i is here chosen to be normal to the plane made by z_i and z_{i-1}. The direction of x_i is arbitrary.

Note that the quantities a_i and α_i are always constant for all i regardless if the joint is prismatic or revolute. θ_i is constant if the joint i is prismatic, and d_i is the i^{th} joint variable. Further d_i is constant if the joint i is revolute, and θ_i is the i^{th} joint variable.

3.4.3 The Raptor manipulators kinematics

The theory explained earlier must now be adapted to fit with the Raptor manipulator. Figure 3.9 shows the chosen reference frames for the arm according to the D-H convention and the right hand rule shown in Figure 3.8.

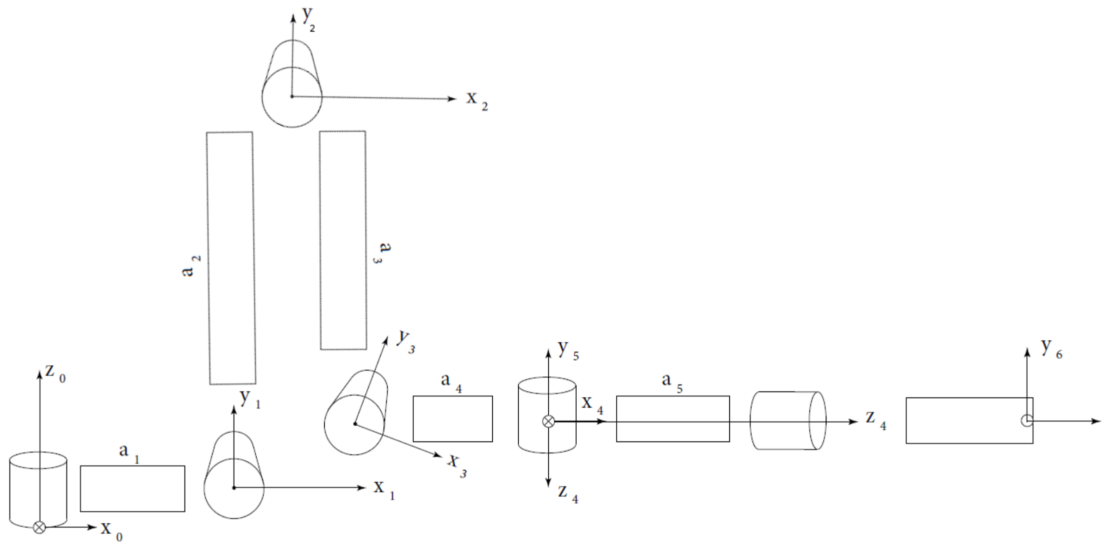


Figure 3.9: Raptor model with frames chosen according to D-H convention [9]

Further, the table below shows the chosen D-H parameters for each link in the manipulator.

Table 3.1: Denavit-Hartenberg parameters

Link #	a_i	α_i	d_i	θ_i
1	a_1	90°	0	θ_1
2	a_2	0°	0	θ_2
3	a_3	0°	0	θ_3
4	a_4	90°	0	θ_4
5	0	-90°	0	θ_5
6	0	0°	d_6	θ_6

Now it is an easy task to put these parameters into (3.18) to obtain all the homogeneous transformation matrices. Together they form the base of all the forward kinematic problems. Of course the size and the complexity of the manipulator will form these into big and complex equations, but still easy to solve since the variables are known.

$$A_1 = \begin{bmatrix} c\theta_1 & 0 & s\theta_1 & a_1c\theta_1 \\ s\theta_1 & 0 & -c\theta_1 & a_1s\theta_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} A_2 = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & a_2c\theta_2 \\ s\theta_2 & c\theta_2 & 0 & a_2s\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} c\theta_3 & 0 & s\theta_3 & a_3c\theta_3 \\ s\theta_3 & 0 & -c\theta_3 & a_3s\theta_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} A_4 = \begin{bmatrix} c\theta_4 & 0 & s\theta_4 & a_4c\theta_4 \\ s\theta_4 & 0 & -c\theta_4 & a_4s\theta_4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_5 = \begin{bmatrix} c\theta_5 & 0 & -s\theta_5 & 0 \\ s\theta_5 & 0 & c\theta_5 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_6 = \begin{bmatrix} c\theta_6 & -s\theta_6 & 0 & 0 \\ s\theta_6 & c\theta_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This gives the transformation matrix

$$\mathbf{T}_0^6 = A_1A_2A_3A_4A_5A_6 \tag{3.19}$$

$$= \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_x \\ r_{31} & r_{32} & r_{33} & d_x \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.20}$$

Where

$$\begin{aligned} r_{11} &= c_6(c_5(c_4(c_1c_2c_3 - c_1s_2s_3) + s_1s_4) + s_5(c_1c_2s_3 + c_1c_3s_2)) \\ &\quad -s_6(s_4(c_1c_2c_3 - c_1s_2s_3) - c_4s_1) \\ r_{21} &= c_6(c_5(c_4(c_2c_3s_1 - s_1s_2s_3) - c_1s_4) + s_5(c_2s_1s_3 + c_3s_1s_2)) \\ &\quad -s_6(c_1c_4 + s_4(c_2c_3s_1 - s_1s_2s_3)) \\ r_{31} &= s_4s_6(c_2s_3 + c_3s_2) \\ r_{12} &= -c_6(s_4(c_1c_2c_3 - c_1s_2s_3) - c_4s_1) \\ &\quad -s_6(c_5(c_4(c_1c_2c_3 - c_1s_2s_3) + s_1s_4) + s_5(c_1c_2s_3 + c_1c_3s_2)) \\ r_{22} &= -c_6(c_1c_4 + s_4(c_2c_3s_1 - s_1s_2s_3)) \\ &\quad -s_6(c_5(c_4(c_2c_3s_1 - s_1s_2s_3) - c_1s_4) + s_5(c_2s_1s_3 + c_3s_1s_2)) \\ r_{32} &= -c_5(c_2c_3 - s_2s_3) - c_4s_5(c_2s_3 + c_3s_2) \end{aligned}$$

$$
\begin{aligned}
r_{13} &= c_5(c_1c_2s_3 + c_1c_3s_2) - s_5(c_4(c_1c_2c_3 - c_1s_2s_3) + s_1s_4) \\
r_{23} &= c_5(c_2s_1s_3 + c_3s_1s_2) - s_5(c_4(c_2c_3s_1 - s_1s_2s_3) - c_1s_4) \\
r_{33} &= -c_5(c_2c_3 - s_2s_3) - c_4s_5(c_2s_3 + c_3s_2) \\
d_x &= a_1c_1 - d_6(s_5(c_4(c_1c_2c_3 - c_1s_2s_3) + s_1s_4) - c_5(c_1c_2s_3 + c_1c_3s_2)) \\
&\quad + a_4c_4(c_1c_2c_3 - c_1s_2s_3) + a_2c_1c_2 + a_4s_1s_4 - a_3c_1s_2s_3 + a_3c_1c_2c_3 \\
d_y &= a_1s_1 - d_6(s_5(c_4(c_2c_3s_1 - s_1s_2s_3) - c_1s_4) - c_5(c_2s_1s_3 + c_3s_1s_2)) + \\
&\quad a_4c_4(c_2c_3s_1 - s_1s_2s_3) + a_2c_2s_1 - a_4c_1s_4 - a_3s_1s_2s_3 + a_3c_2c_3s_1 \\
d_z &= a_2s_2 - d_6(c_5(c_2c_3 - s_2s_3) + c_4s_5(c_2s_3 + c_3s_2)) \\
&\quad + a_3c_2s_3 + a_3c_3s_2 + a_4c_4(c_2s_3 + c_3s_2)
\end{aligned}
$$

And s_i and c_i is a simplification of $sin(\theta_i)$ and $cos(\theta_i)$. Note that this matrix has the same form as shown in (3.12)

$$
\mathbf{T}_0^6 = \begin{bmatrix} \mathbf{R}_0^6 & \mathbf{O}_0^6 \\ 0 & 1 \end{bmatrix} \tag{3.21}
$$

Where $\mathbf{R}_0^6$ expresses the rotation of $o_0x_0y_0z_0$ with respect to $o_6x_6y_6z_6$. And $\mathbf{O}_0^6$ are coordinate vectors that describes the origin. In the attachments is a MATLAB-code that is doing all these calculations.

3.5 Dynamics

The kinematic equations describes the motions and positions of the manipulator. Dynamics on the other hand, describes how these motions occur based on the forces acting on the manipulator. These forces can be buoyancy, drag forces, added mass, waves, loads, collisions, current among others [5]. Many of them were presented and simulated for a dynamic model of an underwater manipulator in the project thesis [27].

In further development of a complete manipulator system it will be natural to have some kind of model-based control system included, like the one developed in the project thesis. There are many different ways to derive these equations such as the Euler-Lagranges method and the Newton-Euler method. To do this, one must know much more about the manipulators physical properties like the mass, inertia, drag coefficients and volume among others. A model based controller of the Raptor FFM has been developed in the master thesis [10], and can be used as a base if the work with this manipulator continues.

Chapter 4

Manipulator description

To be able to create a visualization of a manipulator arm as accurate as possible, one must have all the necessary information about its geometry, measurements and location of joints available. This chapter will present all the collected information that is relevant for this development.

The Raptor FFC Manipulator is designed for use in hostile or harsh environments, also under water. It consists of six revolute joints and a rotatable gripper at the end. It has a total mass of 75 kilograms including the base in air, and the mass under seawater due to buoyancy is 44 kilograms. The maximum lifting capacity is 227 kilograms and the maximum capacity is 91 kilograms if the arm is fully extracted.

See also the attached PDF data sheet and the manuals for more information.

4.1 3D model

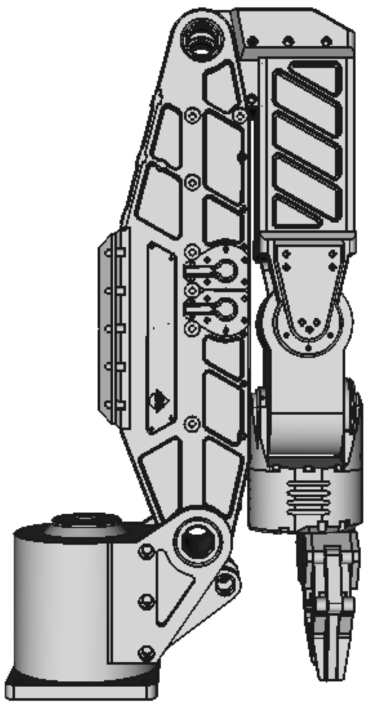


Figure 4.1: 3D model of the Raptor manipulator from Kraft Telerobotics

As mentioned earlier, the 3D CAD file was provided from the manufacturer. The model is the one seen in Figure 4.1. This was convenient because it now was possible to take accurate measurements to obtain more knowledge about the manipulators geometry than what was provided in its data sheets and user manual. It was a detailed CAD model with all the parts separated into sensible parts that was possible to separate and analyse.

Another advantage of having this provided is that the job of developing this model in a CAD-development software would have been a very time consuming task. The model would most likely have ended up being very inaccurate due to lack of important information, and because of the limited amount of time available. A manually created model could have resulted in deviations in the visualization compared to the real manipulator, and the goal is of course that this visualization will be as close to reality as possible.

4.2 Workspace and measurements

In Figure 4.2 and 4.3 drawings of the manipulators workspace and measurements are shown. They are found in the attached data sheets. The measurements are not detailed enough to create a perfect visualization, but they give an overview of the manipulator size. Also the drawings of the manipulators workspace gives a good image on how it is working and how the limitations restricts it from certain motions.

In Section 4.3 the limitations in each joint will be presented. By looking at the drawing of the workspace it is easy to see that all the links has restricted rotations. Compared to the spherical workspace shown in Figure 3.4 it is quite similar, which means that the Raptor manipulator has a restricted articulated configuration.

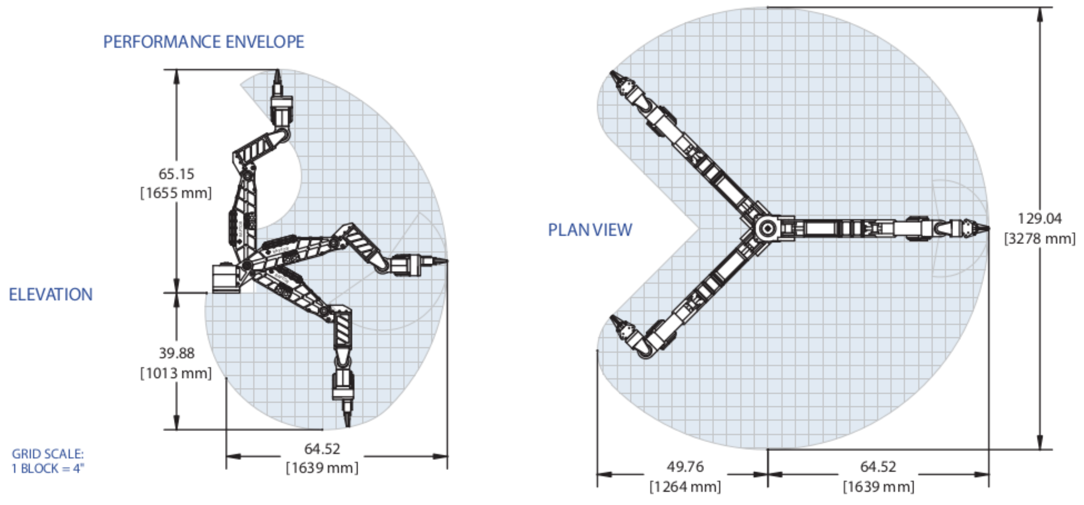


Figure 4.2: Raptor data sheet - workspace [13]

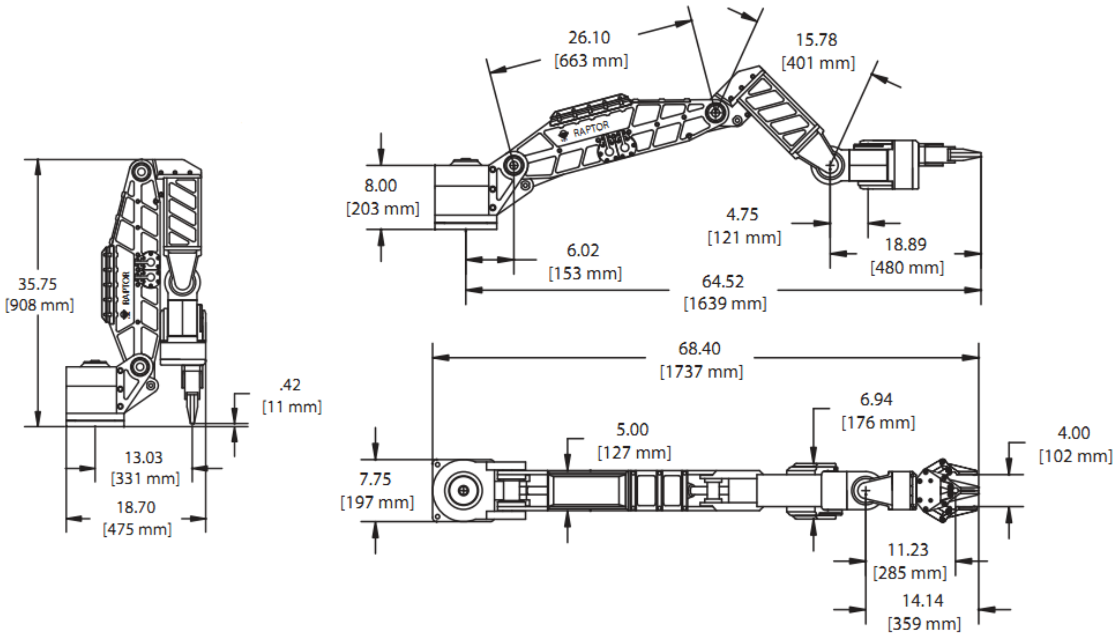
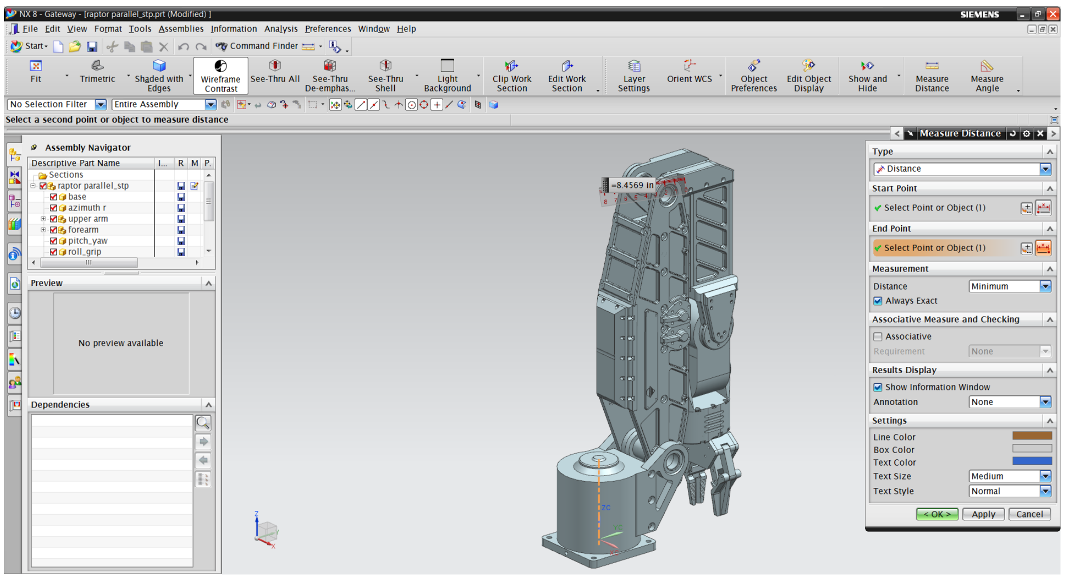


Figure 4.3: Raptor data sheet - Measurements [13]

To be able to visualize the manipulator 3D model it is necessary to have very accurate measurements so that the kinematics that will be developed later becomes perfect. These measurements was made i the program Siemens NX 8, which is an advanced software package used for task like design, analysis and manufacturing [32]. A screenshot of the interface can be seen in Figure 4.4. It has a very intuitive and accurate measurement tool that was used. The necessary measurements was the positions of the joints in the manipulator for each part, and the parts relative position compared to the global coordinate system.



Select a second point or object to measure distance

Measure Distance

Type: Distance

Start Point: Select Point or Object (1)

End Point: Select Point or Object (1)

Measurement
Distance: Minimum
[✓] Always Exact

Associative Measure and Checking
[ ] Associative
Requirement: None

Results Display
[✓] Show Information Window
Annotation: None

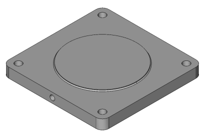
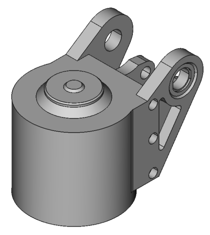
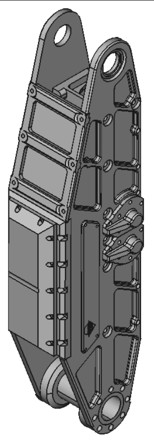
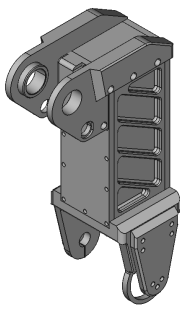
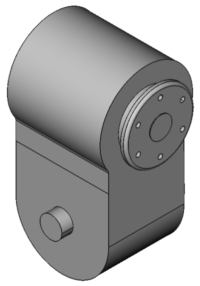
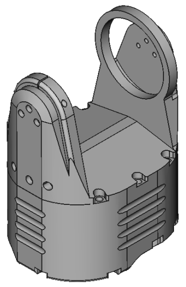
Settings
Line Color
Box Color
Text Color
Text Size: Medium
Text Style: Normal

< OK > Apply Cancel

Figure 4.4: Siemens software NX 8 that was used for measurements

Measurements that was made is shown in Table 4.1 below. The global position is the connection point from the parent link, and the joint position is where the joint is located relative to the previous joint.

Table 4.1: Raptor FFM measurements. All in [mm]

Base link		**Link-1**	
	Global position: (0, 0, 0) Joint position: (0, 0, 0) Note: Global position is parent position		Global position: (0, 0, 0) Joint position: (0, 0, 0)
Link-3		**Link-8**	
	Global position: (152.96, 184.13, 0) Joint position: (152.96, 184.13, 0)		Global position: (158.57, 847.06, 0) Joint position: (5.60, 662.93, 0)
Link-9		**Link-10**	
	Global position: (298.54, 471.36, 0) Joint position: (139.96, -375.69, 0)		Global position: (298.54, 350.74, 0) Joint position: (0.00,-120.62, 2.26)

4.3 Limitations

As mentioned in Section 4.2 the data sheets also provides more accurate limitations for each link. This is important knowledge in a visualization software, because it can be easier to detect errors. Many of the maximum angles will not be possible to obtain in certain conditions due to physical limitations. For instance if the end effector is colliding with one of the other links. In Figure 4.5 and Table 4.2

the different joint limitations are shown. These will be used when testing the visualization software to verify that the manipulator can move as it should within its workspace.

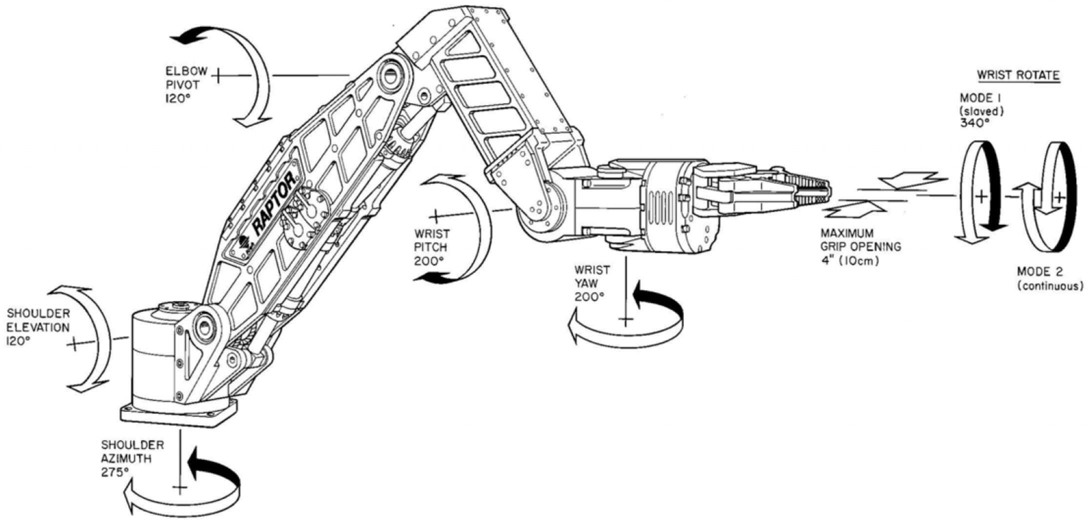


Figure 4.5: Rotation properties for Raptor [11]

Table 4.2: Raptor - Manipulator limits

Maximum range of motion	
Shoulder Azimuth	270°
Shoulder Elevation	120°
Elbow Pivot	120°
Wrist Pitch	200°
Wrist yaw	200°
Wrist Rotate (Slaved mode)	340°
Wrist rotate (Continuous)	0-40 [rpm]
Jaw Opening (parallel acting)	100 [mm]
Jaw opening (intermeshing)	220 [mm]

Chapter 5

System set-up

The target in this thesis is to develop a fully functional visualization system for the Raptor Force Feedback Manipulator. This leads to a variety of different challenges. Choice of visualization tool, software, hardware set-up and the connection between hardware and software is some of them. There are many ways to solve challenges like these. In this chapter the whole process all the way from the beginning of the thesis to the final result will be documented and explained. This work can be, and have been, very time consuming and it is therefore important to have detailed documentation both for further work and for readers in the future. Hopefully, after reading this report, time can be spent on understanding and further development of the system rather than challenges regarding system set-up and software.

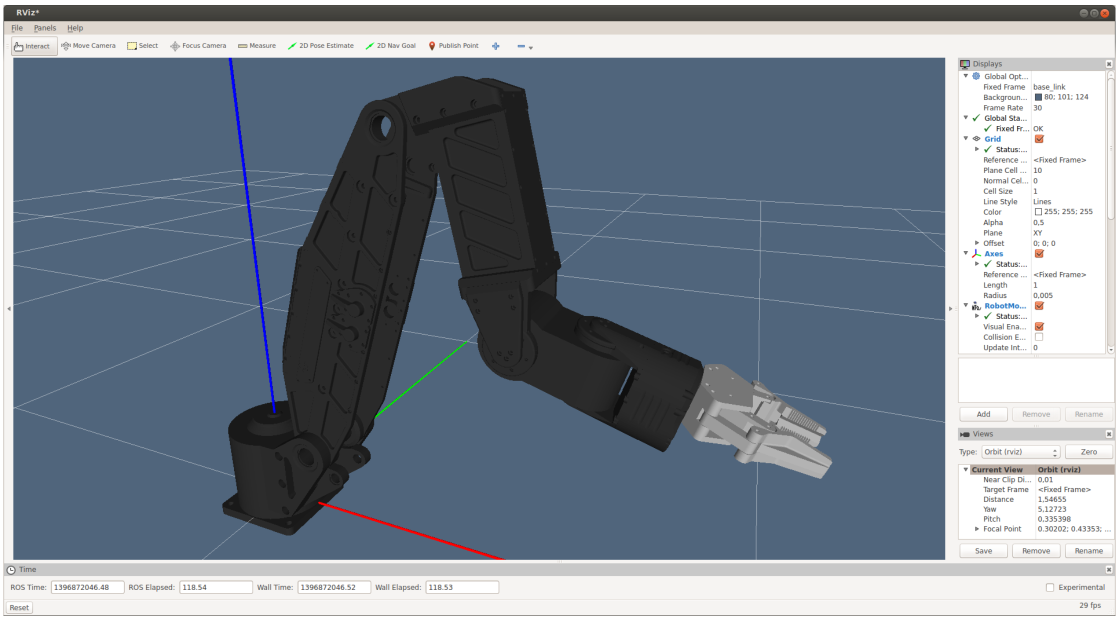
5.1 Software selection

5.1.1 Robot Operation System (ROS)

In the project thesis [27] it was developed a 3D visualization tool for a 2-link manipulator from scratch using the Python Visual package, VPython. The reason why Python was used for this was to have a solution with many opportunities and without limitations that a pre-made software might have. It was also simpler to develop a customized user interface designed for this specific problem. In this thesis the problem has been extended to a more complex manipulator with more details and functionality. As a result of this, the requirements to the software will also be increased. It is possible to do all the software development alone, however an open source software library aimed to develop robot applications will most likely make the development and further work a lot easier. Based on this, it was decided not to develop an application from scratch, but rather use an already existing software library. The software that was found most suitable for this task was the ROS framework [21].

The Robot Operating System is a set of software libraries, tools, and conventions designed specifically for robot application development. It has not been used much for underwater robotics yet. However, there are no reasons why it should not be suited for use in underwater robotics as well. The tools varies from developer tools, mathematical tools, drivers and visualization tools. Its main target is to provide everything that is needed in a robotic project. Also it is open source, which means that it is free for commercial and research use.

The most suitable ROS package for this thesis is the rviz package, which is a visualization package for robots. It has many opportunities for both visualization, monitoring and motion planning. It gives the user a huge amount of possibilities for solving any problems that can occur in a robotic task. The rviz interface with the Raptor model included is shown in Figure 5.1.



Global Opt...	
Fixed Frame	base_link
Backgroun...	80; 101; 124
Frame Rate	30
Global Sta...	
Fixed Fr...	OK
Grid	✓
Status:...	
Reference ...	<Fixed Frame>
Plane Cell ...	10
Normal Cel...	0
Cell Size	1
Line Style	Lines
Color	255; 255; 255
Alpha	0,5
Plane	XY
Offset	0; 0; 0
Axes	✓
Status:...	
Reference ...	<Fixed Frame>
Length	1
Radius	0,005
RobotMo...	✓
Status:...	
Visual Ena...	✓
Collision E...	☐
Update Int...	0

Type: Orbit (rviz)

Current View	Orbit (rviz)
Near Clip Di...	0,01
Target Frame	<Fixed Frame>
Distance	1,54655
Yaw	5,12723
Pitch	0,335398
Focal Point	0.30202; 0.43353; ...

ROS Time: 1396872046.48 ROS Elapsed: 118.54 Wall Time: 1396872046.52 Wall Elapsed: 118.53

Figure 5.1: The rviz ROS package. Here with the Raport Force Feedback Manipulator visualized

5.1.2 3D modelling software

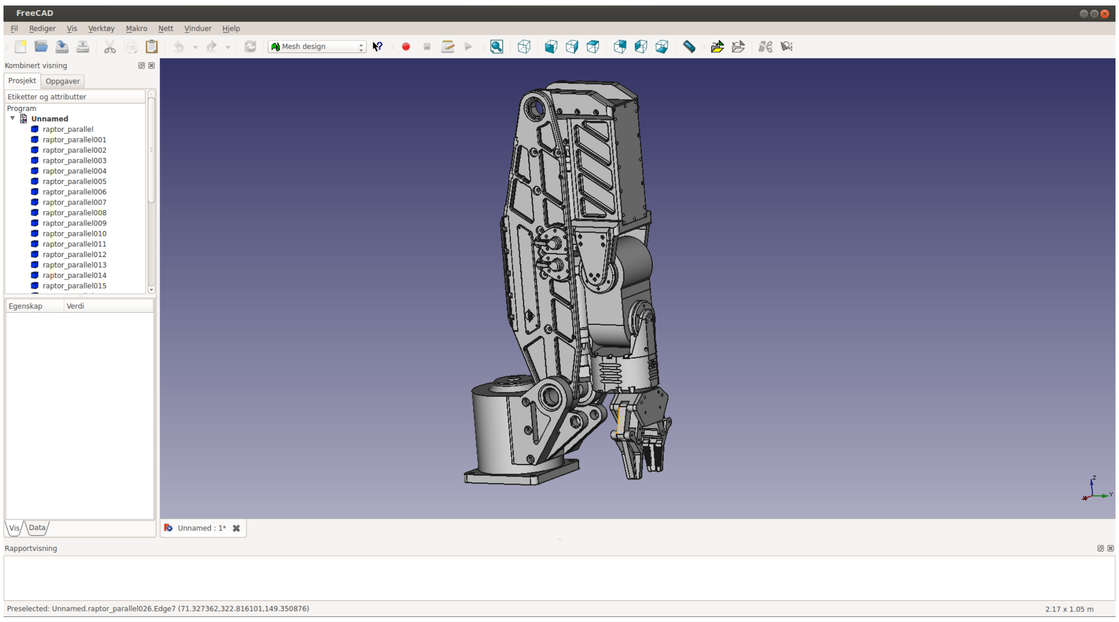
When working with 3D-visualization it is beneficial to have a detailed 3D model. Since today's 3D-software solutions allows extremely good and detailed 3D models, one is able to create visualizations that is very close to the reality. This increases the system quality and its appearance, and can make it easier for the operator that is manoeuvring the visualized manipulator because of its close to real appearance. 3D modelling is a lengthy process, especially if the model has many details. The Raptor FFM has many details and it would have been very time consuming to create a model from scratch. Since the original 3D model was provided from Kraft TeleRobotics, there wasn't found any reason not to use this model in the visualization. However, the model does not work out of the box. It has to be prepared for visualization, meaning that all the parts of the model must be separated so that they can be assigned to their associated reference frame. To do this a combination of the programs NX 8 by Siemens PLM Software [32], FreeCAD[31] and the ROS package rviz [22] was used.

5.2 Software set-up

With a 3D model available and the software chosen, it is time to prepare the model for visualization so that it is ready for connection to a real manipulator. The aim is to have a software that is complete and ready for connection to a physical system without any new serious issues. The first step in the process of developing this software is to split up the existing 3D model into pieces that can be connected in a kinematic chain for visualization.

5.2.1 Preparing the 3D model

The 3D model files that was provided from the manufacturer was formatted in a format called STEP format (Standard for the Exchange of Product model data). The STEP format is a widely used CAD file format. It is used to share 3D models between users regardless of witch CAD-software that is being used. Rviz does not support visualization of STEP-files directly. Therefore it is necessary to convert the model into a supported mesh format, in this case a STL (STereoLithography) file format. To do this the FreeCAD software that is shown in Figure 5.2 was used.



Preselected: Unnamed.raptor_parallel026.Edge7 (71.327362,322.816101,149.350876)   2.17 x 1.05 m

Figure 5.2: FreeCAD 3D modeling software

FreeCAD is a free Open Source 3D CAD modelling software. It is developed as a tool for mechanical engineering and product design. However, it can also be used in many other disciplines as well. The software runs on both Windows, Mac and Linux operating systems, and is therefore very flexible to use for developers that is running on different operating systems. Also it has a big user community so that questions and challenges can be posted and discussed with a huge amount of FreeCAD users around the world. [31]

First the parts must be separated into pieces that describes each link in the manipulator arm. Fortunately the CAD-model was already separated into categories so all that is needed is to identify the different parts of the manipulator and separate it from the rest of the model. An example is shown in Figure 5.3 where the upper arm is separated from the rest of the model. After this, the part can be saved in the desired STL format so that it is ready to be imported into rviz.

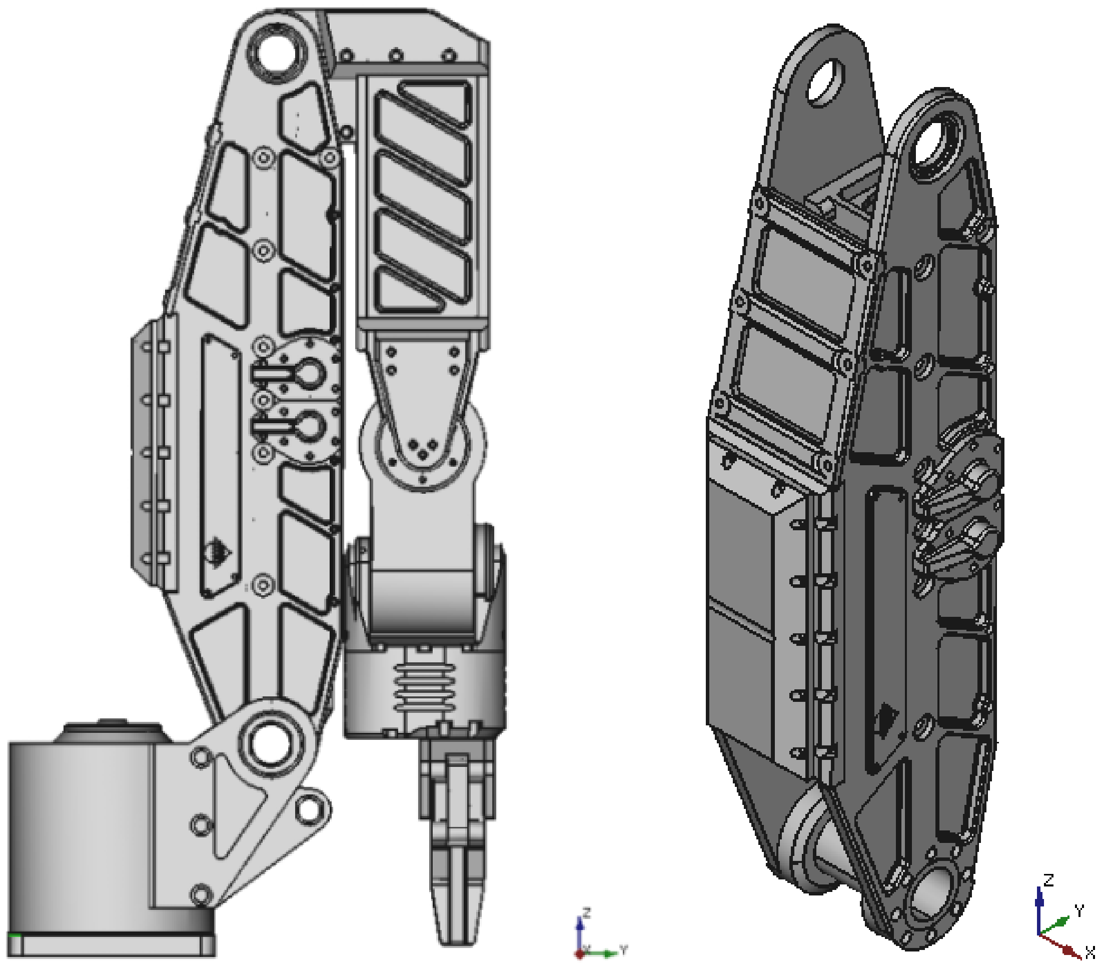


Figure 5.3: Left: the whole model. Right: a separated piece of the manipulator

The separated parts was as listed in Table 4.1. Their names are Base, Azimuth, Upper arm, Fore arm, Wrist Pitch, Wrist yaw and gripper. With these links separated it is possible to visualize the whole manipulator since these are the main parts that is connected through the joints. Hydraulic parts are also included in the original model, but these are not included in the visualization model. This is for simplification, and because including them will have little effect on the end result.

5.2.2 Develop the kinematic chain for rviz

The next step in the process is to describe how these parts are connected in the kinematic chain, so that rviz knows how to determine the kinematics of the arm. To do this it is necessary to make a URDF (Unified Robot Description Format), which is an XML (Extensible Markup Language) format for representing robot models. In this file all the information about the parts and joints are described, so that it can be loaded into rviz [17]. All used information about the manipulator is what was found in Chapter 4.

To explain how a URDF-file work, a simplified excerpt of the developed Raptor URDF is shown below. This part of the file describes the two first links in the manipulator. It defines where they are located and how they are connected to each other.

Code 5.1: URDF-file excerpt

```xml
<robot name="raptor">
<link name="base_link">
    <visual>
        <geometry>
            <mesh filename="file://STL-files/base.stl" scale="0.001 0.001
                0.001"/>
        </geometry>
        <origin xyz="0 0 0.0" rpy="0 0 -${pi/2}" />
        <material name="gray0">
            <color rgba="${color}"/>
        </material>
    </visual>
</link>

<link name="link2">
    <visual>
        <geometry>
            <mesh filename="file://STL-files/link-2.stl" scale="0.001 0.001
                0.001"/>
        </geometry>
        <material name="gray0">
            <color rgba="${color}"/>
        </material>
        <origin xyz="0 0 0" rpy="0 0 -${pi/2}" />
    </visual>
</link>

<joint name="joint0" type="revolute">
    <parent link="base_link"/>
    <child link="link2"/>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <axis xyz="0 0 1" />
    <limit effort="0" velocity="1.0" lower="-${(135/180)*pi}" upper="${(135
        /180)*pi}" />
</joint>
...
</robot>
```

First of all the `robot` tag is defined, with the name of the robot typed in.

Code 5.2: URDF: Define robot

```
<robot name="raptor">
```

Then the links are introduced with the link-tag. In this tag all the information about the link is provided. Its visual properties like geometry, origin and colour is specified. Also the dynamic characteristics of the part can be specified here, but there are no need for dynamic properties in this thesis.

Code 5.3: URDF: `Base_link`

```
<link name="base_link">
    <visual>
        <geometry>
            <mesh filename="file://STL-files/base.stl" scale="0.001 0.001
                0.001"/>
        </geometry>
        <origin xyz="0 0 0.0" rpy="0 0 -${pi/2}" />
        <material name="gray0">
            <color rgba="${color}"/>
        </material>
    </visual>
</link>
```

Note that some of the definitions is given as variables, like the color tag for the `base_link`. The rgba color given as `${color}` where this variable is defined in the beginning of the document. This is called xacro, which is a way to construct shorter and more readable XML files by using macros that expand to larger XML expressions [23]. One can also see that it is possible to do math operations like here where the orientations is defined as a fraction of π (`pi`).

Likewise, `link2` is defined. Then to connect the two links together the first joint, `joint0`, is created.

Code 5.4: URDF: `Joint0`

```
<joint name="joint0" type="revolute">
    <parent link="base_link"/>
    <child link="link2"/>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <axis xyz="0 0 1" />
    <limit effort="0" velocity="1.0" lower="-${(135/180)*pi}" upper="${(135
        /180)*pi}" />
</joint>
```

The parent link is of course the `base_link`, and the child link is `link2`. This is because movement of the `base_link` will affect `link2`, but movement in `link2` will not affect the `base_link`. It continues like this in the URDF-file until all the links and joints are defined. A complete connection chart based on the URDF-file is shown in Figure 5.4. The complete URDF is to be found in the attachments.

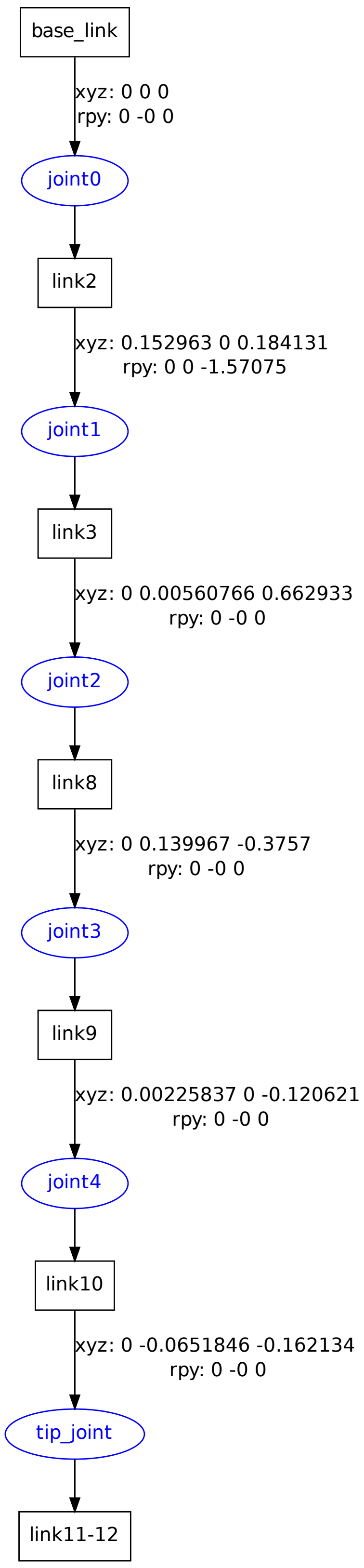


Figure 5.4: Connection chart from Raptors URDF-file

5.2.3 rviz set-up

As known rviz is, among many other things, a tool for visualization of input data. And as long as it knows the kinematic chain of the robot it can visualize the incoming signals. It works out of the box as long as all the inputs are correct. This means that by loading the Raptor URDF-file it will work immediately. However, there are some additional features that should be added when rviz is starting up.

In Code 5.5 below is the developed launch file for this thesis. It is separated into four different parts. The first part is where the parameters are set, like the project-file path and what model to load, before the user interfaces are starting up with these parameters. Next is where the `joint_state_publisher` and topics are defined. A topic is the channel where ROS is communicating with the different applications. In this case the topic name is `raptor`, meaning that this is the topic that the visualization will receive its signals from. How these states are being published and generated will be explained in Section 5.3.3. Further the `robot_state_publisher` is introduced. This publisher connects the states received from the `joint_state_publisher` and applies it to the visualization.

Last part is the camera set-up and start-up procedures. It can be loaded if it is connected to the computer. If it is not, this part should not be included in the launch-file. The camera will be described more in Section 5.2.6.

Code 5.5: Launch-file

```xml
<launch>
    <!--Start model and gui-->
    <arg name="path" value="/home/uav/Desktop/Robot/" />
    <arg name="model" value="$(arg path)urdf/raptor-mesh.urdf.xacro" />
    <arg name="xacrofile" value="/opt/ros/hydro/share/xacro/xacro.py" />
    <arg name="gui" default="True" />
    <param name="robot_description" command="$(arg xacrofile) '$(arg model)'"
        />
    <param name="use_gui" value="$(arg gui)"/>
    <node name="rqt_gui" pkg="rqt_gui" type="rqt_gui" />
    <!--Load and connect correct topic-->
    <node name="joint_state_publisher" pkg="joint_state_publisher" type="
        joint_state_publisher">
        <rosparam param="source_list">[raptor]</rosparam>
    </node>
    <!--Load state publisher-->
    <node name="robot_state_publisher" pkg="robot_state_publisher" type="
        state_publisher" />
    <!--Load Camera-->
    <node name="usb_cam" pkg="usb_cam" type="usb_cam_node" output="screen" >
            <param name="video_device" value="/dev/video0" />
            <param name="image_width" value="800" />
            <param name="image_height" value="600" />
            <param name="pixel_format" value="mjpeg" />
            <param name="camera_frame_id" value="usb_cam" />
            <param name="io_method" value="mmap"/>
    </node>

</launch>
```

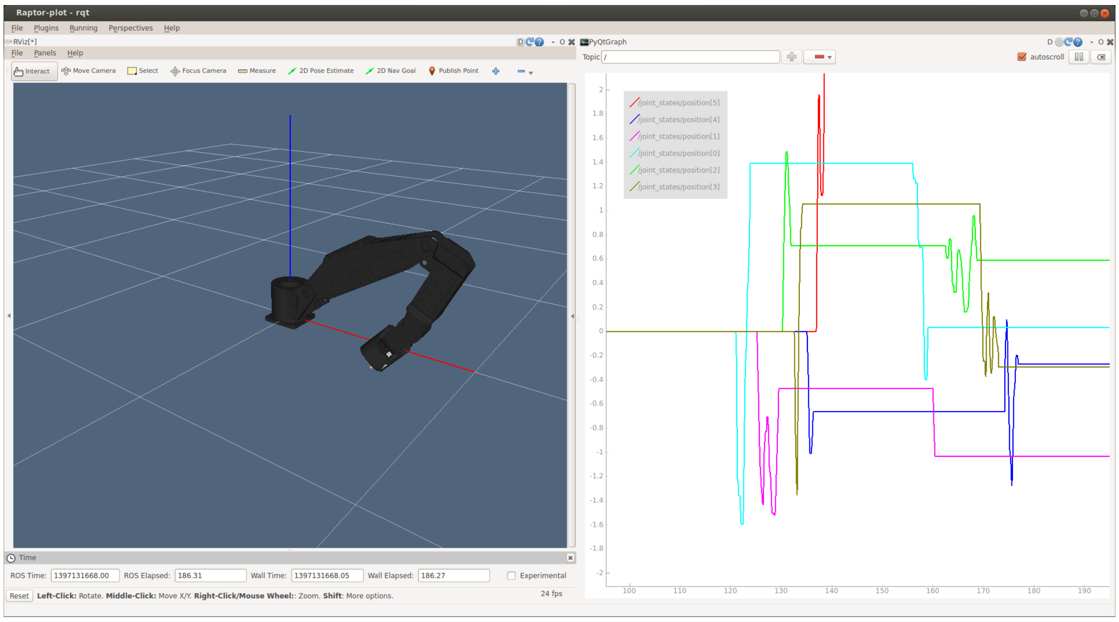
5.2.4 Navigation and use of rviz

Now that the software has been prepared, it is time to test and explain its functionality. To start up the system one must navigate to the root folder of the project in the terminal and execute the launch file launch as shown in Code 5.6 below.

Code 5.6: Start-up command

```
roslaunch "/Robot/launch/display_raptor_mesh.launch"
```

In the launch file it is defined that the `rqt_gui` package is to be opened. This is a package that allows the user to have multiple widgets running at the same time in the same window. It makes it possible to customize the user interface by adding more widgets in the program such as state plotting, and cameras. In Figure 5.5 the window that is appearing when running the launch file is shown.


Raptor-plot - rqt
File Plugins Running Perspectives Help
RViz[*]
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Nav Goal Publish Point
PyQtGraph
Topic /
autoscroll
/joint_states/position[5]
/joint_states/position[4]
/joint_states/position[1]
/joint_states/position[0]
/joint_states/position[2]
/joint_states/position[3]
Time
Experimental
Reset
24 fps

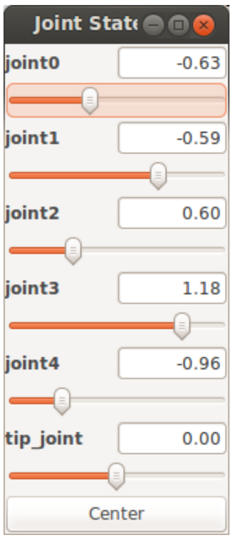
ROS Time: 1397131668.00 ROS Elapsed: 186.31 Wall Time: 1397131668.05 Wall Elapsed: 186.27

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift**: More options.

Figure 5.5: The rqt user interface with rviz and plotter

In this interface more features can easily be added and adjusted as desired by the user. As an example it has in Figure 5.5 been added a plot of all the joint states.

Since there is no real input signals at this moment it is necessary to have a manual state publisher to see how the visualization is working. Fortunately, this is an already existing feature in the ROS package. As seen in Code 5.5 the `joint_state_publisher` is loaded when launching the software with `gui` value set to True. This is a handy interface tool for changing link states easily just by dragging the marker that defines the joint state to the desired state. This way it is easy to test the visualization and search for errors in the URDF-file. The interface of this publisher is shown in Figure 5.6.

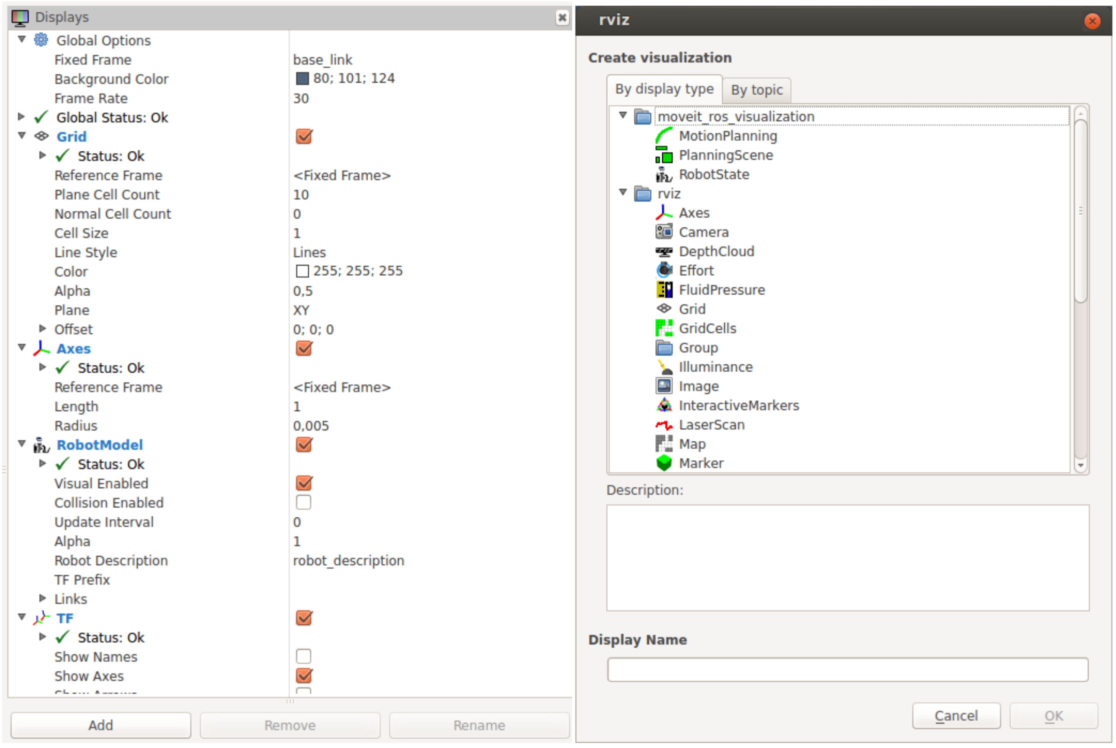


Joint State

joint0	-0.63
joint1	-0.59
joint2	0.60
joint3	1.18
joint4	-0.96
tip_joint	0.00

Figure 5.6: The joint state publisher

The visualization window can easily be customized to fit the users needs and wishes. Just by checking and unchecking the checkboxes that controls the different features it can behave and look as desired. New displays and features is also simple to add, and some of these can be seen in Figure 5.7.



Displays

Global Options	
Fixed Frame	base_link
Background Color	■ 80; 101; 124
Frame Rate	30
✓ Global Status: Ok	
Grid	✓
✓ Status: Ok	
Reference Frame	<Fixed Frame>
Plane Cell Count	10
Normal Cell Count	0
Cell Size	1
Line Style	Lines
Color	□ 255; 255; 255
Alpha	0,5
Plane	XY
Offset	0; 0; 0
Axes	✓
✓ Status: Ok	
Reference Frame	<Fixed Frame>
Length	1
Radius	0,005
RobotModel	✓
✓ Status: Ok	
Visual Enabled	✓
Collision Enabled	□
Update Interval	0
Alpha	1
Robot Description	robot_description
TF Prefix	
Links	
TF	✓
✓ Status: Ok	
Show Names	□
Show Axes	✓

rviz

Create visualization

moveit_ros_visualization
- MotionPlanning
- PlanningScene
- RobotState

rviz
- Axes
- Camera
- DepthCloud
- Effort
- FluidPressure
- Grid
- GridCells
- Group
- Illuminance
- Image
- InteractiveMarkers
- LaserScan
- Map
- Marker

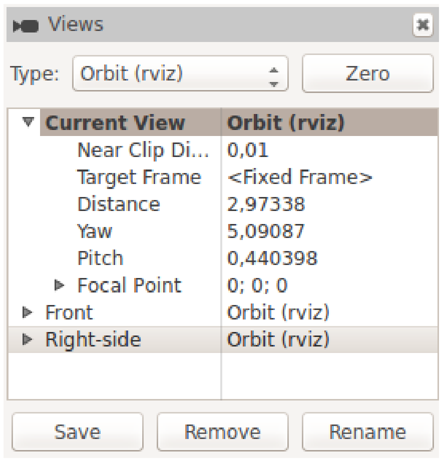
Description:

Display Name

Figure 5.7: rviz - Settings and displays

Different views can be defined and stored for use later as shown in Figure 5.8. Navigation is also very easy with a 3-button mouse; left-, right- and scroll-button [22]. The mouse-buttons have the following functions

- **Right mouse button:** Click and drag to zoom
- **Left mouse button:** Click and drag to rotate
- **Scrollwheel:** Zoom in and out
- **Scroll-button:** Click to move focal point



Current View	**Orbit (rviz)**
Near Clip Di...	0,01
Target Frame	<Fixed Frame>
Distance	2,97338
Yaw	5,09087
Pitch	0,440398
Focal Point	0; 0; 0
Front	Orbit (rviz)
Right-side	Orbit (rviz)

Figure 5.8: Stored views in rviz

A very simple tool-line shown in Figure 5.9 makes it easy to change between modes and do measurements among others.

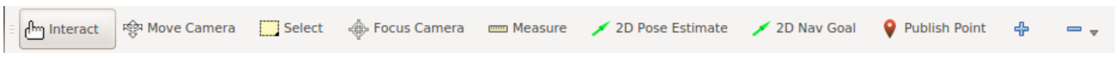


Figure 5.9: Tool-line in rviz

5.2.5 Plotter

As seen in Figure 5.5 a plotter window is added to the interface. It plots the `/joint_states/position` topic where the joint states are published. With this tool it is easy to keep control of the current states and to detect errors in the publisher. If one of the states suddenly freezes, it can be that one of the sensors has lost its signal, or that it is broken. And this should be possible to see in the graphs. It is also possible to go back in the log and see what has happened earlier. The window can be zoomed in and out using the mouse and the axes can be modified individually.

5.2.6 Camera

Figure 5.10: Camera used in the set-up

Even though this visualization software is made to help the operator during operations where the visibility is bad, a camera is of course a necessary equipment in the visualization. With this, it is easy to compare the visualization with what is seen on the camera. Also a camera gives opportunities that can increase the functionality of the visualization. For example to create a collision avoidance system. A calibrated camera can be used to detect objects that endangers the operation. In this task the camera is only included for practical reasons during testing.

5.3 Hardware set-up

A lab located at the Marine Technology Centre has been created for testing and development of the manipulator. Hydraulics, electrical transformers, control units and computers is located here. An overview of the lab is shown in Figure 5.11 below. It is a great advantage to have this lab available for implementation of the visualization system in the future. In this section, all the work done to prepare the software for connection to the real manipulator is described.

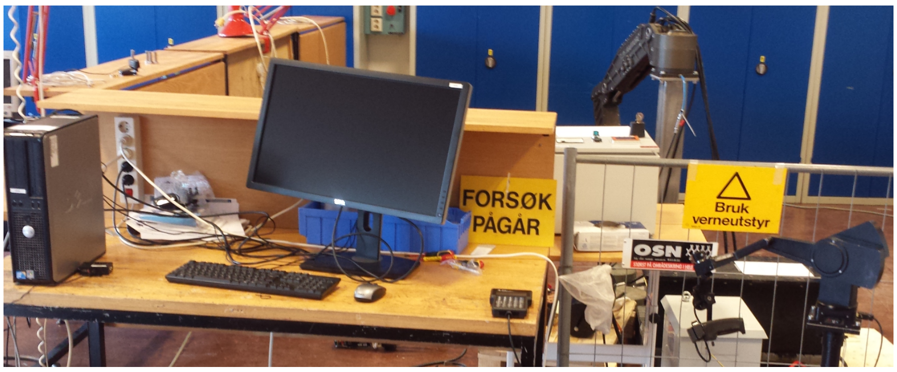


Figure 5.11: Lab-set-up overview at the Marine Technology Centre

5.3.1 Manipulator set-up

As seen in Figure 5.12 below, the simplified version of the system consists of a Control Unit that sends signals to a KMC 9100F Control Chassis. Further the orders are given to the servo driver module and the manipulator should perform the desired tasks. There are of course more components represented like hydraulics and electrical components. Some of them can be seen in Figure 5.11.

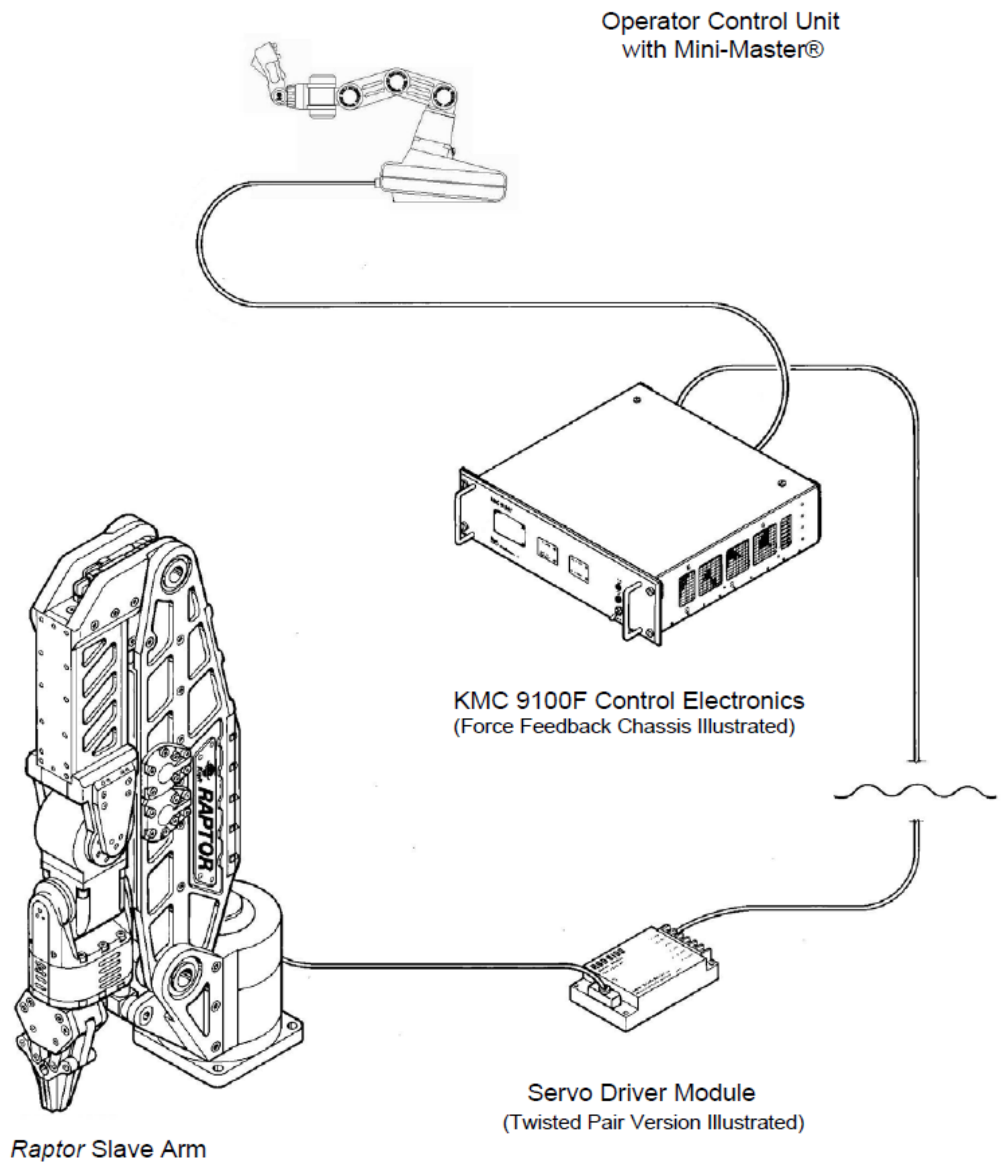

Operator Control Unit
with Mini-Master®
KMC 9100F Control Electronics
(Force Feedback Chassis Illustrated)
RAPTOR
Servo Driver Module
(Twisted Pair Version Illustrated)
Raptor Slave Arm

Figure 5.12: Raptor manipulator with control system [11]

The arm is mounted on a support structure so that it can work in the entire manipulator workspace. For safety reasons is the whole manipulator area fenced, and security procedures has been developed for the lab.

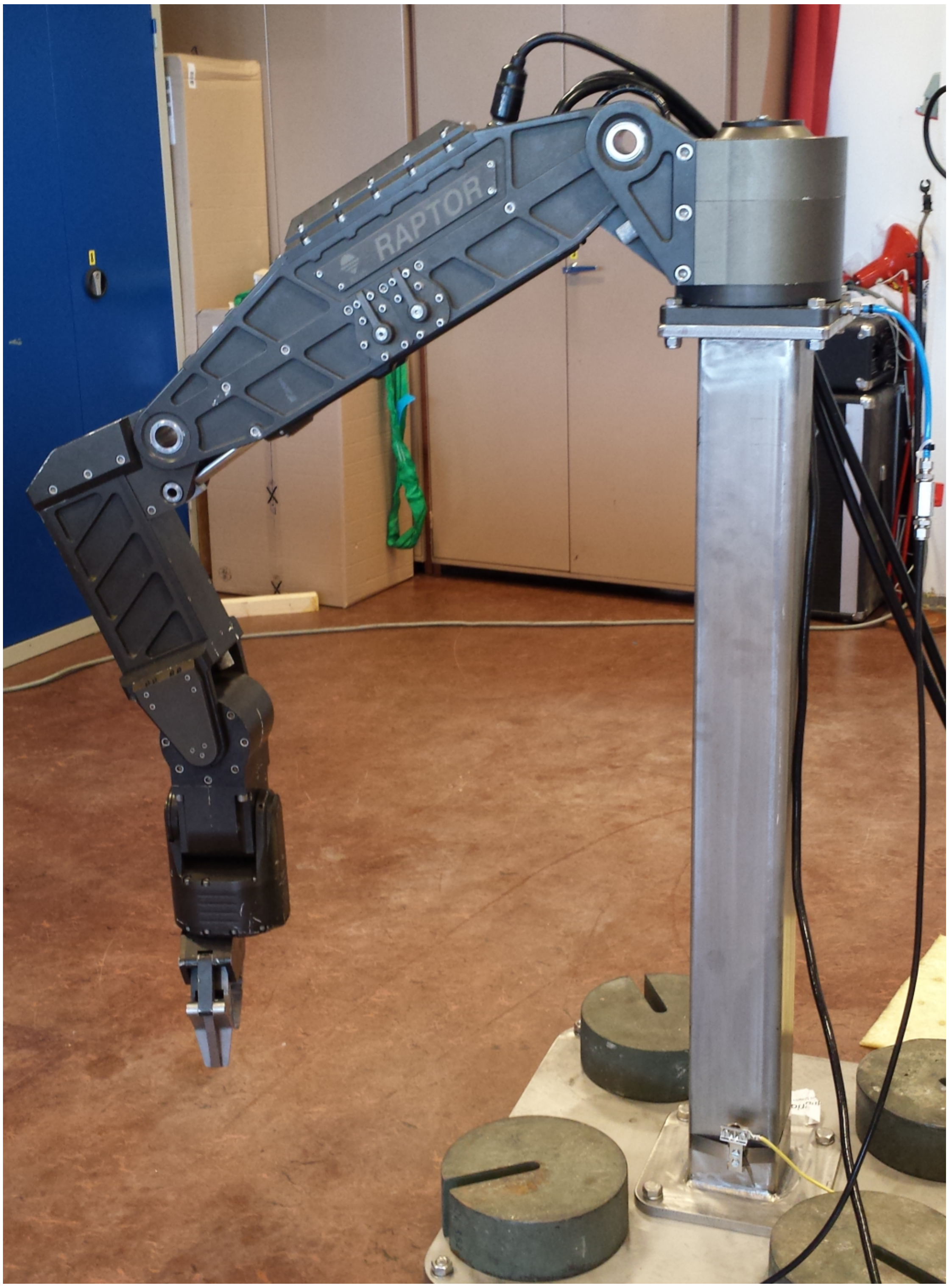


Figure 5.13: The manipulator mounted on the support structure

Since the manipulator was not up and running when the visualization software was developed, it was not possible to test the simulation with real signals from the manipulator. However, it is easy to test the visualization with simulated signals. This way we know that the software works as long as it gets the right input. Below is a staged scenario of how the manipulator together with the visualization will look like.

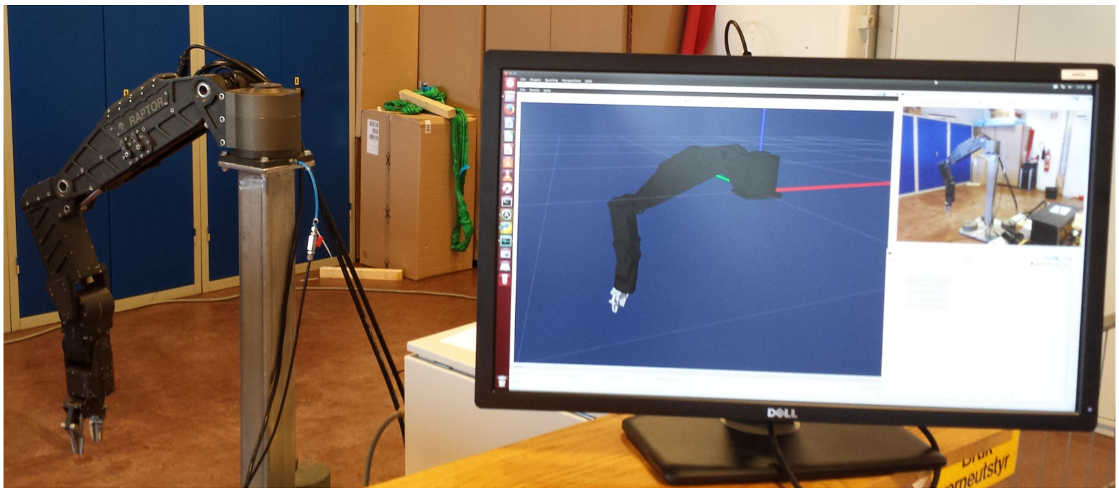


Figure 5.14: Overview of the software and manipulator

5.3.2 Signals

Now that the physical system is described, it is time to explain how the signals is planned to be fed into the software. Signals that is returning from the manipulator while it is controlled, is being transferred to the computer with the visualization software running. This is to be done with serial communication. The Raptor manipulator is using an architecture called RS-485, which is an updated version of RS-232. They are both widely used in robot solutions. For example, RS-485 is used in many devices on ROV SF 30K and ROV Minerva that is owned and operated by NTNU. The RS-485 standard is forwarding signals differentially, which gives the opportunity to send signals over longer distances. It is also experiencing less noise due to its balanced line structure [28].

Serial communication is preferred because it is easy to read serial data into Python which easily can publish the data to the visualization. The Python package used for this is called pySerial [18], and allows communication through the serial port directly in the Python code. It can be implemented simply by importing the `serial` package in the code.

Figure 5.15: pySerial API logo [18]

5.3.3 Programming for state publisher

The pySerial module should be imported into a python script so that the signals can be published. How this would look like will depend on the final connection, and this is therefore not discussed here. However, how data is being fed into to the visualization is important to explain because this is also how the simulations are carried out.

Below is a simple python code that is publishing states to the visualization.

Code 5.7: Joint-state publisher

```python
#!/usr/bin/python

from sensor_msgs.msg import JointState
import rospy
import math
pi = 3.1415
rospy.init_node('raptor', anonymous=True)
publisher = rospy.Publisher("raptor", JointState)
count = 0
count2 = count
while not rospy.is_shutdown():
    raptor = JointState()
    raptor.name.insert(0,"joint0")
    raptor.name.insert(1,"joint1")
    raptor.name.insert(2,"joint2")
    raptor.name.insert(3,"joint3")
    raptor.name.insert(4,"joint4")
    raptor.header.stamp = rospy.Time.now()

    sine    = math.sin(count)
    cosine  = math.cos(count2)
    val1 = sine*pi*0.75
    val2 = -(0.5 + 0.5*sine)*0.5*pi
    val3 = (0.5 + 0.5*sine)*0.5*pi
    val4 = (0.5 + 0.5*cosine)*0.5*pi
    val5 = 0.5*cosine*pi;

    raptor.position.insert(0,val1)  #range: 270 deg
    raptor.position.insert(1,val2)      #range: 120 deg
    raptor.position.insert(2,val3)  #(-(0.5+0.5*sine)*(120/180)*pi)
    raptor.position.insert(3,val4)
    raptor.position.insert(4,val5)

    publisher.publish(raptor)

    count = count+0.01
    count2 = count2+0.02

    rospy.sleep(0.01)
```

What this code does is to publish time varying states into the `raptor`-topic, where each joint is updated every 0.01 second. This topic is set to the main topic in the visualization through the launch file in Code 5.5, and this gives a real-time update of the 3D model as long as the publisher is running. The publisher is everything that is needed to carry out simulations of the visualization in rviz.

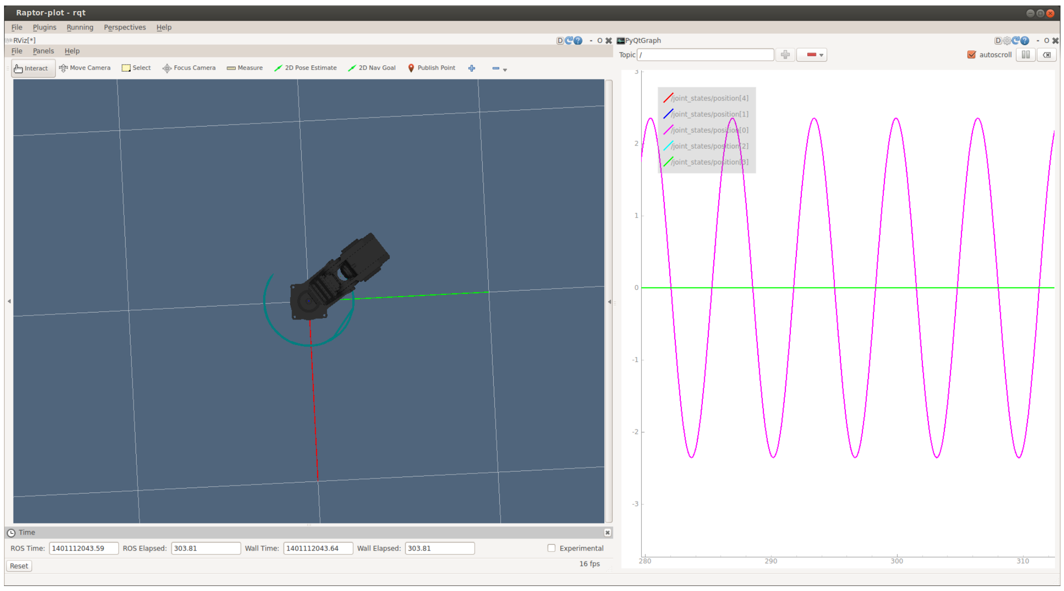
Chapter 6

Results

The software must be tested to verify that the visualization works as it is supposed to do. And the procedures for doing this is as mentioned in Section 5.3.3. Each joint is to be tested separately with a varying input within its limits shown in Table 4.2. As a last test, all joints are changed at the same time just as in a real situation. A tracer is added to the end of the wrist to see what line the end-effector is covering. This will determine if the end effector is moving in the desired pattern or not. It is important to make sure that there is no errors in the kinematics, so that the rotations propagate correctly through the whole arm.

6.1 Shoulder azimuth

Maximum range of motion is 270°. This will correspond to a rotation in the range $(-135°, +135°)$, or $(-\frac{3}{4}\pi, \frac{3}{4}\pi)[rad]$, around the initial position 0°. All other joints are set to 0°. A varying rotation with one cycle every third second is applied.



Topic: /

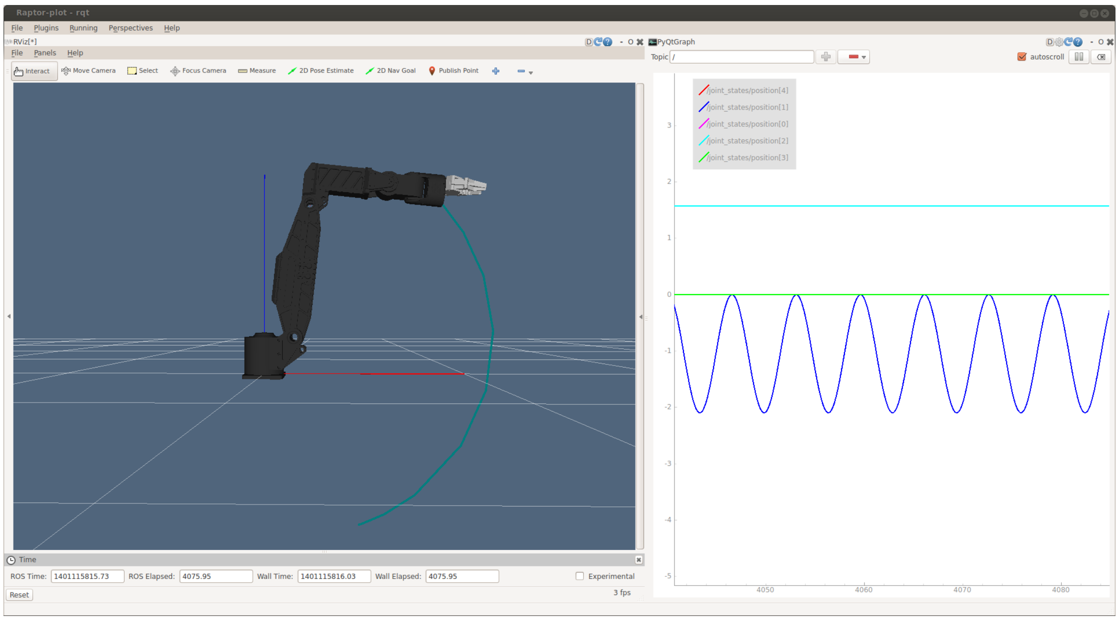
ROS Time: 1401112043.59 ROS Elapsed: 303.81 Wall Time: 1401112043.64 Wall Elapsed: 303.81

Figure 6.1: Test of shoulder azimuth

The result is as expected. There are no deviations or distortions during movement. Since this joint is located at the global centre $(0, 0, 0)$ this is no surprise.

6.2 Shoulder elevation

Maximum range of motion in this joint is 120°. This corresponds to a rotation in the range $(-120°, 0°)$ or $(-\frac{2}{3}\pi[rad], 0[rad])$, where 0° is the initial condition. The Elbow Pivot is set to an angle 90° since the Shoulder Elevation then can rotate within its whole range without any collisions with its own body. All other joints are set to 0°.



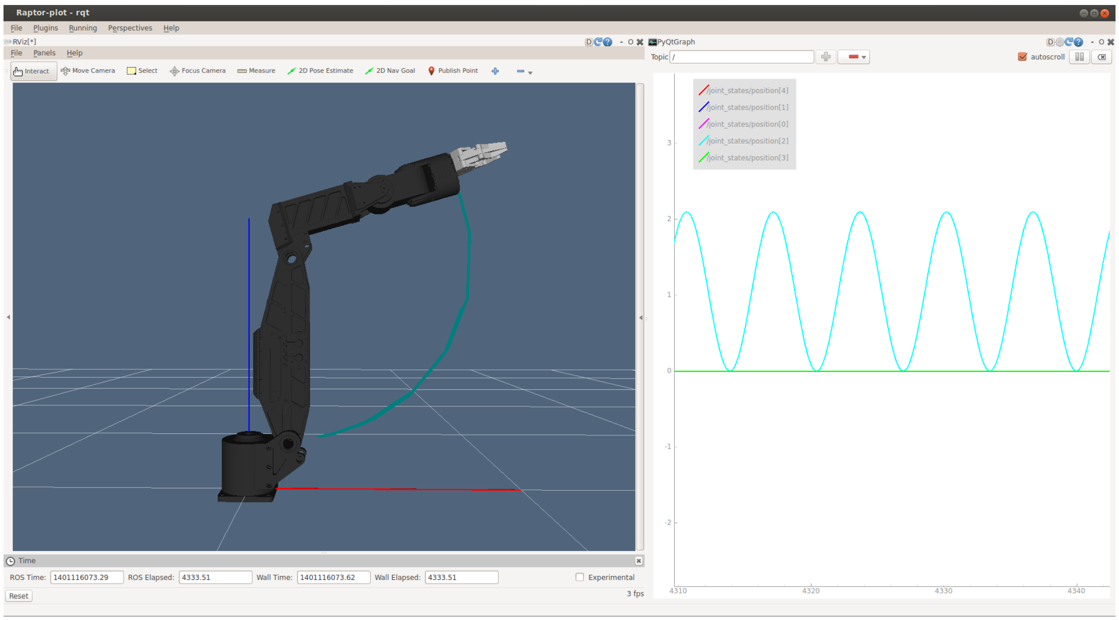
ROS Time: 1401115815.73 ROS Elapsed: 4075.95 Wall Time: 1401115816.03 Wall Elapsed: 4075.95

Figure 6.2: Test of shoulder elevation

Also here is the rotation conducted without deviations or distortions. Note that already in this joint the limitations in the manipulators theoretical workspace becomes visible. In theory, this manipulator could have moved within itself. This is obviously impossible in reality, and a detection procedure for this can be considered in further development.

6.3 Elbow Pivot

The maximum rotation for this joint is 120°. This corresponds to a rotation in the range $(0°, 120°)$ or $(0[rad], \frac{2}{3}\pi[rad])$, where 0° is the initial condition. All other joints are set to 0°.



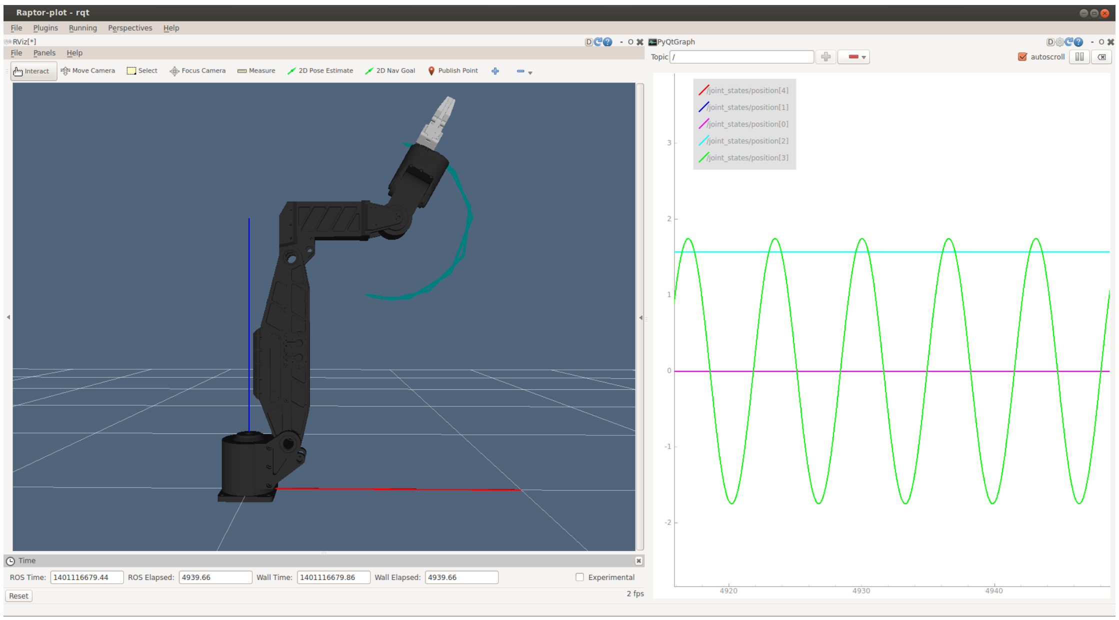
ROS Time: 1401116073.29 ROS Elapsed: 4333.51 Wall Time: 1401116073.62 Wall Elapsed: 4333.51

Figure 6.3: Test of Elbow Pivot

Again, everything works properly. This joint is important because it is the joint that gives the arm its biggest extent and therefore also fully extract the manipulator. The maximal horizontal distance this manipulator can reach is 1.64 meters according to Figure 4.2, and this is obtained when Elbow Pivot is rotated 120°.

6.4 Wrist Pitch

Maximum rotation is 200°. This corresponds to a rotation in the range $(-100°, 100°)$ or $(-\frac{5}{9}\pi[rad], \frac{5}{9}\pi[rad])$, where 0° is the initial condition. The Elbow Pivot is set to an angle 90° for the same reason as mentioned earlier. All other joints are set to 0°.



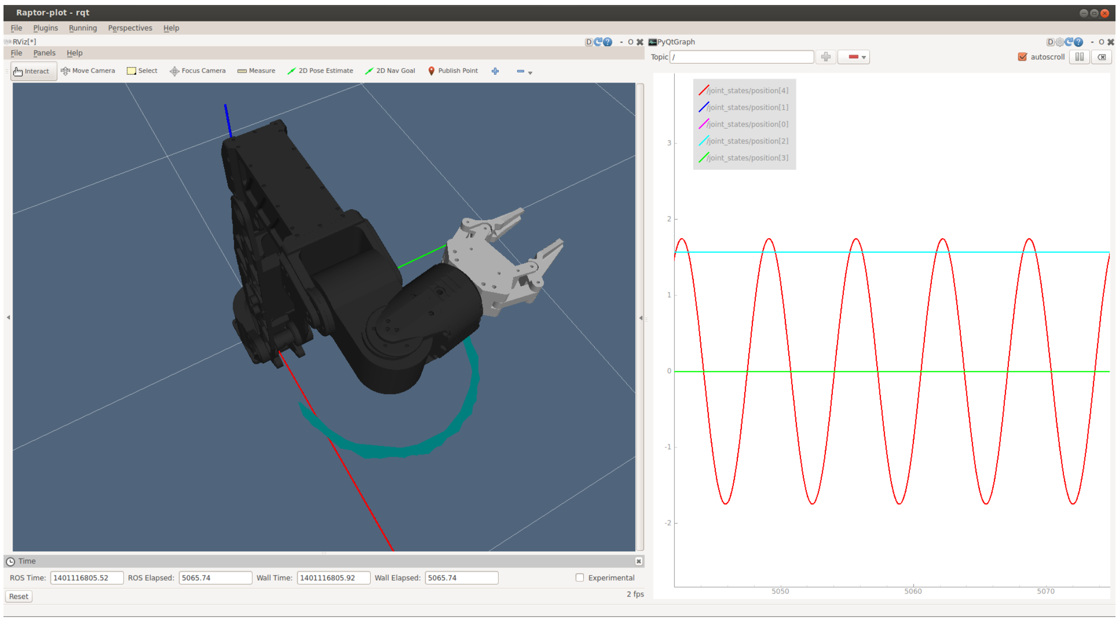
ROS Time: 1401116679.44 ROS Elapsed: 4939.66 Wall Time: 1401116679.86 Wall Elapsed: 4939.66 Experimental

Figure 6.4: Test of Wrist Pitch

No errors was found in this joint either. If the forearm is too close to the upper arm this link can collide with the upper arm. Its range of motion is therefore depending on how the Elbow Pivot is rotated.

6.5 Wrist Yaw

For this joint the rotation range is just like the Wrist Pitch, $(-100°, 100°)$ or $(-\frac{5}{9}\pi[rad], \frac{5}{9}\pi[rad])$. It rotates about the opposite axis from the previous joints. The gripper is also attached to this link. It can rotate in its whole range when all the other joints is in the initial position, but for convenience the Elbow Pivot is set to an angle 90° when running the simulation.



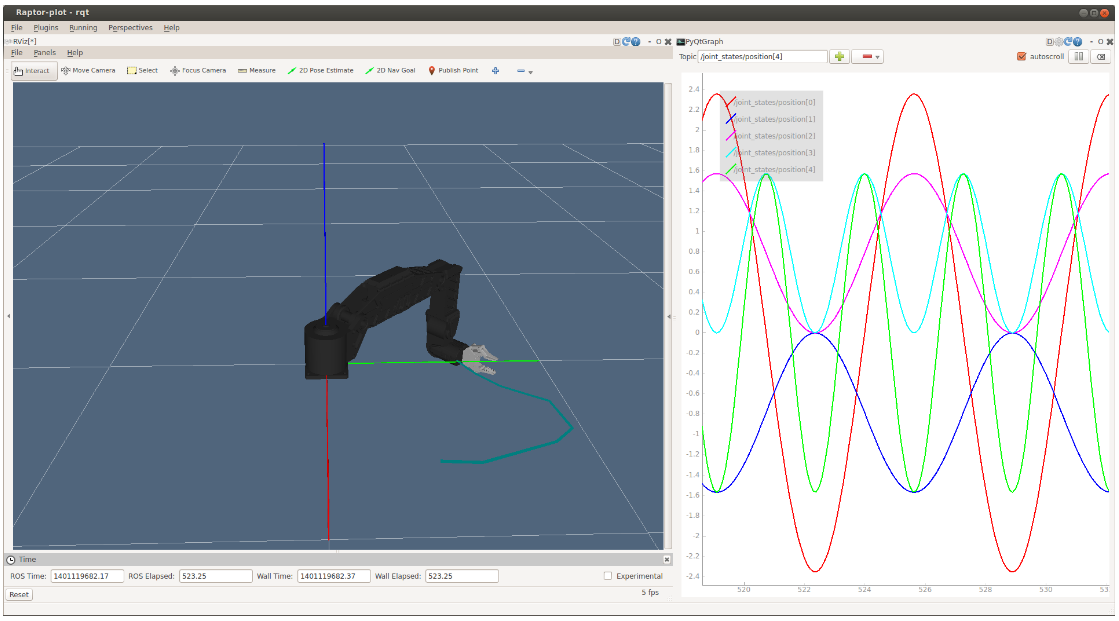
ROS Time: 1401116805.52 ROS Elapsed: 5065.74 Wall Time: 1401116805.92 Wall Elapsed: 5065.74

Figure 6.5: Test of Wrist Yaw

Also here is the joint rotating with no deviations or distortions. This means that the kinematic model implemented into rviz is correct, and the end-effector is correct as long as the state input is correct.

6.6 Rotation in all joints

To test that a situation with movement in all joints are working, a more complex simulation was done. This was done also to test that the software could handle changes in more than just one link at the time. To make sure that the movement was conducted with no collisions, all the joints except the Shoulder Azimuth, the Shoulder Elevation and the Wrist Yaw was varying with positive state angles.



Topic /joint_states/position[4]

ROS Time: 1401119682.17 ROS Elapsed: 523.25 Wall Time: 1401119682.37 Wall Elapsed: 523.25 Experimental

Figure 6.6: Test of motion in all joints

The tracer at the last link shows how the end-effector is changing through time. Simulations was executed with no errors whatsoever. This means that the system most likely is ready for real visualization connection. Even when the states where changed fast and in a big range, the software had no problem to run on a lightly equipped laptop. This is exactly what the software is supposed to do, and the results are therefore satisfying.

Chapter 7

Conclusion and further work

7.1 Conclusion

The objective with this thesis has been to develop and demonstrate a 3D visualization of a underwater robot manipulator arm. The arm was the Raptor Force Feedback Manipulator that is constructed for use in harsh environments such as underwater operations. This leads to a number of challenges regarding the software layout and the hardware robustness.

The first step was to develop and analyse the 3D model, and prepare it for use in a visualization application. Further, the kinematics of this arm had to be derived and implemented into the chosen visualization software. Also a full start-up structure and launch-procedure had to be developed. Procedures for start-up, connection and use also had to be produced. Explanation on how the physical system would work and look like also had to be included. In the end simulations had to be conducted to verify that the software was fully functioning.

A visualization software has been developed in the Robot Operating System framework which is a widely used software in robotic development, but not yet extensively used in underwater robotics. The ROS package used for this is called rviz. The manipulator has been inspected, the theory behind the kinematics has been explained, and a kinematic model has been developed. Further, the manipulator theory has been imported into the visualization software, together with its 3D model. A complete system for visualization including start-up and set-up procedures has been created, and the physical system design has been described and explained to prepare for the process of testing the system on a real manipulator.

Simulations have been executed and this verifies that the visualization works properly and that it is ready for connection to the manipulator-lab that has been established at the Marine Technology Centre. A camera located in the lab has also been implemented into the software to improve the visualization performance and to simplify the development process in the future.

The work with this thesis did not reveal any reasons not to use ROS for visualization of underwater robotics. The software is solid, stable and well equipped. It can be combined with other software solutions, customized by including appropriate packages, and be expanded by creating tailor made extensions for specific problems. Hopefully this thesis will help readers in the future to see the potential in the use of ROS for underwater robotic projects.

7.2 Further work

The visualization software developed in this thesis is designed for monitoring and visualization only. It has no controlling or security functionality whatsoever, and its only task is to provide the operator with necessary and important information that can help to improve and streamline the operation. The model itself should be good enough to be used in a real operation. However, there is room for improvement in the software solution. Firstly a "plug and play" functionality should be developed. This means that the software is fully up and running immediately after it has been started up. Now the visualization software and the state reader and publisher is separated, and must be started manually.

A system that detects obvious errors in the signals should also be implemented. This will help to determine whether the input is reliable or not. For example the limits of the rotations in each links are known. If these limits is being exceeded the operator should get a warning telling that something must be wrong.

The dynamics of the manipulator is not implemented in the software. This limits the possibility to use the software as a control system. However, this is possible to implement by using one of the many extensions available, and it is a natural next step in the process of developing an operation centre.

For further work there should also be a goal to have a fully functional controlling and monitoring system that can use the visual data provided from visual sensors to improve the operation itself. Such functions can be systems designed for collision avoidance, path planning, and feasibility analysis. The system should give the operator a full overview of the operation, the possibility to plan and test steps in the operation and act smart if unforeseen events should occur. A system that can read and analyse visual data and provide this information to the controlling system and the operator can be a great solution to many of the existing and upcoming challenges that the offshore industry is facing.

Bibliography

[1] Karim Abdel-Malek and Harn-Jou Yeh. Atlases of orientability for robotic manipulator arms. *INTERNATIONAL JOURNAL OF ROBOTICS AND AUTOMATION*, 15(4):189–205, 2000.

[2] Gianluca Antonelli. *Underwater robots.* Cham: Springer International Publishing, 2013. ISBN: 3319028766.

[3] Robert D. Christ and Robert L. Wernli Sr. *The ROV manual: a user guide for observation class remotely operated vehicles.* Butterworth-Heinemann, 2007. ISBN: 0750681489.

[4] Mari Carmen Domingo. An overview of the internet of underwater things. *Journal of Network and Computer Applications*, 2012.

[5] Odd Faltinsen. *Sea loads on ships and offshore structures*, volume 1. Cambridge university press, 1993. ISBN: 0521458706.

[6] Thor Inge Fossen. *Handbook of marine craft hydrodynamics and motion control.* John Wiley & Sons, 2011. ISBN: 1119991498.

[7] Pål Johan From, Jan Tommy Gravdahl, and Kristin Ytterstad Pettersen. *Vehicle-Manipulator Systems.* Springer, 2013. ISBN: 1447154622.

[8] Tony Hargreaves. Robot working envelopes. `http://thnet.co.uk/thnet/robots/25.htm`, 2013. Visited 09.06.2014.

[9] Morten Haugen. Modeling and control of rov manipulator, 2011. Project Thesis.

[10] Morten Haugen. Modeling and control of rov manipulator, 2012. Master Thesis.

[11] Kraft TeleRobotics Inc. *Raptor Manipulator System Manual, Part 1.* Kraft TeleRobotics, 2012.

[12] Kraft TeleRobotics Inc. *Raptor Manipulator System Manual,Part 2.* Kraft TeleRobotics, 2012.

[13] Kraft TeleRobotics Inc. Raptor remotely operated force feedback manipulator arm system. `http://krafttelerobotics.com/products/raptor.htm`, 2014. Visited 07.06.2014.

[14] Steven Michael LaValle. *Planning algorithms.* Cambridge university press, 2006. ISBN: 0521862051.

[15] NTNU. Introduksjon til undervannsteknikk. `http://ivt.ntnu.no/imt/systemer/emner/uvtek/`, 2014. Visited 22.01.2014.

[16] NTNU. Rov minerva - research vessel - ntnu. `http://www.ntnu.edu/marine/minerva`, 2014. Visited 01.03.2014.

[17] Jason M. O'Kane. *A Gentle Introduction to ROS*. CreateSpace Independent Publishing Platform, 10 2013. ISBN: 9781492143239.

[18] pySerial. `http://pyserial.sourceforge.net/pyserial_api.html`, 2014. Visited 25.05.2014.

[19] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, 2009.

[20] RobotWorx. Robotic sensors. `http://www.robots.com/education/sensors`, 2014. Visited 09.06.2014.

[21] ROS. Robot operating system webpage. `http://ros.org`, 2014. Visited 07.02.2014.

[22] ROS. rviz - ros wiki. `http://wiki.ros.org/rviz`, 2014. Visited 07.02.2014.

[23] ROS. xacro - ros wiki. `http://wiki.ros.org/xacro`, 2014. Visited 09.02.2014.

[24] Lorenzo Sciavicco and Luigi Villani. *Robotics: modelling, planning and control*. Springer, 2010. ISBN: 1846286417.

[25] Mae L. Seto. *Marine Robot Autonomy*. Springer, 2012. ISBN: 1461456584.

[26] Mark W. Spong and Mathukumalli Vidyasagar. *Robot dynamics and control*. John Wiley & Sons, 2005. ISBN: 0471649902.

[27] Christian H. Sunde. 3d visualization of autonomus marine underwater operations, 2013. Project Thesis.

[28] Linear Technology. Rs485 quick guide. `www.linear.com/docs/29238`, 2014. Visited 07.06.2014.

[29] UCSB. Lecture notes, cs290. `http://excelsior.cs.ucsb.edu/courses/cs290n_cg_modeling/notes/lecture.html`, 2014. Visited 24.04.2014.

[30] Wikipedia. Visualization (computer graphics) — wikipedia, the free encyclopedia, 2013. Visited 02.02.2014.

[31] Wikipedia. Freecad — wikipedia, the free encyclopedia, 2014. Visited 15.03.2014.

[32] Wikipedia. Nx (unigraphics) — wikipedia, the free encyclopedia, 2014. Visited 16.03.2014.

[33] Wikipedia. Orthogonal matrix — wikipedia, the free encyclopedia, 2014. Visited 05.04.2014.

[34] Junku Yuh and Michael West. Underwater robotics. *Advanced Robotics*, 15(5):609–639, 2001.

Appendix A

Attachments

Folder: MATLAB

- ***Kinematics.m***: Calculations of kinematic transformation matrices for Raptor FFM using Denavit-Hartenberg representation.

Folder: Movie

- ***Raptor-movie.mp4***: Movie of navigation in the user interface of rviz

Folder: Poster

- ***Poster.pdf***: Poster presentation of project for the *Master Thesis Poster Exhibition 2014*

Folder: Raptor info

- ***Raptor_PDF.pdf***: Product description of Raptor FFM
- ***Raptor_manual_part_1.pdf***: Raptor manual part 1
- ***Raptor_manual_part_2.pdf***: Raptor manual part 2

Folder: Robot

- **/launch/*display_raptor_mesh.launch***: Software start-up launch file
- **/publisher/*Publisher.py***: State publisher for visualization
- **Folder: STL-files**: Manipulator CAD-files
  - ***base.stl***
  - ***link-2.stl***
  - ***link-3.stl***
  - ***link-8.stl***
  - ***link-9.stl***
  - ***link-10.stl***
  - ***link-11-12.stl***
- **/urdf/*raptor-mesh.urdf.xacro***: URDF-description of Raptor FFM
- ***Raptor.perspective***: Perspective configuration for rqt
- ***raptorconfig-mesh.rviz***: rviz configuration for Raptor