# POLITECNICO DI TORINO

Department of Mechanical and Aerospace Engineering

**Master's Degree in Mechanical Engineering**
ERASMUS+ / PROGRAMME COUNTRIES with UPC - ETSEIB

Master Thesis

## Design and Implementation of 3-RRR Spherical Parallel Robot with Three Coaxial Actuators


Home Supervisor:
Prof. Stefano Pastorelli

I.R.I. Supervisor:
Prof. Federico Thomas

U.P.C. - E.T.S.E.I.B. Supervisor:
Prof. Enric X. Martin

Candidate:
Francesco Morfea

April 2019

# Working Environment

This work has been carried out in the research center *"Institut de Robotica i Informatica Industrial" (I.R.I.)* in cooperation with the *"Universitat Politecnica de Catalunya (U.P.C. - E.T.S.E.I.B.)"*, during the mobility period of Erasmus between U.P.C. and *Politecnico di Torino*.

The *"Institut de Robotica i Informatica Industrial" - I.R.I* is a Joint Research Center of the Spanish Council for Scientific Research (CSIC) and the Technical University of Catalonia (UPC).

The Institute has three main objectives: to promote fundamental research in Robotics and Applied Informatics, to cooperate with the community in industrial technological projects, and to offer scientific education through graduate courses.

The research activities are organized in four research lines. Three of them tackle various aspects of robotics research, including indoor and outdoor human-centered human-safe robotics systems, and the design and construction of novel parallel mechanisms.

During my Erasmus mobility, I've spent six months in the *Kinematics and Robot Design Group*, cooperating with my supervisor, *F. Thomas*, the research engineer *P. Grosch* and the PhD student *A. Shabani*.

The aims of this group are to carry out fundamental researches on design, construction, and motion analysis of complex mechanisms and structures. These devices are parallel manipulators, multi-fingered hands, reconfigurable mechanisms, or cooperating robots, to name a few, but they appear in other domains too, as mechanistic models of locomotive organisms, molecular compounds, or nano-structures.


*To My Family*

# Abstract

This work, entitled *"Design and Implementation of 3-RRR Spherical Parallel Robot with Three Coaxial Actuators"* has had the *scope* to analytically study the kinematics (both inverse and forward one), the workspace and the singularities of a coaxial configuration of a spherical parallel manipulator.

The complete 3D design of the robot has been realised, building it thanks to a 3D printing process called FDM Technology (fused deposition modelling).

Moreover, it has been studied how to use the Dynamixel USB2 hardware and software, which is the interface between the motors and the Pc, to move the motors in Matlab environment. Therefore, it has been firstly used three servomotors on scale as a learning scope, and, after the design of the prototype, it has been installed three servomotors with higher performances, coherently with the scope, the dimensions and the weight of the robot at stake. They have been controlled by an inner Feed Forward Position Control, in order to realise a motion planning in the joint space, through a trapezoidal velocity timing law.

As for the *state of the art*, this thesis has distanced itself from the literature before [10, 9, 12, 4], not using a Denavith-Hartenberg's formulation or a loop equation process, in order to describe the kinematics, but investigating on new method, that could be more efficient in a computational terms, and exploiting its peculiar characteristics and functioning.

For these reasons, it has been developed and implemented a *geometric method* [30] to realise the analytical model of the manipulator. This approach has involved only constant and variable squared distances among the relative fundamental points, after defining the parameters of the robot's architecture. Moreover, this method has several features, such as being Reference Frame free, not managing trigonometrical functions, not having recursive elimination of unknowns and getting one equation in one unknown. It is also versatile, because it could be apply to several mechanisms.

In the end, according to the results of the analytical implementation, it has been decided to realise *two configurations of the robot assembly*, in order to show how the characteristic parameters of the robot influence its behaviour.

# Contents

**Abstract**

**Contents**

**List of Figures**

**List of Tables**

**1 Introduction** **1**

**2 Overview on Parallel Robots** **3**
2.1 Definition . . . 3
2.2 Features and Needs for Robotics . . . 3
2.3 Classifications . . . 4
2.3.1 Planar Robot . . . 5
2.3.2 Spatial Robot . . . 6
2.3.2.1 Translation Manipulator . . . 6
2.3.2.2 Orientation Manipulator . . . 7
2.4 Applications of Spherical Class . . . 10
2.4.1 Laparoscopic Surgery . . . 10
2.4.2 Agile Eye Device . . . 10
2.4.3 Robotics For an AUV . . . 11
2.5 Coaxial Configuration and First Design of the Prototype . . . 12

**3 Inverse Kinematics Analysis** **13**
3.1 Introduction . . . 13
3.2 Matrix Transformation for Each Leg . . . 14
3.2.1 Solving of Inverse Kinematics as the Computation of Euler Angles . . . 14
3.2.1.1 Solving the Two Unique Solutions. . . . 15
3.2.1.2 Solving the Dependent Solution. . . . 15
3.3 Example and Implementation . . . 16

**4 Forward Kinematics Analysis** **19**
4.1 Introduction . . . 19
4.2 Gosselin's Method . . . 22
4.3 Wampler's Method . . . 26
4.4 Thomas' Method . . . 31
4.4.1 Example and Implementation . . . 34
4.4.1.1 Example . . . 34

4.4.1.2 Explanation of Matlab Code . . . . . 38
4.4.2 Assembly Mode and Non Trivial Solutions . . . . . 40
4.4.3 Considerations . . . . . 43

**5 Workspace and Singularities Analysis 45**
5.1 Introduction . . . . . 45
5.2 Workspace Analysis . . . . . 46
5.3 Singularities Analysis . . . . . 51
5.3.1 Type of Singularities in a Parallel Robot . . . . . 51
5.3.2 Jacobian Matrices for 3-RRR Coaxial SPR . . . . . 52
5.4 Design Considerations . . . . . 57

**6 On the Design of Coaxial 3-RRR SPM 59**
6.1 Introduction . . . . . 59
6.2 Used Equipment and Printing Process . . . . . 60
6.3 Design Using Three Motors . . . . . 63
6.4 Mounting . . . . . 70
6.5 Prototype Realisation . . . . . 73
6.6 Future Works . . . . . 76

**7 On the Motors and Control System 77**
7.1 Goals of the Motor Implementation . . . . . 77
7.2 Motor XL-320 . . . . . 78
7.2.1 The Dynamixel System Interface among Drivers, Robot, PC and Programmes . . . . . 78
7.2.2 Model and Features of the Motor . . . . . 79
7.2.3 Functioning, Types of Control and Consideration as For the Motor . . 80
7.2.4 On the Wheel Control Mode . . . . . 81
7.2.4.1 Experiment as for the Law Velocity-PWM . . . . . 81
7.2.4.2 External Feed-Forward Position Control . . . . . 87
7.2.4.3 Consideration and Future Work . . . . . 89
7.2.5 Graphs of the Experiments . . . . . 91
7.2.5.1 FFPC-No Load - Linear Condition . . . . . 91
7.2.5.2 FFPC-Under a Constant Load - Linear Condition . . . . . 92
7.2.5.3 FFPC-Under a Variable Load - Non Linear Condition . . . . 93
7.2.5.4 FFPC-Influence of the Final Time $t_f$ . . . . . 94
7.2.5.5 FFPC-Influence of the Final Angle $q_f$ . . . . . 95
7.3 Motor MX64-AT . . . . . 96
7.3.1 Specification of the Motor MX64-AT . . . . . 96
7.3.2 Model and Feature of the Motor . . . . . 96
7.3.3 Functioning and Types of Control . . . . . 98
7.3.4 Experiments . . . . . 100
7.3.4.1 Considerations on the Experiments . . . . . 101
7.3.4.2 On the Bench With No Load . . . . . 102
7.3.4.3 On the Robot with No Load . . . . . 102
7.3.4.4 On the Robot with Calibrated 500g Load . . . . . 103
7.3.4.5 On the Robot Passing Through Forward Singularity . . . . . 104
7.3.4.6 Influence of the Final Time $t_f$ . . . . . 105
7.3.4.7 Influence of the Total Travel Distance $\Delta q$ . . . . . 106

**8 Costs Analysis** **107**
8.1 Project Budget . . . . . 107

**9 Conclusions** **109**
9.1 Summary of Thesis Contributions . . . . . 109
9.2 Direction for Future Work. . . . . 110

**10 Attachments** **113**
10.1 Appendix A . . . . . 113
10.2 Appendix B . . . . . 117
10.3 Appendix C . . . . . 132

**Acknowledgement** **137**

**Bibliography**

# List of Figures


2.1 Possible Chains for Planar Robot [26]. . . . . 5
2.2 Delta Robot [26]. . . . . 6
2.3 Passive Costrain Mechanism [26]. . . . . 7
2.4 General Architecture of a SPM [9]. . . . . 8
2.5 Shoulder Configuration. . . . . 8
2.6 Coplanar Configuration [9]. . . . . 9
2.7 Coaxial Configuration [9]. . . . . 9
2.8 Laparoscope (left) and Haptic(right) device [24]. . . . . 10
2.9 Agile Eye device [8]. . . . . 11
2.10 AUV [6]. . . . . 11
2.11 Coaxial Configuration. . . . . 12

3.1 Reference Frame EE and Base of the Robot. . . . . 13
3.2 The Eight Solutions of the Inverse Kinematics. . . . . 17

4.1 Identification of the Two Independent Loop Equations. . . . . 20
4.2 Potentiometer Sensor. . . . . 21
4.3 Comparison Between Two Configurations of 3-RRR SPM . . . . . 22
4.4 Spherical Four-Bar Linkages Applied on 3-RRR SPM [4]. . . . . 26
4.5 Spherical Four-Bar Linkages [4]. . . . . 27
4.6 Semi-Graphical Resolution. . . . . 29
4.7 Generic Tetrahedral Strip [30]. . . . . 31
4.8 Reduction of Tetrahedra. . . . . 32
4.9 Schematic Example of the Robot. . . . . 34
4.10 Elimination of Tetrahedra. . . . . 36
4.11 Solutions for the Forward Kinematics. . . . . 37
4.12 Reduction of Tetrahedra. . . . . 38
4.13 Trilateration Process. . . . . 39
4.14 Assembly Mode 1 (left) and Assembly Mode 2 (right). . . . . 40
4.15 Assembly Mode 1 and Solutions 1 and 2. . . . . 41
4.16 Assembly Mode 2 ans Solutions 3 and 4. . . . . 42
4.17 Getting Two Real Solution from a Couple of Complex Conjugate One. . . . . 43

5.1 Characteristics Parameters of the Robot. . . . . 45
5.2 Workspace for Each Leg. . . . . 48
5.3 Whole Workspace General Configuration. . . . . 49
5.4 Whole Workspace Coaxial Configuration. . . . . 49
5.5 Singularities of the Inverse Kinematics. . . . . 55
5.6 Singularities of the Forward Kinematics. . . . . 56

5.7 Singularities of the Third Type. . . . . 56
5.8 Configuration 1 - Top View. . . . . 57
5.9 Configuration 2 - Top View. . . . . 57
5.10 Configuration 2 (left) and Configuration 1 (left) - Front View. . . . . 58

6.1 3-RRR Coaxial SPM. . . . . 59
6.2 uPrint SE (left) - 1200es Series (right). . . . . 60
6.3 Example as for a Limit Orientation. . . . . 61
6.4 Example as for a Limit Orientation. . . . . 62
6.5 Section View. . . . . 65
6.6 Bottom View. . . . . 66
6.7 Top View. . . . . 66
6.8 Driving System Section View. . . . . 67
6.9 Driving System View. . . . . 67
6.10 Robot Assembly View. . . . . 68
6.11 Detail - End Effector Joint Section. . . . . 69
6.12 Detail - Distal Link Joint Section. . . . . 69
6.13 An Overview on the Final Printed Prototype - View 1. . . . . 73
6.14 An Overview on the Final Printed Prototype - View 2. . . . . 74
6.15 An Overview on the Final Printed Prototype - View 3. . . . . 75
6.16 An Overview on the Final Printed Prototype - View 4. . . . . 75
6.17 Concept of the Future Prototype. . . . . 76

7.1 Blocks Diagram of the Part of the System. . . . . 78
7.2 Robotis Motor XL-320 [21]. . . . . 79
7.3 $PWM = 50\ u_2$ . . . . . 83
7.4 $PWM = -50\ u_2$ . . . . . 83
7.5 Velocity with respect to PWM. . . . . 85
7.6 PWM with respect to Velocity. . . . . 85
7.7 Enlargement: Velocity with respect to PWM. . . . . 86
7.8 Enlargement: PWM with respect to Velocity. . . . . 86
7.9 Feed Forward Position Control. . . . . 88
7.10 Cubic Polynomial Timing Law. . . . . 89
7.11 Diagram of Mechatronics Quantities with $G10$ - $t_f = 2s$ - $q_i = 10°$ - $q_i = 100°$ . 90
7.12 FFPC-No Load. . . . . 91
7.13 FFPC-Under a Constant Load: weight 1. . . . . 92
7.14 FFPC-Under a Constant Load: weight 2. . . . . 92
7.15 FFPC-Under a Variable Load: weight 1 (100g). . . . . 93
7.16 FFPC-Under a Variable Load: weight 2 (200g). . . . . 93
7.17 FFPC-Influence of the $t_f = 1s$ . . . . . 94
7.18 FFPC-Influence of the $t_f = 3s$ . . . . . 94
7.19 FFPC-Influence of the $t_f = 3s$ . . . . . 95
7.20 FFPC-Influence of the $q_f = 30s$ . . . . . 95
7.21 Robotis Motor MX-64AT [21]. . . . . 97
7.22 Robotis Motor MX-64AT Performance Graph [21]. . . . . 98
7.23 Robotis Motor MX-64AT Position Control [21]. . . . . 99
7.24 Experiment on Bench Without Load . . . . . 102
7.25 Experiment on the Robot Without Load. . . . . 102
7.26 Experiment on the Robot with Calibrated 500g Load. . . . . 103

7.27 Experiment on the Robot Passing Through Forward Singularity - 1. . . . . 104
7.28 Experiment on the Robot Passing Through Forward Singularity - 2. . . . . 104
7.29 Influence of the parameters of the timing law $t_f = 1s$. . . . . 105
7.30 Influence of the parameters of the timing law $t_f = 3s$. . . . . 105
7.31 Influence of the parameter of the timing law $\Delta q = 50deg$. . . . . 106
7.32 Influence of the parameters of the timing law $\Delta q = 180deg$. . . . . 106

# List of Tables


3.1 Possible Configurations of the Ikine. . . . . 16

4.1 Configurationfs for a 3-RRR SPM. . . . . 23
4.2 Denavit - Hartenberg Parameters . . . . . 23
4.3 4 Solutions of Forward Kinematics. . . . . 36

6.1 3D Printes Used. . . . . 60

7.1 Unit Table Conversion for Motor XL-320. . . . . 80
7.2 Features of the Motor MX64-AT. . . . . 96
7.3 Control Table of MX64-AT Motors. . . . . 97
7.4 Unit Table Conversion - Motor MX64-AT. . . . . 98

# Chapter 1

# Introduction

This thesis, called *"Design and Implementation of 3-RRR Spherical Parallel Robot (S.P.R.) with Three Coaxial Actuators"*, has been carried out in the research center *"Institut de Robotica i Informatica Industrial" (I.R.I.)* in cooperation with the *"Universitat Politecnica de Catalunya (U.P.C. - E.T.S.E.I.B.)"*, during the mobility period of Erasmus between U.P.C. and *Politecnico di Torino*.

It has had the *goal* to investigate on a coaxial configuration of the SPR, given that this symmetric chosen architecture has several benefits with respect to other configurations [9], such as the advantage to use a mechanical transmission with only one motor, the possibility to have a more regular spherical workspace and a better stiffness than a serial wrists, with the same task.

First, it has been defined the characteristic angles of the robot, in order to realise and analytically implement the inverse and forward kinematics. Thanks to this particular configuration, it has been possible to base on the Euler Angles theory the study of the *inverse kinematics*, not following the classic literature [12]. In fact, as it can be found in several papers, it used to treat each chain of the robot as a serial one, identifying each link through the Denavit-Hartenber parameters. Therefore, by trigonometrical transformations and substitutions to make linear the equations, it is possible to achieve the actuator angles, knowing the given orientation of the End Effector (EE). On the contrary, following the Euler Angles ZYX convention, it is easier to obtain the active joint angles, because the needed inputs of the process are only the referent frame (RF) of base and the actuator joints, the chosen zero configuration of the motors and the orientation of the EE.

The following step has been to examine *forward kinematics*, by distancing itself from the state of art [10, 9, 12, 4]. In literature, it is used to obtain the two independent loop equations for the three legs of the robot, starting from the three matrices relationships that characterise the inverse kinematics. However, this process is not the best strategy to solve the forward one, because of different reasons such as the non linear trigonometric system (obtained after manipulating the equations) and the high computational complexity of the algorithm to solve. So, after comparing common resolute theories, it has been developed and implemented a *geometric approach based on the use of only unknown and known squared distances* [30, 27, 31, 13, 28] of characteristic lengths of the robot, leading to obtain a octic linear polynomial without trigonometrical variables. In also this case, this decision has resulted in many advantages, in terms of simplicity, generality, and optimality.

Then, starting from the kinematics analysis, it has been investigated on the influence of the characteristics angles $\alpha_1$, $\alpha_2$ and $\gamma_1$ of the robot on the both *singularities* and *workspace limits* [11, 7, 14, 2]. It has been compared the mobility range of a general configuration ($\alpha_1 = \pi/3,\ \alpha_2 = \pi/2$) to the coaxial prototype, underlining how the workspace increases if $\alpha_1 = \alpha_2 = \pi/2$. On the contrary, studying the Jacobian matrix, it can be stated the number and types of singularities grow up, using these angles. Therefore, it is possible to argue that the optimisation of the mobility range and the singularities is a trade-off; a better workspace carries on more singularities that have to be avoided, during a path motions in the manipulations of the robot. According to these results, it has been taken into account the possibility to realise both the two configurations in order to show how the characteristic parameters of the robot influence its behaviour.

Moreover, it has been physically implemented the robot, by studying how to move the three motors and by realising the final 3D concept of this prototype.

As for the *design of the prototype*, it is based on the use of three electrical motors to actuate the respective degrees of freedom. The driving system is basic and is made of only one gear-mate with 1:1 ratio between each coaxial shaft and the gear motor, in order to realise the prototype very fast. In fact, most of the parts of the robot have been printed, using a process called FDM Technology (fused deposition modelling) and only the support parts and other threaded pivots has been made of in High-Speed Steel, being the backbone of the model. Furthermore, for making the *system reconfigurable and modular*, it has been separated the *transmission system* from the *robot assembly*. This choice has complicated the design, but, at the same time, will allow to change the characteristic angle of the robot, if required, and to reprint it in a cheapest and fastest way.

After getting the final 3D model, it has been defined the *model of the motor* for the mechanical transmission. It has been firstly used three servomotors on a smaller scale (*Robotis Dynamixel XL-320*), in order to learn how to use the Dynamixel USB2 interface to move the motors in Matlab environment.

The goal has been to move the actuators, reading and writing to and from the addresses of the RAM inside the motors. During several tests in different working conditions to evaluate the motor response for achieving the final position of a cubic polynomial timing law, it has been noticed some problems that do not make the motors to reach the desire position. The solution has been to consider the motor as a black-box, with one input and one output and to introduce an *external Feed Forward Position Control*.

However, starting from the results of these experiments, it has been installed a set of three motors Robotis Dynamixel Mx-64 AT with higher performances, coherently with the scope, the dimensions and the mass of the robot at stake.

Overall, the same experiments have been carried out both on bench and on the prototype in order to choose a set of motor parameters to perform the motion of the platform.

# Chapter 2

# Overview on Parallel Robots

Parallel robots are closed-loop mechanisms that have very good performances in terms of accuracy, rigidity and ability to manipulate large loads.

They have many applications in different fields such as astronomy, medicine, camera orienting device. This section want you to give a brief classification of the many categories of the parallel robots, mainly focusing on the *spherical 3-RRR family*, that include the **coaxial configuration**, core of this work.

## 2.1 Definition

General parallel manipulators can be defined as follows: [26]

*"A generalized parallel manipulator is a closed-loop kinematic chain mechanism whose end-effector is linked to the base by several independent kinematic chains."*

It is possible to define different features that identify the parallel manipulator:

- at least two chains support the end-effector. Each of those chains contains at least one simple actuator. There is an appropriate sensor to measure the value of the variables associated with the actuation (rotation angle or linear motion).
- The number of actuators is the same as the number of degrees of freedom of the end-effector.
- The mobility of the manipulator is zero when the actuators are locked.

Therefore, parallel robots can be redefine as follows [26]:

*"Parallel robot is made up of an end-effector with n degrees of freedom,and of a fixed base, linked together by at least two independent kinematic chains. Actuation takes place through n simple actuators."*

## 2.2 Features and Needs for Robotics

By explaining the needs for robotics, it is possible to understand the reason why parallel robots are more efficient than the serial counterpart.

In fact, it is possible to summarise the main feature of the robot, depending on its own application:

- **Accuracy**: it may be most important for *assembly tasks* while the amplitude of motion is less relevant.
- **Dynamics** is also important for *tasks involving a contact* between the robot and its surrounding or for work where execution speed is determinant, like *pick-and-place operations*, in which a robot have to be a very light moving part.
- **Passive compliance**: this concept is important when the end effector is under an external load in terms of force and torque. In general, in these cases there will be slight changes in the pose of the end-effector which are due to backlash in the drive, flexure in the links. This behaviour cannot be observed by the internal sensor of the robot, and cannot be also corrected using the robot control. In the serial robot this concept is called passive compliance and becomes a strong limit in such of applications (machine-tool industry) because it leads to very negative effects. On the other hand, parallel manipulator has a stiffness which is in general much higher than an open-loop structure, so the deformations due to passive compliance is reduced.
- **Active compliance**: for a certain application, for example in some phases of assembly task, the elasticity is also required. By using the controlled actuators, it is possible to adapt the compliance respect to the current application, obtaining a very good stiffness along a certain direction, and soft in the two orthogonal directions.

According to the needs for robotics, mentioned above, this type of robot is interesting for the following reasons:

- **distribution of the load**: a minimum of two chains is necessary to balance the external load.
- **number of sensors**: it is required a minimal number of sensor the closed-loop control of the mechanism.
- **locked position**: when the actuators are locked, the manipulator remains in its position, being an important safety aspect for certain applications, such as medical robotics.
- **better stiffness**, obtained with a small mass of the manipulator allowing high precision and high speed of movements;
- **moment of inertia**: the heavy actuators may often be centrally mounted on a single base platform, the movement of the arm taking place through struts and joints alone. This reduction in mass along the arms permits a lighter arm construction, thus lighter actuators and faster movements. This centralisation of mass also reduces the robot's overall moment of inertia, which may be an advantage for a mobile or walking robot.

## 2.3 Classifications

It is possible to classify the closed-loop mechanisms into two main families: *planar robots*, that have 3 degrees of freedom in the plane and the *spatial one*, that move in the Cartesian space, with 3 or more degrees of freedom, depending on them own application. You can see a complete classification, listed below:

- **Planar Robot**;
- **Spatial Robot**:
  - *3 DOF*:
    - Translation Manipulator;
    - Orientation Manipulator;
    - Mixed Degrees.
  - *4 DOF*;
  - *5 DOF*;
  - *6 DOF*;
  - *Redundant DOF.*

### 2.3.1 Planar Robot

A parallel planar manipulator has an end-effector with three degrees of freedom, two translations (around x-axes and y-axes) and one rotation (with an angle $\vartheta$ around the z axes).

Three chains support the end-effector and they are attached to the end-effector, generally with triangular shape, at three points.

The mobility is zero when the actuators are locked, and that it becomes 3 when all of the actuators are active.

Under these explained conditions, each of the chains is constituted of two rigid bodies linked together by a joint, and that they have a total of three joints.

It is possible to characterise them by the sequence of these three joints, where $R$ stands for revolute joint, while $P$ for the prismatic one. By combining this two types of joints, it can be represented this following sequences (*figure 2.3.1*), able to realise the motion of the triangular platform:

$$RRR,\ RPR,\ RRP,\ RPP,\ PRR,\ PPR,\ PRP$$

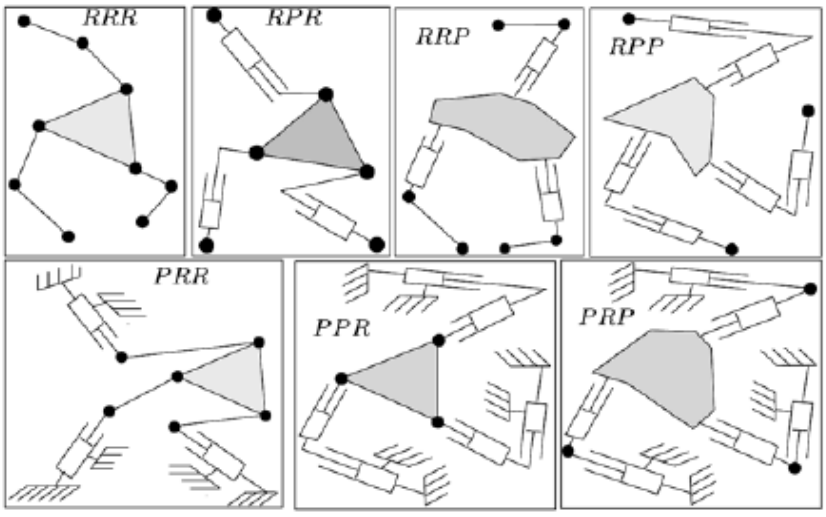



***Figure 2.1:*** Possible Chains for Planar Robot [26].

### 2.3.2 Spatial Robot

In the **spatial robot group** it falls within the *translation manipulator* and the *orientation manipulator*. The typical used joints to realise the chain of these closed-loop mechanisms are listed below, in increasing order of degrees of freedom:

- **R**: revolute;
- **P**: prismatic;
- **Pa**: it stands for parallelogram mechanism;
- **U**: universal;
- **S**: all-and-socket;

#### 2.3.2.1 Translation Manipulator

Manipulators with 3 degrees of freedom in translation are extremely interesting for task of pick-and-place and machining operations.

The most famous and successful robot, in industrial terms, is the **Delta Robot**, firstly studied in the EPFL University and then industrialised by ABB Company.

All the kinematic chains of this robot are of the $RRP_aR$ type: a motor makes a revolute joint rotate about an axis $w$. On this joint is a lever, at the end of which another joint of the $R$ type is set, with axis parallel to $w$. A parallelogram $P_a$ is fixed to this joint, allowing translation in the directions parallel to $w$. At the end of this parallelogram is a joint of the $R$ type, with axis parallel to $w$, and which is linked to the end effector.

It is possible to see both the schematic representation and the ABB robot in the *figure 2.2*.


***Figure 2.2:*** Delta Robot [26].

Other robots of the same family have been developed, but they will not be treated in this work.

#### 2.3.2.2 Orientation Manipulator

Manipulators allowing three rotations around one point are a very good alternative to the serial wrist counterpart.

A first possible category of this group theory results in a **passive constraint mechanism** that allows only rotation of the platform. Three additional kinematics chains will be used to actuate the platform. In *figure 2.3* it presents a wrist using the central mast principle a chain of the *RRPS* types (*left*) and an application with the Vertical Motion Simulator (VMS) of NASA (*right*).

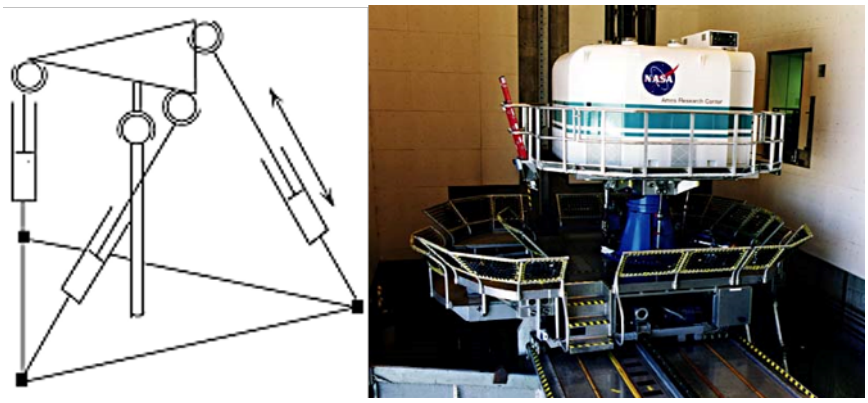


***Figure 2.3:*** Passive Costrain Mechanism [26].

The following subgroup of this orientation family, are the **spherical robots**. This manipulator uses three actuated spherical chains with rotary actuators *RRR*, in which all of the axes converging to a point that is the center of rotation.

It is possible to identify three different configurations in which one of them includes the *coaxial configuration* of the studied prototype:

- **shoulder module**;
- **manipulator with coplanar architecture**;
- **manipulator with coaxial architecture**.

**Shoulder Module**

The manipulators consist of **three identical kinematic sub-chains** connecting the base to a common end effector (EE). On each chain, there is **one fixed actuated** revolute joint whose rotation is associated with angle $\vartheta_i$ , and **two free revolute** joints connecting, respectively, the proximal and distal links and the distal link with the end effector. The rotation of the two free joints are represented by angles $\varphi_i$ and $\sigma_i$ respectively. The characteristic feature is that the axes of rotation of all the joints intersect at a common point called **the geometric center of the manipulator - CM**, the point around which every element of the robot rotates.

The base and the EE can be thought of as **two pyramidal modules** having one vertex in common; this vertex is the CM. The **axes** of the revolute joints of the base and EE are located on the **edges of the pyramids**. For purposes of symmetry, the triangle at the base

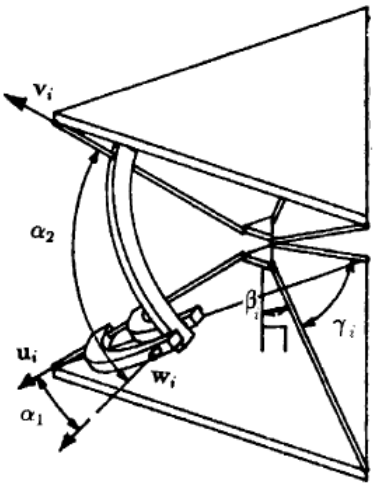



***Figure 2.4:*** General Architecture of a SPM [9].

of each pyramid will be an equilateral triangle. Angle $\gamma_1$ is the angle between two edges of the base pyramid, and angle $\gamma_2$ is the angle between two edges of the EE pyramid.

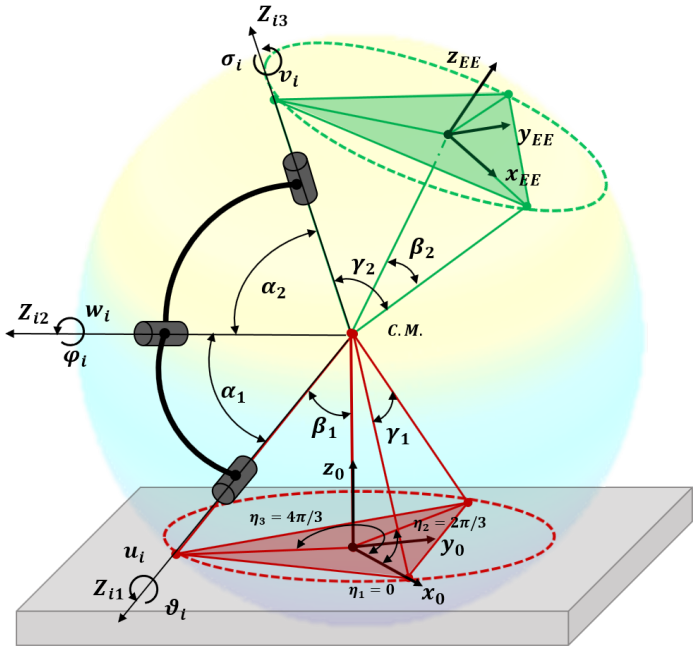



***Figure 2.5:*** Shoulder Configuration.

Furthermore, $\beta_i$ is the angle between one edge and a line passing through the center of the manipulator and perpendicular to the base of the pyramid. It is to be noted that only one of these two angles - $\beta_i$ or $\gamma_i$ - is necessary to uniquely define the base or the platform of a manipulator. Angles $\alpha_1$ and $\alpha_2$ represent the angular lengths associated with the intermediate links. By symmetry, these angles will be the same for each of the sub-chains of the manipulator.

### Coplanar Configuration

The particular feature of this design is that the three revolute on the base and on the platform have coplanar axes. This architecture is in fact a special case of the shoulder module in which angles $\gamma_1 = \gamma_2 = 2\pi/3$. A manipulator of this type is shown in *figure 2.6*

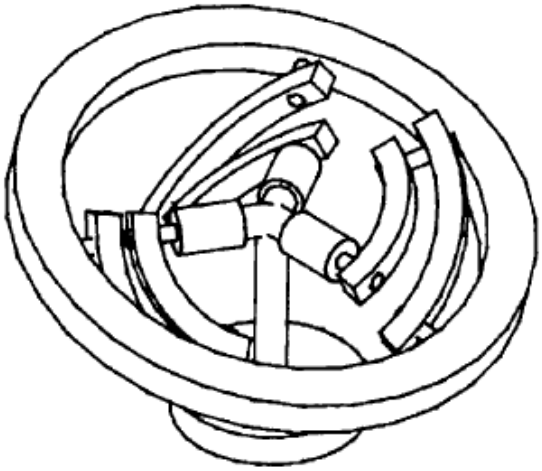


***Figure 2.6:*** Coplanar Configuration [9].

### Coaxial Configuration

The kinematic architecture of Coaxial S.P.M., can be considered as a particular case of the shoulder one, where the angle $\gamma_1 = 0$ and $\gamma_2 = 2\pi/3$.

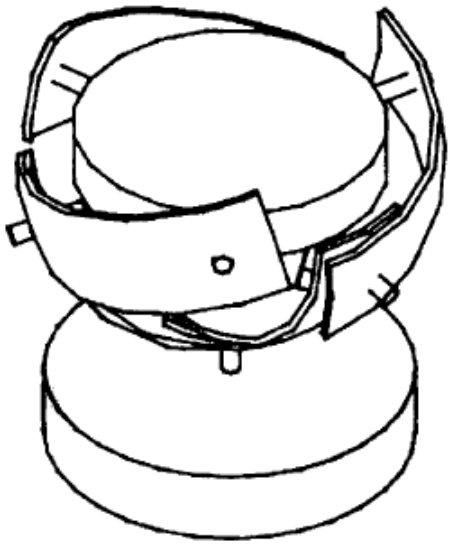


***Figure 2.7:*** Coaxial Configuration [9].

## 2.4 Applications of Spherical Class

As mentioned in the previous section, this family of robot have many applications in different fields such as astronomy, medicine, camera orienting device.

### 2.4.1 Laparoscopic Surgery

One important field regards *medical surgery*, where it has been studied by Temei Li and Shahram Payandeh (2002) [24] a spherical parallel mechanisms for application to **laparoscopic surgery**.

In this application, spherical parallel manipulator has been selected because of the several feature mentioned in the *section 2.2*. They have studied tho different designs for maximizing their workspaces; *a haptic device,* as part of training system, and a *laparoscope holding* mechanism. The laparoscope holding mechanism has to satisfy additional constraints by minimizing the occupied space above the patient.

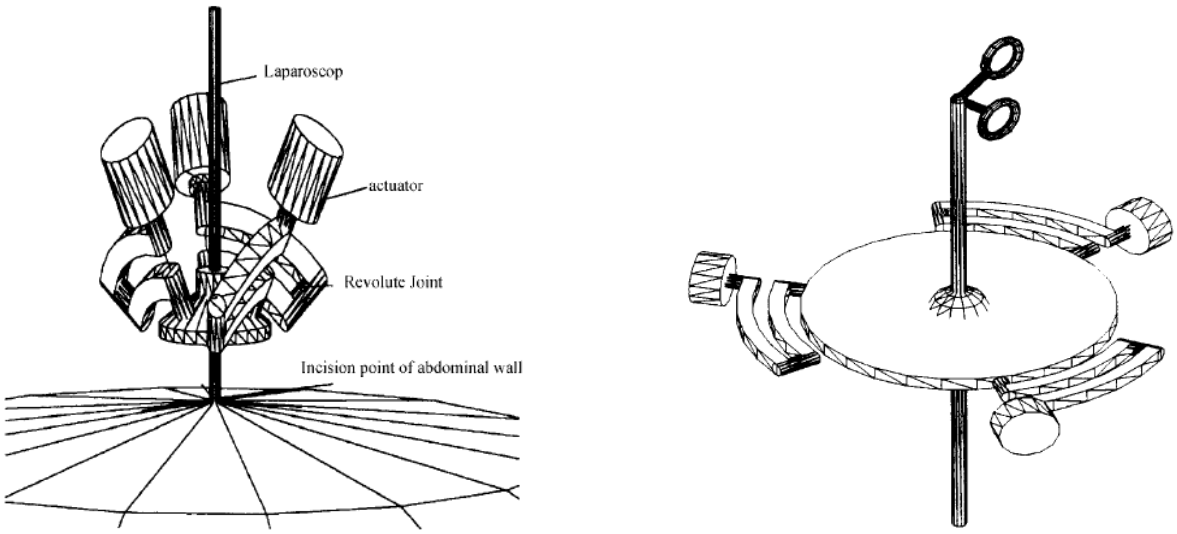



***Figure 2.8:*** Laparoscope (left) and Haptic(right) device [24].

### 2.4.2 Agile Eye Device

Gosselin and Hamel [8] developed a three-degree-of-freedom camera orienting device, called **agile eye**. In fact, as it is referred to, is capable of an orientation workspace larger than that of the human eye. The miniature camera mounted on the end-effector can be pointed within a cone of 140 degrees opening with plus or minus 30 degrees in torsion. The mechanical architecture of the orienting device is based on a 3-RRR spherical parallel manipulator which leads to high-performance dynamics.

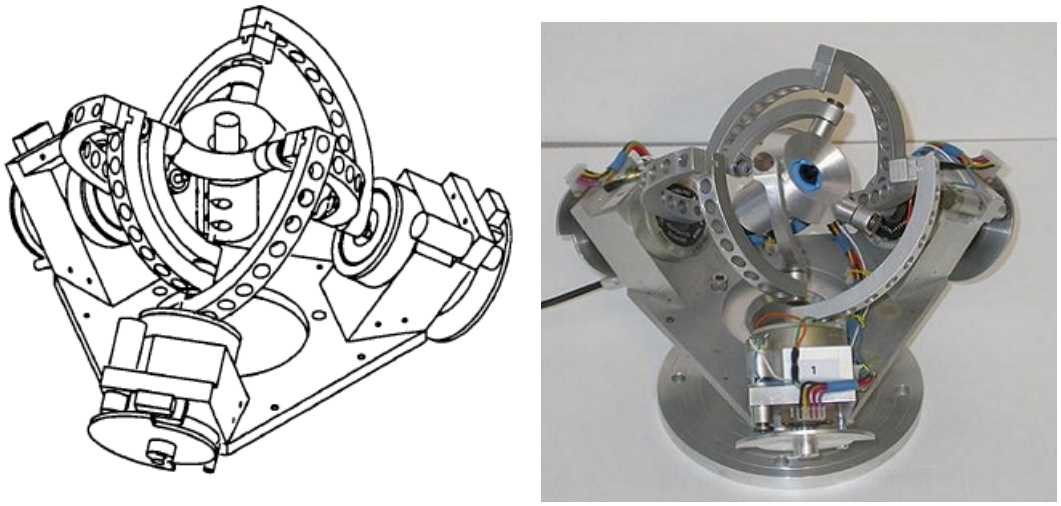


***Figure 2.9:*** Agile Eye device [8].

### 2.4.3 Robotics For an AUV

E. Cavallo and R.C. Michelini [6] developed a cheap **autonomous underwater vehicle - AUV**, SWAN, Submarine Wobble-free Autonomous Navigator, entrusted of extended manoeuvrability for surveying and docking missions with accurate control, both, of trajectory tracking and of attitude keeping.

The joint path and attitude control is obtained by driving the propeller assembly through a *three degrees of freedom parallel kinematics robotic wrist, with coaxial configuration.*

The innovative setting is made possible by the availability of a robotic wrist. The mechanism shall face heavy duty engagements, assuring high stiffness and accuracy. These requirements, quite clearly, lead to a parallel kinematics manipulator, namely, a compact wrist with three driving motors solid to the SWAN hull, actuating a merely rotating effector, which carries the combined propulsion and steering device.

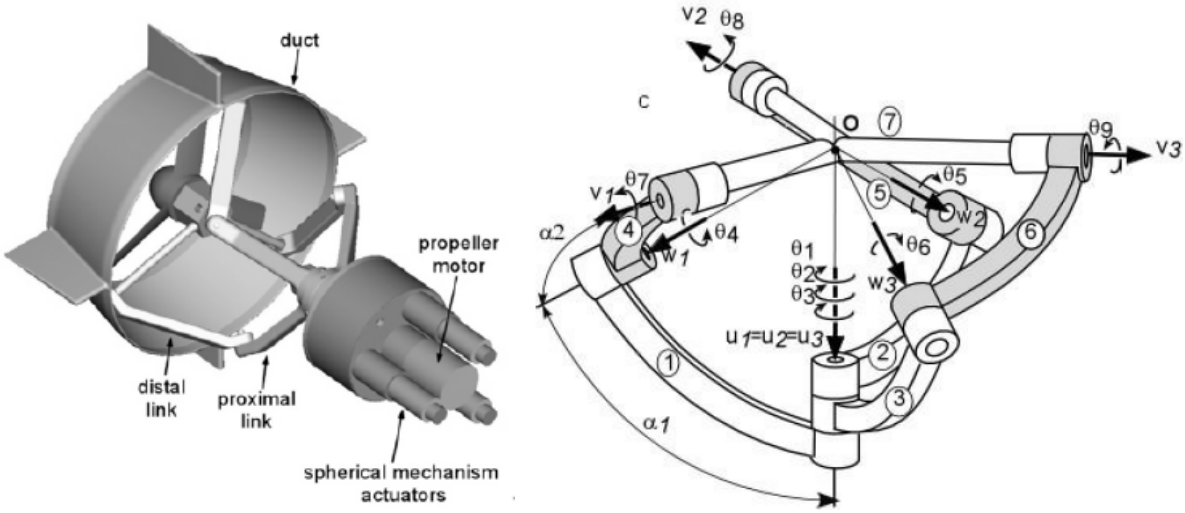



***Figure 2.10:*** AUV [6].

The SWAN principal innovation lies in the combined fixture, used to shelter and to command the propeller. Basically, two parts shall distinguish: *the duct*, with outer compensation vanes or fins and inner screw propeller and the *wrist*, with three co-axial shafts driven by independent motors.

## 2.5 Coaxial Configuration and First Design of the Prototype

The idea of considering a spherical parallel manipulator with coaxial axes in the base pyramid is due to the fact that the coaxial one has a more regular and higher mobility range. It is very interesting its total symmetry too.

The first approximate design has these characteristic angles

- $\alpha_1 = \pi/2$;
- $\alpha_2 = \pi/2$;
- $\gamma_1 = 0$;

The final design will change, at the end of the study as for the forward kinematic, inverse kinematics and workspace. In fact, by doing an analytical implementation of the robot, it will be possible to parametrize these characteristic angles and also to confirm or modify these optimised values.

Starting from the shoulder module, it is possible to develop the scheme of the coaxial configuration, as you can see in *figure 2.11*.

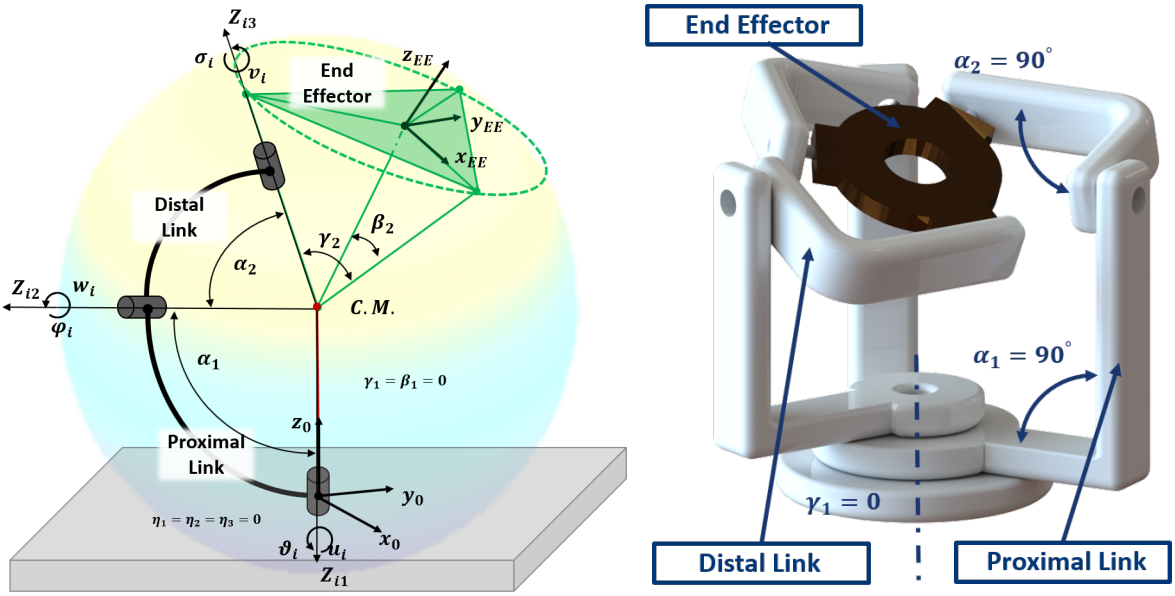



***Figure 2.11:*** Coaxial Configuration.

# Chapter 3

# Inverse Kinematics Analysis

## 3.1 Introduction

To describe the matrix transformations to move an auxiliary referent frame from the base to the EE, it can be noticed that the orientation could be expressed through following rotation matrices, instead of the classical D-H general formulation.

On each chain, there is one fixed actuated revolute joint whose rotation is associated with angle $\vartheta_i$, and two passive revolute joints connecting, respectively, the proximal $\varphi_i$ and distal links and the distal link $\sigma_i$ with the end effector.

All the axes of these revolute joints, intersect in a geometric point, called **center of the mechanism** (CM). The problem is only rotational; no translation is involved, because the work space is spherical, centred on the CM.

The inverse kinematics uses the known orientation of the End effector (EE), to solve joint angles, giving two solutions for the same transformation matrix per leg. So, considering the whole robot, it is obtained **8 analytical solutions** even if not all of them can be accepted because of physical interference among the parts.

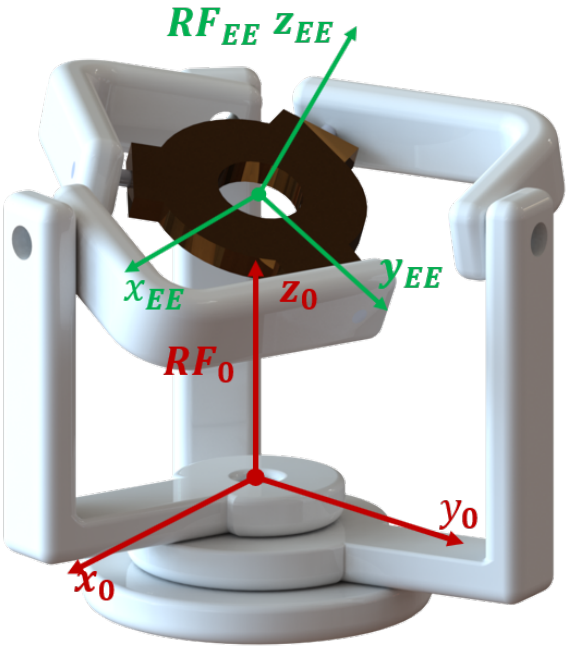



***Figure 3.1:*** Reference Frame EE and Base of the Robot.

In order to literally write the orientation matrix of the EE, referred to each chain, we can use an auxiliary $RF_{aux}$, coincident with the $RF_{Base}$.

Then, for each chain, it is rotated from the base to the EE with follow rotation around respectively the z-axes, y-axes, x-axes, in order to align $RF_{aux}$ to $RF_{E.E}$. In the end, we obtain three matrices to explain the orientation of

EE with respect to each chain of the robot.

## 3.2 Matrix Transformation for Each Leg

By writing the following rotation matrices to align the auxiliary referent frame with EE, it is possible to notice that transformation matrix for each chain can be expressed through the **Euler Angles ZYX** and an extra rotation around *z-axes*, as follows:

$$R = R_z(\alpha) \cdot R_y(\beta) \cdot R_x(\gamma) \qquad \alpha,\ \beta,\ \gamma\ euler\ angles \tag{3.1}$$

$$(1) \quad R = R_z(\vartheta_1) \cdot R_y(\varphi_1) \cdot R_x(\sigma_1) \tag{3.2}$$

$$(2) \quad R = R_z(\vartheta_2) \cdot R_y(\varphi_2) \cdot R_x(\sigma_2) \cdot R_z\left(\frac{2\pi}{3}\right) \tag{3.3}$$

$$(3) \quad R = R_z(\vartheta_3) \cdot R_y(\varphi_3) \cdot R_x(\sigma_3) \cdot R_z\left(\frac{-2\pi}{3}\right) \tag{3.4}$$

$$\tag{3.5}$$

In order to resolve the inverse kinematics of ROLL-PITCH-YAW ANGLES (RPY ), it can be simplified in this way:

$$(1) \quad R_1 = R = R_z(\vartheta_1) \cdot R_y(\varphi_1) \cdot R_x(\sigma_1) \tag{3.6}$$

$$(2) \quad R_2 = R \cdot R_z^T\left(\frac{2\pi}{3}\right) = R_z(\vartheta_2) \cdot R_y(\varphi_2) \cdot R_x(\sigma_2) \tag{3.7}$$

$$(3) \quad R_3 = R \cdot R_z^T\left(\frac{-2\pi}{3}\right) = R_z(\vartheta_3) \cdot R_y(\varphi_3) \cdot R_x(\sigma_3) \tag{3.8}$$

$$\tag{3.9}$$

where de matrix respectively $R_1$, $R_2$, $R_3$ are known.

### 3.2.1 Solving of Inverse Kinematics as the Computation of Euler Angles

The generic matrix can be written in the following way:

$$R = R_z(\vartheta) \cdot R_y(\varphi) \cdot R_x(\sigma) \tag{3.10}$$

$$\tag{3.11}$$

$$R = \begin{bmatrix} c_\varphi \cdot c_\vartheta & c_\vartheta \cdot s_\varphi \cdot s_\sigma - c_\sigma \cdot s_\vartheta & s_\sigma \cdot s_\vartheta + c_\sigma \cdot c_\vartheta \cdot s_\varphi \\ c_\varphi \cdot c_\vartheta & s_\vartheta \cdot s_\varphi \cdot s_\sigma + c_\sigma \cdot c_\vartheta & -s_\sigma \cdot c_\vartheta + c_\sigma \cdot s_\vartheta \cdot s_\varphi \\ -s_\varphi & c_\varphi \cdot s_\sigma & c_\varphi \cdot c_\sigma \end{bmatrix}, \qquad i = 1, 2, 3 \tag{3.12}$$

where $c$ and $s$ represent respectively the functions *cosine* and *sine*.
It can be seen how the the generic $R$ matrix, depend on the value of $\sin\phi$:

- $\sin\varphi \neq \pm 1$: 2 unique solutions;
- $\sin\varphi = \pm 1$: 1 dependent solution;

#### 3.2.1.1 Solving the Two Unique Solutions.

Given R, it be calculated respectively the value of $\varphi$, $\sigma$ and $\vartheta$, obtaining the two sets of angles for the two solutions.

- **First Solution:**

$$\begin{cases} \vartheta_I = a\tan 2\left(\frac{R_{21}}{\cos\varphi}, \frac{R_{11}}{\cos\varphi}\right) \\ \varphi_I = a\sin\left(-R_{31}\right) \\ \sigma_I = a\tan 2\left(\frac{R_{32}}{\cos\varphi}, \frac{R_{33}}{\cos\varphi}\right) \end{cases} \tag{3.13}$$

- **Second Solution:**

$$\begin{cases} \vartheta_{II} = \vartheta_I + \pi \\ \varphi_{II} = \pi - \varphi_I \\ \sigma_{II} = \sigma_I + \pi \end{cases} \tag{3.14}$$

#### 3.2.1.2 Solving the Dependent Solution.

When $\sin\varphi = \pm 1$ the inverse kinematics turns 1 dependent solution. So the literal terms of the generic matrix $R$ change depending on the value of the $\sin\varphi$ and $\cos\varphi$.

So, the generic matrix begins:

- $\sin\varphi = 1$ $(\varphi = \pi/2)$:

$$R = \begin{bmatrix} 0 & c_\vartheta \cdot s_\sigma - c_\sigma \cdot s_\vartheta & s_\sigma \cdot s_\vartheta + c_\sigma \cdot c_\vartheta \\ 0 & s_\vartheta \cdot s_\sigma + c_\sigma \cdot c_\vartheta & -s_\sigma \cdot c_\vartheta + c_\sigma \cdot s_\vartheta \\ -1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -s_{\vartheta-\sigma} & c_{\vartheta-\sigma} \\ 0 & -c_{\vartheta-\sigma} & s_{\vartheta-\sigma} \\ -1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & R_{12}{}' & R_{13}{}' \\ 0 & R_{22}{}' & R_{23}{}' \\ -1 & 0 & 0 \end{bmatrix}$$

- $\sin\varphi = -1$ $(\varphi = 3\pi/2)$:

$$R = \begin{bmatrix} 0 & -c_\vartheta \cdot s_\sigma - c_\sigma \cdot s_\vartheta & s_\sigma \cdot s_\vartheta - c_\sigma \cdot c_\vartheta \\ 0 & -s_\vartheta \cdot s_\sigma + c_\sigma \cdot cs_\vartheta & -s_\sigma \cdot c_\vartheta - c_\sigma \cdot s_\vartheta \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -s_{\vartheta+\sigma} & -c_{\vartheta+\sigma} \\ 0 & c_{\vartheta+\sigma} & -s_{\vartheta+\sigma} \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & R_{12}{}'' & R_{13}{}'' \\ 0 & R_{22}{}'' & R_{23}{}'' \\ 1 & 0 & 0 \end{bmatrix}$$

It is possible to solve the inverse kinematics in these cases using the function $atan2$, obtaining:

- $\sin\varphi = 1$:

$$\sigma = a\tan 2({R_{12}}', -{R_{22}}') + \vartheta \quad \forall\vartheta; \tag{3.15}$$

- $\sin\varphi = -1$:

$$\sigma = a\tan 2(-{R_{12}}'', {R_{22}}'') - \vartheta \quad \forall\vartheta; \tag{3.16}$$

## 3.3 Example and Implementation

Is it possible to show in the figure below, how the RF is coincident, for each chain, plotting the different RF associated to the:

- **Given orientation matrix** of the EE;
- Orientation Matrix to the EE, calculated using the set angles of the **first solution** of the ikine;
- Orientation Matrix to the EE, calculated using the set angles of the **second solution** of the ikine.

In this example, the given matrix $R_{EE}$ is obtained thanks to the follow rotation matrix.

$$R_{EE} = R_x(\pi/3) \cdot R_y(\pi/6) \cdot R_z(\pi/2)$$

In the table, the number 1 represents a possible solution, while the number 0 an impossible one, because of the physical interference among the legs. So, given that the degrees of freedom is defined starting from three fixed frames that are 120° between each other, it is possible to accept the solution only if it is respected this boolean condition:

$$(120° - \vartheta_1 + \vartheta_2 > 30°)(120° - \vartheta_2 + \vartheta_3 > 30°)(120° - \vartheta_3 + \vartheta_1 > 30°) \tag{3.17}$$

where 30° is the limit angle, chosen to not have physical interference among the legs.

| **Possible configuration** | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Sol | $\vartheta_1$ [°] | $\varphi_1$ [°] | $\psi_1$ [°] | $\vartheta_2$ [°] | $\varphi_2$ [°] | $\psi_2$ [°] | $\vartheta_3$ [°] | $\varphi_3$ [°] | $\psi_3$ [°] | Y/N |
| I | 90 | -60 | 30 | 9 | 41 | 55 | -140 | 13 | -64 | 1 |
| II | 270 | 240 | 210 | 9 | 41 | 55 | -140 | 13 | -64 | 0 |
| III | 90 | -60 | 30 | 189 | 139 | 235 | -140 | 13 | -64 | 0 |
| IV | 270 | 240 | 210 | 189 | 139 | 235 | -140 | 13 | -64 | 0 |
| V | 90 | -60 | 30 | 9 | 41 | 55 | 40 | 167 | 116 | 0 |
| VI | 270 | 240 | 210 | 9 | 41 | 55 | 40 | 167 | 116 | 0 |
| VII | 90 | -60 | 30 | 189 | 139 | 235 | 40 | 167 | 116 | 0 |
| VIII | 90 | -60 | 30 | 189 | 139 | 235 | 40 | 167 | 116 | 0 |

***Table 3.1:*** Possible Configurations of the Ikine.

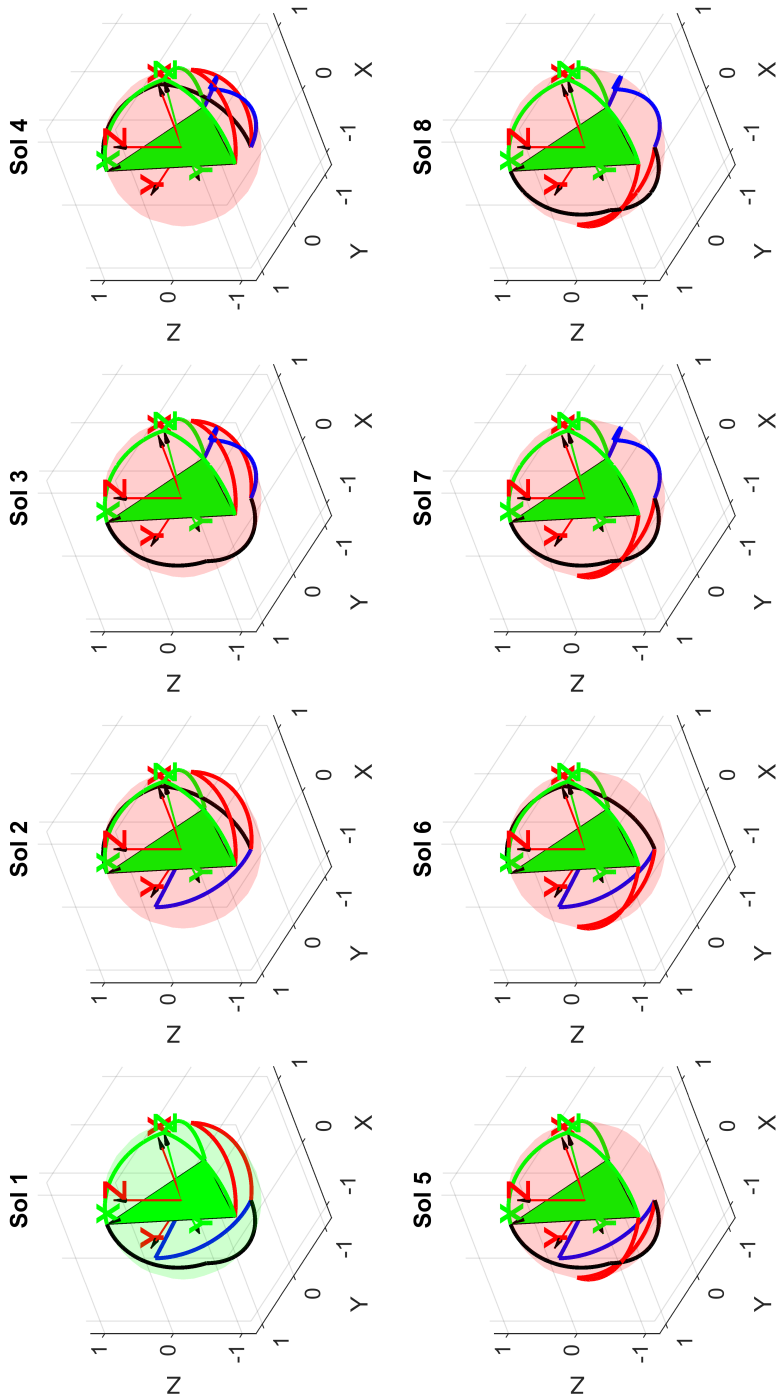



***Figure 3.2:*** The Eight Solutions of the Inverse Kinematics.

# Chapter 4

# Forward Kinematics Analysis

## 4.1 Introduction

In this work, the forward kinematics of a 3-RRR coaxial spherical parallel robot has been investigated and studied, obtained the best process to describe the direct kinematics itself.

First, it was analysed how to obtain the two independent loop equations for the three legs of the robot. So, it was considered the three matrices equations, described in the inverse kinematics and based on the Euler Angles ZYX Convention.

However, not being the good strategy to solve the forward, it has been compared and explained different resolute methods, in order to choose the best one in analytical and physical terms.

In the end, after identifying the best one, it has been elaborated an example in order to show how the algorithm works.

As it has been mentioned above, it has been firstly identified the two independent loop equations. It can be seen in *figure 4.1*, how the three legs lead to three respective loop relationship, but one (3) is a linear combination of the other two $(1, 2)$.

Starting from the base referent frame $R_0 - O_0x_0y_0z_0$, for each leg, it is possible to reach to the End Effector (EE) orientation $R_{EE} - O_{EE}x_{EE}y_{EE}z_{EE}$, by using the rotation matrices $R_z(\vartheta_i)$, $R_y(\varphi_i)$, $R_x(\sigma_i)$ of the Euler Angles ZYX convention.

$$(1) \quad R = R_z(\vartheta_1) \cdot R_y(\varphi_1) \cdot R_x(\sigma_1) \tag{4.1}$$

$$(2) \quad R = R_z(\vartheta_2) \cdot R_y(\varphi_2) \cdot R_x(\sigma_2) \cdot R_z\left(\frac{2\pi}{3}\right) \tag{4.2}$$

$$(3) \quad R = R_z(\vartheta_3) \cdot R_y(\varphi_3) \cdot R_x(\sigma_3) \cdot R_z\left(\frac{-2\pi}{3}\right) \tag{4.3}$$

$$\tag{4.4}$$

Therefore, it can be generated three loop equations, matching the equations in this way:

- **(1) = (2)**:

$$R_z(\vartheta_1) \cdot R_y(\varphi_1) \cdot R_x(\sigma_1) = R_z(\vartheta_2) \cdot R_y(\varphi_2) \cdot R_x(\sigma_2) \cdot R_z\left(\frac{2\pi}{3}\right) \tag{4.5}$$

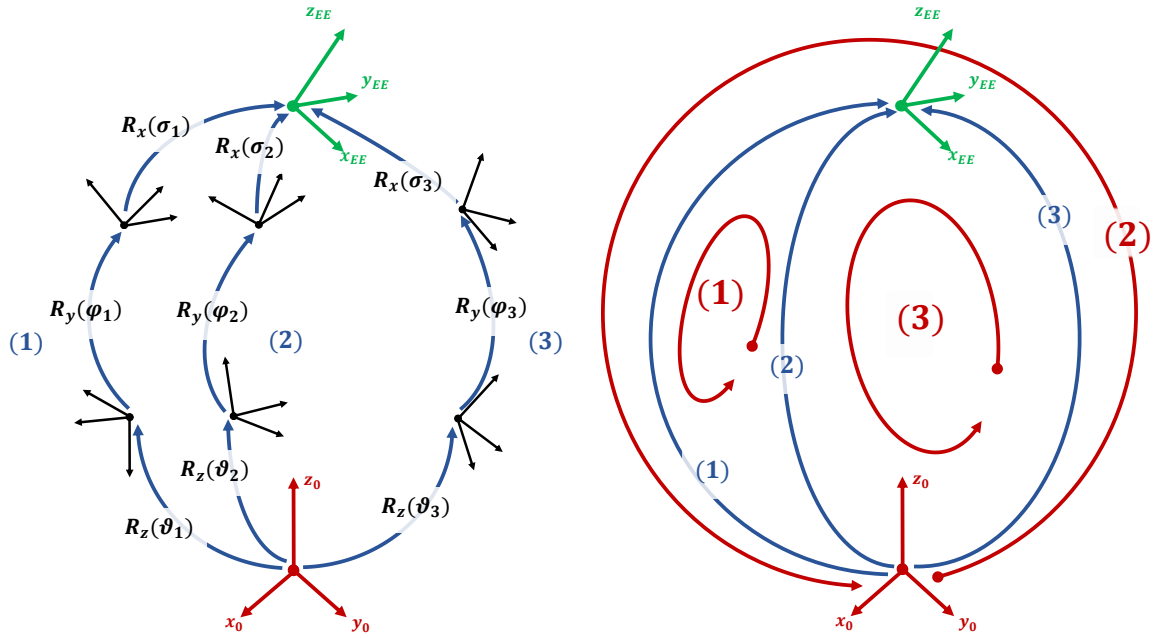



***Figure 4.1:*** Identification of the Two Independent Loop Equations.

.

- **(1) = (3)**:

$$R_z(\vartheta_1) \cdot R_y(\varphi_1) \cdot R_x(\sigma_1) = R_z(\vartheta_3) \cdot R_y(\varphi_3) \cdot R_x(\sigma_3) \cdot R_z\left(\frac{-2\pi}{3}\right) \tag{4.6}$$

- **(2) = (3)**: linear dependent of the other two.

In this case, there are 2 matrices equations that lead to 6 scalar equations ($2 \cdot (3x3)$), while the unknown ($\vartheta_1, \varphi_1, \sigma_1, ..., \sigma_3$) are 9. In order to solve the forward kinematics, it should be rearranged the two loop equations, obtaining two identity matrices ($I_{1-2}$, $I_{1-3}$) on the left side of the relations and took 6 terms, from the right one. Then, it should be chosen 3 angles as an input (eg. $\varphi_1$, $\varphi_2$, $\varphi_3$) to have 6 equations in 6 unknowns.

Given that working with the trigonometric function has a high computational complexity, and it is difficult to simplify the matrix equations, other analytical processes have been investigated to solve the forward kinematics of a 3-RRR coaxial spherical parallel robot, such as:

- **Gosselin's method** [9, 12];
- **Wampler's method** [4]: based on Spherical Four-Bar Linkage theory;
- **Thomas' method** [30]: based on closure polynomial of tetrahedral strip and distance geometry.

Both Gosselin's and Wampler's methods consider two loop equations, that are based on trigonometrical function for describing the orientation of each link with respect to a fixed referent frame. The last one uses unknown and known square distances of characteristic lengths of the robot, leading to a polynomial without trigonometrical variables.

Moreover, in each method, giving both a set of actuator angles ($\vartheta_1$, $\vartheta_2$, $\vartheta_3$) and/or the characteristic parameters and angles as for the architecture of the robot, it is possible to achieve an octic polynomial, in order to solve the roots them relative solutions of the forward kinematics itself.

Therefore, even if these methods use a different starting point in order to define the same solutions of Forward Displacement Analysis (FDA), the scopes and result are the same.

Last but not least, it has been chosen the last method, **closure polynomial for strips of tetrahedra**, for the following reasons:

- **simplicity**: elimination of the trigonometric functions from the algorithm: it is based on only square distances;
- **generality**: the possibility of extending this method to other mechanisms such as 3-RRR SPM; decoupled platform, 4-4 platform with planar base and platform, etc...;
- **optimality**: obtaining the polynomial of the minimal degree, by eliminating the singularity of the formulation in the polynomial expression;
- **moreover, the use of a potentiometer extra-sensor** reading the feedback, it should be used as an input in the analytical process. In this way, considering the read distance like an input, it will be possible to identify the exact solution of the forward kinematics, among all the possible one.


***Figure 4.2:*** Potentiometer Sensor.

.

## 4.2 Gosselin's Method

First of all, *Gosselin's method* starts considering the geometry of a **generic spherical parallel manipulator** and identifies **two symmetric pyramids** with an equilateral triangle as a base, (one for the mobile platform and another one for the fixed base), joined in the vertex, called **center of the mechanism** (CM). The axes of all the revolute parts intersect in the CM point. The **base** and the **platform** are connected thanks to **two links** per leg: the proximal one, linked to the base, and the distal one, linked on the platform (or End Effector, E.E).

In the **coaxial configuration**, it is lost the base pyramid (it degenerates in only one axes), because the three actuators axes are coincident with the z-axes of the whole referent frame. So, the following considerations are valid for all the configurations, even if the characteristic parameters of the robot change. In order to better illustrate the difference between the general arrangement and the coaxial one, it can be see the comparison of the schematic shape and angles in both the *figure 4.3* and the table 4.1.

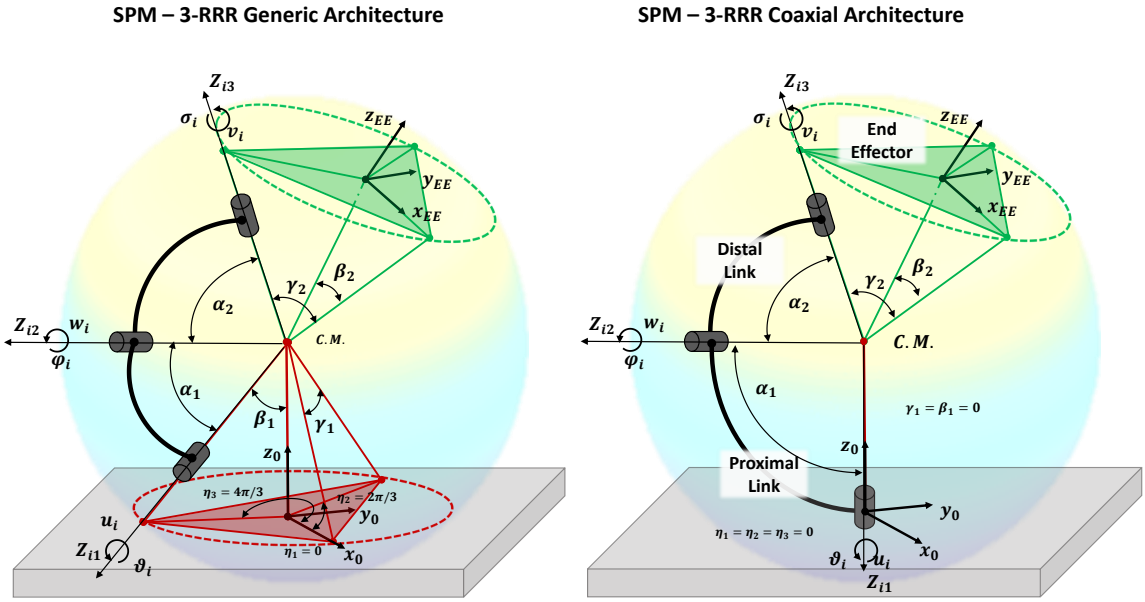



***Figure 4.3:*** Comparison Between Two Configurations of 3-RRR SPM

.

After defining the geometrics parameters for the robot, a fixed referent frame (RF) to the base, a fixed one to the platform and the unit vectors of the axes it is possible to explain the Gosselin's algorithm.

- **Defining of all the characteristic parameters of the robot, through geometric relations**:

$$\sin\beta = \frac{2\sqrt{3}}{3}\sin\left(\frac{\gamma}{2}\right) \tag{4.7}$$

| Configurations | | |
|---|---|---|
| Angles | 3-RRR SPM - Generic | 3-RRR SPM - Coaxial |
| $\vartheta_1$, $\vartheta_2$ , $\vartheta_3$ | d.o.f | d.o.f |
| $\alpha_1$ | chosen | chosen |
| $\alpha_2$ | chosen | chosen |
| $\beta_1$ | chosen | chosen |
| $\beta_2$ | chosen | chosen |
| $\gamma_1$ | calculated, $f(\beta_1)$ | calculated, $f(\beta_1)$ |
| $\gamma_2$ | calculated, $f(\beta_2)$ | calculated, $f(\beta_2)$ |
| $\psi_1$ | calculated, $f(\beta_1, \gamma_1)$ | calculated, $f(\beta_1, \gamma_1)$ |
| $\psi_2$ | calculated, $f(\beta_2, \gamma_1)$ | calculated, $f(\beta_2, \gamma_1)$ |
| $\eta_1$ | 0 | 0 |
| $\eta_2$ | $2\pi/3$ | 0 |
| $\eta_3$ | $4\pi/3$ | 0 |

***Table 4.1:*** Configurationfs for a 3-RRR SPM.

$$\psi = \beta + a\tan\left(\frac{\sin\frac{\gamma}{2}}{\sqrt{3 - 4\sin^2\left(\frac{\gamma}{2}\right)}}\right) \tag{4.8}$$

these geometric relations are valid for both the two pyramids.

- **Identifying of the Denavit-Hartenberg parameters ($\alpha_1$, $\alpha_2$, $\vartheta_i$, $\varphi_i$) and the respective matrices**:

| **Denavit - Hartenberg Parameters** | | | | |
|---|---|---|---|---|
| Link | $\alpha_i$ | $a_i$ | $d_i$ | $q_i$ |
| 1 | 0 | 0 | $\alpha_1$ | $\vartheta_i$ |
| 2 | 0 | 0 | $\alpha_2$ | $\varphi_i$ |

***Table 4.2:*** Denavit - Hartenberg Parameters

$$Q_{i1} = R_z(\vartheta_i) \cdot R_x(\alpha_1) = \begin{bmatrix} \cos(\vartheta_i) & -\cos(\alpha_1)\sin(\vartheta_i) & \sin(\alpha_1)\sin(\vartheta_i) \\ \sin(\vartheta_i) & \cos(\alpha_1)\cos(\vartheta_i) & -\sin(\alpha_1)\cos(\vartheta_i) \\ 0 & \sin(\alpha_1) & \cos(\alpha_1) \end{bmatrix}, \qquad i = 1, 2, 3 \tag{4.9}$$

- **Expressing of vectors $v_1$, $v_2$, $v_3$ with respect to $R_{EE} - O_{EE}x_{EE}y_{EE}z_{EE}$**:

$$[v_1]_{R_{EE}} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad [v_2]_{R_{EE}} = \begin{bmatrix} \sin\gamma_2 \\ 0 \\ \cos\gamma_2 \end{bmatrix} \qquad [v_3]_{R_{EE}} = \begin{bmatrix} \cos\psi_2 \sin\frac{\gamma_2}{2} \\ \sin\psi_2 \\ \cos\psi_2 \cos\frac{\gamma_2}{2} \end{bmatrix} \tag{4.10}$$

- **Using the Euler Angles ZYZ Convention** to express the orientation of the EE with respect to the fixed referent frame $R_0 - O_0x_0y_0z_0$:

$$Q_1 = \begin{bmatrix} \cos\phi_1 & -\sin\phi_1 & 0 \\ \sin\phi_1 & \cos\phi_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad Q_2 = \begin{bmatrix} \cos\phi_2 & 0 & \sin\phi_2 \\ 0 & 1 & 0 \\ -\sin\phi_2 & 0 & \cos\phi_2 \end{bmatrix} \quad Q_3 = \begin{bmatrix} \cos\phi_3 & -\sin\phi_3 & 0 \\ \sin\phi_3 & \cos\phi_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.11}$$

$$Q = Q_1 Q_2 Q_3 = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix} \tag{4.12}$$

these **three angles** $\phi_1$**,** $\phi_2$**,** $\phi_3$ are the unknowns of the analytical process and the whole matrix transformation is used to pass from the $RF_{12}-O_{12}x_{12}y_{12}z_{12}$ to the $RF_{EE}$, chosen for each leg, as a simplification.

- **Referring the RF of the proximal link of each leg** with respect to the fixed RF, by using the $\beta_1$ and $\eta_i$ angles.

$$R_i = \begin{bmatrix} \cos(\eta_i) & \cos(\beta_1)\sin(\eta_i) & \sin(\beta_1)\sin(\eta_i) \\ \sin(\eta_i) & -\cos(\beta_1)\cos(\eta_i) & -\sin(\beta_1)\cos(\eta_i) \\ 0 & \sin(\beta_1) & -\cos(\beta_1) \end{bmatrix}, \qquad i = 1, 2, 3 \tag{4.13}$$

$$\eta_1 = \eta_2 = \eta_3 = 0 \tag{4.14}$$

- **Writing the distal link unit vectors** $w_1$**,** $w_2$**,** $w_3$ with respect to the fixed RF:

$$[w_i]_{R_0} = R_i Q_{i1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad i = 1, 2, 3 \tag{4.15}$$

- **Writing the EE unit vectors** $v_1$**,** $v_2$**,** $v_3$ with respect to the fixed RF:

$$[v_i]_{R_0} = R_i Q_{11} Q [v_i]_{R_{EE}} \qquad i = 1, 2, 3 \tag{4.16}$$

- **Specifying of the geometric constrain to obtain the two loop equations**:

$$w_i \cdot v_i = cos(\alpha_2) \qquad i = 1, 2, 3 \qquad w.r.t. \quad RF_0 \tag{4.17}$$

As a consequence of the set of Euler Angles used in this Cartesian coordinates, the first equation can be simplified as:

$$\cos\phi_2 = \cos\alpha_2 \tag{4.18}$$

The last two equations can be expressed as a function of cosine and sine ($c\phi_1, s\phi_1, c\phi_3, s\phi_3$) and a constant terms, depended on the characteristic angles of the robot, as follows:

$$I_1 \sin\phi_1 + I_2 \cos\phi_1 + I3 = 0 \tag{4.19}$$

$$J_1 \sin\phi_1 + J_2 \cos\phi_1 + J3 = 0 \tag{4.20}$$

$$I_1 = A\sin\phi_3 + B\cos\phi_3 + F \tag{4.21}$$
$$I_2 = G\sin\phi_3 + H\cos\phi_3 + J \tag{4.22}$$
$$I_3 = K\cos\phi_3 + L \tag{4.23}$$
$$J_1 = M\sin\phi_3 + P\cos\phi_3 + Q \tag{4.24}$$
$$J_2 = R\sin\phi_3 + S\cos\phi_3 + T \tag{4.25}$$
$$J_3 = V\sin\phi_3 + W\cos\phi_3 + U \tag{4.26}$$

- **Solving the linear system in function of $s_{\phi_1}$ and $c_{\phi_1}$:**

$$\begin{cases} \sin\phi_1 = \frac{-I_3J_2+I_2J_3}{-I_2J_1+I_1J_2} \\ \\ \cos\phi_1 = \frac{-I_3J_1+I_1J_3}{-I_2J_1+I_1J_2} \end{cases} \tag{4.27}$$

- **Using the trigonometric identity, for obtaining only one unknown ($\phi_3$):**

$$\sin^2\phi_1 + \cos^2\phi_1 = 1 \tag{4.28}$$
$$-I_2^2J_1^2 + I_3^2J_1^2 + 2I_1I_2J_1J_2 - I_1^2J_2^2 + I_3^2J_2^2 - 2I_1I_3J_1J_3 \tag{4.29}$$
$$- 2I_2I_3J_2J_3 + I_1^2J_3^2 + I_2^2J_3^2 = 0 \tag{4.30}$$
$$I_1J_2 \neq I_2J_1 \tag{4.31}$$

- **Substituting the trigonometric function in $s_{\phi_3}$ and $c_{\phi_3}$** with an only parameter $t$ (tan-half), to obtain a linear polynomial of 8 degrees with only one unknown:

$$\cos\phi_3 = \frac{1-t^2}{1+t^2} \qquad \sin\phi_3 = \frac{2t}{1+t^2} \qquad t = \tan\frac{\phi_3}{2} \tag{4.32}$$

- **Rearranging and solving the polynomial to achieve the 8 roots**:

$$\sum_{i=0}^{8} N_i t^i = 0 \qquad i = 1, 2, .., 8 \tag{4.33}$$

## 4.3 Wampler's Method

As it has been said before, SPM does not allow the resolution of the FDA in a closure form, due to the peculiar multi-loop architecture, and because of the non linear trigonometric equations, the resolution of the algorithm has a high complexity. For these reasons, Wampler's approach has been used with the objective of extending the kinematics of a four-bar linkages to the 3-RRR Coaxial SPM. The **advantages** are mainly two:

- *managing* the two loop equations with compact constant coefficients;
- *solving* the equations in a semi-graphically way.

The consideration as for the geometry of the robot in different configurations (eg. general, coaxial) are exactly the same; for this reason, the dissertation should be done on the general architecture of SPM and in literary shape.

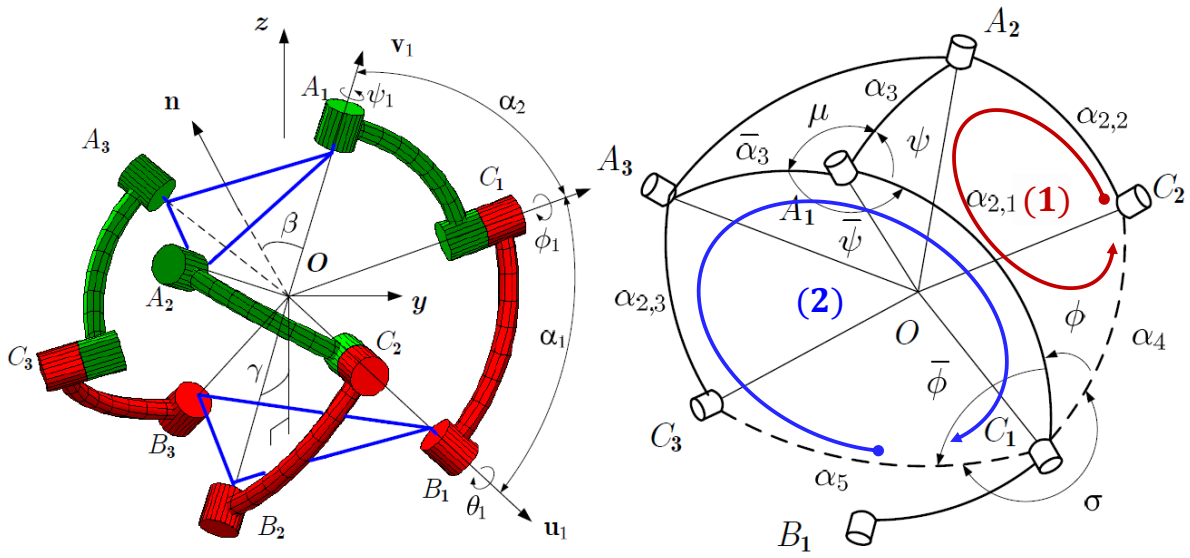



***Figure 4.4:*** Spherical Four-Bar Linkages Applied on 3-RRR SPM [4].

The characteristic parameters of the robot are the same as those already used in the Gosselin's method.

In fact, knowing the orientation of the $u_i$ an $w_i$ axes of the revolute joints, one of the target of the FDA is to **identify the orientation of the $v_i$ axes**, expressing them with respect to fixed RF:

- *using the same three unit vectors of the mobile platform as unknowns*:

$$v_i \cdot v_j = \cos\alpha_3 \qquad i,j = 1,2,3, \qquad i \neq j \qquad ||v_i|| = 1 \tag{4.34}$$

  where $\cos\alpha_3$ is the angle between two distal joints of two different legs of the robot.

- *using the Euler angles of the mobile platform as unknowns*:

$$[v_i]_{R_0} = Q[v_i]_{R_{EE}} \qquad Q = Q(\varphi) \qquad \varphi = [\varphi_1, \varphi_2, \varphi_3]^T \tag{4.35}$$

- *using the actuated joint angles as unknowns*:

$$[v_i]_{R_0} = R_i[v_i]_{R_{EE}} \qquad R_i = R_i(\vartheta_i, \varphi_i, \psi_i) \qquad i = 1, 2, 3 \tag{4.36}$$

  in this case, the actuator angle $\vartheta_i$ are the input and the unknowns are the passive ones of the remained joints.

Before entering in the core of the algorithm, it is possible to do an explanation, considering a schematic spherical four-bar loop, in order to identify the loop equation. Then, this theory will be extended to the SPM, involving two spherical four-bar linkages.

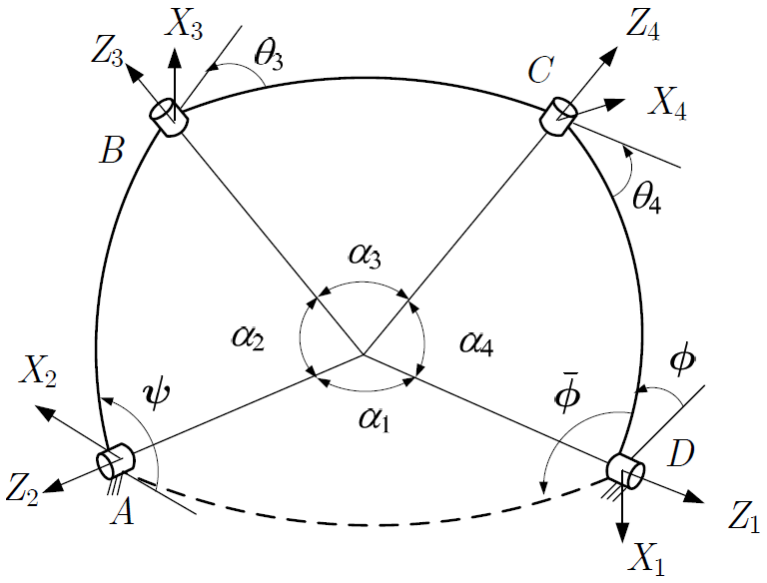



***Figure 4.5:*** Spherical Four-Bar Linkages [4].

As it possible to see in *figure 4.5*, by moving from A-point to D-point, two kinds of rotations are considered in this loop:

- *joint rotation*: it describes the relative orientation (it is constant) of each neighbouring pair of joint axes of a link;
- *side rotation*: that is variable.

The loop equations can be expressed as follows:

$$I = Z_4 S_4 Z_1 S_1 Z_2 S_2 Z_3 S_3 \tag{4.37}$$

$$Z_1 = R_z(\phi) \qquad Z_2 = R_z(\pi - \psi) \qquad Z_3 = R_z(\vartheta_3) \qquad Z_4 = R_z(\vartheta_4)$$
$$S_i = R_x(\alpha_i) \quad i = 1, 2, 3, 4$$

By manipulating the equation, let it write:

$$z^T S_3^T z = z^T S_4 Z_1 S_1 Z_2 S_2 z \qquad z = [0,0,1]^T \tag{4.38}$$

Expliciting the relation below, it has been achieved the final expression, based on constant terms (depending on the characteristic angles) and the unknowns $\phi$ and $\psi$ angles.

$$k_1 + k_2 \cos\psi + k_3 \cos\psi \cos\phi - k_4 \cos\phi + k_5 \sin\psi \sin\phi = 0 \tag{4.39}$$

$$\begin{aligned} k_1 &= c_{\alpha_1} c_{\alpha_2} c_{\alpha_4} - c_{\alpha_3} \\ k_2 &= s_{\alpha_1} s_{\alpha_2} c_{\alpha_4} \\ k_3 &= c_{\alpha_1} s_{\alpha_2} s_{\alpha_4} \\ k_4 &= s_{\alpha_1} c_{\alpha_2} s_{\alpha_4} \\ k_5 &= s_{\alpha_2} s_{\alpha_4} \end{aligned}$$

Therefore, the **resolute algorithm** can be shown, considering two spherical four-bar linkages for the 3-RRR SPM ( $A_1C_1C_2A_2$, $A_1C_1C_3A_3$), as you can see in *figure 4.4*.

- **definition of the unit vectors of the joints**:

$$u_i = \begin{bmatrix} -\sin\eta_i \sin\beta_1 \\ \cos\eta_i \sin\beta_1 \\ -\cos\beta_1 \end{bmatrix} \qquad w_i = \begin{bmatrix} -s_{\eta_i} s_{\beta_1} c_{\alpha_1} + (c_{\eta_i} s_{\vartheta_i} - s_{\eta_i} c_{\beta_1} c_{\vartheta_i}) s_{\alpha_1} \\ -c_{\eta_i} s_{\beta_1} c_{\alpha_1} + (s_{\eta_i} s_{\vartheta_i} - c_{\eta_i} c_{\beta_1} c_{\vartheta_i}) s_{\alpha_1} \\ -c_{\beta_1} c_{\alpha_1} + s_{\beta_1} c_{\vartheta_i} s_{\alpha_1} \end{bmatrix} \tag{4.40}$$

$$[v_1]_{RF_{EE}} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad [v_2]_{RF_{EE}} = \begin{bmatrix} \sin\gamma_2 \\ 0 \\ \cos\gamma_2 \end{bmatrix} \qquad [v_3]_{RF_{EE}} = \begin{bmatrix} \cos\psi_2 \sin\frac{\gamma_2}{2} \\ \sin\psi_2 \\ \cos\psi_2 \cos\frac{\gamma_2}{2} \end{bmatrix} \tag{4.41}$$

- **identification and calculation of the angles on the sphere**:

$$\alpha_4 = \cos^{-1}(w_1 \cdot w_2) \qquad \alpha_4 \in (0, \pi] \tag{4.42}$$

$$\alpha_3 = \cos^{-1}([v_1]_{R_{EE}} \cdot [v_3]_{R_{EE}}) \qquad \alpha_3 \in (0, \pi] \tag{4.43}$$

$$\bar{\alpha_3} = \cos^{-1}([v_1]_{R_{EE}} \cdot [v_3]_{R_{EE}}) \qquad \bar{\alpha_3} \in (0, \pi] \tag{4.44}$$

$$\tag{4.45}$$

$$\begin{cases} \cos\alpha_5 = w_1 \cdot w_3 \\ \sin\alpha_5 = ||w_1 \times w_3|| \end{cases} \tag{4.46}$$

$$\begin{cases} \cos\alpha_{45} = w_2 \cdot w_3 \\ \sin\alpha_{45} = ||w_2 \times w_3|| \end{cases} \tag{4.47}$$

$$\sigma = 2\pi - \cos^{-1}[(\cos\alpha_{45} - \cos\alpha_4 \cos\alpha_5)/(\sin\alpha_4 \sin\alpha_5) \tag{4.48}$$

$$\mu = cos^{-1}[(csc^2\alpha_3)(\cos\alpha_3 - \cos^2\alpha_3)] \qquad \mu \in (0, \pi]; \tag{4.49}$$

$$\bar{\phi} = 2\pi - \phi - \sigma \tag{4.50}$$

$$\bar{\psi} = 2\pi - \psi - \mu \tag{4.51}$$

- **selection of the two loop equations**:
  - **Loop Equation (1)**:

$$k_1 + k_2 \cos\psi + k_3 \cos\psi \cos\phi - k_4 \cos\phi + k_5 \sin\psi \sin\phi = 0 \tag{4.52}$$
$$A_{1(\phi)} c\psi + B_{1(\phi)} s\psi + C_{1(\phi)} = 0 \tag{4.53}$$

    where the constant terms are function of characteristics angles:

$$k_i = F(\alpha_2, \alpha_3, \alpha_4) \qquad i = 1, 2, .., 5$$

  - **Loop Equation (2)**:

$$j_1 + j_2 \cos\bar{\psi} + j_3 \cos\bar{\psi} \cos\bar{\phi} - j_4 \cos\bar{\phi} + j_5 \sin\bar{\psi} \sin\bar{\phi} = 0 \tag{4.54}$$
$$A_{2(\bar{\phi})} c\bar{\psi} + B_{2(\bar{\phi})} s\bar{\psi} + C_{2(\bar{\phi})} = 0 \tag{4.55}$$

    it is possible to express the second equation as a function of $\phi$ and $\psi$, in order to solve the linear system in $\sin\psi$ and $\cos\psi$. Moreover, the constants $j_i$ are function of:

$$j_i = F(\alpha_2, \bar{\alpha_3}, \alpha_5, \mu, \sigma) \qquad j = 1, 2, .., 5$$

- **resolution of the linear system, in function of $\phi$ and $\psi$ angles**:

$$\begin{cases} \sin\psi = \frac{-B_2C_1 + B_1C_2}{-A_2B_1 + A_1B_2} \\ \cos\psi = \frac{-A_2C_1 + A_1C_2}{-A_2B_1 + A_1B_2} \end{cases} \tag{4.56}$$

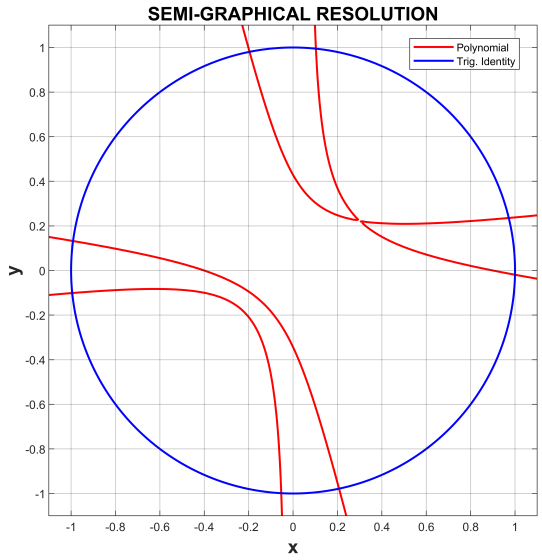



***Figure 4.6:*** Semi-Graphical Resolution.

- **semi-graphical equation solving**: assuming the denominator $-A_2B_1 + A_1J_2 \neq 0$ and evaluating the trigonometric identity, it is possible to obtain:

$$\sin^2\psi_1 + \cos^2\psi_1 = 1 \tag{4.57}$$

$$C_2^2A_1^2 + C_2^2B_1^2 + 2A_2B_2A_1B_1 - B_2^2A_1^2 + A_2^2C_1^2 - 2A_2C_2A_1C_1 \tag{4.58}$$

$$- 2B_2C_2B_1C_1 - A_2^2B_1^2 + B_2^2C_1^2 = 0 \tag{4.59}$$

  Rearraging the new equation in function of $\cos\phi$ and $\sin\phi$, it has been achieved the eight solutions, coming from the intersection with these two curves, as shown in *figure 4.6*:

$$\begin{cases} x = \cos\phi \\ y = \sin\phi \end{cases} \quad \Rightarrow \quad \begin{cases} f(x,y) = 0 \\ x^2 + y^2 = 1 \end{cases} \tag{4.60}$$

- **solving the octic polynomial, using the tan-half identities**:

$$\sum_{i=0}^{8} N_i t^i = 0 \qquad i = 1, 2, .., 8 \tag{4.61}$$

  substituting $s_\phi$ and $c_\phi$ with an only parameter t (tan-half)

$$\cos\phi = \frac{1-t^2}{1+t^2} \qquad \sin\phi = \frac{2t}{1+t^2} \qquad t = \tan\frac{\phi}{2} \tag{4.62}$$

## 4.4 Thomas' Method

As it has been mentioned in the previous section, this method has many advantages in both analytical and physical terms and it is very interesting for these reasons:

- **simplicity**: elimination of the trigonometric functions from the algorithm: it is based on only squared distances;
- **generality**: the possibility of extending this method to other mechanisms such as 3-RRR SPM; decoupled platform, 4-4 platform with planar base and platform, etc...;
- **optimality**: obtaining the polynomial of the minimal degree, by eliminating the singularity of the formulation in the polynomial expression;
- **moreover, the use of a potentiometer extra-sensor**: reading the feedback, it should be used as an input in the analytical process. In this way, considering the read distance like an input, it will be possible to identify the exact solution of the forward kinematics, among all the possible one.

First, let it do a short explanation as for to identify constant and variable distances in a generic strip of tetrahedra.

### Theory of the Tetrahedral strip and Closure Polynomials

A tetrahedral strip is a tetrahedron-tetrahedron truss where any tetrahedron has two neighbours except those in the extremes which have only one.

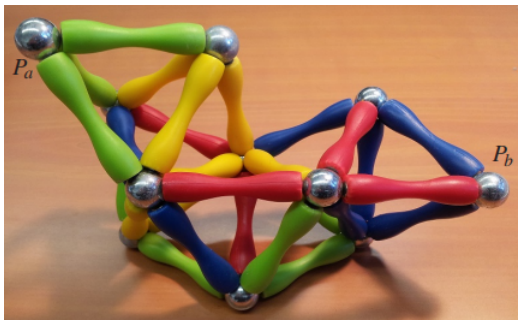



***Figure 4.7:*** Generic Tetrahedral Strip [30].

It is possible to see that each strip has two endpoints $P_a$ and $P_b$, as shown in *figure 4.7* and, for this reason, any rod length cannot be freely chosen. Therefore, unless any of the tetrahedra degenerate, such a truss is rigid.

The distance between the strip endpoints is first derived by iterating a basic operation involving only two neighbouring tetrahedra over the whole strip, leading to a scalar equation containing radical terms.

Given a set of points in the strip, the valid distances between them can be characterized using the theory of **Cayley-Menger determinants** [27, 31, 13], defined for two sets of points $P_{i_1}, ..., P_{i_n}$ and $P_{j_1}, ..., P_{j_n}$. In this case, the two sets of the points are the same and the determinant of these sets of points became $D(i_1, ..., i_n) = D(i_1, ..., i_n; i_1, ..., i_n)$ and it is proportional to the squared volume of the simplex spanned by $P_{i_1}, ..., P_{i_n}$ in $R^{n-1}$.

$$D(i_1,...,i_n;j_1,...,j_n) = 2\left(-\frac{1}{2}\right)^n \begin{vmatrix} 0 & 1 & \dots & 1 \\ 1 & s_{i_1,j_1} & \dots & s_{i_1,j_n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & s_{i_n,j_1} & \dots & s_{i_n,j_n} \end{vmatrix}$$

where $s_{i,j}$ stands for the squaredd distance between $P_i$ and $P_j$.

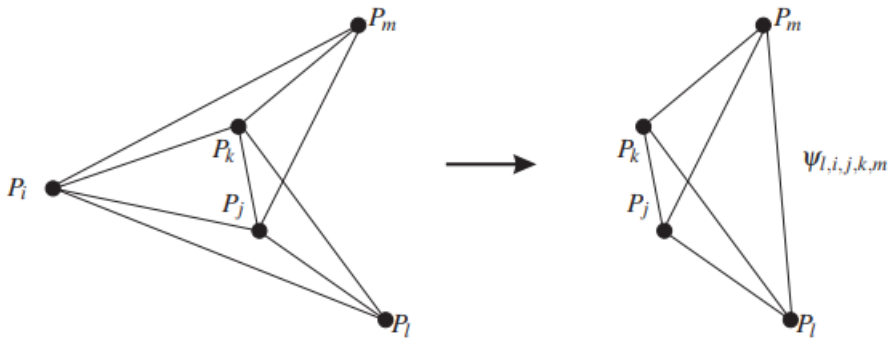



***Figure 4.8:*** Reduction of Tetrahedra.

Considering the two neighbouring tetrahedra, shown in *figure 4.12*. The squared distance between $P_l$ and $P_m$ can be calculated as:

$$s_{l,m} = \frac{2}{D(i,j,k)}\left(D(i,j,k,l;i,j,k,m)\mid_{s_{l,m}=0} \pm\sqrt{D(i,j,k,l)D(i,j,k,m)}\right) \tag{4.63}$$

The $\pm$ sign is due to the two possible solutions depending on the relative orientation between two tetrahedra.

It has been introduced a notation to express these squared distances:

$$s_{l,m} = \Psi_{l,i,k,m} \tag{4.64}$$

If some distances, that are involved in $\Psi_{l,i,ik,m}$, are a unknown, it can be explicated as:

$$s_{l,m} = \Psi_{l,i,k,m}(s_{i,j}) \tag{4.65}$$

Moreover, if one of the points in the set $P_i, P_j, P_k$ does not belong to any other tetrahedron in the strip, it can be removed from it, connecting the point $P_l$ and $P_m$ with a road, with a length in according to the *formulation 4.63*. This reduces the number of tetrahedra in the strip into only two.

After repeating this operation until the strip contains only two tetrahedra, the distance between the tetrahedral strip endpoints is finally obtained as a $2^{n-2}$ - valued function, where $n$ is the number of tetrahedra in the strip, before rearranging.

Once it has been treated the theory at the base of the tetrahedral strip, it is possible to introduce the algorithm, used to solve the roots of the polynomial.

- **defining the squared distances** through the Cayley-Menger determinant;

- **clearing radicals** to obtain a closure condition as a polynomial in terms of a given rod length:

$$a_0 + a_1\sqrt{r} + a_2(\sqrt{r})^2 + a_3(\sqrt{r})^3 + ... = 0 \tag{4.66}$$

the first step consists in taking the numerator of the rational form of the obtained function and then clearing radicals. As radicals will appear nested, they are cleared using an iterative process starting from the outer one. At each step of this process, the expressions involving a radical will have the general form as in *eq. 4.66*, that can be rearranged , considering both of the sign, as follows in the *equation 4.70*:

$$(a_0 + a_2r + a_4r^2...) + \sqrt{r}(a_1r + a_3r + a_5r^2 + ...) = 0 \tag{4.67}$$
$$(a_0 + a_2r + a_4r^2...) - \sqrt{r}(a_1r + a_3r + a_5r^2 + ...) = 0 \tag{4.68}$$
$$\tag{4.69}$$

Given that the process is interested in the roots of both equations, it has been considered the product of them, achieving:

$$(a_0 + a_2r + a_4r^2...)^2 - r(a_1r + a_3r + a_5r^2 + ...)^2 = 0 \tag{4.70}$$

- **eliminating of the singularity of the formulation**, in order to obtain a polynomial of a minimal degree: if a rod with variable length belongs to a shared face, this face degenerates for some values of $s_{i,j}$. When this happens, the three points defining the face get aligned and the tetrahedral strip can be decomposed into two parts so that one can freely rotate with respect the other about the axis defined by these three aligned points. These terms corresponding to these degenerate configurations will appear in the closure polynomial and they can be easily removed by iteratively dividing the closure polynomial by the function that describes the area of this shared face, until the remainder is not null.

### 4.4.1 Example and Implementation

#### 4.4.1.1 Example

In order to apply the technique explained above to solve the kinematics of a 3-RRR Coaxial SPM, in the *figure 4.9* it has been shown a schematic representation with arches and tetrahedra of the kinematics of the robot itself.

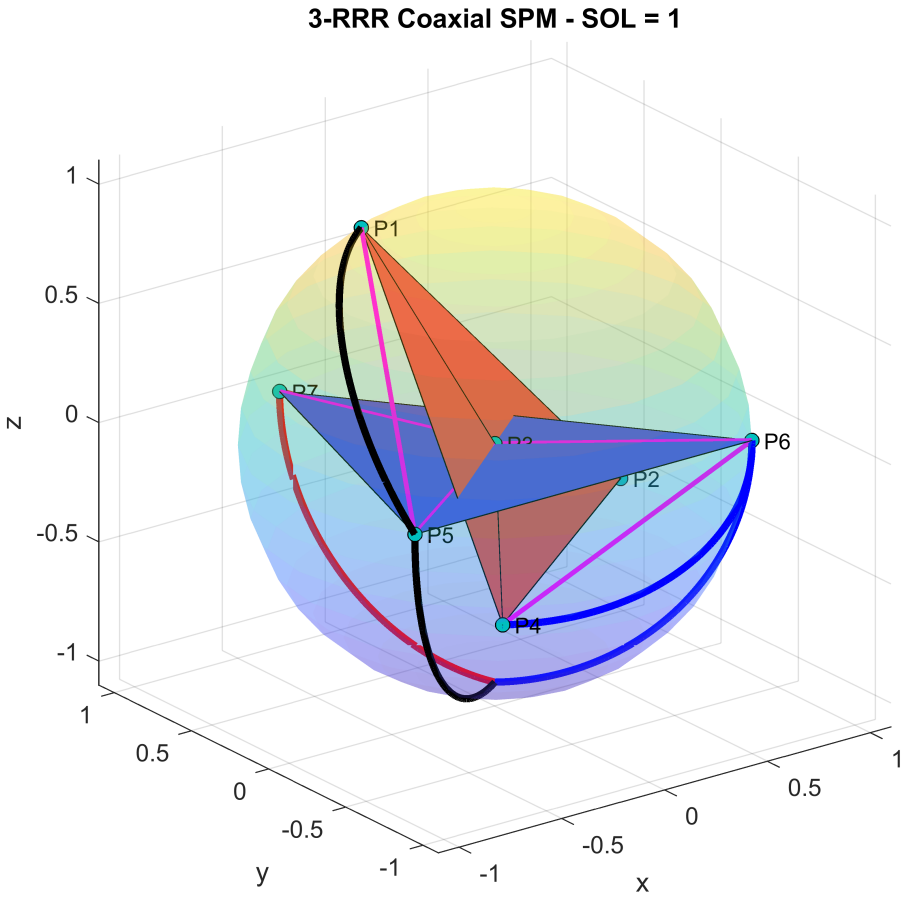



***Figure 4.9:*** Schematic Example of the Robot.

As it can be possible, the **workspace** it has been identified as a sphere, neglecting the limitations due to both the physical constrains and singularities. The **arches** represent the proximal and distal link of each chain of the robot (1° leg-black, 2° leg-blue, 3° leg-red), while the **violet lines** symbolize the squared distances, used in the whole process. The **center of the mechanism (CM)** is the point $P_3$, while the points of the **platform** are $P_1$, $P_2$ and $P_4$. However the points $P_5$, $P_6$ and $P_7$ represent the connection between the proximal and the distal link. As it is possible to see, the **two tetrahedra** that describe the platform (red one) and the fixed base (blue one) orientations degenerate into two triangles, centred in the CM.

In this representation all of the constant squared distances depend on the **radius of the sphere**, that it has been chosen unitary. It follows the $S^{7x7}$ **matrix** of the squared distances

that are involved into the seven points $(P_1, P_2, ..., P_7)$, representative of the mechanism.

$$S = \begin{bmatrix} 0 & s_{12} & s_{13} & s_{14} & s_{15} & s_{16} & s_{17} \\ s_{21} & 0 & s_{23} & s_{24} & s_{25} & s_{26} & s_{27} \\ s_{31} & s_{32} & 0 & s_{34} & s_{35} & s_{36} & s_{37} \\ s_{41} & s_{42} & s_{43} & 0 & s_{45} & s_{46} & s_{47} \\ s_{51} & s_{52} & s_{53} & s_{54} & 0 & s_{56} & s_{57} \\ s_{61} & s_{62} & s_{63} & s_{64} & s_{65} & 0 & s_{67} \\ s_{71} & s_{72} & s_{73} & s_{74} & s_{75} & s_{76} & 0 \end{bmatrix} = \begin{bmatrix} 0 & 3 & 1 & 3 & 2 & ? & ? \\ 3 & 0 & 1 & 3 & ? & ? & 2 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 3 & 3 & 1 & 0 & ? & ? & ? \\ 2 & ? & 1 & ? & 0 & 2.6840 & 2.5176 \\ ? & ? & 1 & ? & 2.6840 & 0 & 3.6383 \\ ? & 2 & 1 & ? & 2.5176 & 3.6383 & 0 \end{bmatrix} \tag{4.71}$$

where the **constant squared distances** have been calculated as follows:

$$\hat{p}_{50} = [r_{sphere}, 0, 0, 1]^T \qquad \hat{p}_{60} = \hat{R}_z\left(\frac{2\pi}{3}\right)\hat{p}_{50} \qquad \hat{p}_{70} = \hat{R}_z\left(\frac{4\pi}{3}\right)\hat{p}_{50} \tag{4.72}$$

$$\hat{R}_1 = \hat{R}_z(\vartheta_1), \quad \vartheta_1 = 15^\circ \qquad \hat{R}_2 = \hat{R}_z(\vartheta_2) \quad \vartheta_2 = 5^\circ \qquad \hat{R}_3 = \hat{R}_z(\vartheta_3) \quad \vartheta_2 = 30^\circ \tag{4.73}$$

$$\hat{p}_5 = \hat{R}_1\hat{p}_{50} \qquad \hat{p}_6 = \hat{R}_2\hat{p}_{60} \qquad \hat{p}_7 = \hat{R}_3\hat{p}_{70} \tag{4.74}$$

$$s_{56} = ||p_5 - p_6||^2 = 2.6840 \qquad s_{57} = ||p_5 - p_7||^2 = 2.5176 \qquad s_{67} = ||p_6 - p_7||^2 = 3.6383 \tag{4.75}$$

$$s_{35} = s_{36} = s_{37} = s_{13} = s_{23} = s_{34} = r^2_{sphere} = 1 \tag{4.76}$$

$$s_{12} = s_{14} = s_{24} = (\sqrt{3}r_{sphere})^2 = 3 \tag{4.77}$$

$$s_{51} = s_{46} = s_{72} = (\sqrt{2}r_{sphere})^2 = 2 \tag{4.78}$$

while the unknowns depend on the squared distance $s_{25}$

$$s_{25} = s_{17} = s_{16} = s_{45} = s_{46} = s_{47} = ? \tag{4.79}$$

$$s_{54} = \Psi_{5,3,2,1,4}(s_{52}) \qquad s_{74} = \Psi_{7,5,3,2,4}(s_{52}; s_{54}) \qquad s_{64} = \Psi_{6,7,5,3,4}(s_{654}; s_{74}) \tag{4.80}$$

This dependence can be seen in the *figure 4.10*, where it is shown the reduction of tetrahedra in number of two, as the theory says.

In the end, it has been attached the solution for the case that has been studied in *figure 4.11* and a *table 4.4.1.1* with the 7 Cartesian coordinates for the set of points.

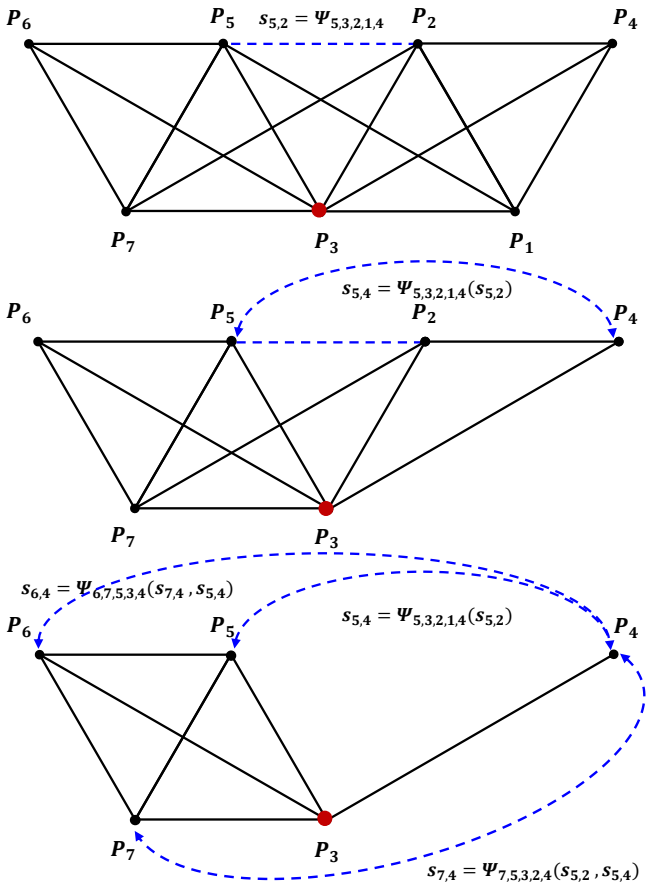



***Figure 4.10:*** Elimination of Tetrahedra.

| **Solution of Forward Kinematic** ($\vartheta_1 = 15°$, $\vartheta_2 = 5°$, $\vartheta_3 = 30°$) | | | | | | |
|---|---|---|---|---|---|---|
| $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_5$ | $P_6$ | $P_7$ |
| -0,3134 | 0,8411 | 0 | -0,5277 | -0,8192 | 0,8192 | -0,3420 |
| 0,4476 | 0,3061 | 0 | -0,7537 | -0,5736 | -0,5736 | 0,9397 |
| 0,8375 | -0,4459 | 0 | -0,3917 | 0 | 0 | 0 |
| -0,3134 | 0,8411 | 0 | -0,5277 | -0,8192 | 0,8192 | -0,3420 |
| 0,4476 | 0,3061 | 0 | -0,7537 | -0,5736 | -0,5736 | 0,9397 |
| -0,8375 | 0,4459 | 0 | 0,3917 | 0 | 0 | 0 |
| 0,3134 | -0,8411 | 0 | 0,5277 | -0,8192 | 0,8192 | -0,3420 |
| -0,4476 | -0,3061 | 0 | 0,7537 | -0,5736 | -0,5736 | 0,9397 |
| 0,8375 | -0,4459 | 0 | -0,3917 | 0 | 0 | 0 |
| 0,3134 | -0,8411 | 0 | 0,5277 | -0,8192 | 0,8192 | -0,3420 |
| -0,4476 | -0,3061 | 0 | 0,7537 | -0,5736 | -0,5736 | 0,9397 |
| -0,8375 | 0,4459 | 0 | 0,3917 | 0 | 0 | 0 |

***Table 4.3:*** 4 Solutions of Forward Kinematics.

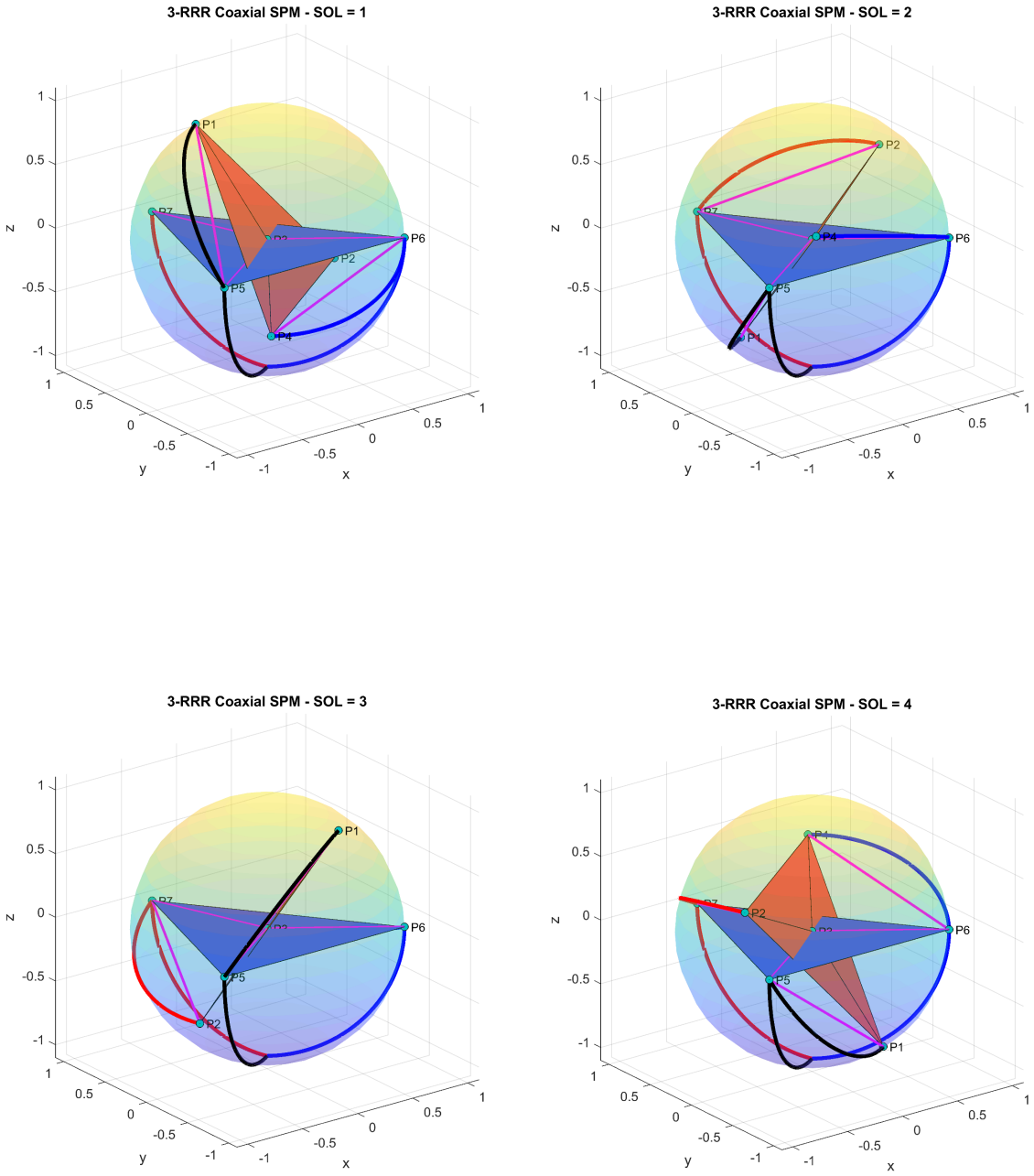



***Figure 4.11:*** Solutions for the Forward Kinematics.

#### 4.4.1.2 Explanation of Matlab Code

As it has been mentioned above, Thomas' method is based on the squared distances and a recursive trilateration, in order to get coordinates after solving the polynomial equations. Moreover, it needs to be better explained the analytical algorithm used to perform the forward kinematics, that you can find in the main `FDA.m`, being in the attachments chapter.

The **inputs** of the method are three actuator angles `M.t1, M.t2, M.t3` and the unit radius of the sphere `M.r_sphere`. Starting from these values and using the trigonometrical relationships on the sphere, it possible to define the known squared distances, computed by the function `square_distances`.

After initialising the matrix `S` as a symbolic one, it has been filled the same matrix with the calculated squared distances, and it has been made symmetric, by the function `Symmetrize`. In this way, `S` with known and unknown squared distances becomes the input for the following step.

Then, it has been defined the **possible strips of tetrahedra** among the 7 points that represent the robot in the spherical workspace. In the line below

`ST= { [3 5 7 2 6] [2 3 5 1 6] [1 2 3 4 6] }`

the cell `ST` collects 3 possible strips defined as $ST = [P_i,\ P_j,\ P_k,\ P_l,\ P_m]$, in which the 2 terminal points $P_l,\ P_m$ do not belong to any other neighbouring tetrahedra, according to the *figure 4.12*. This indexation to find the unknowns, will be useful to solve the polynomial roots in the next steps.

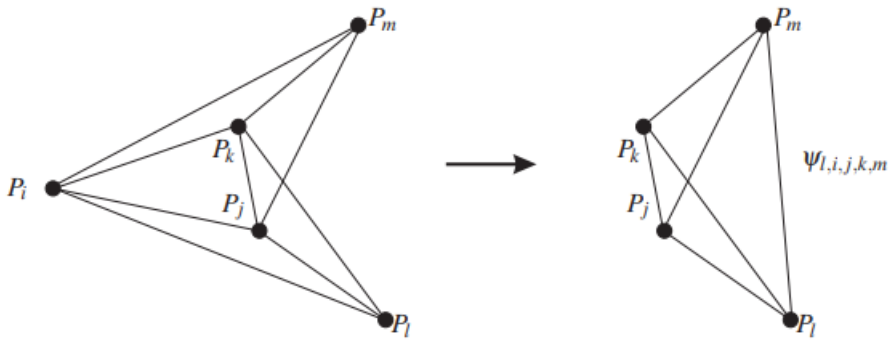



***Figure 4.12:*** Reduction of Tetrahedra.

Another indexation regards **trilateration method** to define the order of the points to get the respective coordinates.

`T= { [5 6 7 3] [5 3 7 2] [5 2 3 1] [1 2 3 4] }`

`SG= { [0 0 0 1] }`

Starting from the three fixed points of the base $P_5,\ P_6,\ P_7$, it is possible to find the following points in the strip as you can see in the *tree diagram 4.13*. Per each four numbers, it is possible to choose the fourth, up or down the base, identified by the previous three numbers,

in the sequence. After computing the tree diagram, it will get 16 branches, corresponding to 16 solutions, in which 8 out of 16 are exactly the symmetrical one of the others. The value

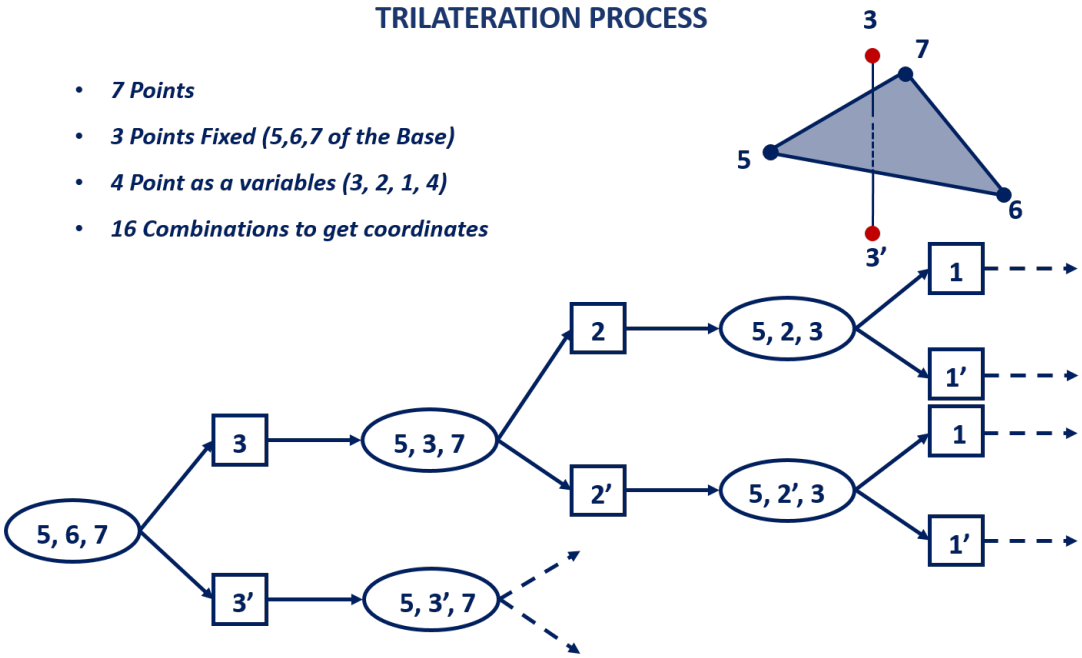



***Figure 4.13:*** Trilateration Process.

`SG=0` or `SG=1` regards the possibility to freely choose or not the detected point. If it is `SG=0`, the point can be both directions, if it `SG=1` the orientation in the Cartesian space is fixed. In this case, only the point $P_4$ is fixed, because is coplanar to $P_1$, $P_2$, being the extremities of the mobile platform.

After the trilateration indexation, it has been **calculated the variable squared distances**, as mentioned in the *formulation 4.63* with the sub-function `FillDistance`. Then, by clearing radicals with the sub-function `EliminateSquareRoot` it is possible to achieve the **univariate closure polynomial** with the function `UCPolynomial`.

The function `SolveSystem` finds the **roots of the polynomial** and returns the roots as a cell array `R` and the respective variables `V` as a row/column of the matrix `S`.

The last step has been to **get the coordinates** for the set of 7 points per each solution, that correspond to the several values of the matrix `S`. The function `GetCoordinates` realises the recursive trilateration process, mentioned before, recalling the sub-functions `FixVariable`, `RTrilaterate` and `Trilaterate`.

In the end, it has been **plotted the 16 solutions** with the function `DrawRobot`, using the set of coordinates, calculated in the previous steps.

### 4.4.2 Assembly Mode and Non Trivial Solutions

Considering Thomas' approach, it is clear how the direct kinematic solutions are obtained by solving a univariate polynomial of degree 16, which means that *there is no way to designate each solution to a particular assembly mode* [5].

A natural sorting criterion is that the direct kinematic solution should be reachable through continuous motion from the initial assembly mode without crossing a singularity.

However, in our case, although we obtained 4 non trivial solutions, it possible to get only 2 solutions out of 4, after choosing an assembly mode.

In fact, by mounting the distal links on the proximal one, it is possible to choose 2 types of assembly mode (*figure 4.14*):

- **counter clockwise**;
- **clockwise**.

Depending on these assembly modes, realising the all of the four solutions is possible only by crossing the singularities among the legs (it means breaking the robot) or by disassembling the robot and rearranging the distal links in the opposite configuration.

This relationship between the solutions of the forward kinematics and the assembly mode is possible to see in the *figure below 4.15, 4.16*.

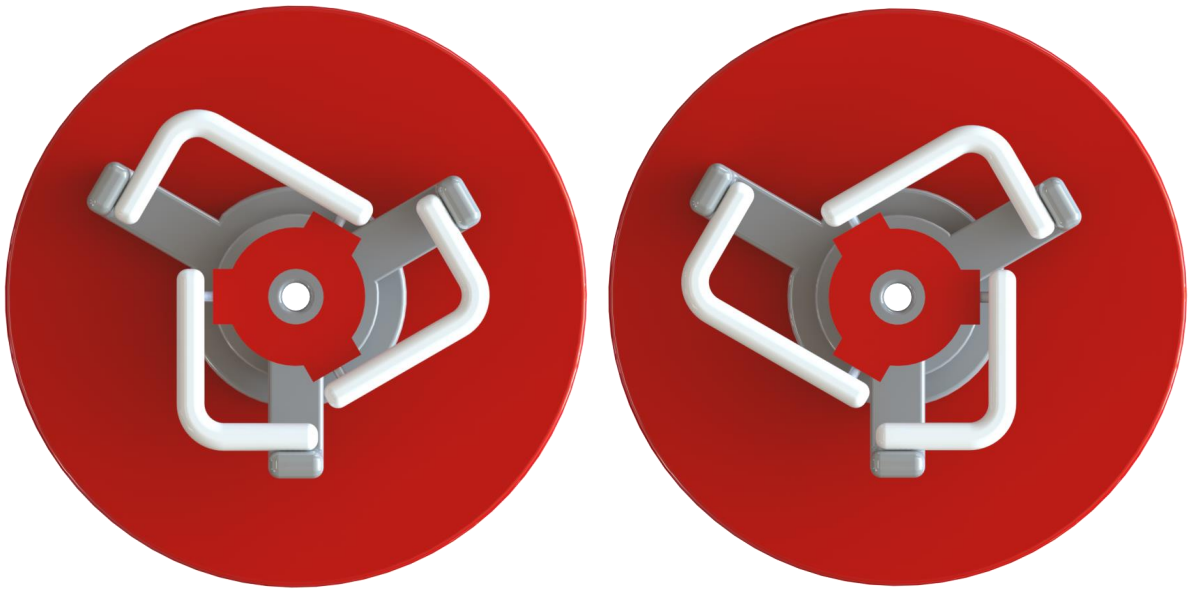


***Figure 4.14:*** Assembly Mode 1 (left) and Assembly Mode 2 (right).

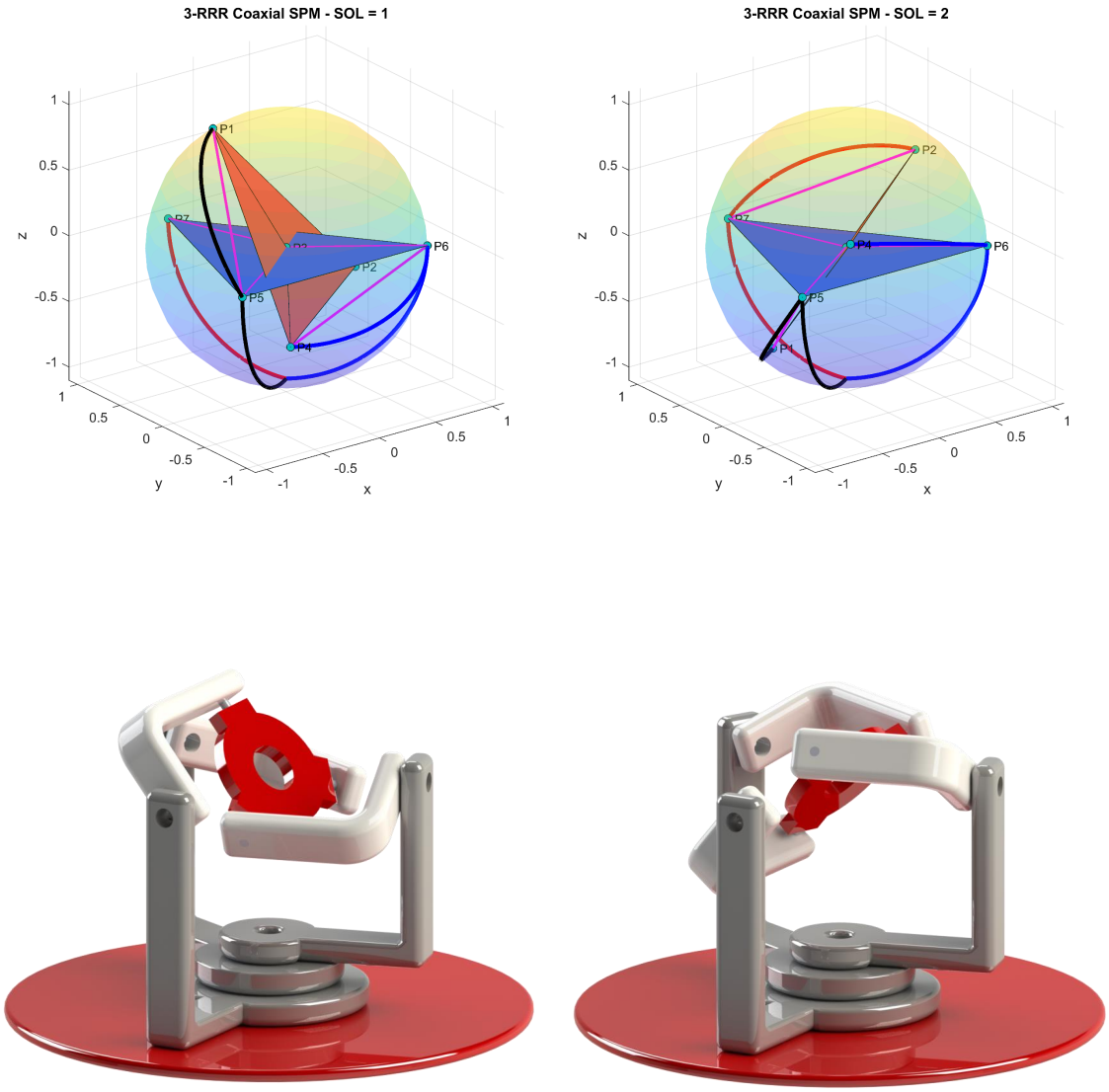



***Figure 4.15:*** Assembly Mode 1 and Solutions 1 and 2.

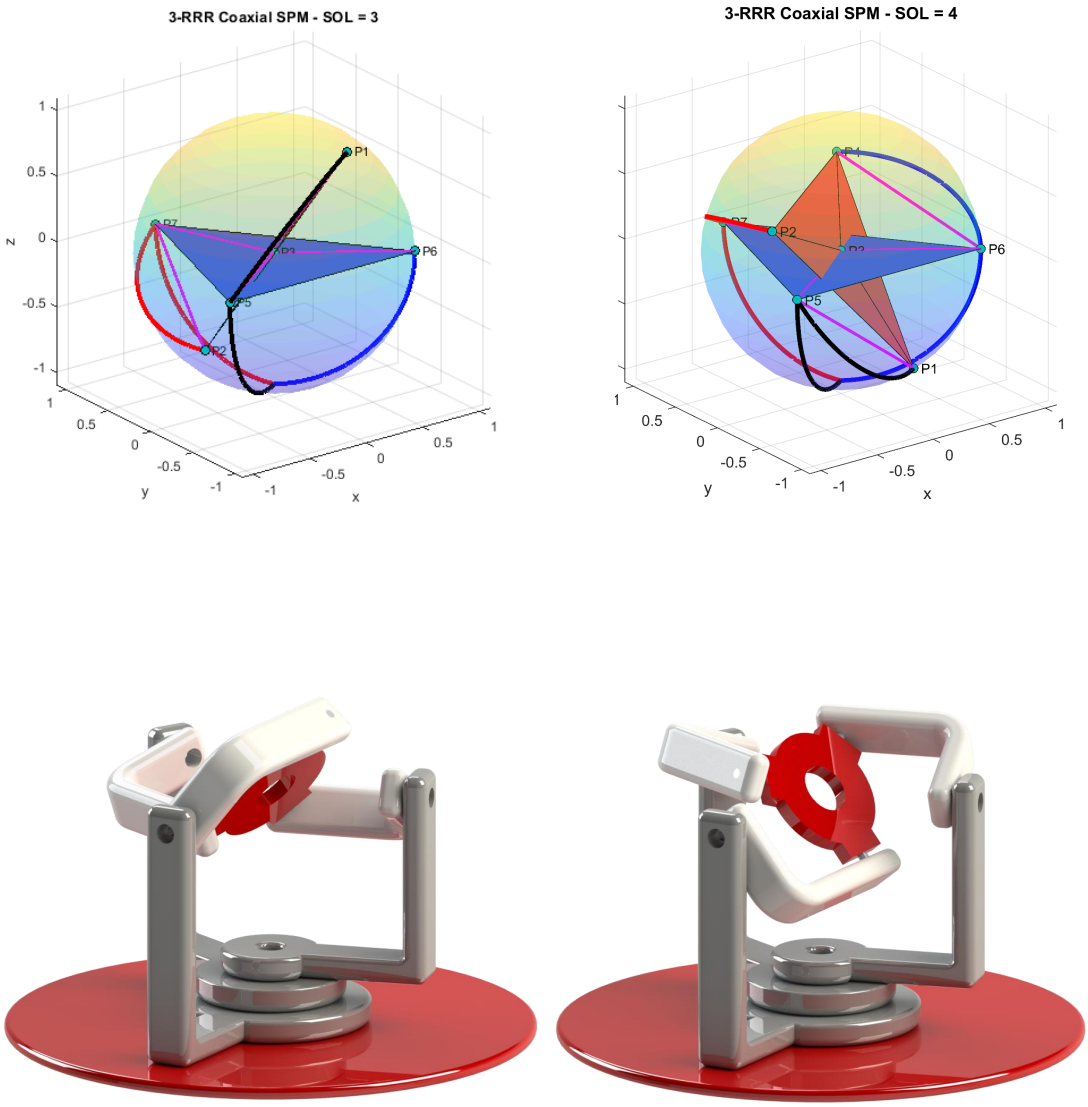



***Figure 4.16:*** Assembly Mode 2 ans Solutions 3 and 4.

### 4.4.3 Considerations

First, **the used algorithm gets 16 solutions**, because the code has been thought to be the most general possible. In fact, after defining the base through the 3 respective points, the positioning of the point $P_3$ (CM) can be choose both upper or under this base.

In our case, given that the two tetrahedra collapse into two triangle with the same barycentre, the point $P_3$ will be always the same positioning. Therefore, 8 solutions out of 16 are exactly the same.

Moreover, the **shown solutions are 4** in this example, because there are complex conjugate solutions for the input parameters $\vartheta_1,\ \vartheta_2,\ \vartheta_3$ [**?**].

However, it is possible **to increase the number of the real solution**, playing with the values of the characteristic parameters of the robot, as it can be seen in *figure 4.17*.

Through this optimisation, one pair of complex conjugate solutions first becomes a double root and finally, by separations, two different real roots.

At the same time, you have to make sure that two of already existing real solutions get too close, becoming before a double solution and after a complex conjugate one.

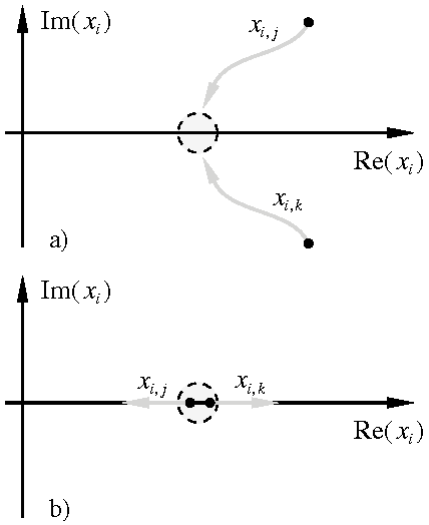



***Figure 4.17:*** Getting Two Real Solution from a Couple of Complex Conjugate One.

# Chapter 5

# Workspace and Singularities Analysis

## 5.1 Introduction

In this chapter it has been treated the relationship between the characteristics parameters of the robot and the both singularities and workspace limits. As you can see in the *figure 5.1*, the angles $\alpha_1$ and $\alpha_2$ are the two that influence either the motion mobility range and the number and types of singularity of the robot.

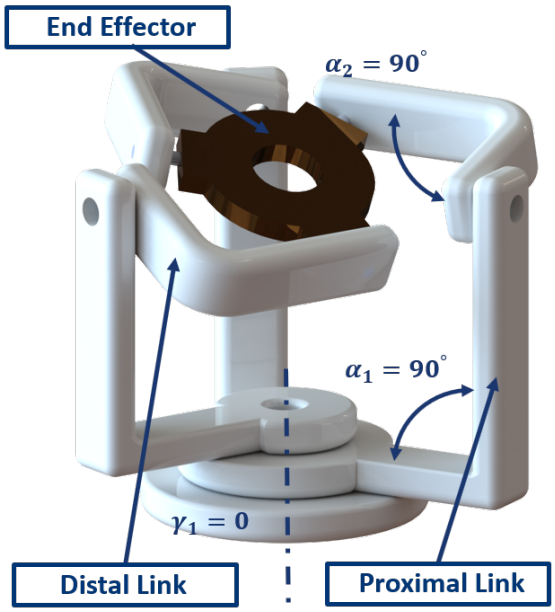



***Figure 5.1:*** Characteristics Parameters of the Robot.

In this case, choosing the angles $\alpha_1 = \alpha_2 = \pi/2$ and $\gamma_1 = 0$ has led to do different considerations, that will be better analysed in the following sections:

- **workspace**: in this way, the coaxial configuration ($\gamma_1 = 0$) led to have the whole range of mobility for each proximal link on the sphere. In theory, they can rotate between $0 - 360$ deg. Moreover, the angles $\alpha_1 = \alpha_2 = \pi/2$ increase the mobility range of the workspace in all of the sphere.

- **number and types of singularities**: if the mobility range increases, the number of singularities grows up too, when the characteristics angles $\alpha_1 = \alpha_2 = \pi/2$.

In the end, it is possible to argue that the optimisation of the mobility range and the singularities is a trade-off; a better workspace carries on more singularities that have to be avoided, during a path motions in the manipulations of the robot.

## 5.2 Workspace Analysis

The mobility region for each leg is given by the set of possible orientations that the end effector can achieve, given the link dimensions of this leg. This region is bounded by the **singularity surface**, which can be found for each leg as the closed surface separating the region where the leg has mobility from that in which it does not. The **global mobility range** is, at the end, the intersection of all these regions [2].

The singularity surface can be firstly obtained, considering the equation of the inverse kinematics of the manipulator, in a geometrical approach.

It is possible to remind the formulation of the inverse kinematics in order to show what kind of condition reduce the workspace mobility.

As we have treated in the chapters before, it possible to express all of the unit vector of all the joint of the robot with respect to a fixed referent frame, attached to the base. Let it to write the expression of the unit vector, related to the characteristic angles of the robot:

$$u_i = \begin{bmatrix} -\sin\eta_i \sin\gamma_1 \\ \cos\eta_i \sin\gamma_1 \\ -\cos\gamma_1 \end{bmatrix} \qquad w_i = \begin{bmatrix} -s_{\eta_i} s_{\gamma_1} c_{\alpha_1} + (c_{\eta_i} s_{\vartheta_i} - s_{\eta_i} c_{\gamma_1} c_{\vartheta_i}) s_{\alpha_1} \\ -c_{\eta_i} s_{\gamma_1} c_{\alpha_1} + (s_{\eta_i} s_{\vartheta_i} - c_{\eta_i} c_{\gamma_1} c_{\vartheta_i}) s_{\alpha_1} \\ -c_{\gamma_1} c_{\alpha_1} + s_{\gamma_1} c_{\vartheta_i} s_{\alpha_1} \end{bmatrix} \tag{5.1}$$

$$v_i^* = \begin{bmatrix} -\sin\eta_i \sin\gamma_2 \\ \cos\eta_i \sin\gamma_2 \\ \cos\gamma_2 \end{bmatrix} \qquad \gamma_2 = \pi/3 \tag{5.2}$$

$$Q = Eul_{ZYX} = R_z(\alpha) R_y(\varphi) R_x(\psi) \tag{5.3}$$

$$v_i = Q {v_i}^* = [x_i,\ y_i,\ z_i]^T \qquad i = 1, 2, 3 \tag{5.4}$$

Using the geometrical constrain equation of the robot, it is possible to achieve the equations of the inverse Kinematics for each leg:

$$w_i \cdot v_i = cos(\alpha_2) \qquad i = 1, 2, 3 \tag{5.5}$$

By substituting the half tangent it is possible to make the equation algebraical, as follows:

$$A_i T_i^2 + B_i T_i + C_i = 0 \qquad T_i = tan(\vartheta_i/2) \qquad i = 1, 2, 3 \tag{5.6}$$

Rearranging the equation it possible to express the coefficients $A,\ B,\ C$, as a function of the characteristic angles and the component of the unit vector $v_i$:

$$A_i = y_i(\sin(\gamma_1)\cos(\alpha_1) - \cos(\gamma_1)\sin(\alpha_1)) - \cos(\alpha_2) \tag{5.7}$$

$$-z_i(\cos(\gamma_1)\cos(\alpha_1) + \sin(\gamma_1)\sin(\alpha_1))$$

$$B_i = x_i \sin(\alpha_1) \tag{5.8}$$

$$C_i = y_i(\sin(\gamma_1)\cos(\alpha_1) + \cos(\gamma_1)\sin(\alpha_1)) - \cos(\alpha_2) \tag{5.9}$$

$$-z_i(\cos(\gamma_1)\cos(\alpha_1) + \sin(\gamma_1)\sin(\alpha_1))$$

By considering the discriminant $\triangle_i = 0$ and simplifying the equation it is possible to obtain the following equations:

$$\triangle_i = B_i^2 - A_i C_i = 0 \tag{5.10}$$

$$[x_i^2 + (y_i\cos(\gamma_1) + z_i\sin(\gamma_1))^2](\sin(\alpha_1))^2 - [(y_i\sin(\gamma_1) - z_i\cos(\gamma_1))\cos(\alpha_1) - \cos(\alpha_2)]^2 \tag{5.11}$$

$$x_i\sin(\eta_i)\sin(\gamma_1) - y_i\cos(\eta_i)\sin(\gamma_1) + z_i\cos(\gamma_1) + D = 0 \qquad i = 1,2,3 \tag{5.12}$$

$$D = -\cos(\alpha_1 \pm \alpha_2) \tag{5.13}$$

Depending on the **value of the constant** $D$, the *equation 5.12* represents the function of two planes in a 3D cartesian space. The possible orientations of the gripper are in the region between the two planes and the intersections of the sphere. The equations *equation 5.12* separates the unit sphere into three regions: the region between the two planes is the attainable workspace and the other two are the immobility regions. The **two intersections** of planes and the sphere are **singularity curves** in the Cartesian space. The conic surface generated by *equation 5.11*, intersects with the unit sphere revealing the coincidence of two equations.

For each leg, you can see in *figure 5.2* the representation of the mobility range in the two case, where in green it has been plotted the sphere, in red the cone and in blue the two plane:

- **generic configuration**: $\alpha_1 = \pi/3,\ \alpha_2 = \pi/2$ and $\gamma_1 = \pi/3$;
- **coaxial configuration**: $\alpha_1 = \alpha_2 = \pi/2$ and $\gamma_1 = 0$.

If we considerer the *equation 5.13*, it is possible to define the **condition** under which the manipulator is **capable of producing all possible rotations**, maximising this function:

$$||D||_{max} = 1 \quad \Leftrightarrow \quad \begin{cases} \alpha_1 + \alpha_2 = \pi \\ \alpha_1 - \alpha_2 = 0 \end{cases} \quad \Rightarrow \quad \alpha_1 = \alpha_2 = \pi/2 \tag{5.14}$$

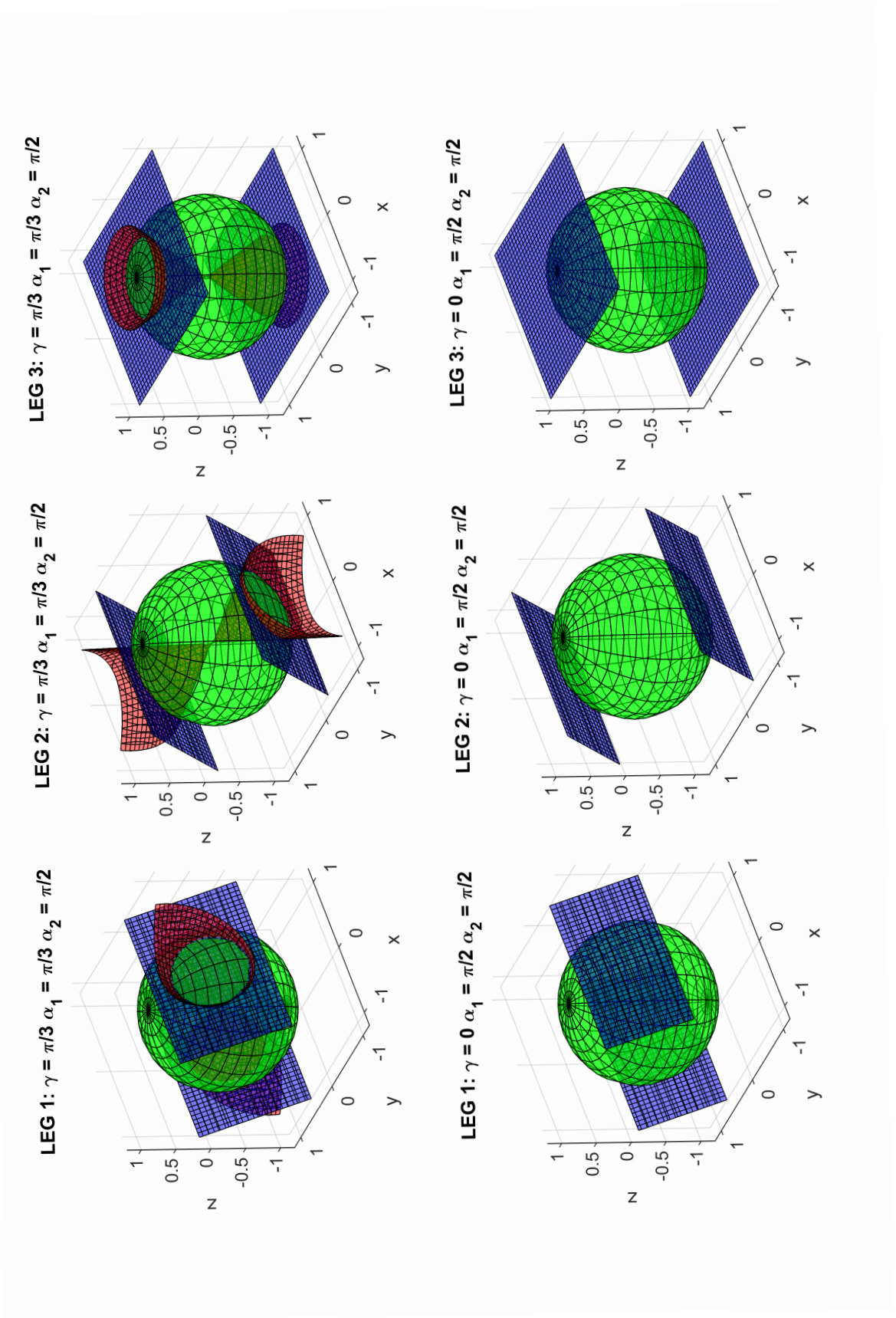



***Figure 5.2:*** Workspace for Each Leg.

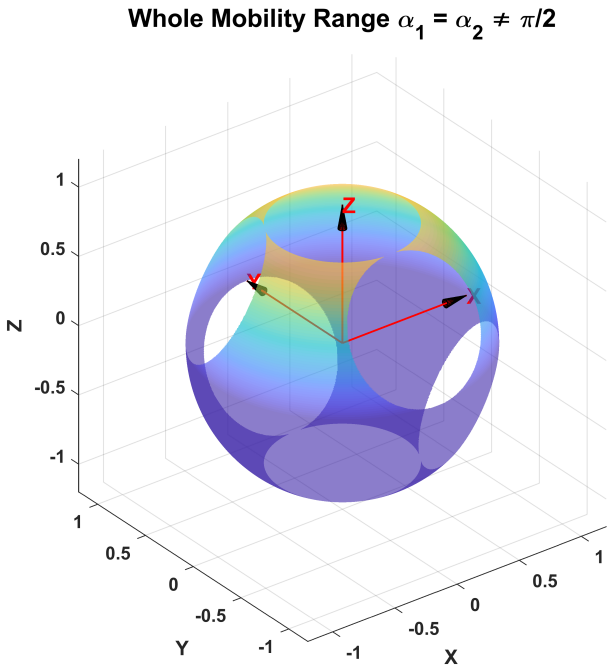



***Figure 5.3:*** Whole Workspace General Configuration.

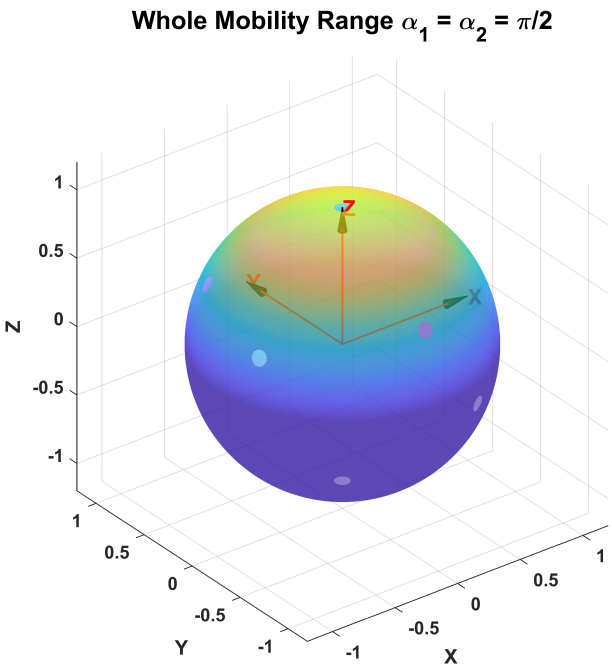



***Figure 5.4:*** Whole Workspace Coaxial Configuration.

The *equation 5.12* is independent from joint variables and relates the orientation to link dimensions only. It means it is possible to describe the capability of rotation in terms of **the architecture parameters of a leg** $\alpha_1,\ \alpha_2,\ \gamma_1$.

The parameter $D$ is very important because it determine the distance from the origin to the boundary plane, evaluating three cases of mobility regions of a single leg, as follows:

- $\alpha_1 \neq \alpha_2$: D has two distinct values, neither equal to the unit radius. The two planes thus intersect with the unit sphere, yielding three regions. The attainable workspace is the region bounded between two planes, while the immobility regions are outside the two planes. There are two singularity curves that are the intersections of the planes and the sphere.

- $\alpha_1 = \alpha_2 \neq \pi/2$: D has two distinct values and one being equal to the unit radius. There is only one plane that intersects the sphere, since the other is tangential to the unit sphere. The spherical surface is divided into two regions by the intersecting plane: one region is the attainable workspace, and the other is the immobility region. Only one singularity curve exists in this case, in addition to a singularity point at the tangential position.

- $\alpha_1 = \alpha_2 = \pi/2$: this last case is exactly the same of this prototype. D takes two identical values equal to the unit radius, which implies two planes are tangential to the unit sphere. The attainable workspace consists of the whole sphere, except the two tangential points. There is no singularity curve, but only two singularity points.

From this analysis, these three cases of singularity curves corresponds to three different kinds of mobility regions, where **the last case maximises the workspace**.

## 5.3 Singularities Analysis

### 5.3.1 Type of Singularities in a Parallel Robot

In this section, it has been analysed the singularities of this parallel robot, which has inputs and output, where the latter represents a set of actuated joints $\theta = [\vartheta_1,\ \vartheta_2,\ \vartheta_3]^T$ and the former is set of angle of the orientation of the End Effector in the Cartesian space $\phi = [\alpha,\ \varphi,\ \psi]$.

Firstly, it is possible to generally describe the different types of singularity in a parallel robot, considering the Jacobian Matrices that appear in the formulations. Jacobian matrix can be seen as the transformation matrix of differential kinematics from joint space to operation space [11, 7].

Let it starts explaining the implicit relationship between inputs and outputs of the robot kinematics and obtaining the equation that describe the input-output speed, by differentiating with respect to the time:

$$F(\theta,\ \phi) = 0 \tag{5.15}$$

$$A\dot{\phi} + B\dot{\theta} = 0 \tag{5.16}$$

$$A = \frac{\partial F}{\partial \phi} \qquad B = \frac{\partial F}{\partial \theta} \tag{5.17}$$

where **A** and **B** are the **Jacobian matrices** $3 \times 3$, depending on both the characteristic parameters of the robot and the configuration, determined by inputs and outputs.

The singularities appear in configurations where either **A** or **B** becomes singular. Therefore, it is possible to classify three kinds of singularity that have different physical interpretations:

- **First Singularity**:

$$det(B) = 0$$

  The corresponding configuration is one in which the chains stand on the boundary limits of the workspace. It means this kind of singularity is related to an **inverse kinematics** problems. It is possible to see this singularity in this way: even if the $det(B) = 0$, there is some non-null values of the actuators rate that correspond a null value of the Cartesian Velocity of the End Effector.

  Generally, these velocity would be orthogonal to the boundary and directed towards the outside of the workspace. In such a configuration, the output link *loses one or more degrees of freedom*, implying that it can resist one or more forces or moments without any torque-or force-at the powered joints.

- **Second Singularity**:

$$det(A) = 0$$

The singular configuration leads the gripper to be locally movable even if all the actuated joints are locked.

In this case, this singularity comes from the **forward kinematics** problem. In the direct kinematic problem, the values of the output variables from given values of the input variables should be obtained. So, it is possible to have a non-null Cartesian rates of the End Effector, even if the the input angle are a null-vector. In such a configuration, the *output link gains one or more degrees of freedom*, implying that the output link cannot resist one or more forces or moments even when all actuators are locked.

- **Third Singularity**:

$$det(B) = det(A) = 0$$

The third kind of singularity occurs when, for certain configurations, both , when **some specific conditions on the linkage parameters are satisfied**. This corresponds to configurations in which the chain can undergo finite motions when its actuators are locked or in which a finite motion of the inputs produces no motion of the outputs, such as a linkage having a constant branch.

### 5.3.2 Jacobian Matrices for 3-RRR Coaxial SPR

Before describing the process to obtain the Jacobian matrices, let it define the unit vectors of all the joints with respect to a fixed frame, attached to the base, as it has been mentioned above in the workspace analysis and in the chapters above.

$$u_i, \quad w_i, \quad v_i \qquad i = 1, 2, 3.$$

So for a certain configuration, after solving the inverse kinematics of a given orientation $R_{EE}$ of the End Effector, it is possible to know the input $\phi$ and output $\theta$ of the process, reminding that the Euler Angles convention ZYX it has been used.

In this case, the *equation 5.15* is the geometrical constrain [14] of the robot and by deriving both sides of time, it is possible to achieve this expression:

$$w_i \cdot v_i = cos(\alpha_2) \qquad i = 1, 2, 3 \tag{5.18}$$

$$\dot{w}_i \cdot v_i + w_i \cdot \dot{v}_i = 0 \tag{5.19}$$

Explicating the equation below, for $i = 1, 2, 3$, it is possible to achieve the system of three equations as follows:

$$\frac{\partial w_i}{\partial \vartheta_1} v_i \dot{\vartheta}_1 + \frac{\partial w_i}{\partial \vartheta_2} v_i \dot{\vartheta}_2 + \frac{\partial w_i}{\partial \vartheta_3} v_i \dot{\vartheta}_3 + \frac{\partial v_i}{\partial \alpha} w_i \dot{\alpha} + \frac{\partial v_i}{\partial \varphi} w_i \dot{\varphi} + \frac{\partial v_i}{\partial \psi} w_i \dot{\psi} = 0 \tag{5.20}$$

$$\begin{cases} a_1\dot{\vartheta}_1 + b_1\dot{\vartheta}_2 + c_1\dot{\vartheta}_3 + e_1\dot{\alpha} + f_1\dot{\varphi} + g_1\dot{\psi} = 0 \\ a_2\dot{\vartheta}_1 + b_2\dot{\vartheta}_2 + c_2\dot{\vartheta}_3 + e_2\dot{\alpha} + f_2\dot{\varphi} + g_2\dot{\psi} = 0 \\ a_3\dot{\vartheta}_1 + b_3\dot{\vartheta}_2 + c_3\dot{\vartheta}_3 + e_3\dot{\alpha} + f_3\dot{\varphi} + g_3\dot{\psi} = 0 \end{cases} \tag{5.21}$$

where the constant terms $a_i,\ b_i,\ c_i,\ e_i,\ f_i,\ g_i$ are:

$$a_i = \frac{\partial w_i}{\partial \vartheta_1} v_i \quad b_i = \frac{\partial w_i}{\partial \vartheta_2} v_i \quad c_i = \frac{\partial w_i}{\partial \vartheta_3} v_i \quad e_i = \frac{\partial v_i}{\partial \alpha} w_i \quad f_i = \frac{\partial v_i}{\partial \varphi} w_i \quad g_i = \frac{\partial v_i}{\partial \psi} w_i$$

Given that the unit vector $w_i$ concerns with only the degree of freedom $\vartheta_i$ and it is independent of $\vartheta_j$ and $\vartheta_k$, the matrices B is a diagonal one. On the contrary, the unit vector $v_i$ concerns with all the three value $\alpha_1,\ \varphi_i,\ \psi_i$.

In general, the constant terms are as a function of the characteristic angles of the robot, the inputs and the output, that have not been written here, to simplify the description.

This system can be translated into a matrix one, as mentioned below:

$$B\dot{\theta} + A\dot{\phi} = 0 \tag{5.22}$$

where **A** and **B** are respectively:

$$B = \begin{bmatrix} a_1 & 0 & 0 \\ 0 & b_2 & 0 \\ 0 & 0 & c_3 \end{bmatrix} \qquad A = \begin{bmatrix} e_1 & f_1 & g_1 \\ e_2 & f_2 & g_2 \\ e_3 & f_3 & g_3 \end{bmatrix} \tag{5.23}$$

If the matrix **A** is not singular it is possible to achieve the transformation matrix of differential kinematics from joint space to operation space:

$$\dot{\phi} = A^{-1}(-B)\dot{\theta} = J\dot{\theta} \tag{5.24}$$

The choice to use the Euler Angle ZYX convention leads it to considerer the Euler angles rate $\dot{\phi}$ as the angular velocity $\omega$ of the End Effector in the Operation space. In fact, the Jacobian Matrix transformation between these two quantities is the Identity matrix, when angular velocity of the body is resolved in fixed frame $R_0 - O_0 x_0 y_0 z_0$. Moreover, you can see how this convention does not introduce a singularity in a formulation, considering only the singularity of the robot.

$${}^0\omega = \begin{bmatrix} {}^0\omega_x \\ {}^0\omega_y \\ {}^0\omega_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \dot{\phi} \tag{5.25}$$

The *equation 5.24* above can be re-written in this way:

$$\omega = J\dot{\theta} \tag{5.26}$$

After this result, it is possible to express the Jacobian matrices in another way, directly considering the Cartesian angular velocity of the End Effector [2]:

$$\dot{w}_i \cdot v_i + w_i \cdot \dot{v}_i = 0 \tag{5.27}$$

$$\dot{v}_i = \omega \times v_i \tag{5.28}$$

$$\dot{w}_i = (u_i \times w_i)\dot{\vartheta}_i \tag{5.29}$$

Substituting this expressions in the differential equation, the matrices **A** and **B** are:

$$A = \begin{bmatrix} (w_1 \times v_1)^T \\ (w_2 \times v_2)^T \\ (w_3 \times v_3)^T \end{bmatrix} \qquad B = \begin{bmatrix} (w_1 \times u_1) \cdot v_1 & 0 & 0 \\ 0 & (w_2 \times u_2) \cdot v_2 & 0 \\ 0 & 0 & (w_3 \times u_3) \cdot v_3 \end{bmatrix} \tag{5.30}$$

This other way to express the Jacobian matrices leads to explain in a physical way what the different kinds of singularity mean for a spherical parallel manipulator, as listed below:

- **First Singularity**:

  $$det(B) = 0 \qquad (w_i \times u_i) \cdot v_i = 0$$

  It stands on the boundary of the workspace and it is related to the inverse kinematics. When this condition are verified, vector $u_i$, $w_i$, $v_i$ are coplanar, as you can see in the *figure 5.5*. When such a configuration is attained, a certain set of velocities of the gripper cannot be produced. At the end, It means the robot loses a dof of mobility, as represented in *figure 5.5* for each leg.

- **Second Singularity**:

  $$det(A) = 0 \qquad (w_i \times v_i) = 0$$

  This singularity appears when non zero angular velocities of the gripper are possible even if the three motors are locked. As you can see in *figure 5.10*, is it impossible to control the platform when it is an horizontal configuration, due to the value of the actuator angles $\vartheta_1 = \vartheta_2 = \vartheta_3$.

  This condition is when the three unit vectors $w_i$ and $v_i$ with $i = 1, 2, 3$ are coplanar and depend on the forward kinematics. In each case, a velocity of the gripper that leaves the actuators at rest is possible, and there exists a torque, which when applied to the gripper, could not be balanced and controlled by the actuators.

- **Third Singularity**:

  $$det(A) = det(B) = 0 \qquad \alpha_1 = \alpha_2 = \pi/2$$

  This singularity appears when the inverse and forward kinematics singularities meet. For this prototype it is means that all the proximal links collide into only one, as shown in *figure 5.7*, even if it is impossible to physically achieve because of the interference among the parts.

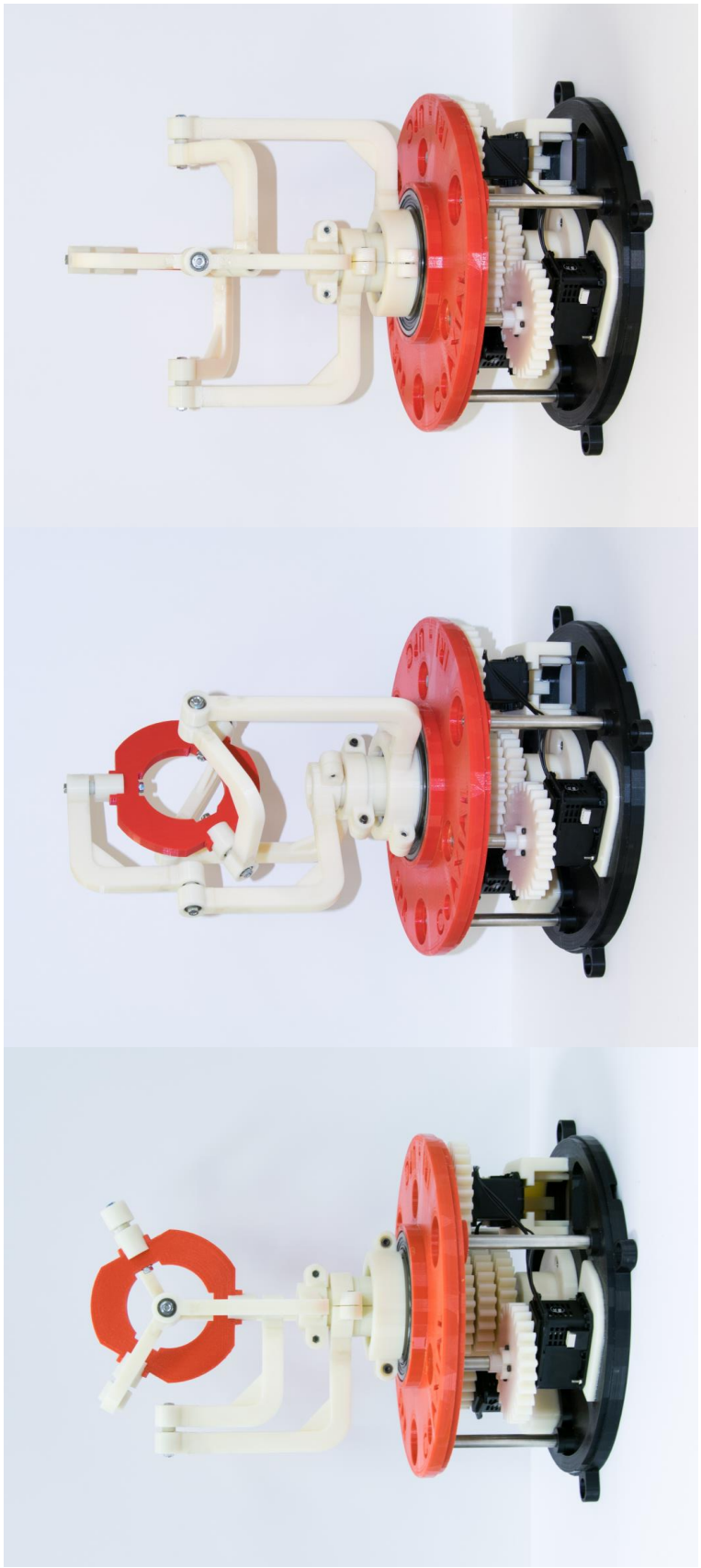


***Figure 5.5:*** Singularities of the Inverse Kinematics.

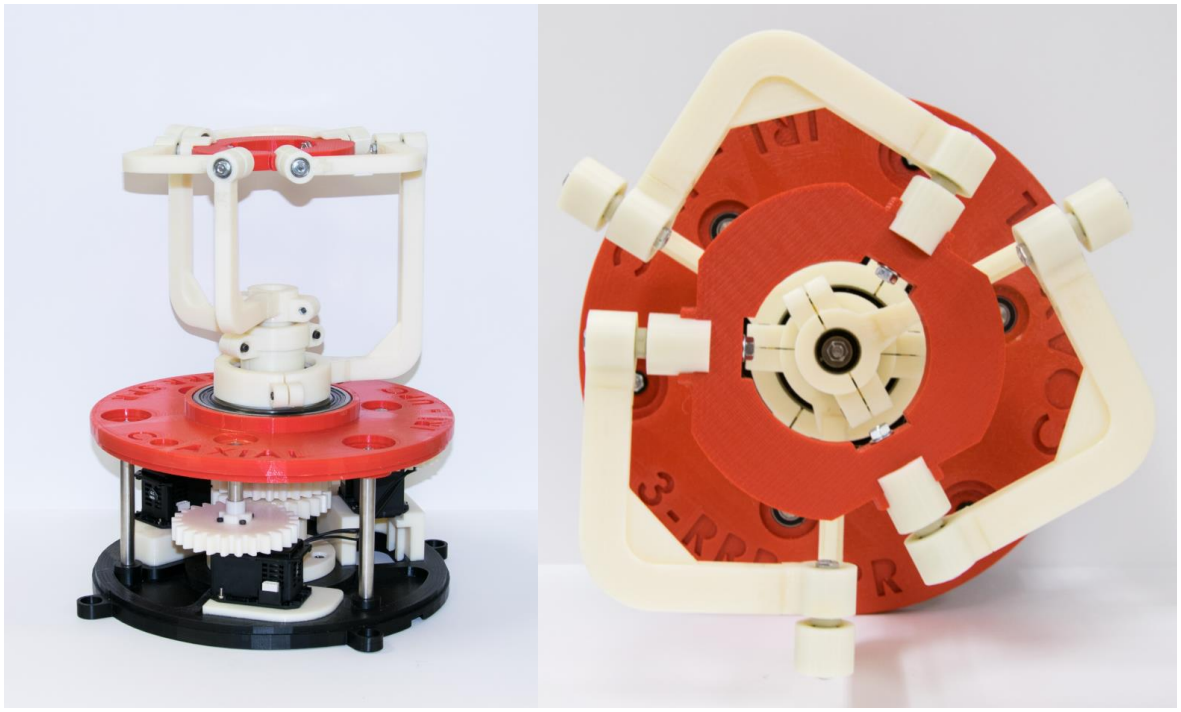


***Figure 5.6:*** Singularities of the Forward Kinematics.

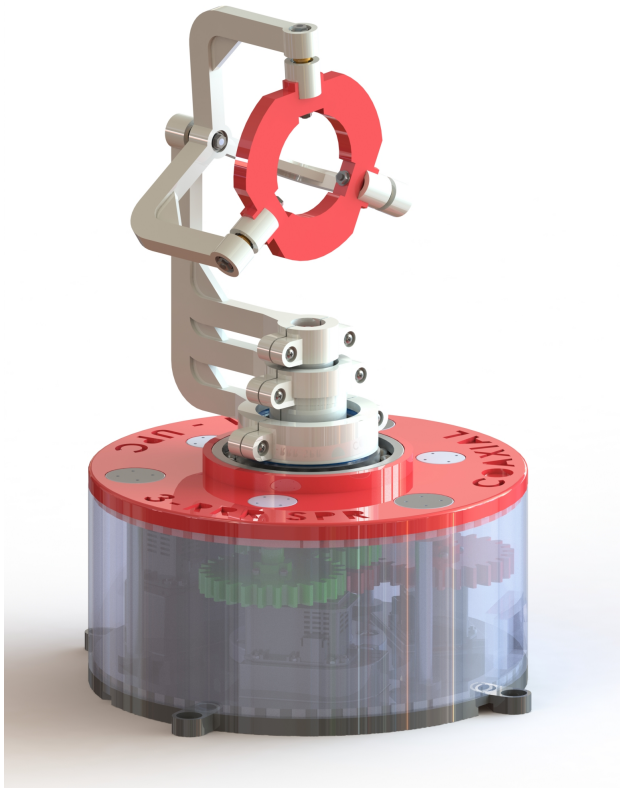


***Figure 5.7:*** Singularities of the Third Type.

## 5.4 Design Considerations

In the sections above It has been analysed and compared the difference between this two configurations in terms of singularities and workspace:

- **First Configuration**: $\alpha_1 = 90\ deg$, $\alpha_2 = 90\ deg$.
- **Second Configuration**: $\alpha_1 = 60\ deg$, $\alpha_2 = 90\ deg$.

Moreover, it is also important to show how the physical implementation of the prototype leads to different considerations in mechanical way.

By **keeping the same dimension of the End Effector**, you can see how in the configuration 2 the *whole workspace* increases. This aspect comes to, as a consequence, an higher dimension of the distal and proximal links, making more flexure problem under load.

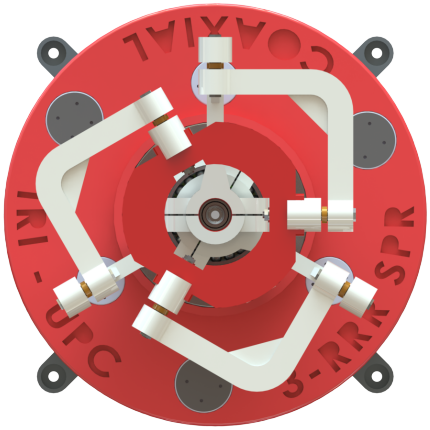



***Figure 5.8:*** Configuration 1 - Top View.

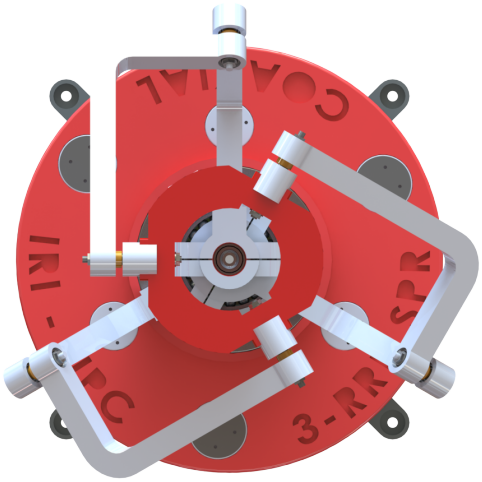



***Figure 5.9:*** Configuration 2 - Top View.

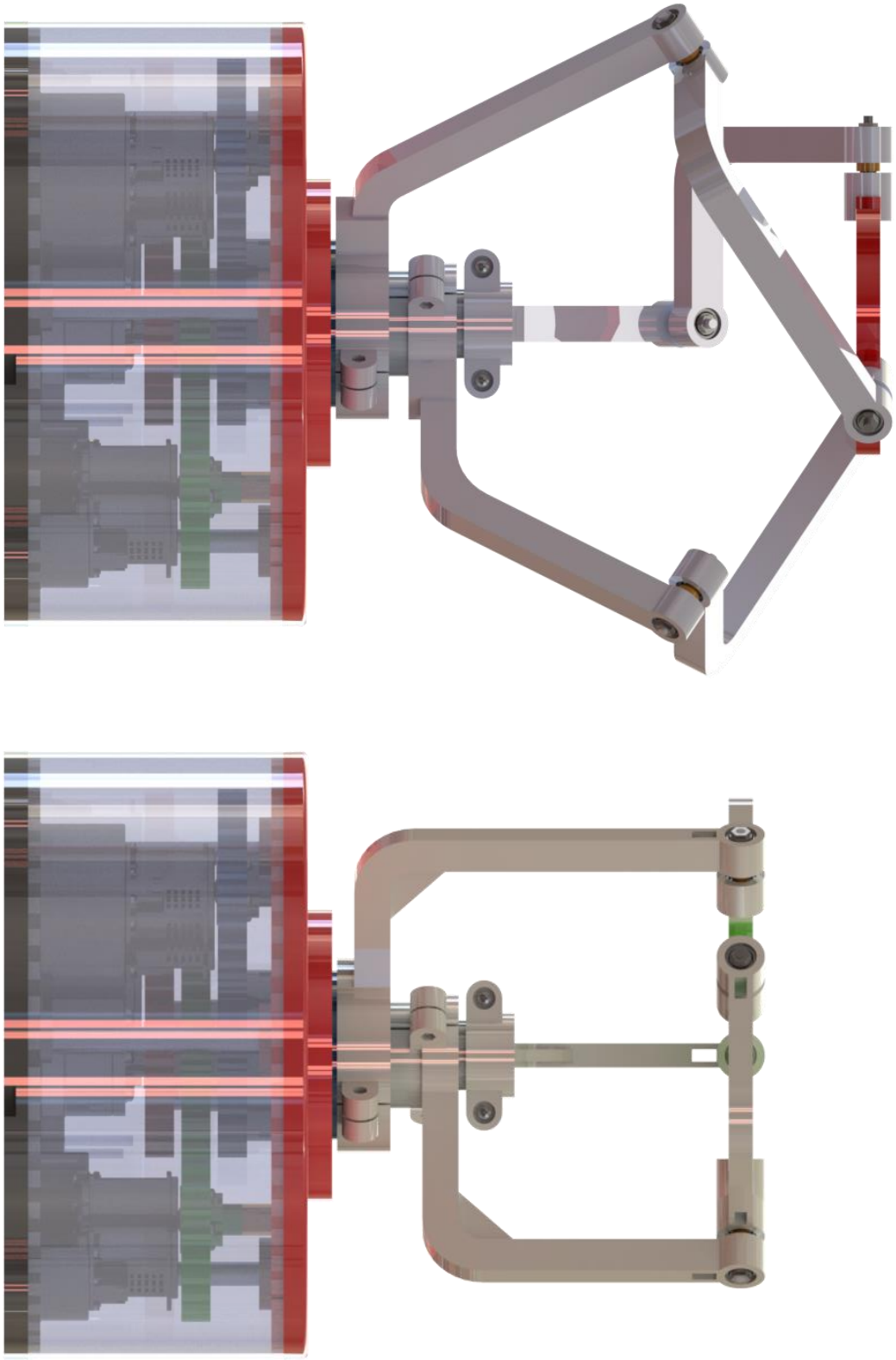


***Figure 5.10:*** Configuration 2 (left) and Configuration 1 (left) - Front View.

# Chapter 6

# On the Design of Coaxial 3-RRR SPM

## 6.1 Introduction

The first design of the 3-RRR Coaxial Spherical Parallel Motor is based on the use of three servomotors to actuate the respective degrees of freedom and a very simple driving system made of only one gear mates with 1:1 ratio, in order to realise the prototype very fast. In fact, most of the parts of the robot will be printed using *ABS plastics* (P430-Nat, 1.04 $g/cm^3$) and only the support parts and other threaded pivots will be in *High Speed Steel*.

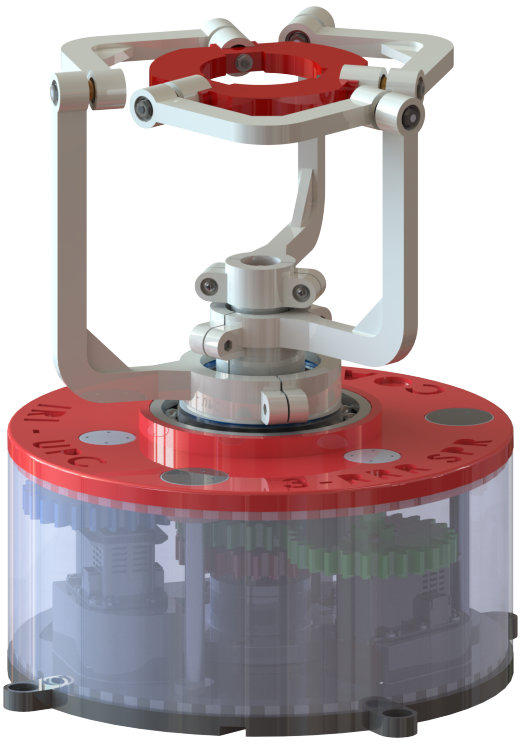


***Figure 6.1:*** 3-RRR Coaxial SPM.

For these reasons, it has been preferred to not complicate the transmission mechanism in order to focused to both the functioning and the optimization of the robot. Moreover, this concept is *reconfigurable and modular*, allowing to rearrange the robot configuration, changing the characteristic angles and printing only the legs of the robot in a easy and cheap way, so far as this does not change the transmission mechanism of the power.

At the same time, it will be postponed, as a future work, a new prototype with a complex transmission, using only one motor, three electro-brakes and a system with pulley and belt to move the robot, in order to reduce the operative costs of the robot itself.

## 6.2 Used Equipment and Printing Process

In order to print the CAD parts two different 3D Printers have been used, namely in the following *table 6.1* and shown in *figures 6.2*

| Company | 3D Printer | Building Size | Model Material | Layer Thinness |
|---|---|---|---|---|
| Proto3000 | 1200es Series | 254 x 254 x 305 mm | ABSplus-P430 | 0,330 mm |
| Stratasys | uPrint SE | 203 x 152 x 152 mm | ABSplus-P430 | 0,254 mm |

***Table 6.1:*** 3D Printes Used.

***Figure 6.2:*** uPrint SE (left) - 1200es Series (right).

Even if the tolerance of the two printers are more or less the same, it has been preferred to print several parts (eg. shafts, gears, etc...) for which the precision and the mechanical properties are highly important with the **uPrint SE** model, because the result of printing has been better than the other, after several attempts. The *base, the cover and the upper-base* have been printed with the **1200es Series** model, because of the needed dimensions of these parts.

The 3D-printer process uses the FDM$^{TM}$ Technology (fused deposition modelling), building the 3D model and its support material, layer by layer, from the bottom to the top on a removable modelling base.
The accuracy on the flat plane is 0.01 mm and the mechanical strength in this plane is bigger than in the other planes.

It was necessary to take into account of this printing features in the design of each model, making compromises among the optimization of the design, the limit and capability of the

3D printer and the functioning of the mechanism. This aspect leads it to complicate the drawing of the parts, splitting them into different ones, making the 3D printer to do its best.

It is possible to see in the *figure 6.3 and 6.4* below, different orientations for the same part, in order to show how the quality of a part can change, by only adapting the design.

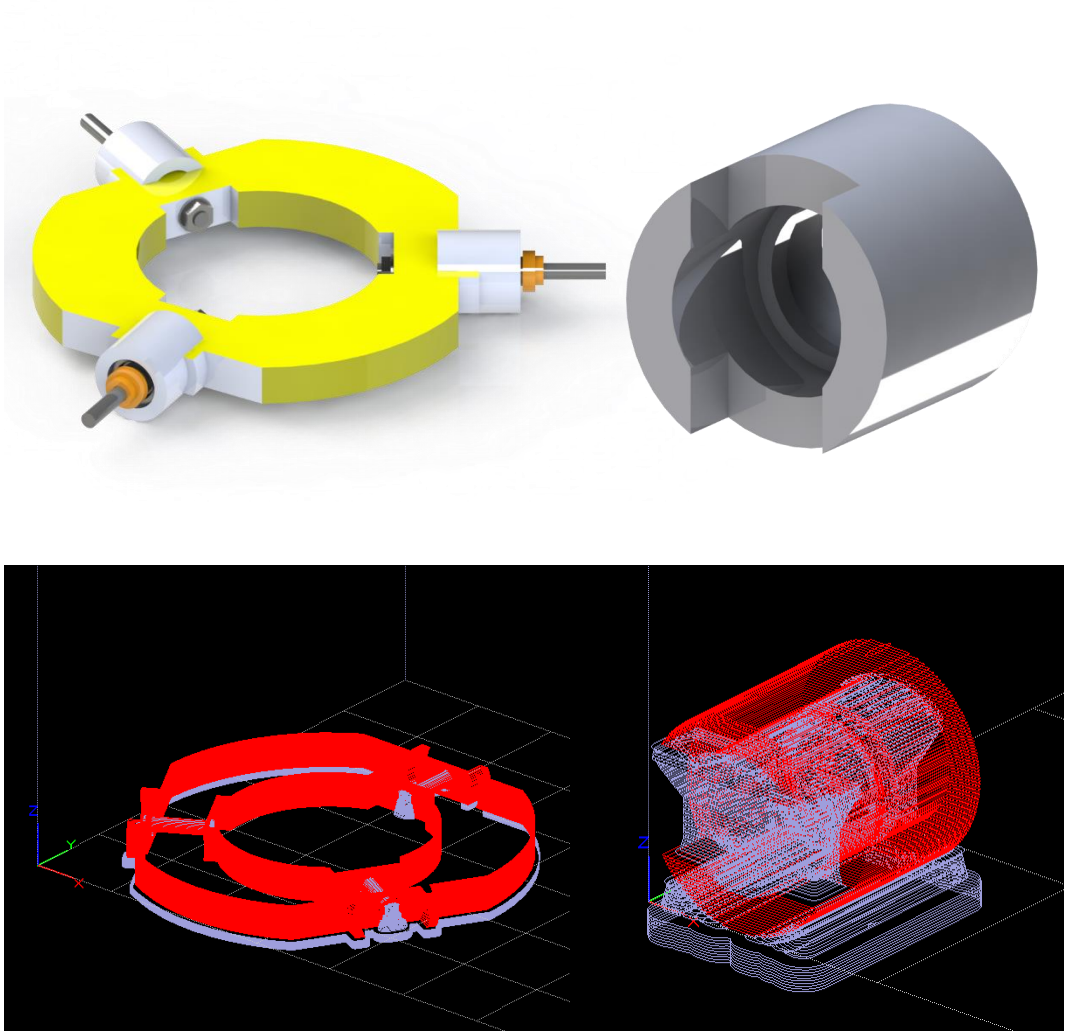


***Figure 6.3:*** Example as for a Limit Orientation.

A software, namely, **CatalystEx** provides to transform the CAD model (*part.sdlprt*), converted before in a *part.stl*, into a printed one, layer by layer as shown in *figure 6.3 and 6.4*. In the first orientation of both the two parts (End Effector on the left side, Housing bBearing on the right one) it has been obtained a bad surface and mechanical quality of either the cylinder and of the hole.

Moreover, there is more accuracy on the *plane xy* than on the *planes zx* and *zy* and the circle of the cylinder will be approximated with the precision of the layer thickness.

For these reason, it has been preferred to realise the End Effector and the respective housing for the bearing into two separated parts, saving quality and mechanical characteristics of the parts themselves, but also materials and waste of time.

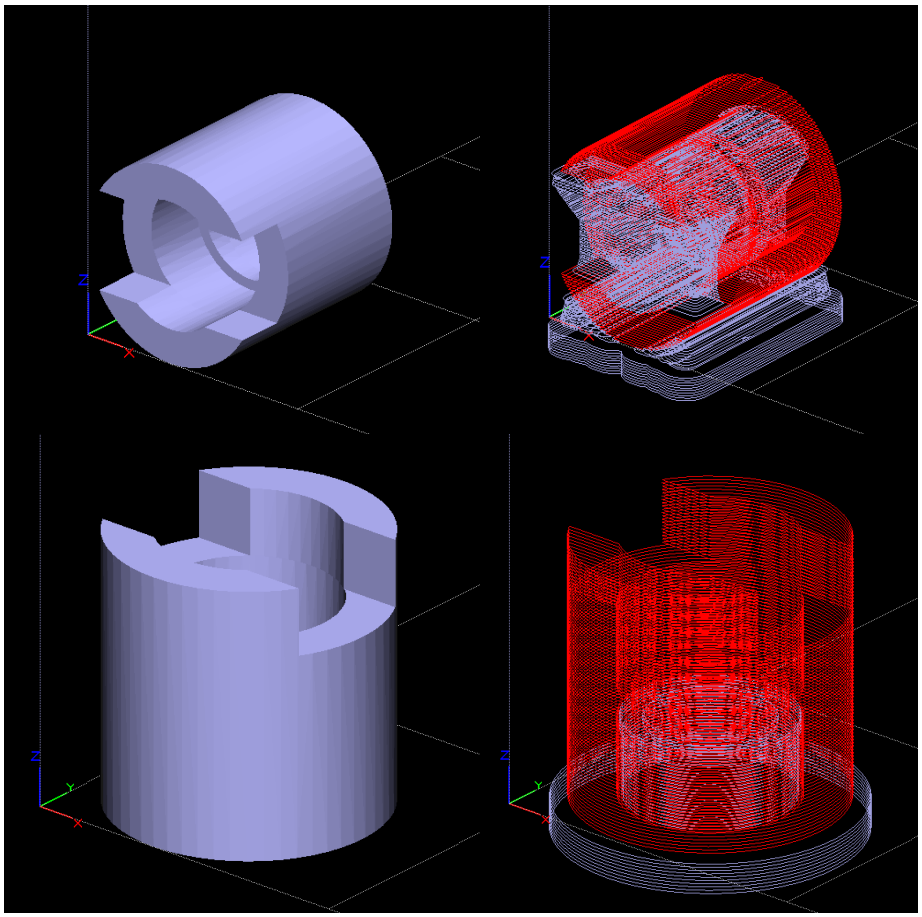


***Figure 6.4:*** Example as for a Limit Orientation.

## 6.3 Design Using Three Motors

As it can be seen in the *figure 6.3*, the **three motors** are fixed to the **base** using the screw-nut mate (M2.5) and they are arranged in a *symmetrical pattern*. In the middle of the base, it has been allocated, through the **support**, the mechanism composed of the **three coaxial shafts** with the respective couple of radial ball bearing. In the bottom, each shaft has an **in-built gear** that has been mated with the respective motor one.

Therefore, the gears are connected with the motor thanks to the relative **servo-horn** through a scree mate M2.5 (*figure 6.8*). To prevent the longitudinal bend due to the radial force generated during the engaging of the gear teeth, a **pivot** connects each gear to the u**pper platform**. It is fixed with respect to the gear, while it has been mated with a bearing to the upper platform, in order to allow the rotation of the gear with the pivot itself. A **bearings** it has been inserted on the upper-base, to prevent the same longitudinal bend for the shaft mechanism.

The **cover** with the upper platform close the transmission mechanism thanks to three threaded pivots (M5) and the respective mate screw-nut.

For making the **system reconfigurable**, it has been separated the **transmission system** from the **robot assembly** (*figure 6.5*). This choice has complicated the design, but, at the same time, allows to change the characteristic angle of the robot and to reprint it in a cheapest and fastest way.

Moreover, the **legs of the robot** have been used to do the **bearing arrangement** of the shaft mechanism (*figure 6.8*). In fact, through the building of the respective shaft abutment shoulder, the using of spacers and the particular realization of the terminal part of the leg it has been possible to complete the shaft bearing housing system of the shaft-mechanism.

As for the legs, the **distal links** and the **End Effector** (E.E.), these bodies have been divided into different parts, because them shape is very complex to realize in a single part and there are several driven orientation in contrast with each other, during the creation of the layer by the 3D printer (*figure 6.5, 6.7*). For these reasons, it has been preferred to separate that bodies to save the mechanical quality and characteristics such as stiffness, bend, torsion and density even though the design appear tricky. In particular, the head, the leg and the top have been fixed together in the mounting by using acetone, that is able to dissolve the ABS, jointing these parts and making only one body.

In the end, the realisation of the three independent degrees of freedom and the respective relative rotation among the legs, arms and the E.E. it has been possible thank to a **couple of ball bearing**, inserted in each head of these bodies and a pivot that realises this connection 8 *figure 6.11, 6.12*). The choice to use a couple of bearing has been made in order to reduced sensitivity the misalignment of the pivots during the mounting of the robot.

It follows in the table below, the **bill of material**s of the assembly.

| **BOM - Bill Of Materials** | | | |
|---|---|---|---|
| **ITEM NO.** | **PART NUMBER** | **MATERIAL** | **QTY.** |
| 1 | Base | ABS - Printed | 1 |
| 2 | Motor Dynamixel 64T | Engineering Plastic | 3 |
| 3 | Dynamixel-64T-Servohorn | High-Speed Steel | 3 |
| 4 | ISO 7045 - M2.5 x 10 - Z - 10S | Commercial Screw | 16 |
| 5 | ISO - 4035 - M2.5 - S | Commercial Nut | 24 |
| 6 | ISO - 4034 - M5 - N | Commercial Nut | 10 |
| 7 | Pivot-Base M5 | High-Speed Steel | 3 |
| 8 | Support | ABS - Printed | 1 |
| 9 | ISO 7045 - M2,5 x 16 - Z - 16S | Commercial Screw | 8 |
| 10 | ISO 4762 M3 x 8 - 8N | Commercial Screw | 3 |
| 11 | Washer ISO 7090 - 5 | Commercial Washer | 7 |
| 12 | ISO 7045 - M5 x 20 - Z - 20S | Commercial Screw | 4 |
| 13 | Spacer-Shaft1 | ABS - Printed | 1 |
| 14 | Shaft1 | ABS - Printed (m3, z26) | 1 |
| 15 | Spacer-Shaft2 | ABS - Printed | 1 |
| 16 | Shaft2 | ABS - Printed (m3, z26) | 1 |
| 17 | Spacer-Shaft3 | ABS - Printed | 1 |
| 18 | Shaft3 | ABS - Printed (m3, z26) | 1 |
| 19 | SKF - 61800 - 14,DE,NC,14-68 | Commercial Bearing | 1 |
| 20 | SKF - 61805 - 22,DE,NC,22-68 | Commercial Bearing | 2 |
| 21 | SKF - 61908 - 20,DE,NC,20-68 | Commercial Bearing | 2 |
| 22 | Flange-Pin | High-Speed Steel | 1 |
| 23 | Pin | High-Speed Steel | 1 |
| 24 | ISO - 4161 - M5 - N | Commercial Nut | 1 |
| 25 | Washer ISO 7089 - 4 | Commercial Washer | 16 |
| 26 | ISO 7045 - M4 x 16 - Z - 16S | Commercial Screw | 4 |
| 27 | Upper-Base | ABS - Printed | 1 |
| 28 | Bearings RS Pro 618 - 9979 | Commercial Bearing | 3 |
| 29 | Top | ABS - Printed | 3 |
| 30 | Cover | ABS - Printed | 1 |
| 31 | Semileg1-2 | ABS - Printed | 1 |
| 32 | Semileg1-1 | ABS - Printed | 1 |
| 33 | Semileg1-3 | ABS - Printed | 1 |
| 34 | Housing-Bearing-Leg | ABS - Printed | 9 |
| 35 | Bearings RS Pro 618 - 9890 | Commercial Bearing | 24 |
| 36 | Spacer-head | ABS - Printed | 12 |
| 37 | ISO - 4036 - M4 - N | Commercial Nut | 16 |
| 38 | ISO - 4032 - M3 - W - N | Commercial Nut | 6 |
| 39 | ISO 8738 - 3 | Commercial Washer | 6 |
| 40 | ISO 7045 - M3 x 16 - Z - 16S | Commercial Screw | 6 |
| 41 | Pivot Head M4 | High-speed steel | 3 |
| 42 | Spacer-Pivot-Bearing | ABS - Printed | 6 |
| 43 | Semileg2-2 | ABS - Printed | 1 |
| 44 | Semileg2-1 | ABS - Printed | 1 |
| 45 | Semileg2-3 | ABS - Printed | 1 |
| 46 | Semileg3-2 | ABS - Printed | 1 |
| 47 | Semileg3-1 | ABS - Printed | 1 |
| 48 | Semileg3-3 | ABS - Printed | 1 |
| 49 | Gear-Motor | ABS - Printed (m3, z26) | 3 |
| 50 | ISO 7045 - M2.5 x 12 - Z - 12S | Commercial Screw | 12 |
| 51 | Pivot-R1 | High-Speed Steel | 1 |
| 52 | Spacer-Pivot-R1 | ABS - Printed | 1 |
| 53 | Pivot-R2 | High-Speed Steel | 1 |
| 54 | Spacer-Pivot-R2 | ABS - Printed | 1 |
| 55 | Pivot-R3 | High-Speed Steel | 1 |
| 56 | EE | ABS - Printed | 1 |
| 57 | Housing-Bearing-EE | ABS - Printed | 3 |
| 58 | Pivot-EE M4 | High-speed steel | 3 |
| 59 | Arm | ABS - Printed | 3 |
| 60 | SKF - 6014 - 18,DE,NC,18-68 | Commercial Bearing | 1 |
| 61 | Top-Bearing | ABS - Printed | 3 |
| 62 | Tap | ABS - Printed | 2 |
| 63 | Washer ISO 7093-5 | Commercial Washer | 1 |
| 64 | SKF - 6300 - 6,DE,NC,6-68 | Commercial Bearing | 1 |

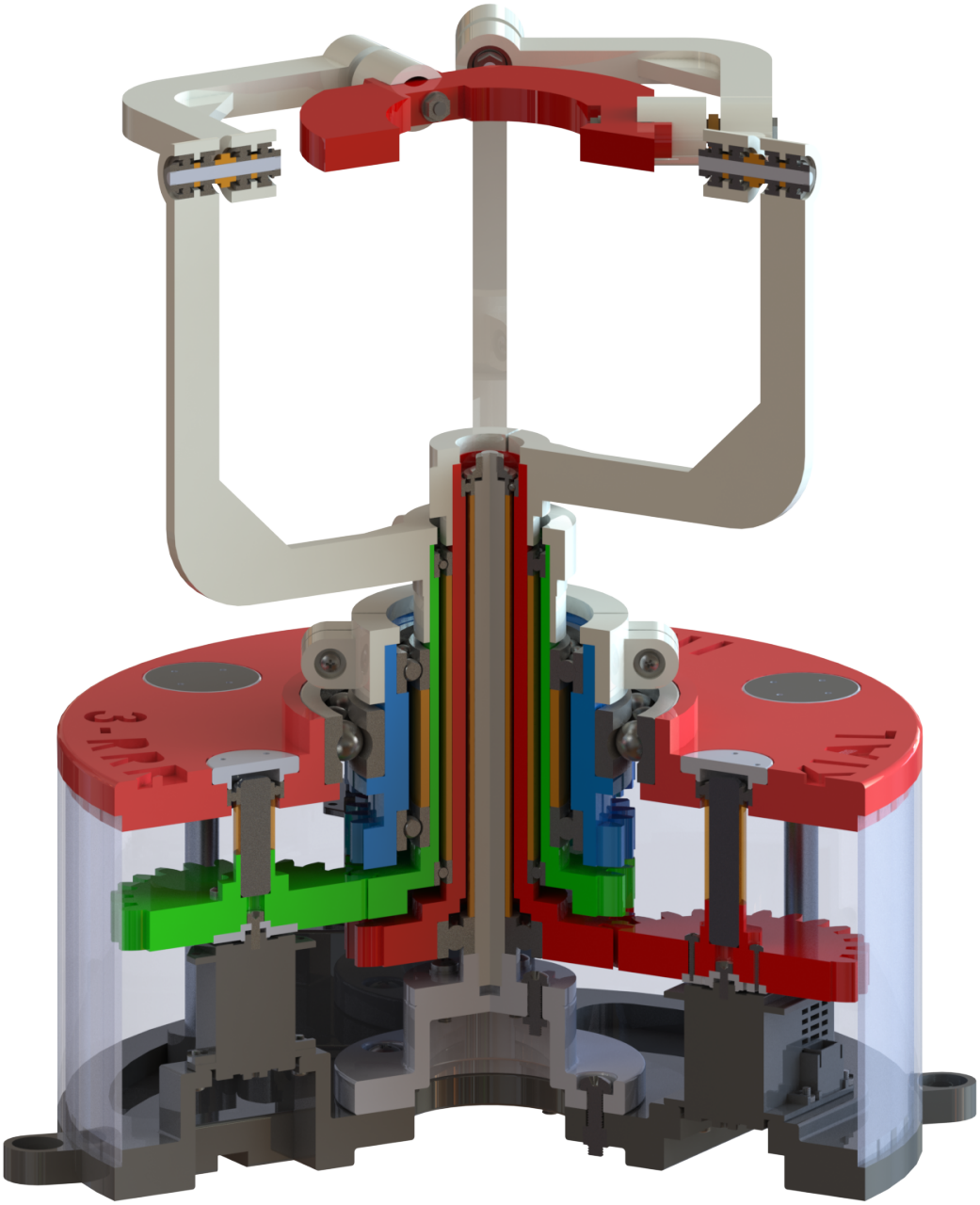



***Figure 6.5:*** Section View.

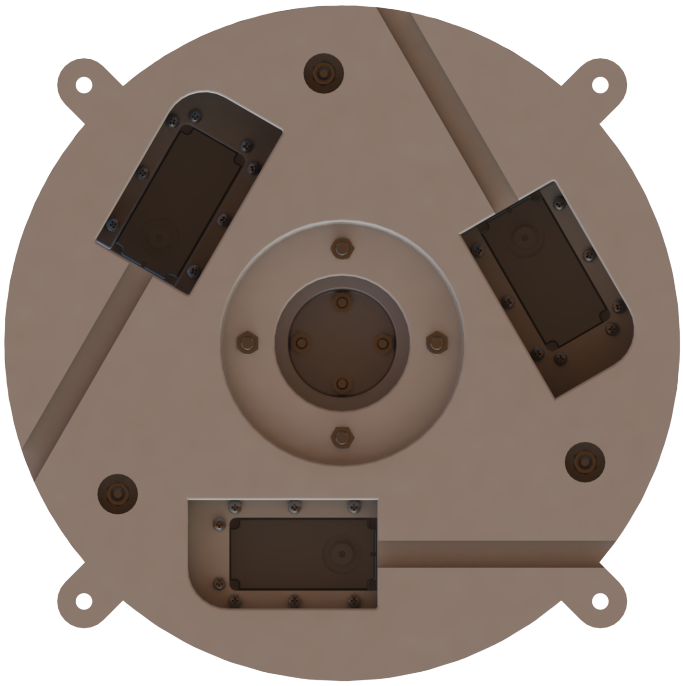


***Figure 6.6:*** Bottom View.

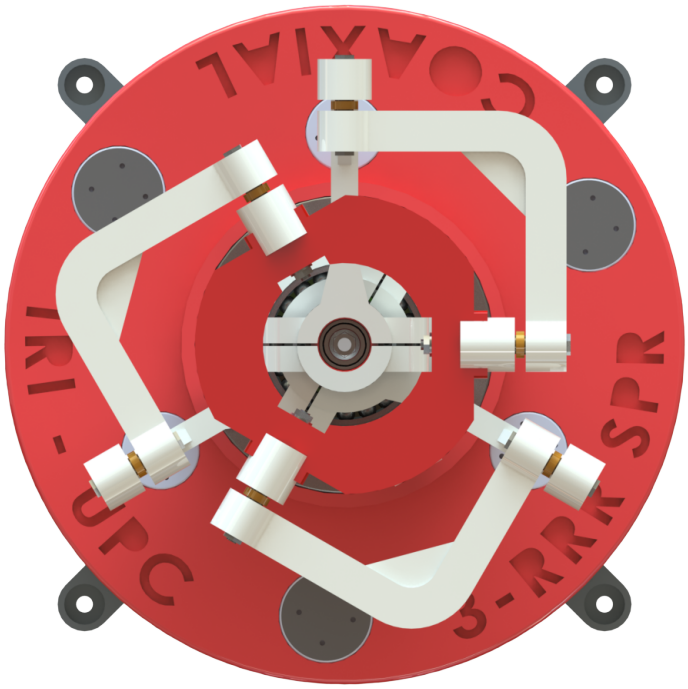



***Figure 6.7:*** Top View.

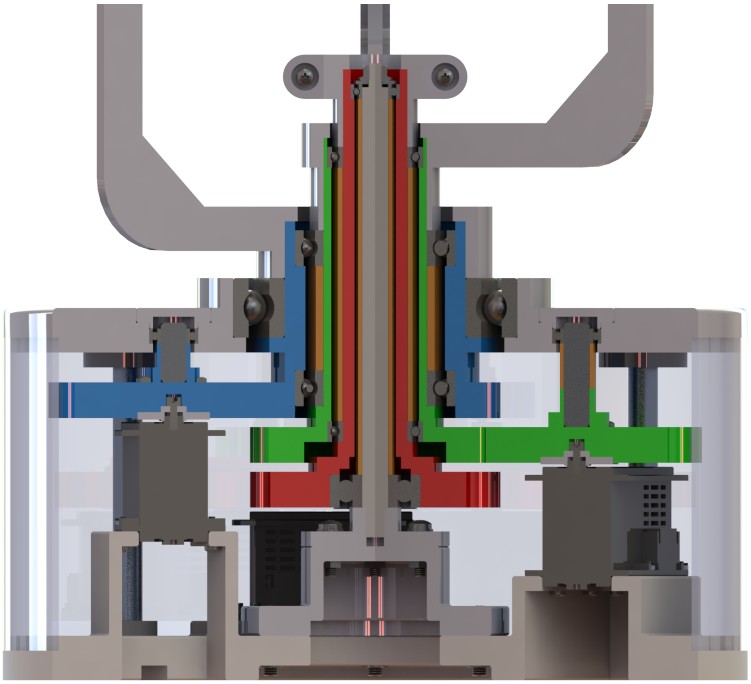


***Figure 6.8:*** Driving System Section View.

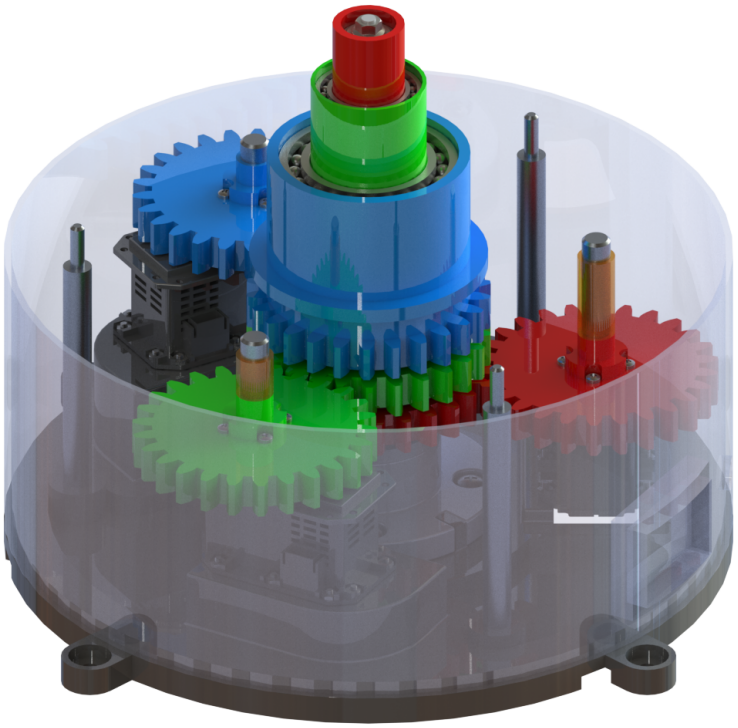


***Figure 6.9:*** Driving System View.

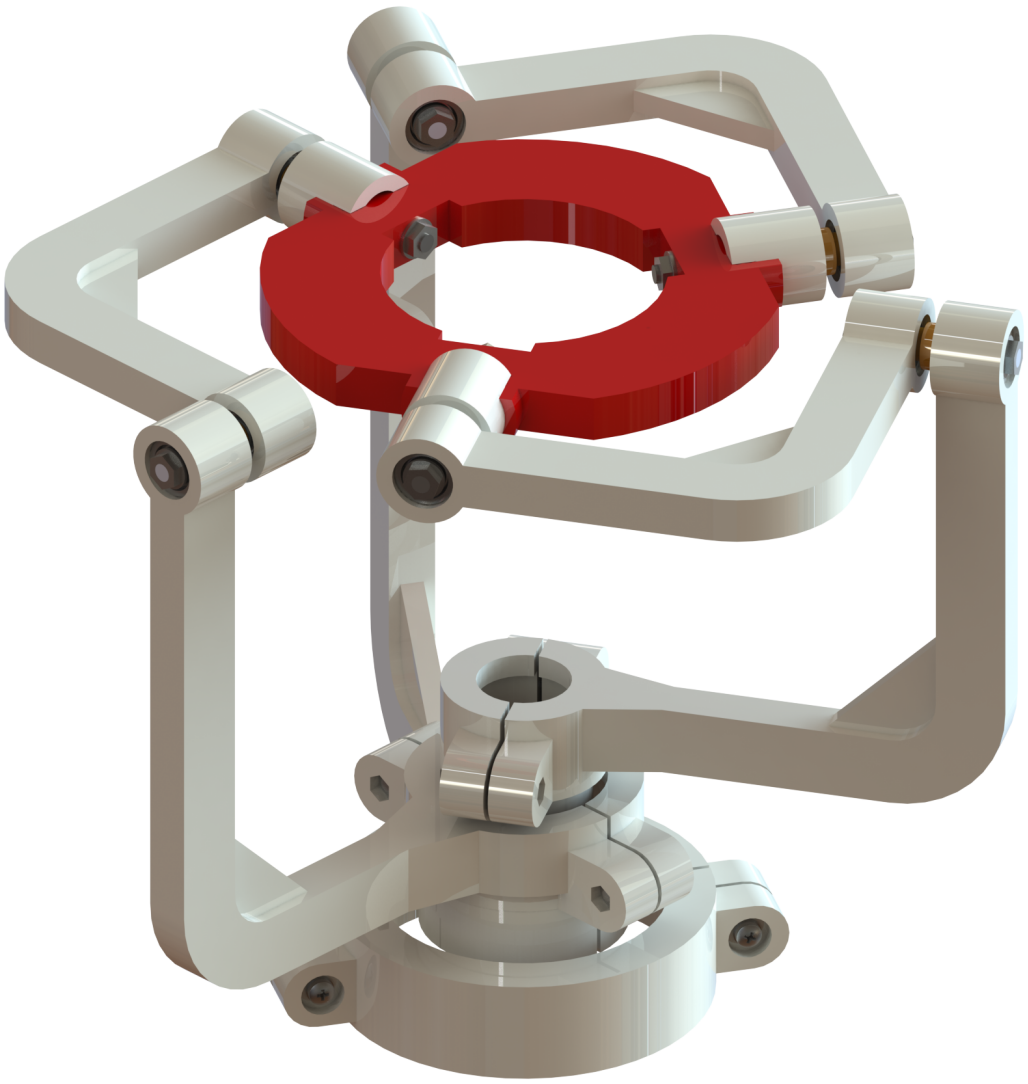


***Figure 6.10:*** Robot Assembly View.

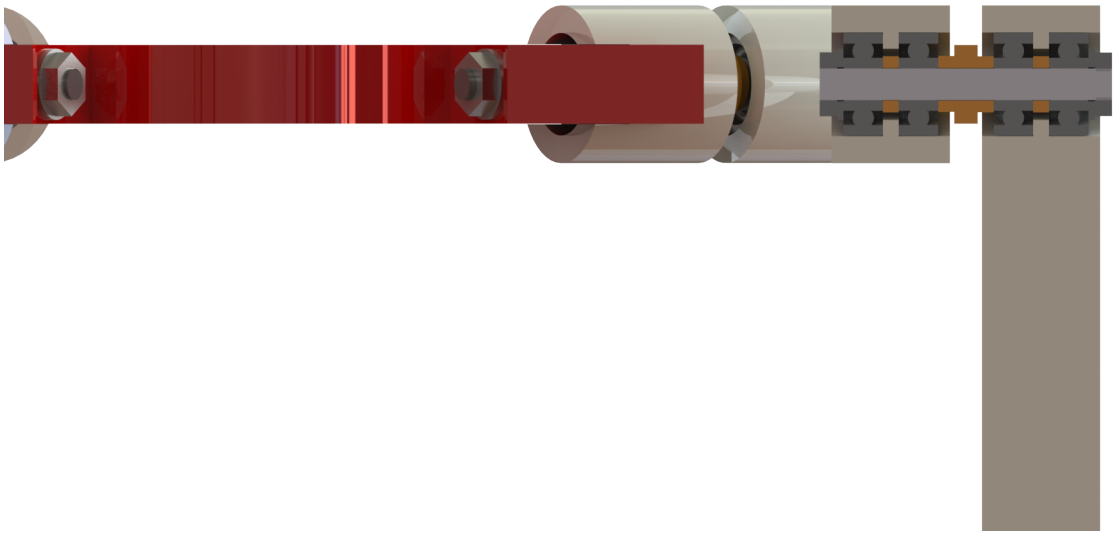


***Figure 6.11:*** Detail - End Effector Joint Section.

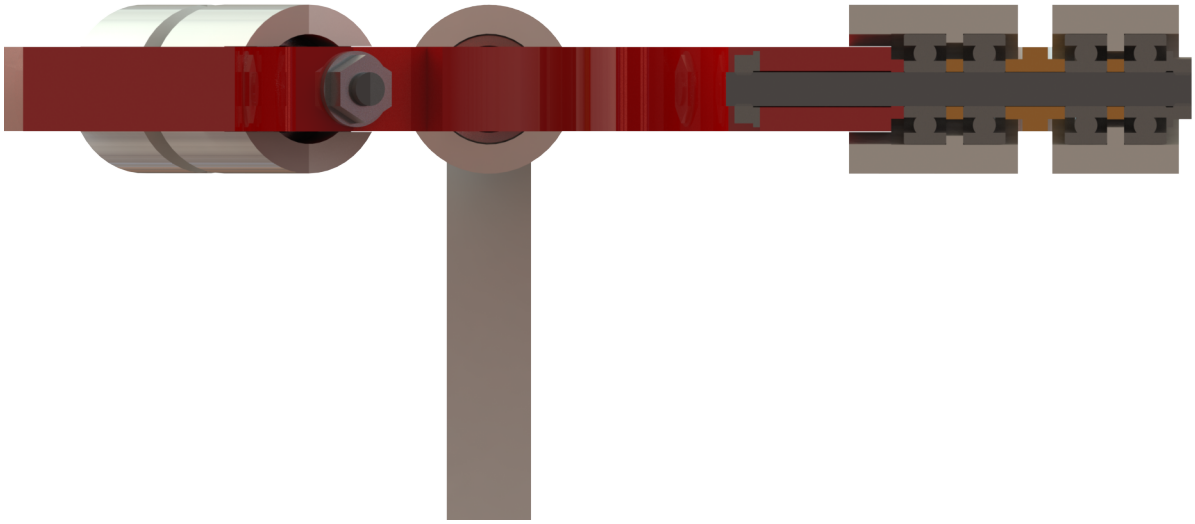


***Figure 6.12:*** Detail - Distal Link Joint Section.

## 6.4 Mounting

In order to establish the mounting process, it has been described the different phases of it, dividing the assembly in two macro-phases:

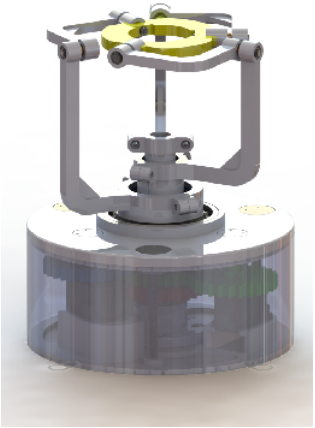
- **assembly of the transmission system;**
- **assembly of the robot.**

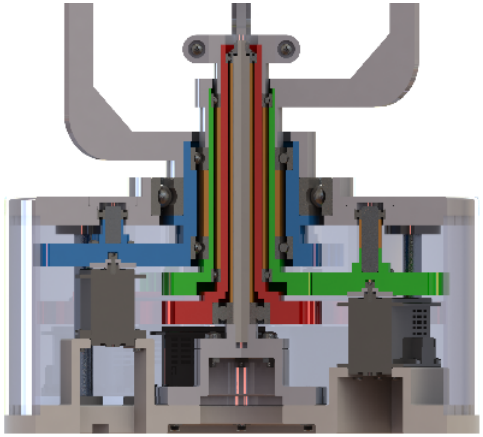
The several phases are written in the following itemize.

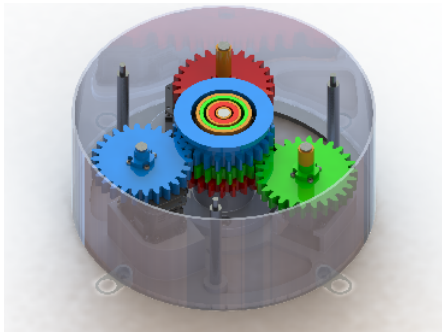
- **TRANSMISSION MECHANISM**:
  - **Sub-assembly Motor-base**:
    - insert the hexagonal nuts M5 (6) in the respective housing of the base (1);
    - put the base (1) on the table;
    - take the pivot-base (7) and screw them on the base;
    - put the three motors Dynamixel MX-64T (2) with the respective servo-horns (3) on the base, securing them with 8 screws (4) and hexagonal nuts (5);
    - fix the support (8) on the base with screw (12 ), hexagonal nuts (6) and washers (11), after referring it with two tap (65);
    - make the pin assembly, by screwing the flange-pin (22) and the pin (23);
    - mate the pin assembly (22-23) with the support (8) through screw (26), hexagonal nuts (37) and washers (25);
  - **Sub-assembly Shaft Mechanism**:
    - insert the Bearing (64) on the pin assembly (22-23-24);
    - mate shaft1 (14) with the bore diameter of bearing (64);
    - insert the spacer-shaft1 (13) in the pin (22);
    - put the Bearing (19) in the housing bore diameter of the shaft1 (14);
    - put the conic washer (25) and screw the Nut (26) on the thread of the pin (22);
    - insert the Bearing (20) on the shaft1 diameter (14);
    - insert the spacer-shaft2 (15) in the shaft1 (14);
    - mate shaft2 (16) with the bore diameter of bearing (20);
    - put the Bearing (20) in the housing bore diameter of the shaft2 (16);
    - insert the Bearing (21) on the shaft2 diameter (16);
    - mate shaft3 (18) with the bore diameter of bearing (21);
    - insert the spacer-shaft3 (17) in the shaft2 (16);
    - put the Bearing (21) in the housing bore diameter of the shaft3 (18);
  - **Sub-assembly Motor Gear**:
    - force the pivot-R1 (51), pivot-R2 (53) and pivot-R3 (55) on the respective gear motor (49);
    - insert the spacer (52, 54, 56) in the respective gear pivot (51, 53, 55);
    - mate the gears (49) on the servo-horn (3) of the motor fixed with screws (4);
    - realisation of the mechanical mate between shaft and gear (ratio 1:1);

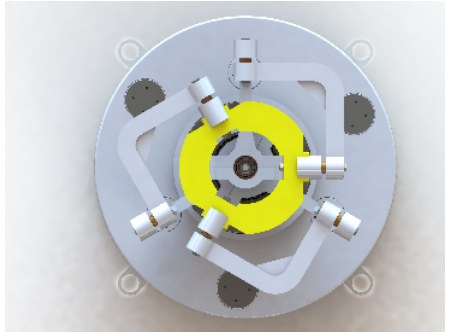
  - **Sub-assembly Upper Base**:
    * put the cover (30) on the base (1);
    * put the Bearings (28) in the housing bore diameter of the holes of the upper base (27);
    * mate the upper-base (27) with cover (30) and base (1), centring the pivots of the gears (51, 53, 55) with the respective inner diameter of the bearings (28);
    * insert the bearings (60), in order to allow thee relative rotation between the upper-base and the shaft mechanism;
  - insert the nut (6) in the housing of the upper base (27);
  - put the top bearings (61) on the upper base (27);
  - close the mechanism system, by assembling through the screw-nut mate (6,11 ,7), (conic washer are included);
  - cover the holes of the upper base (31) with the three top (29) the housing of the previous nuts.

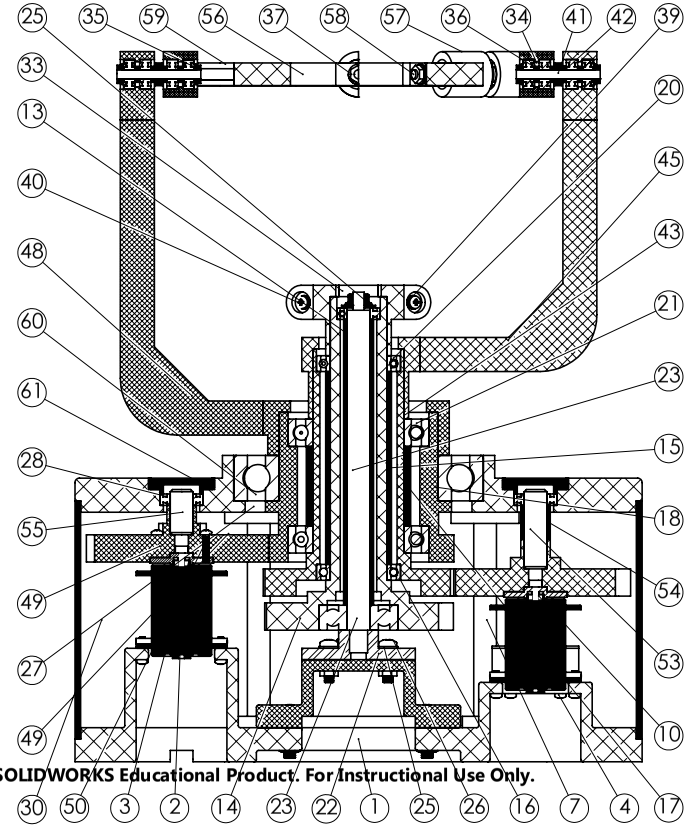
- **3-RRR COAXIAL SPM**:
  - **Sub-assembly Leg Robot (X3)**:
    * fix the semileg#-2 (31) with semileg#-3 (33), through acetone;
    * fix the head-leg (34) with semileg#-2 (31), through acetone;
    * insert the two Bearing (35) in the head (34), by putting the relative spacers (36);
    * inset the respective threaded pivot (43) and screw the pivot with nut (39) and washer (40);
    * mate the semileg#-2 (32) and the semileg#-1 (33) with screws (42), washers (41) and nuts (40) on the top of the respective shaft, locking the respective bearings too;
  - **Sub-assembly Arm Robot (X3)**:
    * fix the two head-arm (32) with the arm (60), through acetone;
    * insert the two Bearing (35) in the head (34), by putting the relative spacers (36);
    * mate the arm assembly with the respective leg one through the head and realise the mate pivot - bearings, by putting the spacer pivot bearing (43) screwing the nut (37) and washer (25) in the head of the arm;
  - **Sub-assembly E.E.**:
    * insert the two Bearing (35) in the head-EE (57) by putting the relative spacers (36);
    * fix with acetone the head-EE (57) with the EE (56) face (X3);
    * insert the threaded pivot (58) for the EE and screw both the nuts (37) an washer (25) of the head-EE (57) and head-arm (33), after putting the spacer pivot bearing (42).












| Num. | Description | Qua.nty | Num. | Description | Qua.nty |
|---|---|---|---|---|---|
| 1 | Base | 1 | 34 | House_Bearing_Leg | 9 |
| 2 | 64T | 3 | 35 | SKF - 624 - Full,SI,NC,Full_68 | 24 |
| 3 | Dynamixel_64T_servohorn | 3 | 36 | Bushing_head | 12 |
| 4 | ISO 7045 - M2.5 x 10 - Z - 10S | 16 | 37 | ISO - 4036 - M4 - N | 16 |
| 5 | ISO - 4035 - M2.5 - S | 24 | 38 | ISO - 4032 - M3 - W - N | 6 |
| 6 | ISO - 4034 - M5 - N | 12 | 39 | ISO 8738 - 3 | 6 |
| 7 | Pivot_Base | 3 | 40 | ISO 7045 - M3 x 16 - Z - 16S | 6 |
| 8 | Support | 1 | 41 | Pivot | 3 |
| 9 | ISO 7045 - M2.5 x 16 - Z - 16S | 8 | 42 | Spacer_Pivot_Bearing | 6 |
| 10 | ISO 4762 M3 x 8 - 8N | 3 | 43 | Semileg2_2 | 1 |
| 11 | Washer ISO 7090 - 5 | 7 | 44 | Semileg2_1 | 1 |
| 12 | ISO 7045 - M5 x 20 - Z - 20S | 4 | 45 | Semileg2_3 | 1 |
| 13 | Bushing | 1 | 46 | Semileg3_2 | 1 |
| 14 | Shaft1-1 | 1 | 47 | Semileg3_1 | 1 |
| 15 | Bushing2 | 1 | 48 | Semileg3_3 | 1 |
| 16 | Shaft2-1 | 1 | 49 | Gear_Motor | 3 |
| 17 | Bushing3 | 1 | 50 | ISO 7045 - M2.5 x 12 - Z - 12S | 12 |
| 18 | Shaft3-1 | 1 | 51 | Pivot_R1 | 1 |
| 19 | SKF - 61800 - 14,DE,NC,14_68 | 1 | 52 | Spacer_Pivot_R1 | 1 |
| 20 | SKF - 61805 - 22,DE,NC,22_68 | 2 | 53 | Pivot_R2 | 1 |
| 21 | SKF - 61908 - 20,DE,NC,20_68 | 2 | 54 | Spacer_Pivot_R2 | 1 |
| 22 | Flange_Pin | 1 | 55 | Pivot_R3 | 1 |
| 23 | Pin | 1 | 56 | EE | 1 |
| 24 | ISO - 4161 - M5 - N | 1 | 57 | House_Bearing_EE | 3 |
| 25 | Washer ISO 7089 - 4 | 16 | 58 | Pivot_EE | 3 |
| 26 | ISO 7045 - M4 x 16 - Z - 16S | 4 | 59 | Arm | 3 |
| 27 | Upper_Base | 1 | 60 | SKF - 6014 -18,DE,NC,18_68 | 1 |
| 28 | SKF - 61800 - 14,SI,NC,14_68 | 3 | 61 | Top_Bearing | 3 |
| 29 | Top | 3 | 62 | Tap | 2 |
| 30 | Cover | 1 | 63 | Washer ISO 7093 - 5 | 1 |
| 33 | Semileg1_2 | 1 | 64 | SKF - 6300 - 6,DE,NC,6_68 | 1 |
| 32 | Semileg1_1 | 1 | | | |
| 33 | Semileg1_3 | 1 | | | |

DESCRIPTION Robot_3D_Printer

STUDENT Francesco Morfea

DATE 28/01/2019

SCALE 1:2

SURFACE TEXTURE ( )

WEIGHT (Kg)

0.5x45°

UNI EN 22768-mK

I.R.I. U.P.C.-E.T.S.E.I.B. Politecnico di Torino

## 6.5 Prototype Realisation

After achieving the 3D model, the next step has been to print the whole robot; in the *figures 6.5,6.5,6.5, 6.5* it is possible to see different views of the **prototype**, in order to give you an idea of the final result.

***Figure 6.13:*** An Overview on the Final Printed Prototype - View 1.

In the end, it is possible to do some considerations as for the realisation of the prototype. The choice to use *ABS*, as the main material, leads to rebuild several parts, according to a functional design, during the realisations.

Considering the *robot assembly*, the prototype has been achieved an enough stiffness and

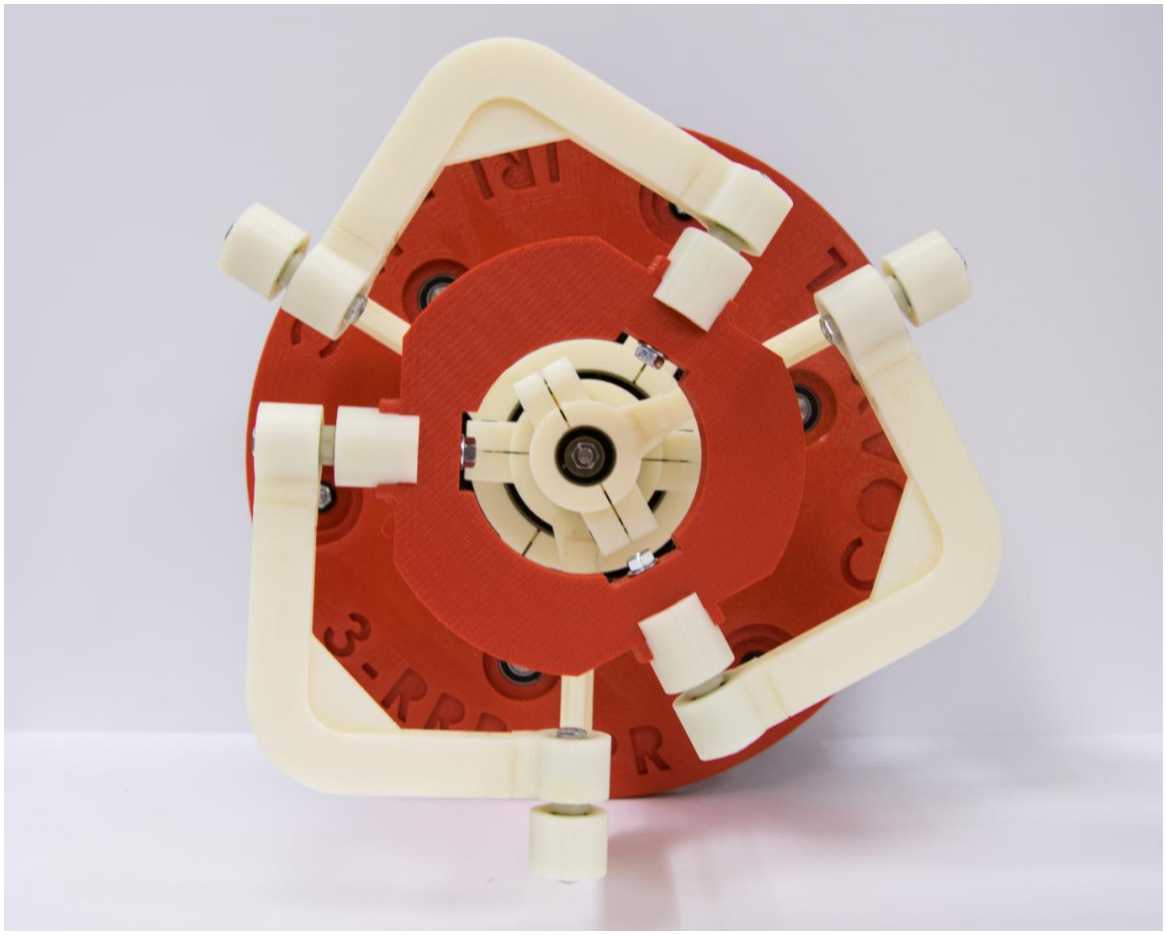


***Figure 6.14:*** An Overview on the Final Printed Prototype - View 2.

the friction is negligible for this application. There are, of course, misalignment errors due to the limits of the 3d printer, in term of tolerance, and a imperfect mounting process, because some parts have been stuck together, losing the exact reference surfaces.

The moving parts that perform the manipulation have a mass, negligible comparing with the driving system, according to the choice to study the robot in an only static way.

Moreover, as for the *mechanical transmission assembly*, the three motors have the capability to control the platform with an acceptable accuracy, bearing the inner forces and the several momentum. As well, the play between the tooth of the gear mate it can be considerer not so relevant.

However, it is possible to suggest precautions to increase the stiffness of the robot assembly, if it need to be reprinted. In the distal and proximal links, it would be optimal to insert a liner made of a carbon fibre. This would be raise the stiffness of the links, leaving more or less unchanged the mass of the moving parts.

Overall, the response of the system, subjected to different working conditions, is acceptable according to the prototype could be improved, and it can be considered a good result in the concise working time, spent in the research center.

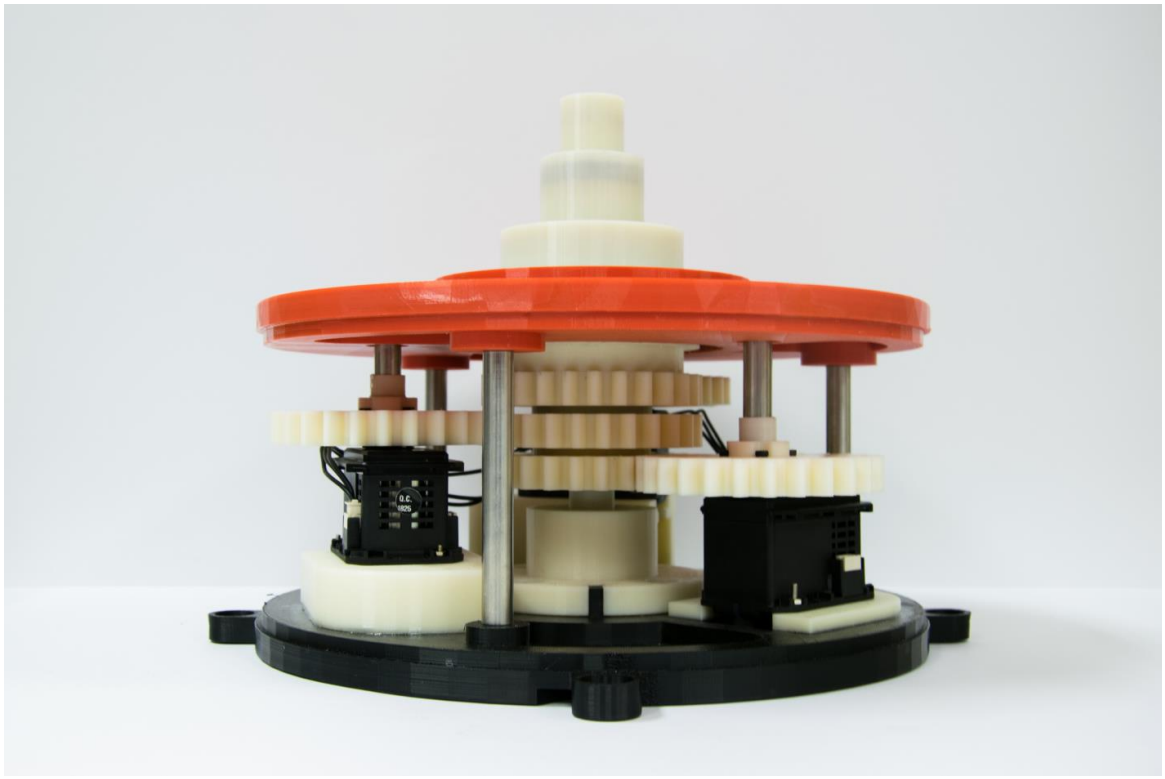


***Figure 6.15:*** An Overview on the Final Printed Prototype - View 3.


***Figure 6.16:*** An Overview on the Final Printed Prototype - View 4.

## 6.6 Future Works

Moreover, starting from the mentioned design, one goal of the future work could be to define a final concept by using only only one motor, in order to reduce the operative costs of the robot itself. As it can be seen, the transmission is composed of three electro-brakes and a system with pulley and belt to move the robot, even if the last power system will be determined at the end of the project itself. At the moment, the three electro-brakes have the task to stop the rotation of the relative shafts when each leg achieve the needed angle of rotation. This mechanism uses the encoder of the motor, which allows to send on-line the On-Off information to the electro-brakes, depending on the status of the control. It follows the *figure 6.17* as for the concept of the next prototype.

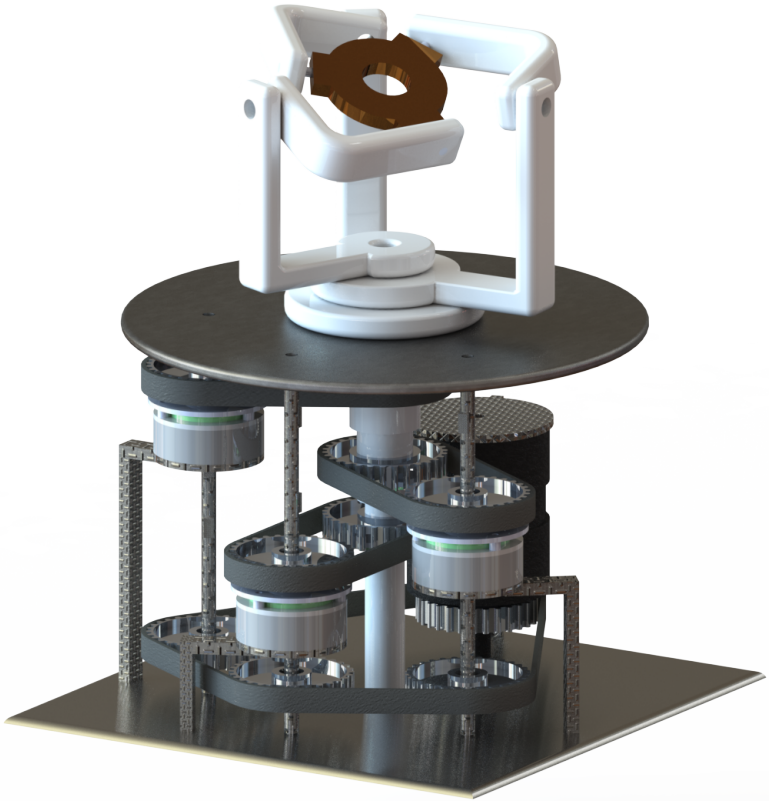


***Figure 6.17:*** Concept of the Future Prototype.

# Chapter 7

# On the Motors and Control System

## 7.1 Goals of the Motor Implementation

In this chapter, after physically implementing the robot, it has been studied the way how to move the three motors. As for the *driving system* it has been firstly used servomotor (*Robotis Dynamixel XL-320)* on a smaller scale, in order to study how to use the USB2 Dynamixel hardware and software, to achieve the final position of a cubic polynomial timing law.

The **scope** of this section has been to move the actuators, following a cubic position timing law, in a Matlab environment. These servomotors are used to work, receiving and sending information through the appropriate addresses of the RAM inside the motors. Moreover, it is possible to control them in joint mode (position control) or in wheel mode (velocity control). The bridge between of the computer language and the motors one is given by the USB2 Dynamixel hardware and software, allowing the exchange of instructions in the both directions.

During several tests in different working conditions to **evaluate the motor response**, it have been noticed some problems that do not allow the motor to achieve the desire position. The *inner PID has not been enough* to follow the timing position law step by step and it has been identified a *dead band* for a low PWM percentage , responsible of a starting delay and, as a consequence, of a steady state error.

The **solution** has been to consider the *motor as a black-box*, with one input (Moving Speed address) and one output (Present Position address), and to introduce an **external Feed Forward Position Control**. This way has been only a tool to find the fastest resolution to these problems, in order to perform the kinematics.

Moreover, **starting from the results of these experiments**, it has been decided to realise a prototype with **three motors Robotis Dynamixel Mx-64 AT**, having a bigger size and higher performances, coherently with the scope, the dimensions and the weight of the robot at stake, getting experiments on the bench and on the robot too. This leads to choose a set of motor parameters to perform the motion of the platform.

Overall, it has been considered the system **only in a static working condition**, given that the revolute mass is negligible, comparing on the stator parts; motors, that have been used, are able to balance the torque and the inner forces. The dynamic study of the system has been postponed as a future work.

## 7.2 Motor XL-320

### 7.2.1 The Dynamixel System Interface among Drivers, Robot, PC and Programmes

In order to move and to control the motor Robotis Dynamixel Xl-320 through the computer, it is important to do a brief explanation as for both the hardware and software that work together. It is possible to describe the Robot System as in the figure below:

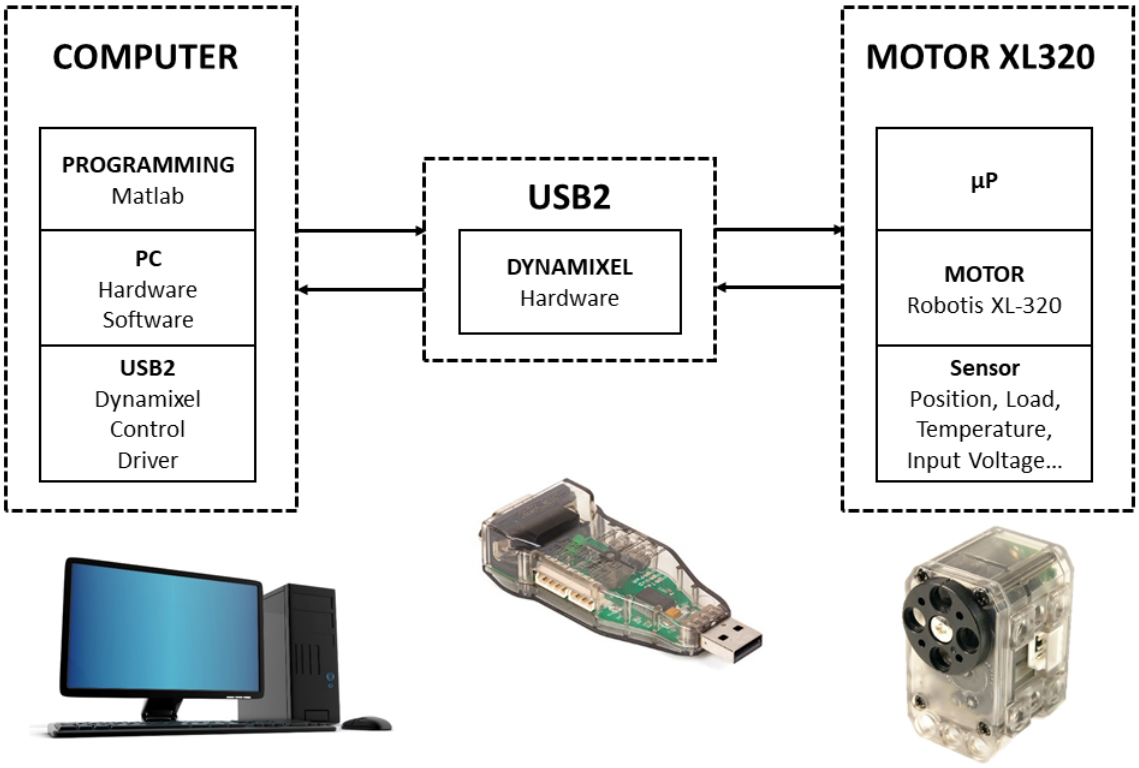



***Figure 7.1:*** Blocks Diagram of the Part of the System.

First, you can see that the informations and the instructions, travel in the both two directions:

- **from the pc to the robot**: by using *Matlab*, as a program to control and move the motor, the *Dynamixel Control Driver*, intalled in the PC, allows to translate the instructions from the Matlab language to a .ddl one, in order to elaborate them inside the *USB2 Dynamixel Hardware*. This one, through the .dll function is able to read and write into the addresses of the RAM of the motor. In the end, the *Motor* with its sensor, its $\mu P$ and its drivers can elaborate the instructions and can move the axes, according to these information themselves.

- **from the robot to the pc**: in the same way, the informations start from the sensors and the addresses of the RAM, and with an inverse process arrive on the PC, ready to be elaborated in the Matlab code.

### 7.2.2 Model and Features of the Motor

The Model Robotis XL-320 has the following specifications, collected in this table.

| **Features Of the Motor** | |
|---|---|
| **Item** | **Specification** |
| Baud rate | 7343 bps - 1Mbps |
| Resolution | 0.29 ° |
| Running Degree | 0 °-300 ° |
| Weight | 16.7 g |
| Dimensions (WxHxD) | 24mm x 36mm x 27mm |
| Gear Ratio | 238:1 |
| Stall Torque | 0.39 Nm @ 7.4V, 1.1A |
| No Load Speed | 114 rpm @ 7.4V, 1.1A |
| Opering Temperature | -5°C - +70°C |
| Input Voltage | 6V - 8.4V (Recommended 7.4) |
| Protocol Type | 2.0 |
| Physical Connection | TTL |
| ID | 0 -252 |
| Feedback | Position, Temperature, Load, Input Voltage, etc... |
| Material | Engineering Plastic |

***Figure 7.2:*** Robotis Motor XL-320 [21].

The Control Table is a structure of data implemented in the DYNAMIXEL. Users can read a specific data to get status of the DYNAMIXEL with READ instruction packets, and modify data as well to control DYNAMIXEL with WRITE instruction packets. The address is a unique value when accessing a specific data in the control table with instruction packets. In order to read or write data, users must designate a specific address in the instruction packet.

It is possible to explain how the control table works:

- **Area (EEPROM, RAM)**: the control table is divided into 2 Areas. Data in the RAM area is reset to initial values when the power is reset (Volatile). On the other hand, data in the EEPROM Area is maintained even when the DYNAMIXEL is powered off (non-volatile). Data in the EEPROM Area can only be written to if Torque Enable is cleared to 0 (Off).
- **Size of the data**: it varies from 1 to 4 bytes, depends on their usage.

- **Type of Access**: the control table has two different access properties. *RW* property stands for read and write access permission while *R* stands for read only access permission. Read only property (*R*) is generally used for measuring and monitoring purpose, and read write property (*RW*) is used for controlling DYNAMIXEL.

- **Initial Value**: is restored to initial values when the DYNAMIXEL is turned on. Default values in the EEPROM area are initial values of the DYNAMIXEL (factory default settings).

As it is explained above, the control and the managing of the motor through the PC, pass from the reading and the writing of the addresses of the RAM, in either the volatile part and the non-volatile one.

In order to understand which address it was used, it is represented in a table, the main used addresses of the whole **Control Table** of the Motor.

| **CONTROL TABLE of EEPRON AREA** | | | | | | |
|---|---|---|---|---|---|---|
| **Address** | **Size [byte]** | **Data Name** | **Access** | **Initial Value** | **Min** | **Max** |
| 3 | 1 | ID | RW | 1 | 0 | 252 |
| 4 | 1 | Baud Rate | RW | 3 | 0 | 3 |
| 11 | 1 | Control Mode | RW | 2 | 1 | 2 |
| **CONTROL TABLE of RAM AREA** | | | | | | |
| **Address** | **Size [byte]** | **Data Name** | **Access** | **Initial Value** | **Min** | **Max** |
| 24 | 1 | Torque Enable | W | 0 | 0 | 1 |
| 30 | 2 | Goal Position | W | - | 0 | 1023 |
| 32 | 2 | Moving Speed | W | - | 0 | 2047 |
| 37 | 2 | Present Pos. | R | - | 0 | 1023 |
| 39 | 2 | Present Speed | R | - | 0 | 1023 |

For each address it is also possible to read or write value in range that is connected to the physical range of the respective physical quantities, as reported in the *table 7.1*.

| **UNITS TABLE CONVERSION** | | | | |
|---|---|---|---|---|
| **Physical Quantities** | **Physical Range** | **Address Range** | **Name Unit** | **Units** |
| Voltage | 1-10 V | 10-100 | $u_1$ | 0.1 V |
| PWM | 0-100% | 0-1023 | $u_2$ | 0.1 % |
| Velocity | 0-114 rpm | 0-1023 | $u_3$ | 0.111 rpm |
| Position | 0-300 deg | 0-1023 | $u_4$ | 0.29 deg |

***Table 7.1:*** Unit Table Conversion for Motor XL-320.

### 7.2.3 Functioning, Types of Control and Consideration as For the Motor

The **functioning** of the motor depends on the type of control, but it used to work with a **Modulation of the Power - PWM**, writing in the address of the Moving Speed (32), when the control mode is in wheel mode. Modulating this percentage means to write an amount of the maximum power, given that the current is fixed.

The Robotis XL-320 Motor has two types of control: **Joint Mode (2)** and **Wheel Mode (1)**. Depending on the value that it is written in the Control Mode address (11), it is possible to write and read different quantities in the respective addresses of the RAM:

- **Joint Mode**: with this type of control, it is possible to write the read the *angular position* of the motor, in order to control it. It is not possible to control the velocity, but to only set the limit of the maximum of the velocity, which the motor uses to arrive to the goal position, writing this value in the Moving Speed address.
- **Wheel Mode**: in this status, it is possible to control the velocity, by modulating the Power from zero to the maximum. The value in this address move from 0 $u_2$ to 2047 $u_2$, where:
  - From 0 to 1023 (CCW), it is possible to modulate the power from 0 to the maximum; for example writing 512 means to used 50% of the power;
  - From 1024 to 2047 (CW), it is possible to modulate the power in the clock wise direction.

It is important to do several consideration about the value that can be read in both two controls of the motor:

- **Write Goal Position and Read Present Position**: it is allowed to write and read from 0° (0 $u_4$) to 300° (1023 $u_4$) even if the motor does not have any kind of physical constrain. The range between $300° - 360°$ is an invalid angle, so the value read from the present position address (37) has no physical sense in this range;
- **Write in Moving Speed and read in Present Speed address**: the consideration done before is worth only in a **NO LOAD CONDITION**. The direct proportionality between the power and velocity is only true without any external loads, because, on the contrary, the power is used both to compensate this disturb and to move the axis of the motor at this desire velocity.
- **Read Present Speed**: it is not possible to read the velocity in wheel mode, it must be calculated as a derivation of the position; for these reason it was necessary to do an **experiment in order to find the law** between the PWM written in the Moving Speed and the respective desire velocity for this amount of the power.

### 7.2.4 On the Wheel Control Mode

#### 7.2.4.1 Experiment as for the Law Velocity-PWM

As stated in the previous chapter, it is impossible to read the present speed in the wheel mode control. For this reason, starting from the present position, the velocity has been calculated as the incremental ratio of two following readings of the position. So, for each value included in the range of $PWM = [-200 : 200]$ $u_2$, 5 experiments have been carried out where the process has involved in these steps:

- **Reading and storing** the present position during the time of the experiment, step by step.
- **Calculating the velocity**.
- **Plotting the velocity**.
- **Calculating the mean**, corresponded to the respective value of the Voltage, using only the data after the transient state (*figure 7.3, 7.4*).

After these several experiments, done for all of the chosen PWM values, the collected data has been elaborated and plotted in order to find the law between the Velocity and the % of PWM. It is possible to show different graphs about the experiments.

By analysing both the data and graph of $Vel\text{-}PWM$ (*figure 7.5*), it is clear that the law it is more or less proportional, even if the **function** in not continuous but a **piecewise** one.

In fact it is possible to determine the different value of the gain $K$:

$$K_v = \begin{cases} 0.81 & \text{if } PWM = [0:200]\ u_2 \\ 0.71 & \text{if } PWM = [-100:0)\ u_2 \\ 0.78 & \text{if } PWM = [-200:-100)\ u_2 \end{cases} \tag{7.1}$$

Moreover, watching the enlargement graph for the lowest value of the PWM (*figure 7.7*), it can be seen the existence of a **dead band**, where for these value, the motor is not able to move its axis.

Therefore, in order to finding the final relationship between the desire velocity and the respective PWM that comes more nearer to the real behaviour of the motor, it has been considered a **safety zone** from $PWM = [-20:20]\ u_2$, in which the motor does not move.

Finally, the inverse function as for the graphs, shown in the *figure 7.6 and 7.8*, becames:

$$PWM = f(Vel) = \begin{cases} 1.24 \cdot Vel & \text{if } Vel = (20:200]\ ^\circ/s \\ 16 & \text{if } Vel = [0:20]\ ^\circ/s \\ -14 & \text{if } Vel = [-20:0]\ ^\circ/s \\ 1.41 \cdot Vel & \text{if } Vel = [-100:-20)\ ^\circ/s \\ 1.28 \cdot Vel & \text{if } Vel = [-200:-100)\ ^\circ/s \end{cases} \tag{7.2}$$

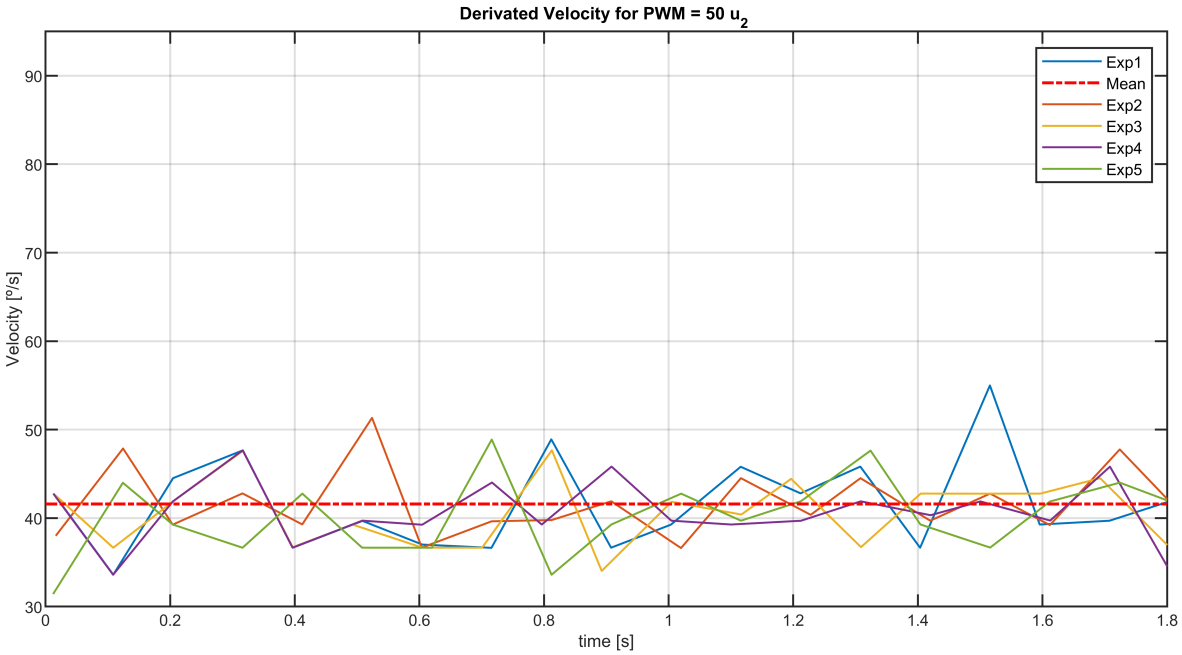



***Figure 7.3:*** $PWM = 50\ u_2$

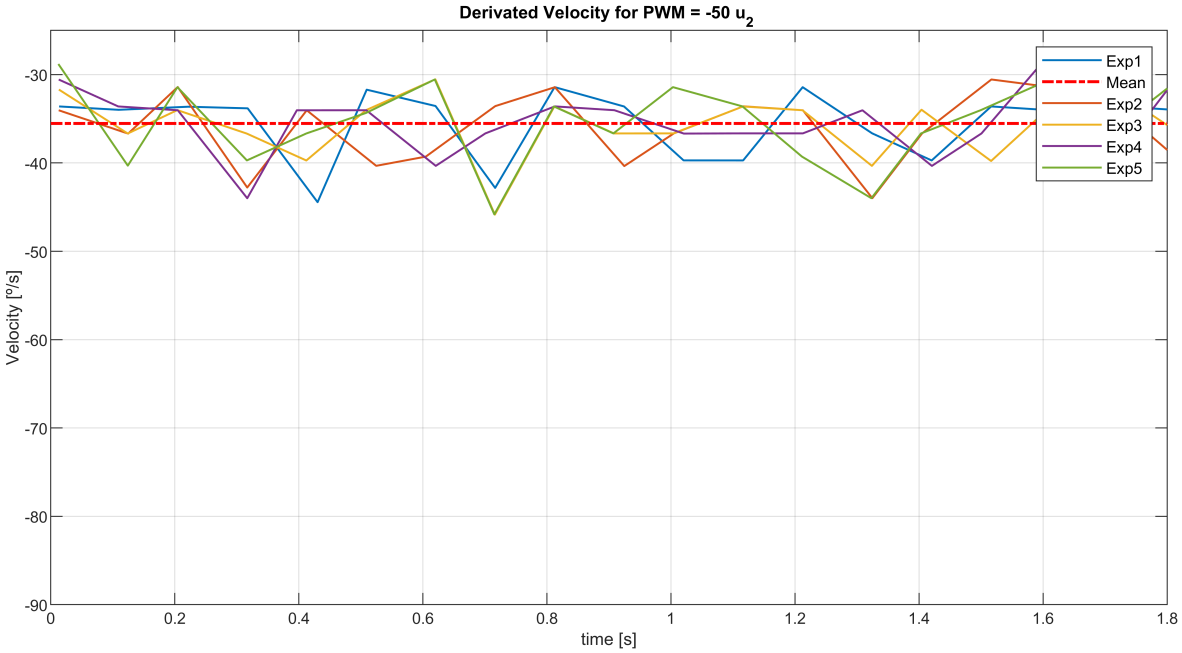



***Figure 7.4:*** $PWM = -50\ u_2$

| Gain Experiment | | | | | | | |
|---|---|---|---|---|---|---|---|
| **PWM ($u_2$)** | **Exp1** | **Exp2** | **Exp3** | **Exp4** | **Exp5** | **Mean** | **Gain** |
| -200 | -158,36 | -158,32 | -161,23 | -157,84 | -158,67 | -158,88 | 0,79 |
| -190 | -151,48 | -152,57 | -148,09 | -149,53 | -151,40 | -150,61 | 0,80 |
| -180 | -143,80 | -141,39 | -142,39 | -143,45 | -141,87 | -142,58 | 0,80 |
| -170 | -134,07 | -133,02 | -132,35 | -133,31 | -130,65 | -132,68 | 0,79 |
| -160 | -125,76 | -124,55 | -126,17 | -125,90 | -125,88 | -125,65 | 0,79 |
| -150 | -114,34 | -116,78 | -116,68 | -116,82 | -116,15 | -116,15 | 0,76 |
| -140 | -107,26 | -107,04 | -106,11 | -109,57 | -109,56 | -107,91 | 0,77 |
| -130 | -102,30 | -99,68 | -100,47 | -101,80 | -100,37 | -100,92 | 0,79 |
| -120 | -92,41 | -92,45 | -96,16 | -88,88 | -92,07 | -92,40 | 0,77 |
| -110 | -82,01 | -86,85 | -83,93 | -80,27 | -81,57 | -82,93 | 0,75 |
| -100 | -77,57 | -74,22 | -74,62 | -72,45 | -77,24 | -75,22 | 0,78 |
| -90 | -66,65 | -69,24 | -67,43 | -70,62 | -69,01 | -68,59 | 0,74 |
| -80 | -58,64 | -58,02 | -59,46 | -56,69 | -58,84 | -58,33 | 0,73 |
| -70 | -48,84 | -51,98 | -53,75 | -48,60 | -52,40 | -51,11 | 0,70 |
| -60 | -42,06 | -42,92 | -44,91 | -40,11 | -44,35 | -42,87 | 0,70 |
| -50 | -36,16 | -36,33 | -36,36 | -36,18 | -36,21 | -36,25 | 0,72 |
| -40 | -28,52 | -25,07 | -29,93 | -25,60 | -28,11 | -27,44 | 0,71 |
| -30 | -20,64 | -18,37 | -18,70 | -22,34 | -19,93 | -20,00 | 0,69 |
| -20 | -10,42 | -11,02 | -11,84 | -10,66 | -11,03 | -11,00 | 0,52 |
| -15 | -6,00 | -5,69 | -5,90 | -5,90 | -5,90 | -5,88 | 0,74 |
| -10 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,74 |
| 10 | 8,73 | 8,81 | 9,45 | 9,46 | 9,05 | 9,10 | 0,87 |
| 15 | 11,71 | 11,44 | 11,48 | 11,64 | 13,43 | 11,94 | 0,78 |
| 20 | 15,32 | 16,66 | 15,62 | 15,15 | 17,75 | 16,10 | 0,77 |
| 30 | 24,33 | 24,28 | 25,45 | 25,20 | 24,51 | 24,76 | 0,81 |
| 40 | 33,40 | 33,50 | 33,32 | 33,47 | 32,32 | 33,20 | 0,83 |
| 50 | 41,99 | 41,36 | 40,74 | 41,07 | 40,31 | 41,09 | 0,84 |
| 60 | 48,98 | 48,77 | 48,73 | 48,94 | 48,38 | 48,76 | 0,82 |
| 70 | 57,01 | 56,07 | 57,04 | 56,58 | 56,94 | 56,73 | 0,81 |
| 80 | 64,54 | 64,01 | 66,04 | 64,84 | 64,77 | 64,84 | 0,81 |
| 90 | 72,05 | 72,68 | 72,99 | 72,64 | 73,13 | 72,70 | 0,80 |
| 100 | 80,83 | 81,14 | 81,33 | 80,15 | 80,63 | 80,82 | 0,81 |
| 110 | 87,94 | 89,40 | 87,09 | 90,39 | 88,56 | 88,68 | 0,80 |
| 120 | 96,51 | 94,17 | 98,57 | 96,71 | 95,54 | 96,30 | 0,80 |
| 130 | 105,03 | 104,48 | 104,72 | 104,94 | 104,76 | 104,79 | 0,81 |
| 140 | 112,31 | 110,86 | 110,75 | 111,11 | 111,41 | 111,29 | 0,80 |
| 150 | 120,33 | 119,65 | 121,51 | 120,01 | 121,39 | 120,58 | 0,80 |
| 160 | 129,20 | 128,14 | 127,37 | 127,67 | 128,71 | 128,22 | 0,81 |
| 170 | 137,82 | 135,67 | 136,69 | 136,56 | 134,48 | 136,24 | 0,81 |
| 180 | 146,25 | 145,76 | 144,41 | 145,75 | 144,03 | 145,24 | 0,81 |
| 190 | 150,62 | 151,38 | 150,99 | 151,97 | 151,54 | 151,30 | 0,79 |
| 200 | 161,73 | 158,45 | 161,04 | 159,19 | 157,97 | 159,68 | 0,81 |
| Positive | | | | | | | 0,81 |
| Negative -100/0 | | | | | | | 0,71 |
| Negative -200/-100 | | | | | | | 0,78 |
| Continuous Function | | | | | | | 0,77 |

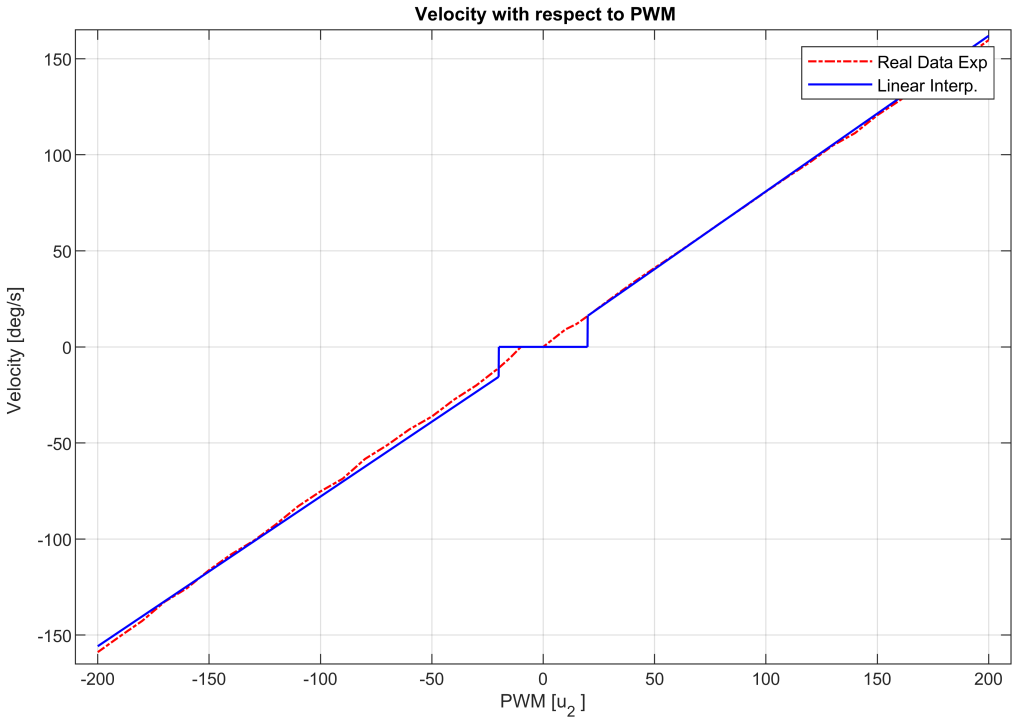



***Figure 7.5:*** Velocity with respect to PWM.

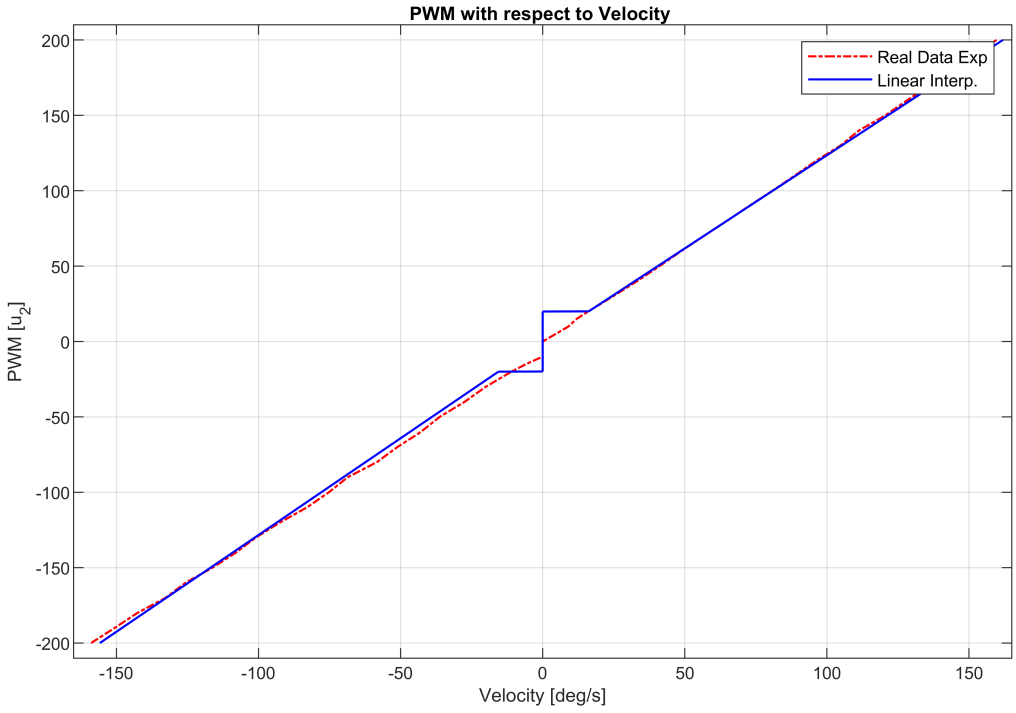



***Figure 7.6:*** PWM with respect to Velocity.

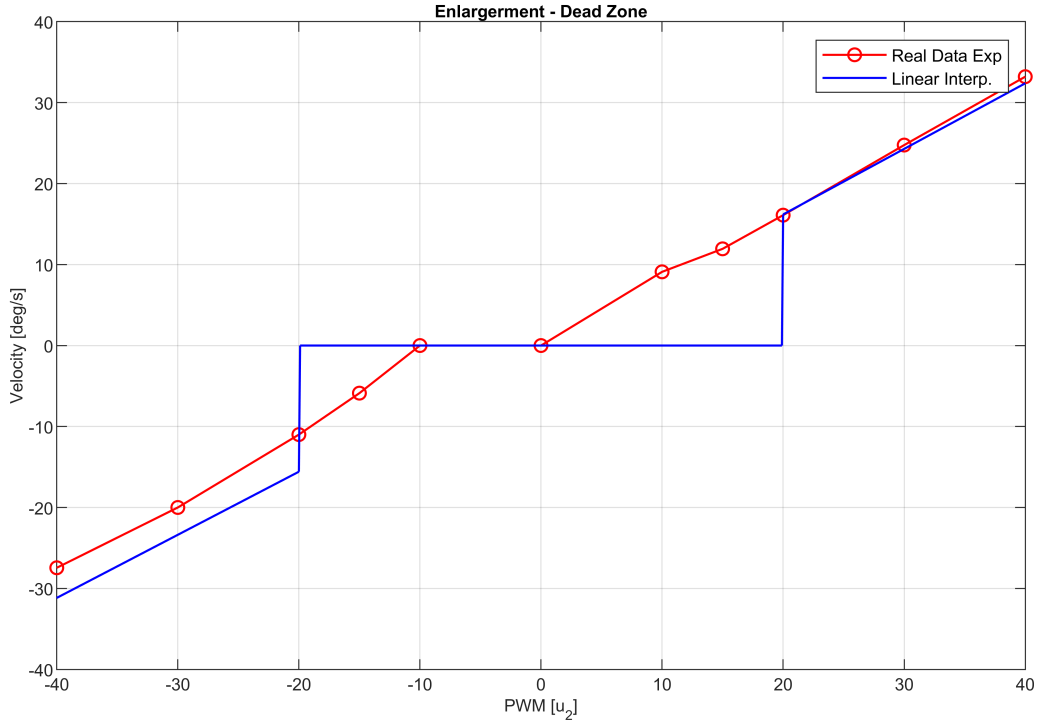



***Figure 7.7:*** Enlargement: Velocity with respect to PWM.

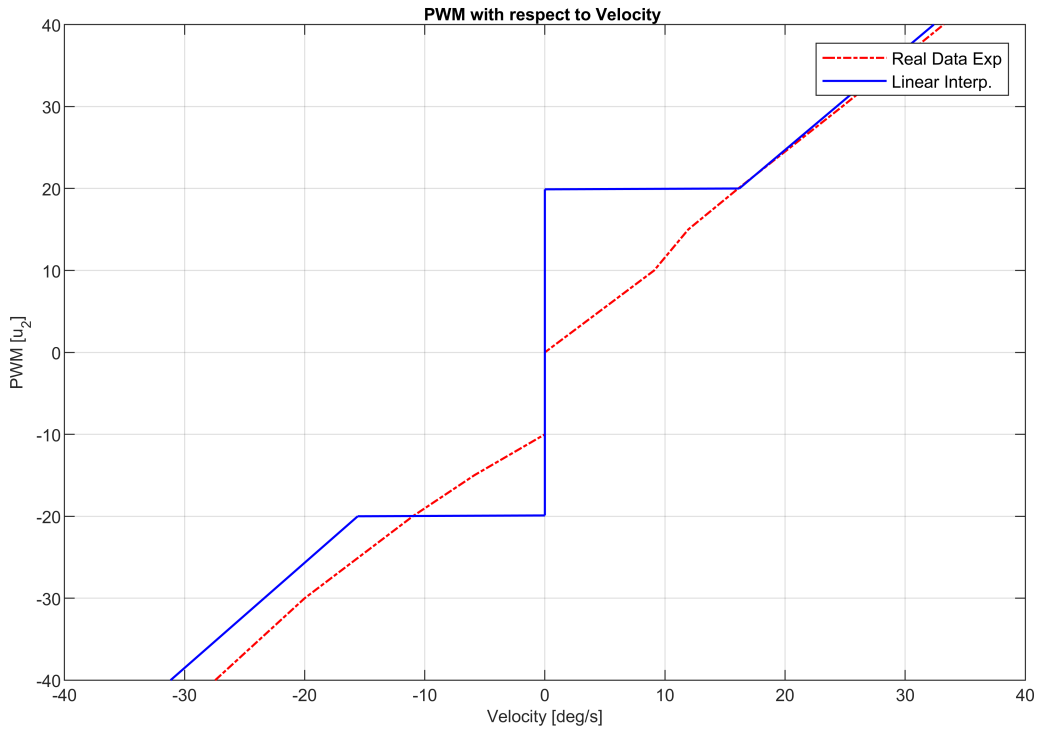



***Figure 7.8:*** Enlargement: PWM with respect to Velocity.

#### 7.2.4.2 External Feed-Forward Position Control

The goal of this type of control is to follow the theoretical curve of the position, for a given timing trajectory, evaluating in real time the error of position between the desire position and the real one, as shown in *figure 7.11*. The algorithm works in this way, for each step:

- **Reading the position** $\vartheta_{FB}$;
- **Calculating the error**: $Err = \vartheta_{SET} - \vartheta_{FB}$;
- **Proportional Control**: in order to obtain a compensated velocity;
- **Compensating the desire velocity**: $\dot{\vartheta} = \dot{\vartheta_{SET}} + \dot{\vartheta_{Err}}$;
- **Converting** this value in a PWM one, with the respective gain.
- **Writing** in the moving speed address this value.

In each step, if the error is positive, the written PWM in the motor is more than the desire one, to get readier the motor itself and to follow better the position curve.

It can be noted that the proportional control of the position is enough for this application, while if the application requests more accuracy an implementation of PID controller, is better in order to have the desire behaviour.

For understanding how the motor works with this control, several experiments have been done, using a Cubic Polynomial Timing Law, shown in the *figure 7.10*:

- **Finding the best proportional gain K**, in different conditions:
  - **no load**: linear, without any kind of load;
  - **under a constant load**: linear, increasing the value of the weight;
  - **under a variable load**: non linear because of the changing in magnitude and direction of the Momentum, increasing the value of the weight.
- **Evaluating the motor behaviour for different value of maximum velocity**, depended on:
  - **the final time** $t_f$;
  - **the final position** $q_f$.

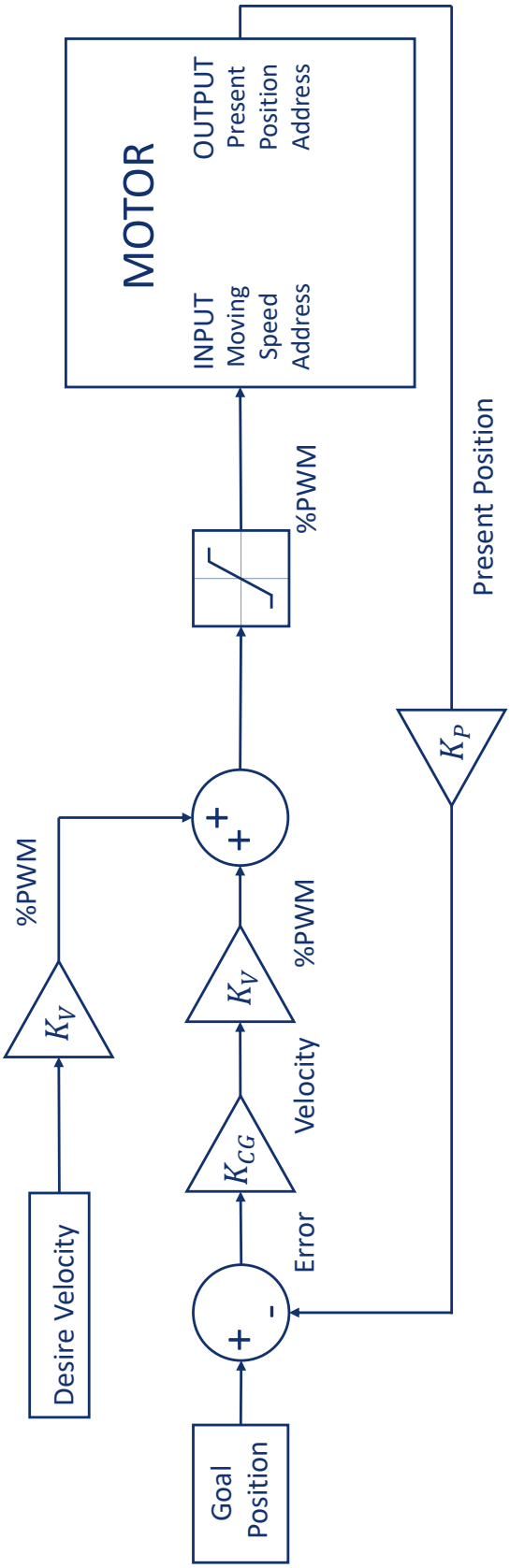



***Figure 7.9:*** Feed Forward Position Control.

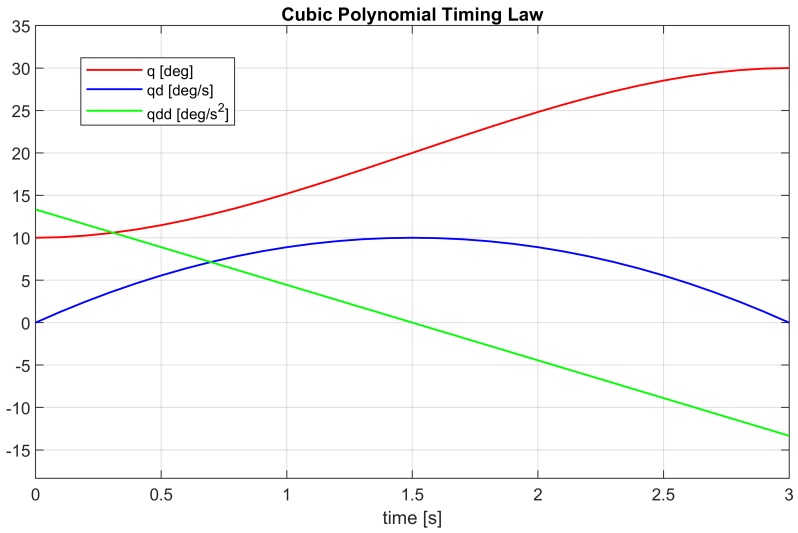



***Figure 7.10:*** Cubic Polynomial Timing Law.

#### 7.2.4.3 Consideration and Future Work

First, analysing all of the graphs, it can be seen how there is **a start delay in all of the graphs**, due to a low speeds, that correspond to a low percentage of voltage to modulate the power. For this range, there is a **dead band**, analysed before, that makes the motor not ready to follow the cubic timing law. In this first approach, this phenomenon has been neglected.

By analysing the **linear response without any load** in the *figure 7.12*, it is clear that it is not necessary an higher proportional gain to follow the theoretical curve, but the $K_P = 4$ is enough to obtain a good approximation.

On the contrary, **the linear response with a growing constant load** (shown in *figure 7.13,7.14* needs an higher gain in order to reflect the actual behaviour. However, increasing the constant load, the behaviour is more or less the same, due to the linear situation in which the motor works.

Moreover, seeing the graphs in *figure 7.15 and 7.16*, **under a non linear condition**, the motor response changes due to both the variations of the gains and the weight. Only with the maximum gain $K = 10$, it is possible to have got a good compromise, while for low gain the behaviour cannot be accepted. It is easy to understand how the error grows up when the gain goes down and, on equal gain, the error is bigger when the non-linear load increases.

In the end, it is possible to analyse the **dependence to the parameters final time $t_f$ and $\Delta q = q_f - q_i$** on the shape of the cubic timing law. For high value of $\Delta q$ and low $t_f$, the velocity of the law increases, making the motor readier (*figure 7.19,7.18,7.20*). On the other hand, for low value of $\Delta q$ and high $t_f$, the motor response is not so good and acceptable, either for the delay in the first steps and for the real shape of the position that is far from the theoretical one.

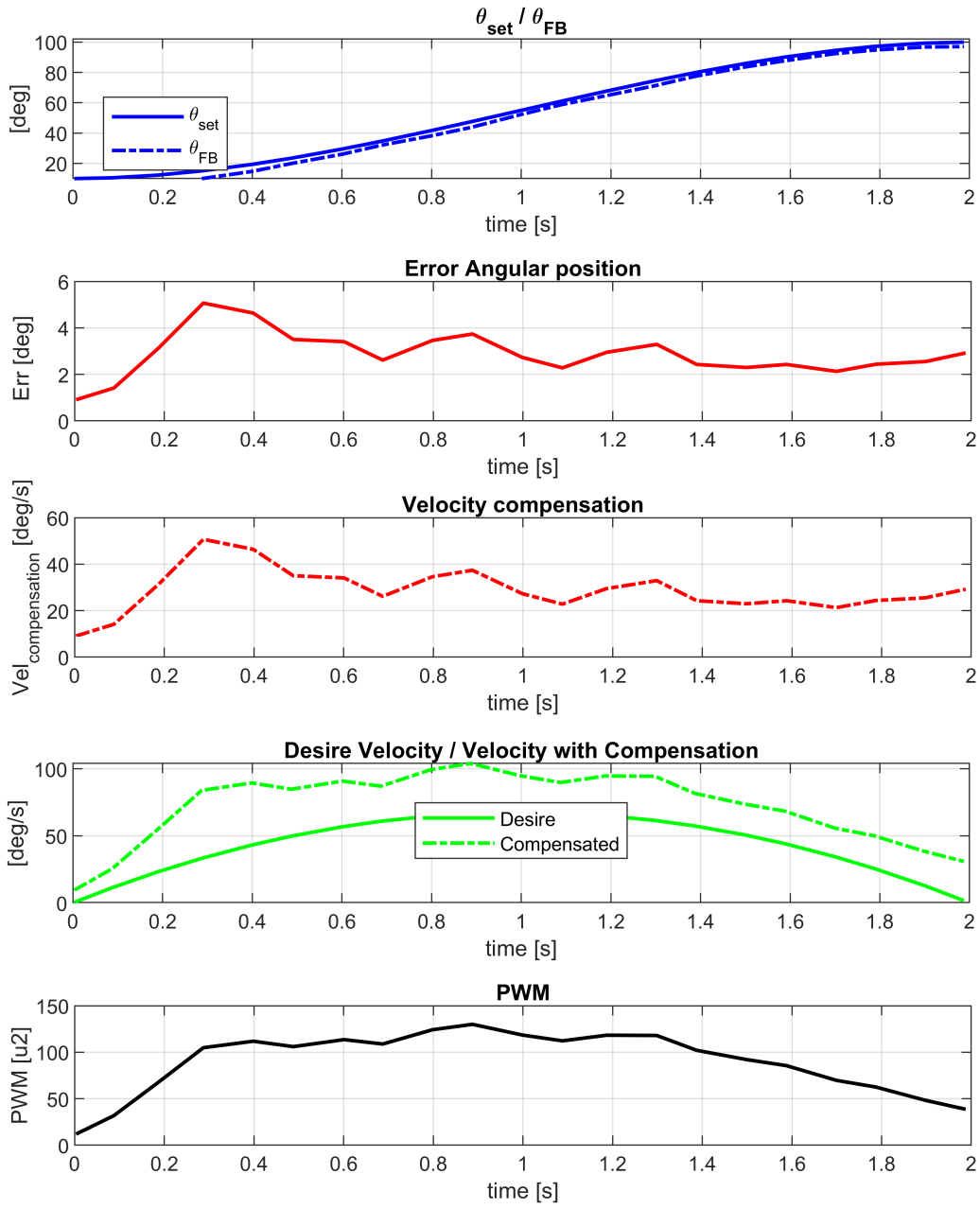



***Figure 7.11:*** Diagram of Mechatronics Quantities with $G10$ - $t_f = 2s$ - $q_i = 10°$ - $q_i = 100°$ .

### 7.2.5 Graphs of the Experiments

#### 7.2.5.1 FFPC-No Load - Linear Condition

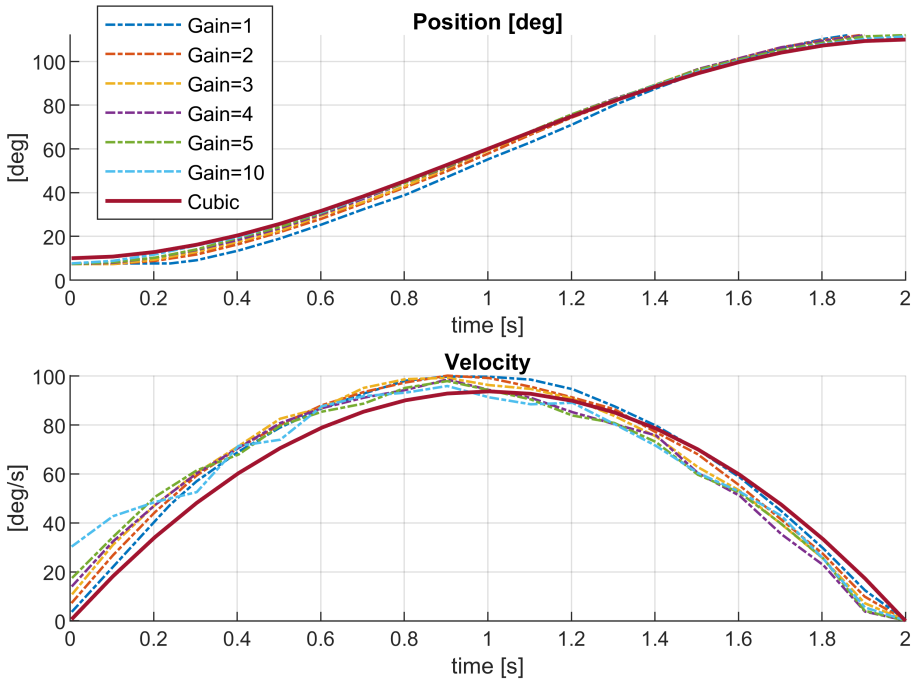



***Figure 7.12:*** FFPC-No Load.

#### 7.2.5.2 FFPC-Under a Constant Load - Linear Condition

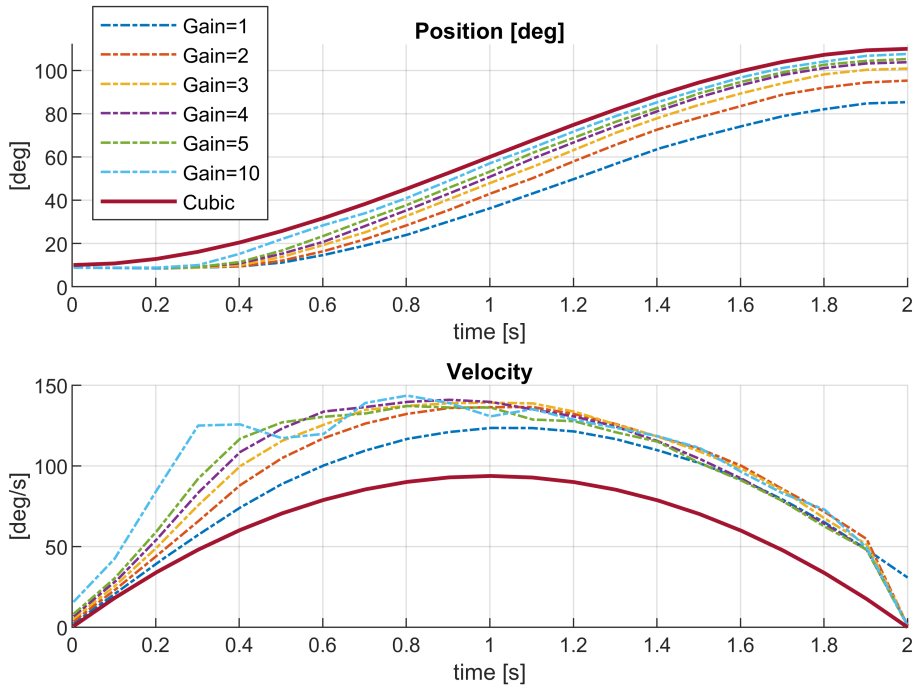



***Figure 7.13:*** FFPC-Under a Constant Load: weight 1.

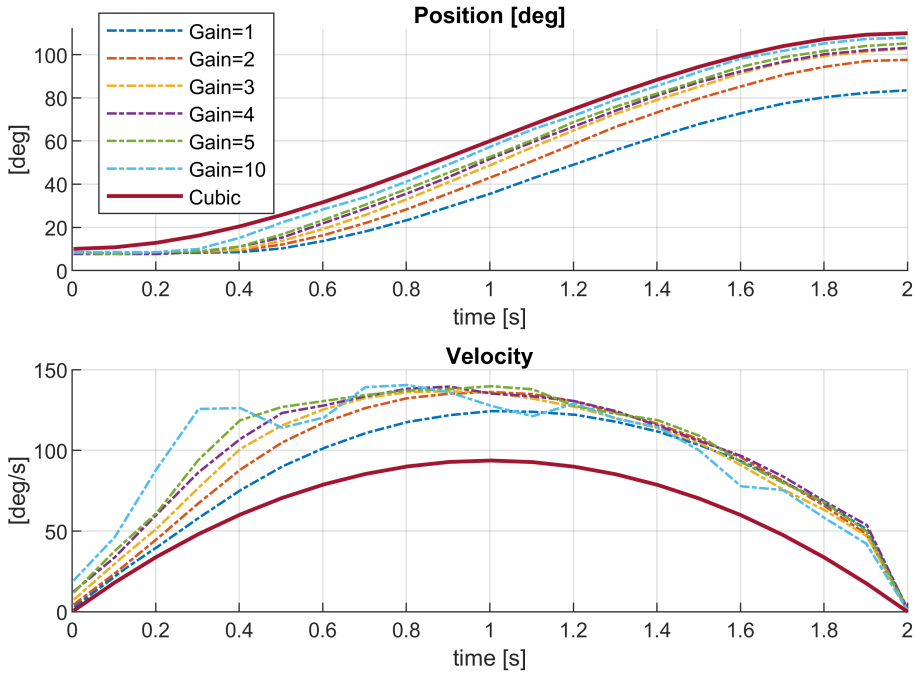



***Figure 7.14:*** FFPC-Under a Constant Load: weight 2.

#### 7.2.5.3 FFPC-Under a Variable Load - Non Linear Condition

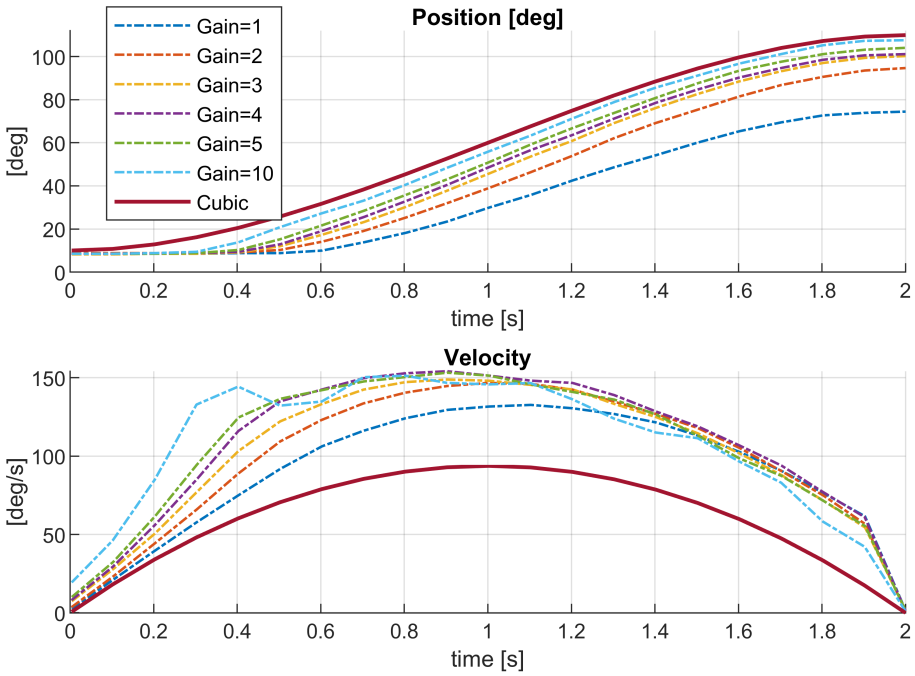



***Figure 7.15:*** FFPC-Under a Variable Load: weight 1 (100g).

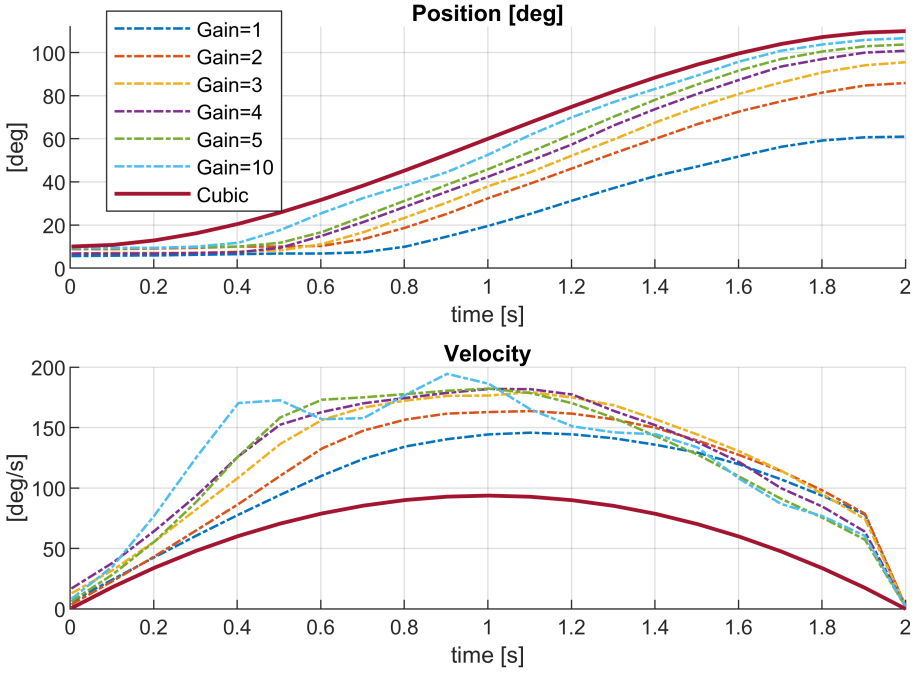



***Figure 7.16:*** FFPC-Under a Variable Load: weight 2 (200g).

#### 7.2.5.4 FFPC-Influence of the Final Time $t_f$

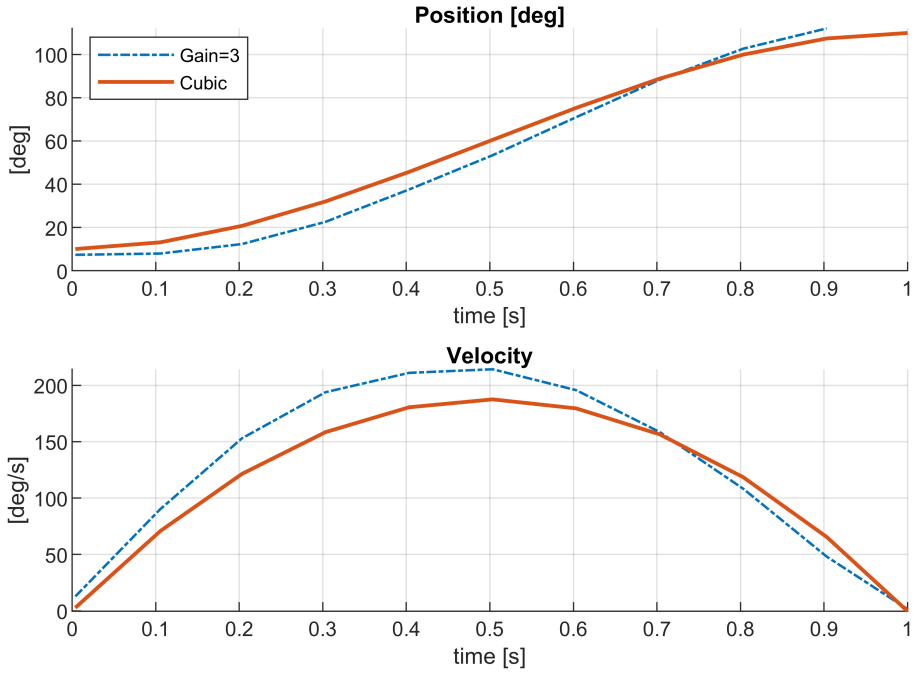



***Figure 7.17:*** FFPC-Influence of the $t_f = 1s$

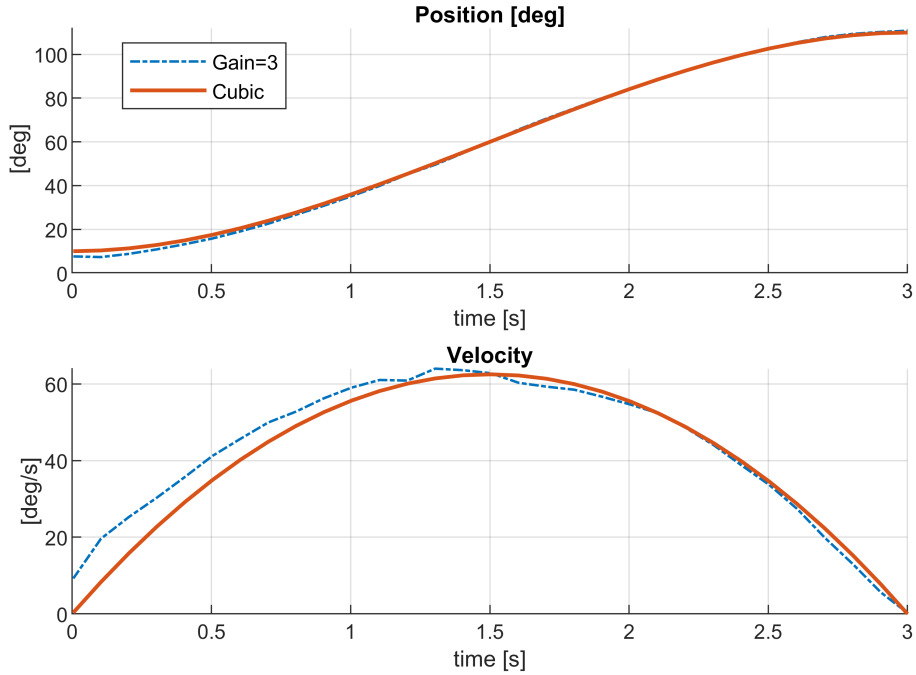



***Figure 7.18:*** FFPC-Influence of the $t_f = 3s$

#### 7.2.5.5 FFPC-Influence of the Final Angle $q_f$

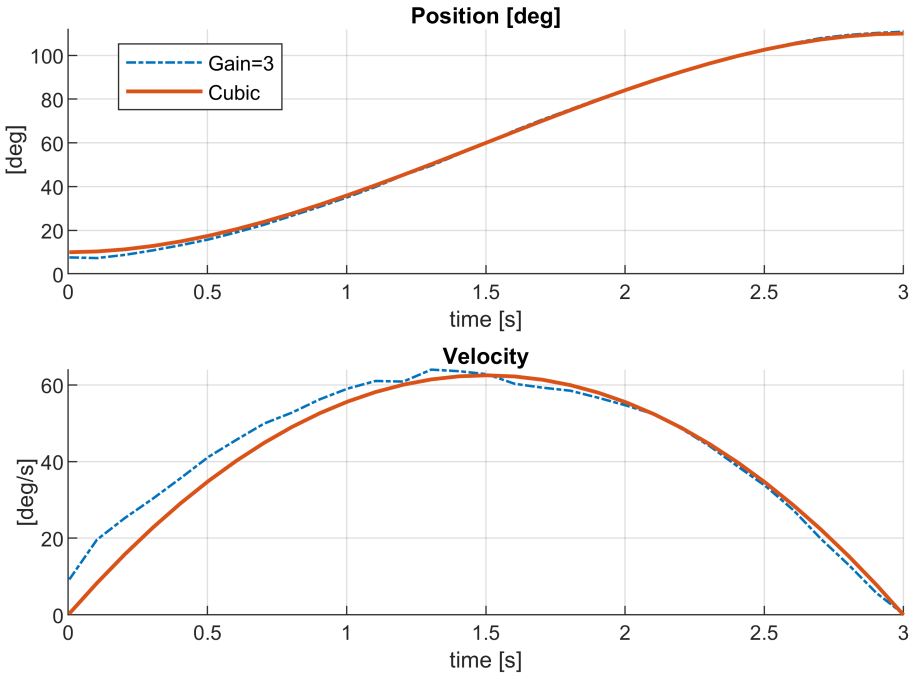



***Figure 7.19:*** FFPC-Influence of the $t_f = 3s$

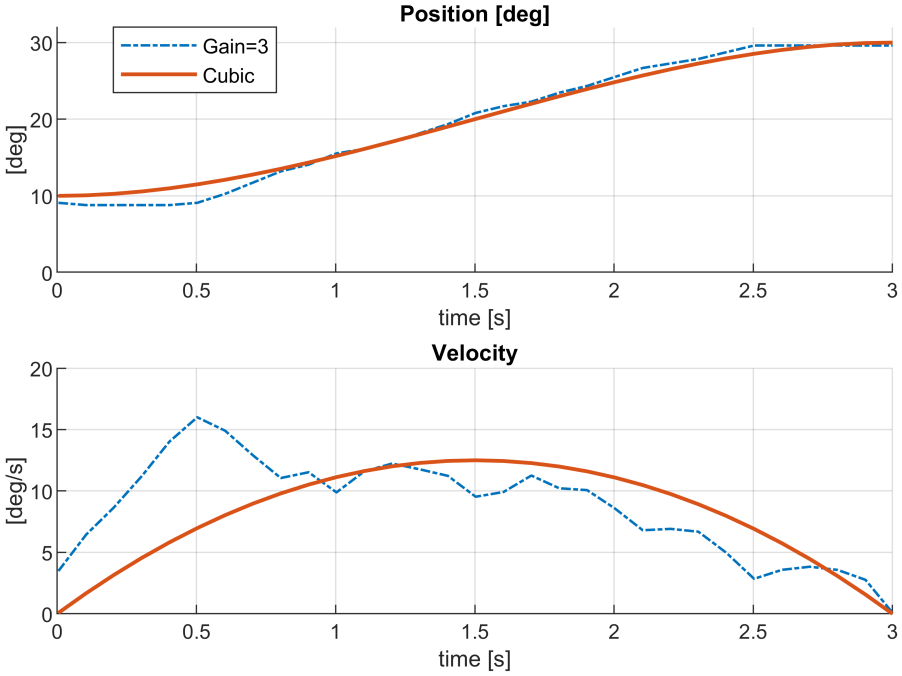



***Figure 7.20:*** FFPC-Influence of the $q_f = 30s$

## 7.3 Motor MX64-AT

After learning how to use the dynamixel interface in order to move the motors in Matlab environment, as detailed in the *section 7.2*, it has been worked with the dynamixel motors **MX64-AT** (mounted in the final prototype), leading several experiments both on the bench and on the robot.

### 7.3.1 Specification of the Motor MX64-AT

In the physical implementation of the prototype it has been used the same hardware and software to move and control the motors with the computer, already treated in *section 7.2*. In this mechatronic system only the motors have been changed, because this application has required more powerful features.

### 7.3.2 Model and Feature of the Motor

The model Robotis MX64-AT has the following specifications, collected in the *table 7.2*.

The informations and instructions, that travel in both two directions, are managed by the same *Control Table System*.

It is a structure of data implemented in the DYNAMIXEL. Users can read a specific data to get status of the DYNAMIXEL with READ instruction packets, and modify data as well to control DYNAMIXEL with WRITE instruction packets. The address is a unique value when accessing a specific data in the control table with instruction packets. In order to read or write data, users must designate a specific address in the instruction packet. The used addresses of the *EEPROM AREA* and *RAM AREA* have been collected in the *table 7.3*.

| **Features Of the Motor** | |
|---|---|
| **Item** | **Specification** |
| Baud rate | 9600 bps - 4.5Mbps |
| Control Algorithm | PID CONTROL |
| Degree of Precision | 0.088 ° |
| Running Degree | 0 °-360 ° |
| Weight | 135 g |
| Dimensions (WxHxD) | 40.2mm x 61.1mm x 41mm |
| Gear Ratio | 200:1 |
| Stall Torque | 6.0 Nm @ 12V, 4.1A |
| No Load Speed | 63 rpm @ 12V, |
| Opering Temperature | -5°C - +80°C |
| Input Voltage | 10V - 14.8V (Recommended 7.4) |
| Protocol Type | 2.0 |
| Physical Connection | TTL |
| ID | 0 -252 |
| Feedback | Position, Temperature, Load, Input Voltage, etc... |
| Material | Engineering Plastic |

***Table 7.2:*** Features of the Motor MX64-AT.

For each address it is possible to read or write value in range that is connected to the physical range of the respective physical quantities, as reported in the *table 7.4*.


***Figure 7.21:*** Robotis Motor MX-64AT [21].

| **CONTROL TABLE of EEPRON AREA** | | | | | | |
|---|---|---|---|---|---|---|
| **Address** | **Size [byte]** | **Data Name** | **Access** | **Initial Value** | **Min** | **Max** |
| 7 | 1 | ID | RW | 1 | 0 | 252 |
| 8 | 1 | Baud Rate | RW | 1 | 0 | 7 |
| 11 | 1 | Operating Mode | RW | 3 | 0 | 16 |
| 13 | 1 | Protocol Version | RW | 2 | 1 | 2 |
| 24 | 4 | Moving Threshold | RW | 10 | 0 | 1023 |
| **CONTROL TABLE of RAM AREA** | | | | | | |
| **Address** | **Size [byte]** | **Data Name** | **Access** | **Initial Value** | **Min** | **Max** |
| 64 | 1 | Torque Enable | RW | 0 | 0 | 1 |
| 76 | 2 | Velocity I Gain | RW | 1920 | 0 | 16383 |
| 78 | 2 | Velocity P Gain | RW | 100 | 0 | 16383 |
| 80 | 2 | Position D Gain | RW | 0 | 0 | 16383 |
| 82 | 2 | Position I Gain | RW | 0 | 0 | 16383 |
| 84 | 2 | Position P Gain | RW | 850 | 0 | 16383 |
| 88 | 2 | FF 2nd Gain | RW | 0 | 0 | 16383 |
| 90 | 2 | FF 1st Gain | RW | 0 | 0 | 16383 |
| 104 | 4 | Goal Velocity | RW | - | 0 | 1023 |
| 108 | 4 | Profile Acceleration | RW | 0 | 0 | 32767 |
| 112 | 4 | Profile Velocity | RW | 0 | 0 | 1023 |
| 116 | 4 | Goal Position | RW | 850 | 0 | 4095 |
| 124 | 2 | Present PWM | R | 850 | 0 | 850 |
| 126 | 2 | Present Current | R | - | 0 | 1941 |
| 128 | 4 | Present Velocity | R | - | 0 | 1023 |
| 132 | 4 | Present Position | R | - | 0 | 4095 |
| 136 | 4 | Velocity Trajectory | R | - | 0 | 1023 |
| 140 | 4 | Position Trajectory | R | - | 0 | 1023 |

***Table 7.3:*** Control Table of MX64-AT Motors.

| UNITS TABLE CONVERSION | | | |
|---|---|---|---|
| **Physical Quantities** | **Physical Range** | **Address Range** | **Units** |
| Voltage | 9.5-16.0 V | 95-160 | 0.1 V |
| PWM | 0-100% | 0-850 | 0.118 % |
| Current | 0-6.5A | 0-1941 | 3.36 mmA |
| Acceleration | - | 0-32736 | 214.577 rpm2 |
| Velocity | 0-234 rpm | 0-1023 | 0.229 rpm |
| Position | 0-360 deg | 0-4095 | 0.088 deg |
| Velocity I Gain | 0-0.25 | 0-16383 | 65536 |
| Velocity P Gain | 0-128 | 0-16383 | 128 |
| Position D Gain | 0-1023 | 0-16383 | 16 |
| Position I Gain | 0-0.25 | 0-16383 | 65536 |
| Position P Gain | 0-128 | 0-16383 | 128 |
| FF 2nd Gain | 0-4095 | 0-16383 | 4 |
| FF 1st Gain | 0-4095 | 0-16383 | 4 |

***Table 7.4:*** Unit Table Conversion - Motor MX64-AT.

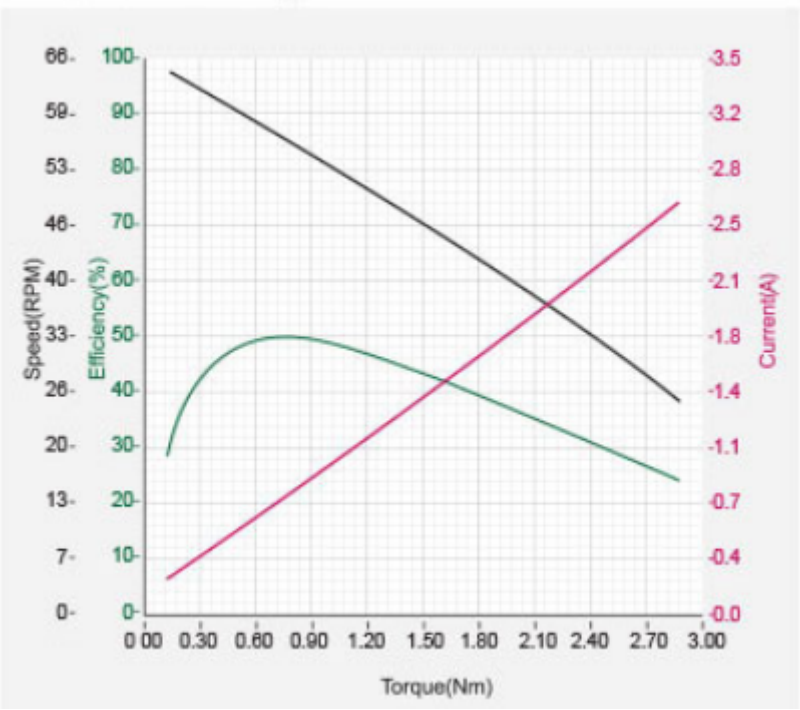



***Figure 7.22:*** Robotis Motor MX-64AT Performance Graph [21].

### 7.3.3 Functioning and Types of Control

This motor has the possibility to be controlled in several control modes (current , velocity, position, pwm, etc...) However, it has been chosen to base the control system on the **position control mode**.

Differently than the previous motor (for which has been necessary to implement an external FFPC), this model has an **inner feed forward position control**.

In fact, it is possible to adapt and optimise the position control, writing in the address of

the Feed Forward Gain, depending on the current application.

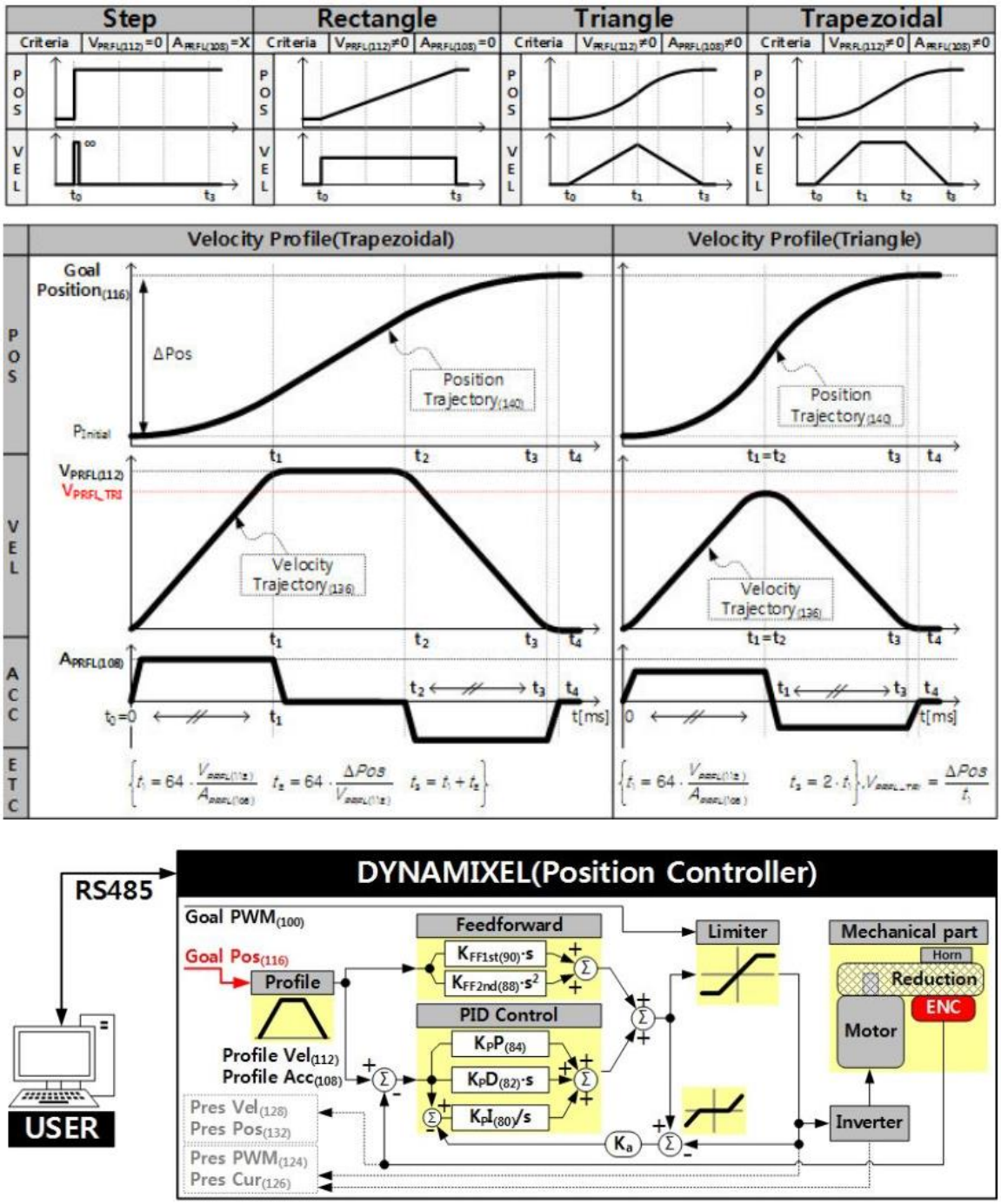



***Figure 7.23:*** Robotis Motor MX-64AT Position Control [21].

The first step is to *set the the profile of acceleration/deceleration control method* to reduce vibration, noise and load of the motor by controlling dramatically changing velocity and acceleration. It is also called **Velocity Profile** as it controls acceleration and deceleration

based on velocity. Dynamixel provides 4 different types of Profile, as it possible to see in *figure 7.23*. Profiles are usually selected by a combination of *Profile Velocity (112)* and *Profile Acceleration (108)*. Triangular and Trapezoidal Profiles exceptionally consider total travel distance($\Delta_{Pos}$, the distance difference between target position and current position) as an additional factor. For convenience, Profile Velocity (112) is abbreviated to $V_{PRFL}$ and Profile Acceleration(108) is abbreviated to $V_{PRFL}$.

After setting the profile of velocity, the instruction from the user is received by Dynamixel, it takes following steps until driving the horn:

1. An Instruction from the user is transmitted via Dynamixel bus, then registered to *Goal Position (116)*.

2. Goal Position (116) is converted to target position trajectory and target velocity trajectory by *Profile Velocity (112)* and *Profile Acceleration (108)*.

3. The target position trajectory and target velocity trajectory is stored at *Position Trajectory (140)* and *Velocity Trajectory (136)* respectively.

4. Feedforward and PID controller calculate *PWM output* for the motor based on target trajectories.

5. *Goal PWM (100)* sets a limit on the calculated PWM output and decides the final PWM value.

6. The final PWM value is applied to the motor through an Inverter, and the horn of Dynamixel is driven.

7. Results are stored at *Present Position (132)*, *Present Velocity (128)*, *Present PWM (124)* and Present Current (126).

### 7.3.4 Experiments

In order to understand the motor response in several working situations it has been led different experiment on both the bench and the robot assembly, as follows:

- **on the bench with no load**;
- **on the robot with no load**;
- **on the robot with a calibrated 500g load on the middle of the platform**;
- **on the robot passing through the forward singularities**;
- **evaluating the influence of the parameters of the timing law**:
  - *final time* $t_f$;
  - *total travel distance* $\Delta_{pos}$.

#### 7.3.4.1 Considerations on the Experiments

First, analysing all of the graphs, it can be seen how it is always possible to achieve the **goal position**, with an error of $0.088deg$, compatible to the resolution of the encoder. It may mean the motors are overestimated for the applications, because they are not affected by the non linearities of the system, working more or less as a linear one, although it is not the same.

By comparing the **response on bench** (*figure 7.24*) with the **response on the robot** (*figure7.25*), without any load, it can be seen how the behaviour of the motors are more or less the same. It is possible to affirm that the revolute masses on the robot assembly do not affect the motors response, in terms of disturbs, as it has been assumed, by neglecting the dynamic of the revolute masses themselves.

Moreover, by putting **a calibrated mass on the middle of the End Effector** (*figure 7.26*), the motor behaviour does not change so much, because the *weight force* is always on the center of the mechanism CM.

However, it is different if we **pass from the forward singularity**, in which it is impossible to control the platform. In fact, depending on how the platform orients itself, because of imbalance of the masses, we can observe *a peak* in the trend of the velocity, that is arbitrary and out of control. In *figure 7.27* the peak is down, meaning that arbitrary orientation of the platform brakes the actuator, introducing a disturb, while in the *figure 7.28* the peak is up, and in this case this uncontrolled rotation helps the motors.

Moreover, due to this uncontrolled motions of the platform, **it fluctuates between the final position**, achieved by the actuators, influencing the final time in which all the system is stable. This problem affects the entire motion, making a delay in the following step of the motion planning and the delay also depends on the flection of the distal and the proximal links, due to the external load.

Furthermore, it is important to investigate on the limits of the characteristic parameters of the trapezoidal timing law. In fact, if we **fix the final position** and decrease the time of the motion (*figure 7.29, 7.30*), as a consequence the velocity increases, putting the system into oscillation between the final position, because of the brusque stop. This case is similar to the fluctuation of the platform around the singularity, mentioned before.

By **fixing the time and increase the final position** (*figure 7.31, 7.32*) is comparable to the previous case, beacuse of the instability of the system.

In the end, **to reduce vibration, noise and load of the motor**, it is recommended to plan the motion, in order to be as fluid is possible and to choose a set of parameters, respecting this mechanical limits, due to a building process mainly made of ABS plastic.

#### 7.3.4.2 On the Bench With No Load

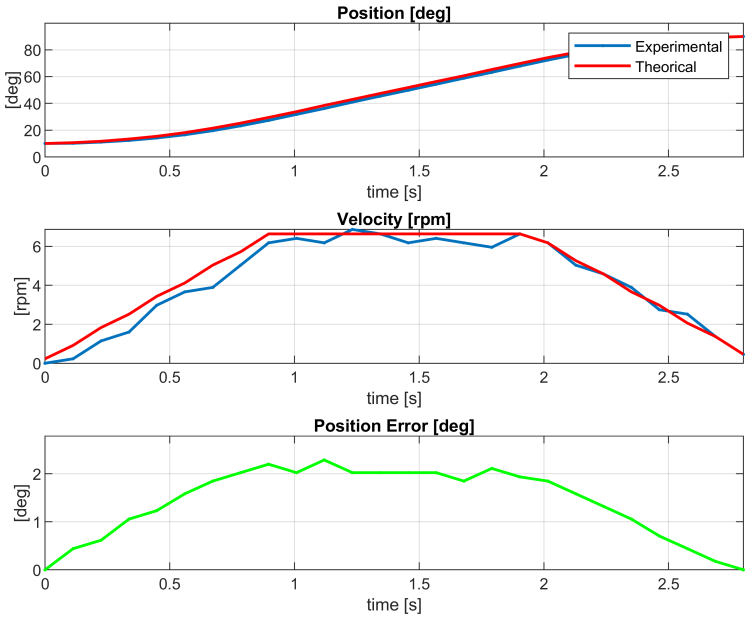



***Figure 7.24:*** Experiment on Bench Without Load

#### 7.3.4.3 On the Robot with No Load

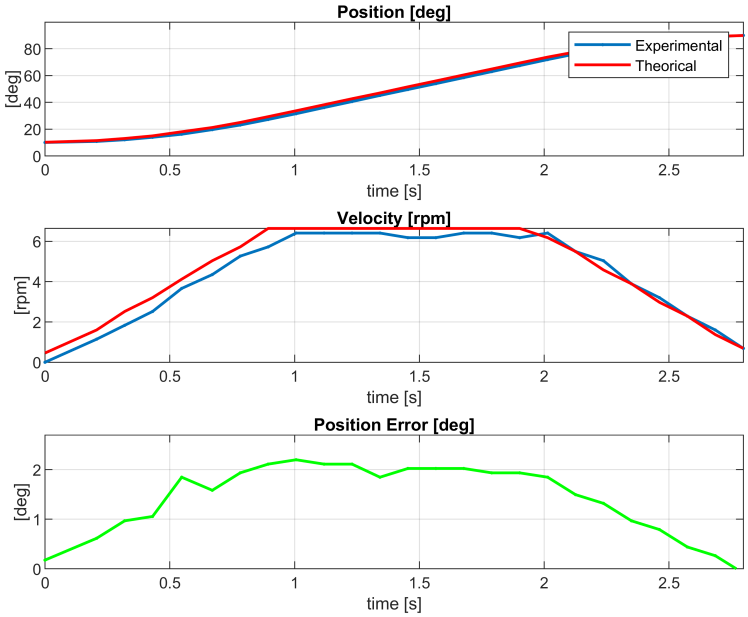



***Figure 7.25:*** Experiment on the Robot Without Load.

#### 7.3.4.4 On the Robot with Calibrated 500g Load

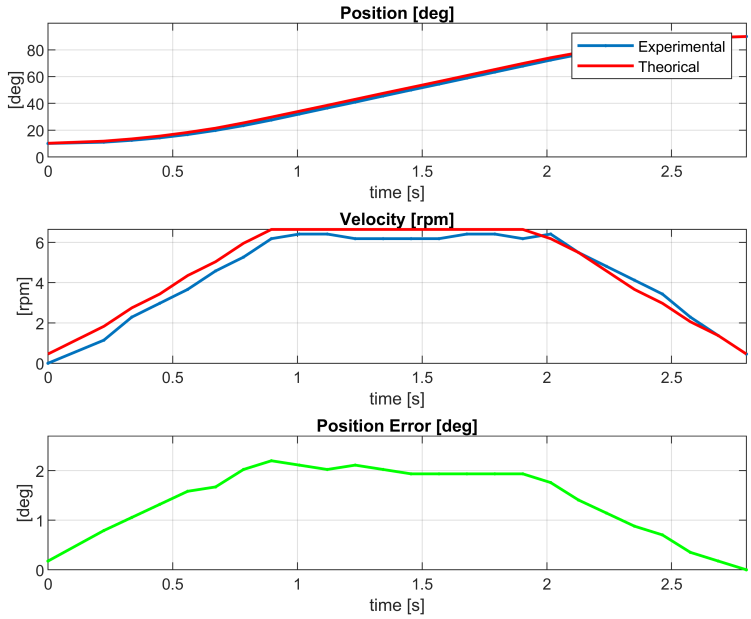



***Figure 7.26:*** Experiment on the Robot with Calibrated 500g Load.

#### 7.3.4.5 On the Robot Passing Through Forward Singularity

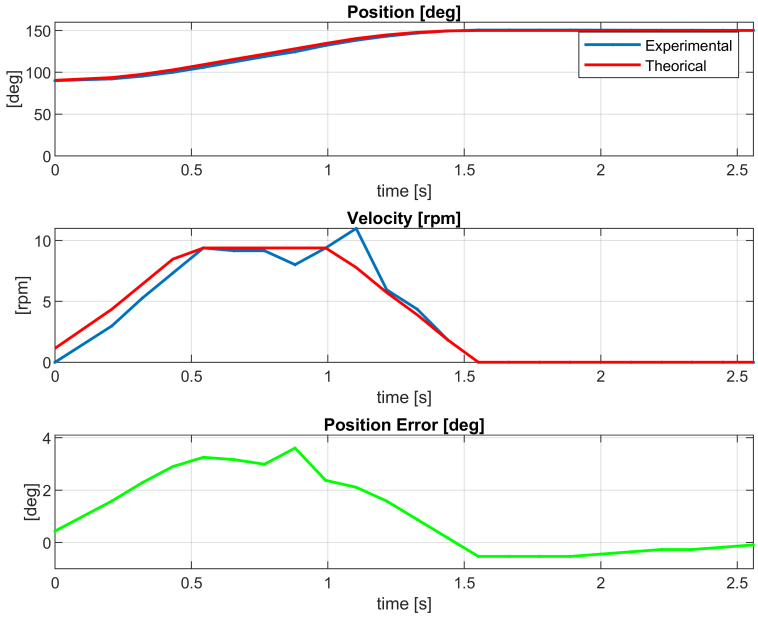



***Figure 7.27:*** Experiment on the Robot Passing Through Forward Singularity - 1.

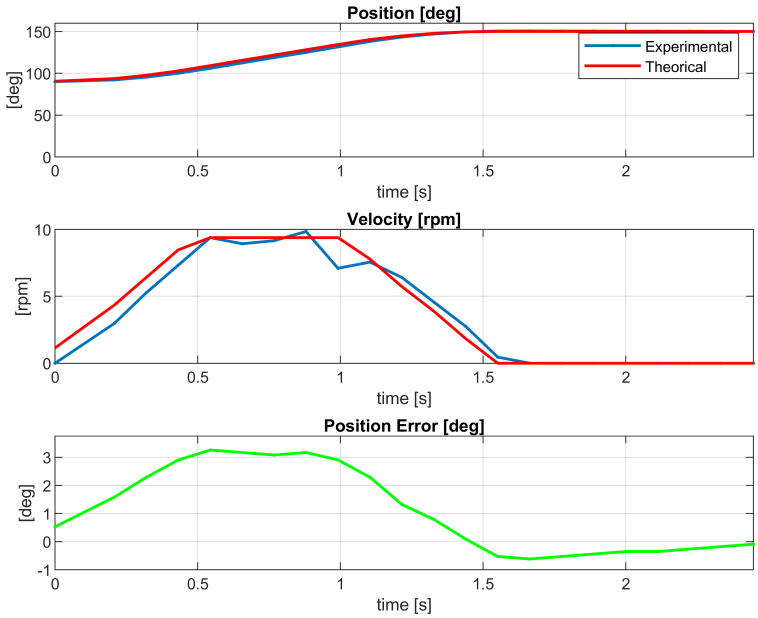



***Figure 7.28:*** Experiment on the Robot Passing Through Forward Singularity - 2.

#### 7.3.4.6 Influence of the Final Time $t_f$

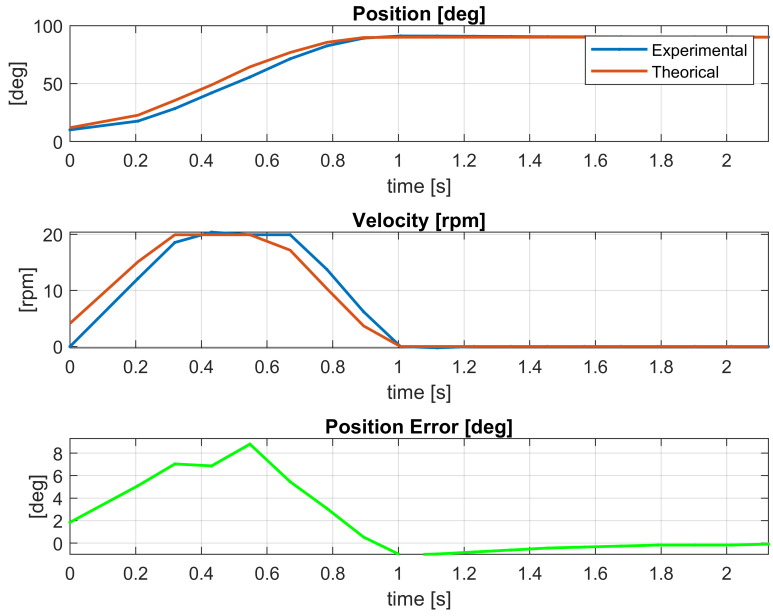



***Figure 7.29:*** Influence of the parameters of the timing law $t_f = 1s$.

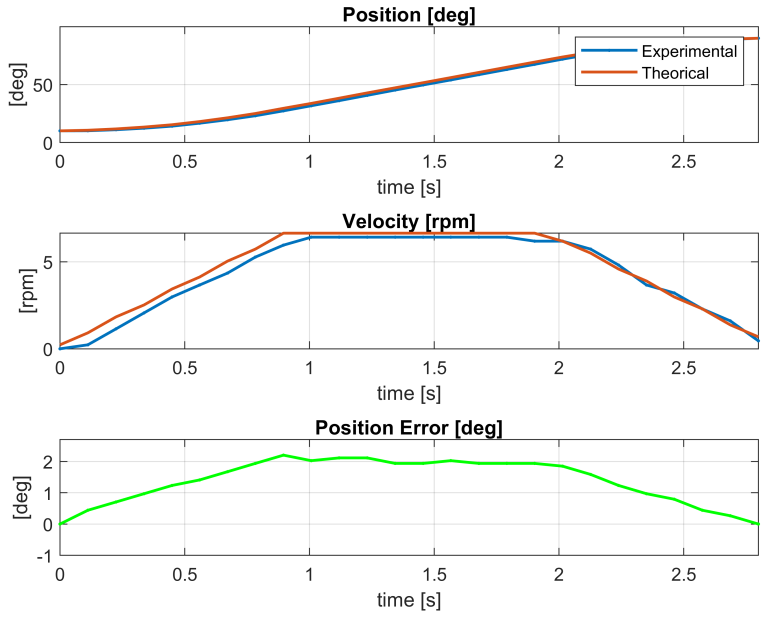



***Figure 7.30:*** Influence of the parameters of the timing law $t_f = 3s$.

#### 7.3.4.7 Influence of the Total Travel Distance $\Delta q$

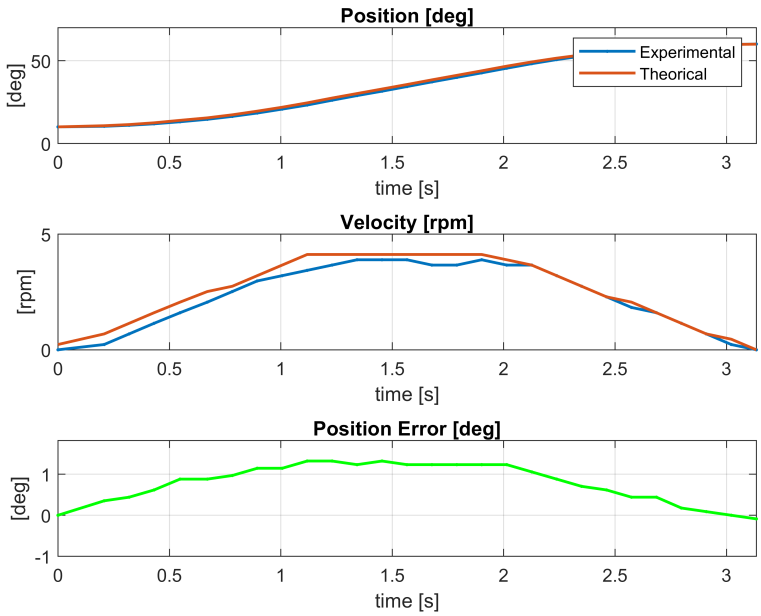



***Figure 7.31:*** Influence of the parameter of the timing law $\Delta q = 50deg$.

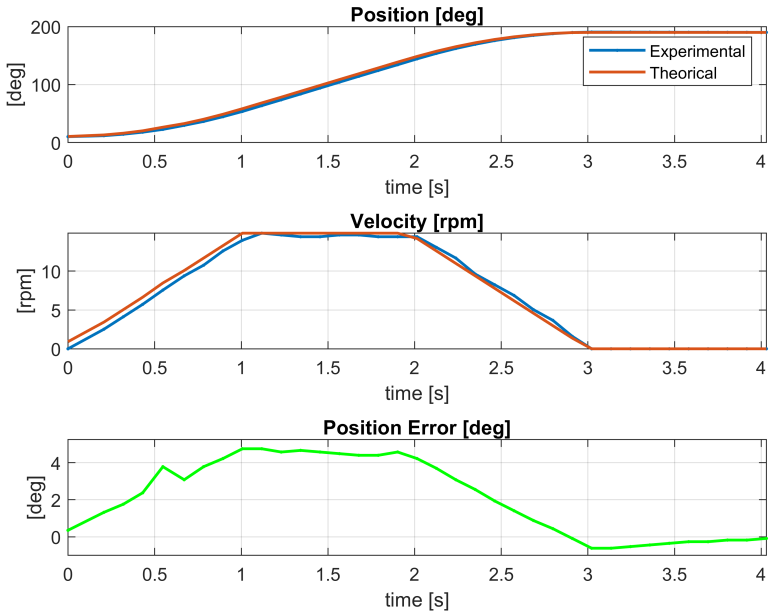



***Figure 7.32:*** Influence of the parameters of the timing law $\Delta q = 180deg$.

# Chapter 8

# Costs Analysis

This chapter details the budget associated with all the components that are involved in the building the 3-RRR Coaxial SPM. Due to the fact that the project is based on the implementation of a new prototype, the associated costs do not take into account for a possible commercialization. For this reason, a viability economic analysis has not been carried out.

## 8.1 Project Budget

As explained in the previous chapters, all the mechanism components have been printed by using the FDM$^{TM}$ Technology (fused deposition modelling) in ABS-P430. Therefore, for the printed components a general assessment has been estimate, involving cost of material, energy supply and labour cost.
For the commercial component, reference was made to the *"https://es.rs-online.com/web/"*, except for the Motors, bought from the seller website.

| **Material Cost** | | | |
|---|---|---|---|
| **ComponentS** | **Unit Cost** | **Quantity** | **Cost** (€) |
| Bearings (SKF, RS Components) | - | - | 174 |
| High-Speed Steel Bars | 30 € | 2 | 60 |
| Dynamixel MX-64AT | 353 € | 3 | 1059 |
| ABS-P430 Components | 82 €/$Kg$ | Estimate | 350 |
| **Total Costs** | | | 1643 |

| **Additional Costs** | | | |
|---|---|---|---|
| **Cost Items** | €/h | **Hours (h)** | **Cost** (€) |
| Engineering Manpower | 12 | 750 | 9000 |
| 3D printer amortization | 5 | 125 | 625 |
| Laboratory tools amortization | 0,6 | 750 | 450 |
| **Total Costs** | | | 10075 |

| **Costs Analysis** | |
|---|---|
| Material Cost | 1643 € |
| Additional Cost | 10075€ |
| **Total cost** | 11718 € |

# Chapter 9

# Conclusions

In this concluding chapter it has been summarized the contributions of this thesis and discussed the important directions of future work.

## 9.1 Summary of Thesis Contributions

This work firstly has had the *scope* to investigate on the coaxial configuration of a spherical parallel manipulator, by studying its kinematics (inverse and forward one), the relative singularities and the mobility range of the spherical workspace and by developing the 3D model design and the modelling the control of the servomotors. Both the two cores of the work, the analytical study and the physical implementation, have been carried out in parallel, leading to change and to rearrange the final proposal of the thesis itself.

One of the goal of this research has been to *analytically implement the kinematics* of this robot, in a new way, distancing itself from the common literature [10, 9, 12, 4], as it has been debated in the relative chapter above. The implementation of Thomas' method [30, 27, 31, 13, 28] for this particular architecture has led to obtain the expected results with a process, which turned out to be the best one for simplicity, generality, and optimality. In fact, it has been possible to solve the forward kinematics with a linear polynomial equation that has the minimal degree, for all the set of actuator angles, chosen as the input of the process.

Then, starting from the kinematics analysis, it has been investigated the influence of the characteristics angles $\alpha_1$, $\alpha_2$ and $\gamma_1$ of the robot on the both *singularities* and *workspace limits* [11, 7, 14, 2]. It has been compared the mobility range of a general configuration ($\gamma_1 = \pi/3,\ \alpha_1 = \pi/3,\ \alpha_2 = \pi/2$) to the coaxial prototype, underlining how the workspace increases if $\gamma_1 = 0,\ \alpha_1 = \alpha_2 = \pi/2$. On the contrary, studying the Jacobian matrix, it can be stated the number and types of singularities grow up, using these angles. Therefore, it is possible to argue that the optimisation of the mobility range and the singularities is a trade-off; a better workspace carries on more singularities that have to be avoided, during a path motion in the manipulations of the robot. This result could be taken a possible optimisation of the design into account, listed as a future work. According to these results, it has been taken into account the possibility to realise both the two configurations in order to show how the characteristic parameters of the robot influence its behaviour. In fact, the optimal characteristic angles could be $\alpha_1 = 60\ deg$, $\alpha_2 = 90\ deg$ and $\gamma_1 = 0\ deg$, allowing to achieve reasonable orientations, that cannot be reach with the existing prototype.

As for the *3D model of the robot*, the first concept has been developed as a design with

only one motor, the use three electro-brakes and a system of pulley-belt to move the robot. Nevertheless, in order to actually build it during the working period on the research center, it has been preferred to not complicate the driving system and to only focus on the functioning, leaving out a complex prototype as a future task. Therefore *the final prototype* is based on the use of three servomotors to actuate the respective degrees of freedom with a basic transmission system made of only one gear mates with 1:1 ratio between each coaxial shaft and the gear motor.

Moreover, during the development of the concept there were some issues not only in terms of optimisation of the design, but also as for the printing process. The 3D drawing has been continuously updated, taking into account the limits of the 3D printer and the supplies of materials and mechanical components, in the research center. This aspect has led to sacrifice, sometimes, the best drawing solution for respecting these constrains, justifying several choices and decisions that could be unusual for a traditional best practice in a design terms.

In the end, it is possible to do some *considerations* as for the realisation of the prototype. The choice to use *ABS*, as the main material, leads to rebuild several parts, according to a functional design, during the realisations.

Considering the *robot assembly*, the prototype has been achieved a good stiffness and the friction is negligible for this application. There are, of course, misalignment errors due to the limits of the 3d printer, in term of tolerance, and a imperfect mounting process, because some parts have been stuck together, losing the exact reference surfaces.

The moving parts that perform the manipulation have a mass, negligible comparing with the driving system, according to the choice to study the robot in an only static way.

Moreover, as for the *mechanical transmission assembly*, the three motors have the capability to control the platform with an acceptable accuracy, bearing the inner forces and the several momentum. As well, the play between the tooth of the gear mate it can be considerer not so relevant.

However, it is possible to *suggest precautions* to increase the stiffness of the robot assembly, if it need to be reprinted. In the distal and proximal links, it would be optimal to insert a liner made of a carbon fibre. This would be raise the stiffness of the links, leaving more or less unchanged the mass of the moving parts.

Overall, the response of the system, subjected to different working conditions, is acceptable according to the prototype could be improved, and it can be considered a good result in the concise working time, spent in the research center.

## 9.2 Direction for Future Work.

As mentioned in the previous section, in order to manage the time available, different points and initial proposals have been listed for future works, as follows:

- *studying the new configuration of the robot assembly, in analytical way*: after the workspace and singularity studies, it has been noticed that it may be better to rebuilt the robot assembly, changing the characteristic angles, with the following value $\alpha_1 = 60\ deg$, $\alpha_2 = 90\ deg$ and $\gamma_1 = 0\ deg$. This will be reduce the mobility range, but, at the same time, it will increase the number of the possible orientations of the end effector.

- *doing a dynamic analysis of the system*: even if the static approximation of the system works in an acceptable way for a prototype, it should be better to study the dynamic

in order to validate the dimensions of the revolute mass and to linearly identify the system.

- *introducing a non linear control*: given that the system is non linear and the inner PID has been enough to control the motors, it should be appropriate to model a non linear-control. However, it should be necessary to control the motors at the same time, with a Multiple-Input Multiple-Output (MIMO) Processes.

- *optimising the design transmission with only one motor*: a first concept has been drown, favouring, later, a concept with three motor, for the reasons already explained. However, it should be interesting to optimise this other prototype, because it could be reduced the operative cost of the robot, from industrial standpoint.

- *putting a camera on the End Effector platform, for realising a path motion*: it could be relevant to investigate on the behaviour of the robot when the camera follows a ball in a track. It could be possible, in this case, to extract the following orientation of the robot, to rebuilt the path motion realised by the robot, for analysing it.

# Chapter 10

# Attachments

## 10.1 Appendix A

### Inverse Kinematics

```
%% Initialization code
clear;
close all;
clc;
%% Given matrix
Ree = rotx(pi/3)*roty(pi/6)*rotz(pi/2);

%% Calculation and drawing the eight solution
figure()
% 1°
subplot(241)
[eul1(1,:), eul2(1,:), eul3(1,:)] = DrawRobot(Ree, 1, 1, 1);
title('Sol 1');
% 2°
subplot(242)
[eul1(2,:), eul2(2,:), eul3(2,:)] = DrawRobot(Ree, 2, 1, 1);
title('Sol 2');
% 3°
subplot(243)
[eul1(3,:), eul2(3,:), eul3(3,:)] = DrawRobot(Ree, 1, 2, 1);
title('Sol 3');
% 4°
subplot(244)
[eul1(4,:), eul2(4,:), eul3(4,:)] = DrawRobot(Ree, 2, 2, 1);
title('Sol 4');
% 5°
subplot(245)
[eul1(5,:), eul2(5,:), eul3(5,:)] = DrawRobot(Ree, 1, 1, 2);
title('Sol 5');
% 6°
subplot(246)
[eul1(6,:), eul2(6,:), eul3(6,:)] = DrawRobot(Ree, 2, 1, 2);
title('Sol 6');
% 7°
subplot(247)
[eul1(7,:), eul2(7,:), eul3(7,:)] = DrawRobot(Ree, 1, 2, 2);
title('Sol 7');
% 8°
```

```
subplot(248)
[eul1(8,:), eul2(8,:), eul3(8,:)] = DrawRobot(Ree, 1, 2, 2);
title('Sol 8');
Sol=rad2deg([eul1 eul2 eul3 zeros(8,1)]);
Dof=[Sol(:,1) Sol(:,4) Sol(:,7) Sol(:,10)];
%% Extraxt the degrees of freedom from the eigth solutions.
for i=1:8
    if (sign(Sol(i,1))==sign(Sol(i,4)))&&(sign(Sol(i,4))==sign(Sol(i,7)))
        if (Sol(i,1)<Sol(i,4)-15)&&(Sol(i,4)<Sol(i,7)-15)
            % Configuration without interference
            Sol(i,end)=1;
            subplot(2,4,i)
            [x,y,z] = sphere;
            surf(x,y,z, 'EdgeColor', 'none', 'Facecolor', 'g','FaceAlpha', 0.1);
        else
            subplot(2,4,i)
            [x,y,z] = sphere;
            surf(x,y,z, 'EdgeColor', 'none', 'Facecolor', 'r','FaceAlpha', 0.1);
        end
    elseif (sign(Sol(i,1))==sign(Sol(i,4)))&&(sign(Sol(i,4))~=sign(Sol(i,7)))
        if Sol(i,1)<Sol(i,4)-15
            % Configuration without interference
            Sol(i,end)=1;
            subplot(2,4,i)
            [x,y,z] = sphere;
            surf(x,y,z, 'EdgeColor', 'none', 'Facecolor', 'g','FaceAlpha', 0.1);
        else
            subplot(2,4,i)
            [x,y,z] = sphere;
            surf(x,y,z, 'EdgeColor', 'none', 'Facecolor', 'r','FaceAlpha', 0.1);
        end
    elseif (sign(Sol(i,1))~=sign(Sol(i,4)))&&(sign(Sol(i,4))==sign(Sol(i,7)))
        if Sol(i,4)<Sol(i,7)-15
            % Configuration without interference
            Sol(i,end)=1;
            subplot(2,4,i)
            [x,y,z] = sphere;
            surf(x,y,z, 'EdgeColor', 'none', 'Facecolor', 'g','FaceAlpha', 0.1);
        else
            subplot(2,4,i)
            [x,y,z] = sphere;
            surf(x,y,z, 'EdgeColor', 'none', 'Facecolor', 'r','FaceAlpha', 0.1);
        end
    else
        Sol(i,end)=0;
    end
    i
    Sol(i,end)
    Dof(i,1:end-1)
end
Sol
Dof=[Sol(:,1) Sol(:,4) Sol(:,7) Sol(:,10)]
Dof_physical=[Dof(:,1), -120+Dof(:,2), 120+Dof(:,3)]
```

```
function [eul1, eul2, eul3] = DrawRobot(T, cf1, cf2, cf3)
%UNTITLED4 Summary of this function goes here
%   Detailed explanation goes here

```

```
% cf=1 flag, first solution
% cf=2 flag, second solution [eulII(1) eulII(2) eulII(3)]=...
% [eulI(1)+pi, pi-eulI(2), eulI(3)+pi]

eul1 = rot2eulZYX(T);
eul2 = rot2eulZYX(T*rotz(-2*pi/3));
eul3 = rot2eulZYX(T*rotz(2*pi/3));

trplot(eye(3), 'color', 'r', 'arrow', 'text_opts', ...
    {'FontSize', 12, 'FontWeight', 'bold'});
hold on;

trplot(T, 'color', 'g', 'arrow', 'text_opts', ...
    {'FontSize', 12, 'FontWeight', 'bold'});
hold on;

% [x,y,z] = sphere;
% surf(x,y,z, 'EdgeColor', 'none', 'Facecolor', 'interp','FaceAlpha', 0.4);

%%% First leg

if cf1 == 1
    R1 = rotz(eul1(1));
    DrawSphericalArc([0 0 -1], R1(:,2)', 'k');
    R2 = R1*roty(eul1(2));
    DrawSphericalArc(R1(:,2)', R2(:,1)', 'k');
    R3 = R2*rotx(eul1(3));
    DrawSphericalArc(R2(:,1)', R3(:,3)', 'g');
end
if cf1 == 2
    R1 = rotz(eul1(1)+pi);
    DrawSphericalArc([0 0 -1], R1(:,2)', 'k');
    R2 = R1*roty(pi-eul1(2));
    DrawSphericalArc(R1(:,2)', R2(:,1)', 'k');
    R3 = R2*rotx(eul1(3)+pi);
    DrawSphericalArc(R2(:,1)', R3(:,3)', 'g')
    eul1=[eul1(1)+pi, pi-eul1(2), eul1(3)+pi];
end

%%% Second leg

if cf2 == 1
    R1 = rotz(eul2(1));
    DrawSphericalArc([0 0 -1], R1(:,2)', 'b');
    R2 = R1*roty(eul2(2));
    DrawSphericalArc(R1(:,2)', R2(:,1)', 'b');
    R3 = R2*rotx(eul2(3));
    DrawSphericalArc(R2(:,1)', R3(:,3)', 'g');
end

if cf2 == 2
    R1 = rotz(eul2(1)+pi);
    DrawSphericalArc([0 0 -1], R1(:,2)', 'b');
    R2 = R1*roty(pi-eul2(2));
    DrawSphericalArc(R1(:,2)', R2(:,1)', 'b');
    R3 = R2*rotx(eul2(3)+pi);
    DrawSphericalArc(R2(:,1)', R3(:,3)', 'g');
    eul2=[eul2(1)+pi, pi-eul2(2), eul2(3)+pi];
end
```

```
64
65 %%% Third leg
66
67 if cf3 == 1
68     R1 = rotz(eul3(1));
69     DrawSphericalArc([0 0 -1], R1(:,2)', 'r');
70     R2 = R1*roty(eul3(2));
71     DrawSphericalArc(R1(:,2)', R2(:,1)', 'r');
72     R3 = R2*rotx(eul3(3));
73     DrawSphericalArc(R2(:,1)', R3(:,3)', 'g');
74 end
75
76 if cf3 == 2
77     R1 = rotz(eul3(1)+pi);
78     DrawSphericalArc([0 0 -1], R1(:,2)', 'r');
79     R2 = R1*roty(pi-eul3(2));
80     DrawSphericalArc(R1(:,2)', R2(:,1)', 'r');
81     R3 = R2*rotx(eul3(3)+pi);
82     DrawSphericalArc(R2(:,1)', R3(:,3)', 'g');
83     eul3=[eul3(1)+pi, pi-eul3(2), eul3(3)+pi];
84 end
85 end
```

```
1  function eul = ikine(rotm)
2  if rotm(3,1) < 1
3      if rotm(3,1) > -1
4          % case 1: if r31 ~= +-1
5          % Solution with positive sign. It limits the range of the values
6          % of theta_y to (-pi/2, pi/2):
7          eul(1) = atan2(rotm(2,1), rotm(1,1)); % theta_z
8          eul(2) = asin(-rotm(3,1));            % theta_y
9          eul(3) = atan2(rotm(3,2), rotm(3,3)); % theta_x
10     else
11         % case 2: if r31 = -1
12         % theta_x and theta_z are linked --> Gimbal lock:
13         % There are infinity number of solutions for
14         % theta_x - theta_z = atan2(-r23, r22).
15         % To find a solution, set theta_x = 0 by convention.
16         eul(1) = -atan2(-rotm(2,3), rotm(2,2));
17         eul(2) = pi/2;
18         eul(3) = 0;
19     end
20 else
21     % case 3: if r31 = 1
22     % Gimbal lock: There is not a unique solution for
23     % theta_x + theta_z = atan2(-r23, r22),
24     % by convention, set theta_x = 0.
25     eul(1) = atan2(-rotm(2,3), rotm(2,2));
26     eul(2) = -pi/2;
27     eul(3) = 0;
28 end
29 end
```

## 10.2 Appendix B

### Forward Kinematics

```
%% Cleaning Code
clear
close all
clc
%% Calculation of the Square Distances Matrix S
% Initialization of the matrix of squared distances according
% to Nico's data: Section V of
% http://www.iri.upc.edu/people/thomas/papers/ICRA2013.pdf

S=sym('S',[7,7]); % using 7 points (3+4)

% Platform has 4 points 1,2,3,4
% Base has 3 points 5 6 7
% The translated point is '3'

M = struct();

M.t1 = deg2rad(15);
M.t2 = deg2rad(5);
M.t3 = deg2rad(30);

M.r_sphere = 1;

[M] = square_distances(M);

% Square distances of platform
S(1,2)= M.s12;
S(1,4)= M.s14;
S(2,4)= M.s24;

% Square distances between platform and CM
S(1,3)= M.s13;
S(2,3)= M.s23;
S(3,4)= M.s34;

% Square distances among the legs and the arms
S(1,5)= M.s15; % leg
S(2,7)= M.s27; % leg
S(4,6)= M.s46; % leg

% Square distances among the arms and the CM
S(3,5)=  M.s35; % leg
S(3,6)=  M.s36; % leg
S(3,7)=  M.s37; % leg


% Square distances among the head of the legs
S(5,6)= M.s56;
S(5,7)= M.s57;
S(6,7)= M.s67;

%% Ensure that the input matrix is symmetric
S=Symmetrize(S);
```

```
54
55 %% Define the strip of tetrahedra
56 ST={[3 5 7 2 6] [2 3 5 1 6] [1 2 3 4 6]};
57
58 % The trilateration sequence to generate coordinates
59 T={[5 6 7 3] [5 3 7 2] [5 2 3 1] [1 2 3 4]};
60
61 % Sign of the tetrahedra in the trilateration sequence
62 SG=[0 0 0 1];
63
64 %% Determine the univariate closure polynomial and its roots
65 P=UCPolynomial(S,ST);
66
67 %% Find the roots of the polynomial
68 [R,V]=SolveSystem(S,P);
69
70 %% Obtain coordinates from the roots
71 C = GetCoordinates(S,T,SG,R,V,1e-4);
72 %% Plot the results
73 % DrawRobot(C,M);
74
75 %%  Rearranging Solution
76 Sol1 = C{1};
77 Sol1 = Sol1 - Sol1(:,3);
78
79 Sol2 = C{2};
80 Sol2 = Sol2 - Sol2(:,3);
81
82 Sol3 = C{9};
83 Sol3 = Sol3 - Sol3(:,3);
84
85 Sol4 = C{10};
86 Sol4 = Sol4 - Sol4(:,3);
87
88 M.Sol = [Sol1
89        Sol2
90        Sol3
91        Sol4];
92 filename = 'Solution.xlsx';
93 xlswrite(filename,M.Sol)
94 %% Remove intermediate variables
95 clear V;
```

```
1  function  [M] = square_distances(M)
2
3  R1 = trotz(M.t1);
4  R2 = trotz(M.t2);
5  R3 = trotz(M.t3);
6  p50 = transl(M.r_sphere,0,0);
7  p50 = p50(:,4);
8  p60 = trotz(2*pi/3)*p50;
9  p70 = trotz(4*pi/3)*p50;
10
11 p5 = R1*p50;
12 p5 = p5(1:3);
13
14 p6 = R2*p60;
15 p6 = p6(1:3);
```

```
16
17 p7 = R3*p70;
18 p7 = p7(1:3);
19
20 % Square distances among the head of the leg
21 M.s56 = (norm(p5 - p6))^2;
22 M.s57 = (norm(p5 - p7))^2;
23 M.s67 = (norm(p6 - p7))^2;
24
25 % Square distances from the center of the mechanism CM
26 M.s35 = M.r_sphere^2;
27 M.s36 = M.r_sphere^2;
28 M.s37 = M.r_sphere^2;
29
30 M.s13 = M.r_sphere^2;
31 M.s23 = M.r_sphere^2;
32 M.s34 = M.r_sphere^2;
33
34 % Square distances among the edges of the mobile platform
35 M.s12 = (sqrt(3)*M.r_sphere)^2;
36 M.s14 = (sqrt(3)*M.r_sphere)^2;
37 M.s24 = (sqrt(3)*M.r_sphere)^2;
38
39 % Square distances among the links
40 M.s46 = (M.r_sphere*sqrt(2))^2;
41 M.s15 = (M.r_sphere*sqrt(2))^2;
42 M.s27 = (M.r_sphere*sqrt(2))^2;
43
44 end
```

```
1  % Ensures that a matrix is symmetric
2  %
3  % Inputs:
4  %       S: The matrix with only the upper triangular part filled.
5  %
6  % Outputs:
7  %       S: The compleated matrix.
8  %
9  function S=Symmetrize(S)
10   [np,np1]=size(S);
11   if np~=np1
12     error('Non squared input distance matrix');
13   end
14   for i=1:np
15     S(i,i)=0;
16     for j=1:i-1
17       S(i,j)=S(j,i);
18     end
19   end
```

```
1  % Computes the univariate closure polynomial for a problem
2  % defined by
3  %     S: The matrix of squared distances.
4  %     T: The strip of tetrahedra.
5  %
6  % The outputs are:
```

```
7  %     P: The univariate closure polynomial.
8  %     R: The roots of the univariate closure polynomial.
9  %
10 function P=UCPolynomial(S,ST)
11
12   % closure squared distance (save it before computing it analytically)
13   cs=S(ST{end}(4),ST{end}(5));
14
15   % Number of steps in the trilateration sequence
16   nt=size(ST,2);
17
18   % Auxiliar variables
19   DS = sym('DS', [1, nt]); % Symbol used to represent the square root
20   DD = sym('DD', [1, nt]); % The contents of the square root
21   A  = sym('DS', [1, nt]); % The expression of the refence area (if variable)
22
23   fprintf('Chaining over the strip  : ');
24   tic;
25
26   for i=1:nt
27     st=ST{i};
28     n=nt-i+1;
29     [S,DD(n),A(n)]=FillDistance(S,st(1),st(2),st(3),st(4),st(5),DS(n));
30   end
31
32   % closure condition
33   c=S(ST{end}(4),ST{end}(5))-cs;
34   [P,~] = numden(c);
35
36   fprintf('%u seconds\n',toc);
37
38   fprintf('Removing square roots    : ');
39   tic;
40
41   % Eliminate the square roots
42   for i=1:nt
43     if DD(i)~=0 % If the square root is not null
44       P=EliminateSquareRoot(P,DS(i),DD(i));
45     end
46   end
47
48   fprintf('%u seconds\n',toc);
49
50   fprintf('Removing non-null factors: ');
51   tic;
52
53   % Remove the non-null factors (variable areas)
54   for i=1:nt
55     if ~isempty(symvar(A(i))) % if the area is variable
56       P=RemoveFactor(P,A(i));
57     end
58   end
59   fprintf('%u seconds\n',toc);
60
61   % simplify
62
63   fprintf('Normalizing              : ');
64   tic;
65   lcof=feval(symengine,'lcoeff',P);
```

```
66    P=P/lcof;
67    fprintf('%u seconds\n',toc);
```

```
1  % Solves a system of equations
2  %
3  % Inputs:
4  %        S: The matrix of squared distances. Used to get the row/column of
5  %           the variables in the system.
6  %        P: The system of equations (a cell array).
7  % Outputs:
8  %        R: The roots. A cell array. R{i}.v is the i-th valid value for the
9  %           first variable and R{i}.VS is a root structure with the valid
10 %           values for the remaining variables, assuming the first is
11 %           already assigned. Essentially R is recursive structure.
12 %        V: The variables (row/column in S for each variable)
13 %
14 function [R,V]=SolveSystem(S,P)
15
16   n=size(P,2); % number of equations
17   fprintf('Solving the system       : ');
18   tic;
19   switch n
20     case 1
21       if iscell(P)
22         E=P{1};
23       else
24         E=P;
25       end
26
27       v=symvar(E);
28       if length(v)~=1
29         error('A single equation with more than one variable');
30       end
31       [r,c]=find(S==v,1);
32       % sol=vpa(roots(sym2poly(E)),15);
33       sol=feval(symengine,'numeric::polyroots',expand(E));
34       ns=length(sol);
35       R=cell(1,ns);
36       for i=1:ns
37         R{i}.v=sol(i);
38         R{i}.VS={};
39       end
40
41       V={[r c]};
42
43     case 2
44       v1=symvar(P{1});
45       v2=symvar(P{2});
46       v=union(v1,v2);
47
48       if length(v)~=2
49         error('Two equation with less/more than two variables');
50       end
51
52       if length(v1)==1 || length(v2)==1
53
54         if length(v1)==1
55           P1=P{1};
```

```
56            P2=P{2};
57          else
58            P1=P{2};
59            P2=P{1};
60          end
61
62          % Solve P{1}, replace in P{2}, and solve
63          %sol1=vpa(roots(sym2poly(P1)),15);
64          sol1=feval(symengine,'numeric::polyroots',expand(P1));
65          ns1=length(sol1);
66
67          R=cell(1,ns1);
68
69          for i=1:ns1
70            R{i}.v=sol1(i);
71            if isreal(sol1(i)) && sol1(i)>=0
72              P2s=subs(P2,v1,sol1(i));
73              %sol2=vpa(roots(sym2poly(P2s)),15);
74              sol2=feval(symengine,'numeric::polyroots',expand(P2s));
75              ns2=length(sol2);
76              R{i}.VS=cell(1,ns2);
77              for j=1:ns2
78                R{i}.VS{j}.v=sol2(j);
79                R{i}.VS{j}.VS={};
80              end
81            else
82              R{i}.VS={};
83            end
84          end
85
86        else
87
88          % Compute the resultant in one variable and solve
89          R1=feval(symengine,'polylib::resultant',P{1},P{2},v1(1));
90          % sol1=vpa(roots(sym2poly(R1)),15);
91          sol1=feval(symengine,'numeric::polyroots',expand(R1));
92
93          % Now the resultant in the other variable and solve
94          R2=feval(symengine,'polylib::resultant',P{1},P{2},v1(2));
95          % sol2=vpa(roots(sym2poly(R2)),15);
96          sol2=feval(symengine,'numeric::polyroots',expand(R2));
97
98          % Patch the results
99          ns1=length(sol1);
100         ns2=length(sol2);
101         R=cell(1,ns1);
102         for i=1:ns1
103           R{i}.v=sol1(i);
104           if isreal(sol1(i)) && sol1(i)>=0
105             R{i}.VS=cell(1,ns2);
106             for j=1:ns2
107               R{i}.VS{j}.v=sol2(j);
108               R{i}.VS{j}.VS={};
109             end
110           else
111             R{i}.VS={};
112           end
113         end
114
```

```
115         end
116
117         [r1,c1]=find(S==v(1),1);
118         [r2,c2]=find(S==v(2),1);
119         V={[r1 c1] [r2 c2]};
120
121       otherwise
122         error('Can not solve systems with more than 2 equations');
123     end
124     fprintf('%u seconds\n',toc);
```

```
1  function DrawSol(M)
2  ns=length(M.Sol(:,1))/3;
3  k=1;
4  for i=1:ns
5  %     pp=C{i};
6  %     pp=pp-pp(:,3);
7      pp = M.Sol(k:k+2,:);
8      CreateFigure(pp);
9
10     % The legs
11     line([pp(1,3);pp(1,5)],[pp(2,3);pp(2,5)],[pp(3,3);pp(3,5)], ...
12         'LineWidth', 2, 'Color', [1 0 1]);
13     line([pp(1,3);pp(1,6)],[pp(2,3);pp(2,6)],[pp(3,3);pp(3,6)],  ...
14         'LineWidth', 2, 'Color', [1 0 1]);
15     line([pp(1,3);pp(1,7)],[pp(2,3);pp(2,7)],[pp(3,3);pp(3,7)],  ...
16         'LineWidth', 2, 'Color', [1 0 1]);
17     line([pp(1,1);pp(1,5)],[pp(2,1);pp(2,5)],[pp(3,1);pp(3,5)],  ...
18         'LineWidth', 2, 'Color', [1 0 1]);
19     line([pp(1,2);pp(1,7)],[pp(2,2);pp(2,7)],[pp(3,2);pp(3,7)],  ...
20         'LineWidth', 2, 'Color', [1 0 1]);
21     line([pp(1,4);pp(1,6)],[pp(2,4);pp(2,6)],[pp(3,4);pp(3,6)],  ...
22         'LineWidth', 2, 'Color', [1 0 1]);
23
24
25     % The plagform
26     h1=fill3(pp(1,[1 2 4])', pp(2,[1 2 4])', pp(3,[1 2 4])', ...
27         [0.9 0.3 0.3]); set(h1,'facealpha',1)
28     h1=fill3(pp(1,[1 3 4])', pp(2,[1 3 4])', pp(3,[1 3 4])', ...
29         [0.9 0.3 0.3]); set(h1,'facealpha',1)
30     h1=fill3(pp(1,[1 2 3])', pp(2,[1 2 3])', pp(3,[1 2 3])', ...
31         [0.9 0.3 0.3]); set(h1,'facealpha',1)
32     h1=fill3(pp(1,[2 3 4])', pp(2,[2 3 4])', pp(3,[2 3 4])',  ...
33         [0.9 0.3 0.3]); set(h1,'facealpha',1)
34
35     % The base
36     h2=fill3(pp(1,5:7)', pp(2,5:7)', pp(3,5:7)', [0.3 0.3 0.9]); ...
37         set(h2,'facealpha',1)
38
39
40     Sol=pp;
41     p0=[0, 0, -M.r_sphere];
42     p1=Sol(:,1);
43     p2=Sol(:,2);
44     p3=Sol(:,3);
45     p4=Sol(:,4);
46     p5=Sol(:,5);
47     p6=Sol(:,6);
```

```
48      p7=Sol(:,7);
49
50
51      % Leg1
52      [x,y,z] = sphere;
53      %     surf(x,y,z, 'EdgeColor', 'none', 'Facecolor', 'g','FaceAlpha', 0.1);
54      surf(x,y,z, 'EdgeColor', 'none','FaceAlpha', 0.25);
55      DrawSphericalArc(p0, p5', 'k');
56      DrawSphericalArc(p5', p1', 'k');
57
58      % Leg2
59      DrawSphericalArc(p0, p6', 'b');
60      DrawSphericalArc(p6', p4', 'b');
61
62
63      % Leg2
64      DrawSphericalArc(p0, p7', 'r');
65      DrawSphericalArc(p7', p2', 'r');
66      xlim([-1.1 1.1]);
67      xlabel('x');
68      ylim([-1.1 1.1]);
69      ylabel('y');
70      zlim([-1.1 1.1]);
71      zlabel('z');
72      view([-45,-60,30])
73      title(['3-RRR Coaxial SPM - SOL = ',num2str(i)]);
74      k=k+3;
75  end
76  end
```

```
1   % Creates a nes figure to draw a robot.
2   %
3   % Inputs
4   %            pp: The points on the robot. Represented by a circle and a
5   %                label.
6   %             f: [optionla] Handler of the figure. Only used when
7   %                redrawing.
8   %             H: [optional] Handler of the figure elements. If provided the
9   %                figure is just re-drawn. Otherwise a new figure is
10  %                created.
11  %
12  % Outputs:
13  %             f: Handler of the figure.
14  %             H: Handler of the figure elements.
15
16  function  [f,H]=CreateFigure(pp,varargin)
17
18      if isempty(varargin)
19        redraw=false;
20        H=[];
21      else
22        redraw=true;
23        f=varargin{1};
24        H=varargin{2};
25      end
26
27      if redraw
28        figure(f);
```

```
29      nh=length(H);
30      set(H(1),'Xdata',pp(1,:),'Ydata',pp(2,:),'Zdata',pp(3,:));
31      for i=2:nh
32        set(H(i),'Position',pp(:,i-1));
33      end
34    else
35      f=figure;
36      hold on;
37      %grid minor;
38      grid on;
39      axis equal;
40      set(gcf, 'Renderer', 'opengl');
41
42      % The points
43      H=scatter3(pp(1,:)',pp(2,:)', pp(3,:)', 'MarkerEdgeColor','k', ...
44          'MarkerFaceColor',[0 .75 .75]);
45
46      % Labels for the points
47      np=size(pp,2);
48      for i=1:np
49        h=text(pp(1,i), pp(2,i), pp(3,i),['  ' sprintf('P%u',i)], ...
50            'HorizontalAlignment','left','FontSize',9);
51        H=[H h];
52      end
53
54      % Point of view
55      view([1 1 1]);
56    end
57
58    % Axis dimensions
59    up=max(pp,[],2);
60    lo=min(pp,[],2);
61    sz=(up-lo)*0.1;
62    axis([lo(1)-sz(1) up(1)+sz(1) lo(2)-sz(2) up(2)+sz(2) lo(3)-sz(3) ...
63        up(3)+sz(3)]);
```

```
1  function DrawSphericalArc(pt1, pt2, color)
2  %UNTITLED2 Summary of this function goes here
3  %   Detailed explanation goes here
4
5
6  pt1 = pt1/norm(pt1);
7  pt2 = pt2/norm(pt2);
8
9  if abs(dot(pt1,pt2)) == 1
10     return;
11 end
12
13 pt3 = cross(pt1, pt2);
14 pt4 = cross(pt3, pt1);
15
16 R = [pt1; pt4/norm(pt4); pt3/norm(pt3)]';
17
18 ang = atan2(norm(pt3),dot(pt1,pt2));
19
20 samples =100;
21
22 oldpt = R(:,1);
```

```
23
24 for i = 0:1:samples
25     RT = R*rotz((i/samples)*ang);
26     newpt = RT(:,1);
27     plot3([oldpt(1) newpt(1)], [oldpt(2) newpt(2)], [oldpt(3) newpt(3)], ...
28         color,'LineWidth',3);
29     oldpt=newpt;
30 end
31
32
33 end
```

```
1  % Eliminates a square root from an expression
2  %
3  % Inputs:
4  %     P: The expression with the square root
5  %    DS: The sympol representing the square root in P.
6  %    DD: The expression in the square root (DS^2=DD)
7  %
8  % Outputs:
9  %     P: The expression without the square root.
10 %
11 function P=EliminateSquareRoot(P,DS,DD)
12   [co,te]=coeffs(P,DS);
13   nc=size(co,2);
14   if nc>1
15     O=0;
16     E=0;
17     for j=1:nc
18       d=feval(symengine,'degree',te(j));
19       %d=nc-j;
20       if mod(d,2)==0
21         % even
22         E=E+co(j)*DD^(d/2);
23       else
24         % odd
25         O=O+co(j)*DD^((d-1)/2);
26       end
27     end
28     P=DD*O^2-E^2;
29   end
```

```
1  % Determines the coordinates of the points given
2  %
3  %     S: The initial squared distance matrix.
4  %     T: The trilateration sequence.
5  %    SG: Sign of each tetrahedron, if any.
6  %     R: The roots. This is a recursive structure defined in SolveSystem.
7  %     V: The variables (fiven as row/columns of S). Also defined in
8  %        SolveSystem.
9  % error: Tolerance to accept a solution.
10 %    pt: [optional] If true we print timing information. If not provided it
11 %        is set to true.
12
13 % Outputs
14 %    C: Set of valid coordinates for the points (one valid set per row).
```

```
15 %   vs: Valid solutiosn. Solutions that lead to a valid coordenalization.
16 %
17 function [C,vs]=GetCoordinates(S,T,SG,R,V,error,varargin)
18
19   if isempty(varargin)
20     pt=true;
21   else
22     pt=varargin{1};
23   end
24
25   % Ensure that the input matrix is symmetric
26   S=Symmetrize(S);
27
28   % The set of valid coordinates (so far empty)
29   C={};
30   vs={};
31
32   if pt
33     fprintf('Generating coordinates   : ');
34     tic;
35   end
36
37   % Fix the variables and generate coordinates
38   [C,vs]=FixVariable(S,T,SG,R,V,1,vs,C,[],error);
39
40   if pt
41     fprintf('%u seconds\n',toc);
42   end
43
44 end
45
46 % Recursive routine to fix the values of the different variables and then
47 % trilaterate.
48 %
49 % Input
50 %      S: The input matrix with know squared distances.
51 %      T: The trilateraion sequence.
52 %     SG: The sign of each trilatertion step, if any.
53 %      R: The root structure. Recursive structure with values for the
54 %         different variables in the problem.
55 %      V: The variables in the problem (given as row/columns of S).
56 %     cv: The variable being assigned at the current recursion level.
57 %      C: The current set of solutions.
58 %     vs: Valid solutions. Solutions that lead to a valid coordenalization.
59 %     cs: Current assigment of variables. Initially empty.
60 %  error: The tolerance.
61 %
62 % Outputs:
63 %      C: The new set of solutions.
64 %     vs: Valid solutiosn. Solutions that lead to a valid coordenalization.
65 %
66 function [C,vs]=FixVariable(S,T,SG,R,V,cv,C,vs,ca,error)
67
68   nv=size(V,2);
69   if cv<=nv
70     % We still have variables to be assigned
71
72     r=V{cv}(1);
73     c=V{cv}(2);
```

```
74      n=size(R,2);
75      for i=1:n
76        v=R{i}.v;
77        if (~isempty(R{i}.VS) || cv==nv) && v>=0 && isreal(v)
78          S(r,c)=v;
79          S(c,r)=v;
80          [C,vs]=FixVariable(S,T,SG,R{i}.VS,V,cv+1,C,vs,[ca v],error);
81        end
82      end
83
84    else
85      % Once all variables are assigned, trilaterate
86
87      % number of points to trilatrate
88      np=size(T,2)+3;
89
90      % coordinates of the points
91      pp=zeros(3,np);
92
93      % The first three points are always the same
94      i=T{1}(1);
95      j=T{1}(2);
96      k=T{1}(3);
97
98      pp(:,i) = [0;0;0];
99      pp(:,j) = [sqrt(S(i,j));0;0];
100     phi = acos((S(i,j)+S(i,k)-S(j,k))/(2*sqrt(S(i,j)*S(i,k))));
101     pp(:,k) = double([sqrt(S(i,k))*cos(phi);sqrt(S(i,k))*sin(phi);0]);
102
103     % fixed poitns so far
104     fixed=false(1,np);
105     fixed([i j k])=true;
106
107     [C,vs]=RTrilaterate(S,fixed,pp,1,T,SG,C,vs,ca,error);
108   end
109
110 end
111
112 % Recursive trilateration. Used when all variables are fixed.
113 %
114 % Input
115 %      S: The input matrix with know squared distances.
116 %  fixed: Points already fixed.
117 %     pp: Coordinates determined so far (for the fixed points).
118 %  level: Current trilateration level
119 %      T: The trilateraion sequence
120 %     SG: The sign of each trilatertion step, if any.
121 %      C: The current set of solutions.
122 %     vs: Valid solutions so far.
123 %     ca: Current assigment of variables.
124 %  error: The tolerance.
125 %
126 % Outputs:
127 %      C: The new set of coordinates.
128 %     vs: The new set of solutions.
129 %
130 function [C,vs]=RTrilaterate(S,fixed,pp,level,T,SG,C,vs,ca,error)
131
132   np=size(S,1);
```

```
133   i=T{level}(1);
134   j=T{level}(2);
135   k=T{level}(3);
136   l=T{level}(4);
137
138   if SG(level)==0
139     ndx=[-1 1];
140   else
141     ndx=SG(level);
142   end
143
144   for sg=ndx
145     coord=Trilaterate(pp(:,i),pp(:,j),pp(:,k),S(l,i),S(l,j),S(l,k),sg);
146
147     %ok=(norm(imag(coord))<error);
148     ok=(norm(imag(coord))<1e-6);
149     coord=real(coord);
150     u=1;
151     while ok && u<=np
152       if fixed(u) && (u~=i) && (u~=j) && (u~=k) && (u~=l) && ...
153               isempty(symvar(S(l,u))) && (S(l,u)>0)
154         ok=(abs(double(norm(coord-pp(:,u))-sqrt(S(l,u))))<error);
155       end
156       u=u+1;
157     end
158     if ok
159       fixed(l)=true;
160       pp(:,l)=real(coord);
161       if level==size(T,2)
162         C=[C pp];
163         vs=[vs ca];
164       else
165         [C,vs]=RTrilaterate(S,fixed,pp,level+1,T,SG,C,vs,ca,error);
166       end
167     %else
168     %  fprintf('Error (level->%u   d->%u   e->%f)\n', ...
169     % level,u-1,abs(double(norm(coord-pp(:,u-1))-sqrt(S(l,u-1)))));
170     end
171   end
172 end
```

```
1  % Removes a factor from an expression
2  %
3  % Inputs
4  %      P: The expression to simplify
5  %      A: The factor to remove
6  %
7  % Outputs:
8  %      P: The simplified expression without the factor.
9  %
10
11 function P=RemoveFactor(P,A)
12
13   %A2=A^2; % Assuming that they always appear in pairs
14   P=expand(P);
15   ok=~isempty(symvar(A)) && ~isempty(symvar(P));
16   while ok
17     Q=feval(symengine,'divide',P,A);
```

```
    if Q(2)==0
      P=Q(1);
    else
      ok=false;
    end
  end
```

```
% Determines the position of a point given the distance to 3 fixed points.
% This is a trilateratin in Cartesian space.
% Note that two solutions are possible.
%
% Inputst
%     p1: The coordinates of the first reference point.
%     p2: The coordinates of the second reference point.
%     p3: The coordinates of the third reference point.
%     d1: Squared distance from p1 to the point to locate.
%     d2: Squared distance from p2 to the point to locate.
%     d3: Squared distance from p3 to the point to locate.
%     sg: Sign of the trilateration (+1/-1)
%
% Outputs:
%      p: The coordinates of the trilaterated point.
%
function p=Trilaterate(p1,p2,p3,d1,d2,d3,sg)
% http://en.wikipedia.org/wiki/Trilateration

  r1=double(d1);
  r2=double(d2);
  r3=double(d3);

  vx=p2-p1;
  d=norm(vx);
  vx=vx/d;

  v=p3-p1;
  i=(vx'*v);
  vy=v-i*vx;
  vy=vy/norm(vy);
  j=vy'*v;

  vz=cross(vx,vy);

  x=(r1-r2+d^2)/(2*d);
  y=(r1-r3+i^2+j^2)/(2*j)-i*x/j;
  d=r1-x^2-y^2;

  z=sqrt(d); % This may be imaginary

  R=[vx vy vz];

  p=R*[x;y;sg*z]+p1;
```

```
% Determines the distance l-m in function of the distance between
% {i,j,k,l,m}.
%
% Inputs:
```

```
%          S: The matrix of squared distances. Some entries are constants
%             and others are symbolic expressions.
%          i: Index of the first point.
%          j: Index of the second point.
%          k: Index of the third point.
%          l: Index of the fourth point.
%          m: Index of the fifth point.
%         DS: Symbol to represent the square root that may appear in
%             the s_lm expression.
%
% Output:
%          S: The updated matrix, with the computed s_lm entry.
%         DD: The expression inside the square root (may be 0).
%          A: Area of the {i,j,k} triangle.
%
function [S,DD,A]=FillDistance(S,i,j,k,l,m,DS)

  T=[ 0 1      1       1
      1 0      S(i,j)  S(i,k)
      1 S(i,j) 0       S(j,k)
      1 S(i,k) S(j,k)  0 ];
  A=det(T);

  x=sym('x');
  B=[ 0 1       1      1      1      1;
        1 0      S(i,j)  S(i,k) S(i,l) S(i,m)
        1 S(i,j) 0       S(j,k) S(j,l) S(j,m)
      1 S(i,k) S(j,k)  0      S(k,l) S(k,m)
      1 S(i,l) S(j,l)  S(k,l) 0      x
      1 S(i,m) S(j,m)  S(k,m) x      0];
  sol = solve(det(B));

  [N1,D1]=numden(sol(1));
  [N2,~]=numden(sol(2));

  DD=((N1-N2)/2)^2; % The discriminant

  if DD==0
      S(l,m)=sol(1); % Both solutions are equal
  else
      S(l,m)=((N1+N2)/2+ DS)/D1; % Substitute the square root by a symbol
  end
  S(m,l)=S(l,m);
```

```
write1ByteTxRx(dxl.port_num, dxl.PROTOCOL_VERSION, ...
    dxl.DXL_ID, dxl.ADDR_PRO_TORQUE_ENABLE, dxl.TORQUE_DISABLE);
dxl.comm_result = getLastTxRxResult(dxl.port_num, dxl.PROTOCOL_VERSION);
dxl.error = getLastRxPacketError(dxl.port_num, dxl.PROTOCOL_VERSION);
if dxl.comm_result ~= dxl.COMM_SUCCESS
    fprintf('%s\n', getTxRxResult(dxl.PROTOCOL_VERSION, dxl.comm_result));
elseif dxl.error ~= 0
    fprintf('%s\n', getRxPacketError(dxl.PROTOCOL_VERSION, dxl.error));
else
    fprintf('Dynamixel TORQUE_DISABLE has been successfully disconnected \n');
end
```

```
write1ByteTxRx(dxl.port_num, dxl.PROTOCOL_VERSION,...
    dxl.DXL_ID, dxl.ADDR_PRO_TORQUE_ENABLE, dxl.TORQUE_ENABLE);
dxl.comm_result = getLastTxRxResult(dxl.port_num, dxl.PROTOCOL_VERSION);
dxl.error = getLastRxPacketError(dxl.port_num, dxl.PROTOCOL_VERSION);
if dxl.comm_result ~= dxl.COMM_SUCCESS
    fprintf('%s\n', getTxRxResult(dxl.PROTOCOL_VERSION, dxl.comm_result));
elseif dxl.error ~= 0
    fprintf('%s\n', getRxPacketError(dxl.PROTOCOL_VERSION, dxl.error));
else
    fprintf('Dynamixel TORQUE_ENABLE has been successfully connected \n');
end
```

## 10.3 Appendix C

### Workspace and Singularity Analysis

```
%% implicit plot
clear
close all
clc

w = struct();

figure

subplot(231)
w.n1 = 0;
w.a1 = pi/3;
w.a2 = pi/2;
w.g = pi/3;
w.int = [-1.1 1.1 -1.1 1.1 -1.1 1.1];
[w] = plot_workspace(w);
title('LEG 1: \gamma = \pi/3 \alpha_1 = \pi/3 \alpha_2 = \pi/2')
hold off

subplot(232)
w.n1 = 0;
w.a1 = pi/3;
w.a2 = pi/2;
w.g = -pi/3;
w.int = [-1.1 1.1 -1.1 1.1 -1.1 1.1];
[w] = plot_workspace(w);
title('LEG 2: \gamma = \pi/3 \alpha_1 = \pi/3 \alpha_2 = \pi/2')
hold off

subplot(233)
w.n1 = 2*pi/3;
w.a1 = pi/3;
w.a2 = pi/2;
w.g = 0;
w.int = [-1.1 1.1 -1.1 1.1 -1.1 1.1];
[w] = plot_workspace(w);
title('LEG 3: \gamma = \pi/3 \alpha_1 = \pi/3 \alpha_2 = \pi/2')
hold off

subplot(234)
```

```
w.n1 = 0;
w.a1 = pi/2;
w.a2 = pi/2;
w.g = pi/3;
w.int = [-1.1 1.1 -1.1 1.1 -1.1 1.1];
[w] = plot_workspace(w);
title('LEG 1: \gamma = 0 \alpha_1 = \pi/2 \alpha_2 = \pi/2')
hold off
subplot(235)
w.n1 = 0;
w.a1 = pi/2;
w.a2 = pi/2;
w.g = -pi/3;
w.int = [-1.1 1.1 -1.1 1.1 -1.1 1.1];
[w] = plot_workspace(w);
title('LEG 2: \gamma = 0 \alpha_1 = \pi/2 \alpha_2 = \pi/2')
hold off

subplot(236)
w.n1 = 2*pi/3;
w.a1 = pi/2;
w.a2 = pi/2;
w.g = 0;
w.int = [-1.1 1.1 -1.1 1.1 -1.1 1.1];
[w] = plot_workspace(w);
title('LEG 3: \gamma = 0 \alpha_1 = \pi/2 \alpha_2 = \pi/2')
hold off


%% Plot the Whole Mobility Range Generic Configuration
clear
figure(2);
trplot(eye(3), 'color', 'r', 'arrow', 'text_opts', ...
    {'FontSize', 12, 'FontWeight', 'bold'});
hold on;

N = 1000;
radius = 0.5;

[x,y,z] = sphere(N);

w.n1 = 0;
w.a1 = pi/3;
w.a2 = pi/2;
w.g = pi/3;
D1 = -cos(w.a1+w.a2);
D2 = -cos(w.a1-w.a2);

for i=1:N+1
    for j=1:N+1

        if x(i,j)*sin(w.n1)*sin(w.g) - y(i,j)*cos(w.n1)*sin(w.g)...
                + z(i,j)*cos(w.g) + D1 <0
            x(i,j) = NaN;
            y(i,j) = NaN;
            z(i,j) = NaN;
        end
        if x(i,j)*sin(w.n1)*sin(w.g) - y(i,j)*cos(w.n1)*sin(w.g)...
                + z(i,j)*cos(w.g) + D2 >0
```

```
100             x(i,j) = NaN;
101             y(i,j) = NaN;
102             z(i,j) = NaN;
103         end
104     end
105 end
106
107 w.n1 = +pi/3;
108 w.a1 = pi/3;
109 w.a2 = pi/2;
110 w.g = -pi/3;
111
112 for i=1:N+1
113     for j=1:N+1
114
115         if x(i,j)*sin(w.n1)*sin(w.g) - y(i,j)*cos(w.n1)*sin(w.g)...
116                 + z(i,j)*cos(w.g) + D1 <0
117             x(i,j) = NaN;
118             y(i,j) = NaN;
119             z(i,j) = NaN;
120         end
121         if x(i,j)*sin(w.n1)*sin(w.g) - y(i,j)*cos(w.n1)*sin(w.g)...
122                 + z(i,j)*cos(w.g) + D2 >0
123             x(i,j) = NaN;
124             y(i,j) = NaN;
125             z(i,j) = NaN;
126         end
127     end
128 end
129 w.n1 = 2*pi/3;
130 w.a1 = pi/3;
131 w.a2 = pi/2;
132 w.g = 0;
133
134 for i=1:N+1
135     for j=1:N+1
136
137         if x(i,j)*sin(w.n1)*sin(w.g) - y(i,j)*cos(w.n1)*sin(w.g)...
138                 + z(i,j)*cos(w.g) + D1 <0
139             x(i,j) = NaN;
140             y(i,j) = NaN;
141             z(i,j) = NaN;
142         end
143         if x(i,j)*sin(w.n1)*sin(w.g) - y(i,j)*cos(w.n1)*sin(w.g)...
144                 + z(i,j)*cos(w.g) + D2 >0
145             x(i,j) = NaN;
146             y(i,j) = NaN;
147             z(i,j) = NaN;
148         end
149     end
150 end
151 surf(x,y,z, 'EdgeColor', 'none',...
152     'Facecolor', 'interp','FaceAlpha', 0.6);
153 axis square
154 title('Whole Mobility Range \alpha_1 = \alpha_2 \neq \pi/2');
155
156 %% Plot the Whole Mobility Range Coaxial Configuration
157 clear
158 figure(3);
```

```
159 trplot(eye(3), 'color', 'r', 'arrow', 'text_opts', ...
160     {'FontSize', 12, 'FontWeight', 'bold'});
161 hold on;
162
163 N = 1000;
164 radius = 0.5;
165
166 [x,y,z] = sphere(N);
167
168 w.n1 = 0;
169 w.a1 = deg2rad(87);
170 w.a2 = pi/2;
171 w.g = pi/3;
172 D1 = -cos(w.a1+w.a2);
173 D2 = -cos(w.a1-w.a2);
174
175 for i=1:N+1
176     for j=1:N+1
177
178         if x(i,j)*sin(w.n1)*sin(w.g) - y(i,j)*cos(w.n1)*sin(w.g)...
179                 + z(i,j)*cos(w.g) + D1 <0
180             x(i,j) = NaN;
181             y(i,j) = NaN;
182             z(i,j) = NaN;
183         end
184         if x(i,j)*sin(w.n1)*sin(w.g) - y(i,j)*cos(w.n1)*sin(w.g)...
185                 + z(i,j)*cos(w.g) + D2 >0
186             x(i,j) = NaN;
187             y(i,j) = NaN;
188             z(i,j) = NaN;
189         end
190     end
191 end
192
193 w.n1 = +pi/3;
194 w.a1 = deg2rad(87);
195 w.a2 = pi/2;
196 w.g = -pi/3;
197
198 for i=1:N+1
199     for j=1:N+1
200
201         if x(i,j)*sin(w.n1)*sin(w.g) - y(i,j)*cos(w.n1)*sin(w.g)...
202                 + z(i,j)*cos(w.g) + D1 <0
203             x(i,j) = NaN;
204             y(i,j) = NaN;
205             z(i,j) = NaN;
206         end
207         if x(i,j)*sin(w.n1)*sin(w.g) - y(i,j)*cos(w.n1)*sin(w.g)...
208                 + z(i,j)*cos(w.g) + D2 >0
209             x(i,j) = NaN;
210             y(i,j) = NaN;
211             z(i,j) = NaN;
212         end
213     end
214 end
215 w.n1 = 2*pi/3;
216 w.a1 = deg2rad(87);
217 w.a2 = pi/2;
```

```
w.g = 0;

for i=1:N+1
    for j=1:N+1

        if x(i,j)*sin(w.n1)*sin(w.g) - y(i,j)*cos(w.n1)*sin(w.g)...
                + z(i,j)*cos(w.g) + D1 <0
            x(i,j) = NaN;
            y(i,j) = NaN;
            z(i,j) = NaN;
        end
        if x(i,j)*sin(w.n1)*sin(w.g) - y(i,j)*cos(w.n1)*sin(w.g)...
                + z(i,j)*cos(w.g) + D2 >0
            x(i,j) = NaN;
            y(i,j) = NaN;
            z(i,j) = NaN;
        end
    end
end
surf(x,y,z, 'EdgeColor', 'none',...
    'Facecolor', 'interp','FaceAlpha', 0.6);
axis square
title('Whole Mobility Range \alpha_1 = \alpha_2 = \pi/2');
```

```
function [w] = plot_workspace(w)
cone = @(x,y,z) (x.^2 +(y*cos(w.g) + z*sin(w.g)).^2)*(sin(w.a1))^2 ...
- ((y*sin(w.g) - z*cos(w.g))*cos(w.a1)-cos(w.a2)).^2;
% Planes
D1 = -cos(w.a1+w.a2);
D2 = -cos(w.a1-w.a2);
plane1 = @(x1,y1,z1) x1*sin(w.n1)*sin(w.g)...
    - y1*cos(w.n1)*sin(w.g) + z1*cos(w.g) + D1 ;
plane2 = @(x2,y2,z2) x2*sin(w.n1)*sin(w.g)...
    - y2*cos(w.n1)*sin(w.g) + z2*cos(w.g) + D2 ;
% Sphere
[X,Y,Z] = sphere;
c = fimplicit3(cone, w.int, 'EdgeColor', 'none',...
    'Facecolor', 'r','FaceAlpha', 0.5);
hold on
surf(X,Y,Z,'EdgeColor', 'none','Facecolor',...
    'g','FaceAlpha', 0.5);
hold on
fun1 = fimplicit3(plane1, w.int, 'Facecolor',...
    'b','FaceAlpha', 0.5);
hold on
fun2 = fimplicit3(plane2, w.int, 'Facecolor',...
    'b','FaceAlpha', 0.5);
hold on
axis square
xlim([-1.2 1.2])
xlabel('x')
ylim([-1.2 1.2])
ylabel('y')
zlim([-1.2 1.2])
zlabel('z')
end
```

# Acknowledgements

I would first like to thank my thesis advisor *Prof. F. Thomas* of the I.R.I. Institute. The door to Prof. Thomas office was always open whenever I had a little trouble or had a question about my project or writing.

I could not forget *Prof. P. Grosch* for his availability, kindness and patience for backing me up on the mechanical and practical work. He was my mentor and observing him up close and listening to his advices were absolutely a career path.

Moreover, I would also like to mention the input of the *PhD student A. Shabani*, because talking to him was always an incentive to improve my work.

I would also like to acknowledge *Prof. S. Pastorelli*, my home supervisor at Politecnico di Torino, for his willingness when I had a little trouble or doubt, during the planning and the development of my project.

Finally, I must express my very profound gratitude to my family and to my friends for providing me with unfailing support and continuous encouragement throughout my years of study and the last step of this landmark achievements. Thank you.

Francesco

# Bibliography


[1] Shaoping Bai. Optimum design of spherical parallel manipulators for a prescribed workspace. *Mechanism and Machine Theory*, 45(2):200–211, 2010.

[2] Shaoping Bai, Michael R Hansen, and Torben O Andersen. Modelling of a special class of spherical parallel manipulators with euler parameters. *Robotica*, 27(2):161–170, 2009.

[3] Shaoping Bai, Michael R Hansen, and Torben O Andersen. Modelling of a special class of spherical parallel manipulators with euler parameters. *Robotica*, 27(2):161–170, 2009.

[4] Shaoping Bai, Michael R Hansen, and Jorge Angeles. A robust forward-displacement analysis of spherical parallel robots. *Mechanism and Machine Theory*, 44(12):2204–2216, 2009.

[5] Ilian A Bonev, Damien Chablat, and Philippe Wenger. Working and assembly modes of the agile eye. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2317–2322. IEEE, 2006.

[6] E Cavallo and R Michelini. A robotic equipment for the guidance of a vectored thrustor auv. 1:1–6, 2004.

[7] Clement Gosselin and Jorge Angeles. Singularity analysis of closed-loop kinematic chains. *IEEE transactions on robotics and automation*, 6(3):281–290, 1990.

[8] Clément M Gosselin and J-F Hamel. The agile eye: a high-performance three-degree-of-freedom camera-orienting device. pages 781–786, 1994.

[9] Clément M Gosselin and Eric Lavoie. On the kinematic design of spherical three-degree-of-freedom parallel manipulators. *The International Journal of Robotics Research*, 12(4):394–402, 1993.

[10] CM Gosselin and Jorge Angeles. The optimum kinematic design of a spherical three-degree-of-freedom parallel manipulator. *Journal of mechanisms, transmissions, and automation in design*, 111(2):202–207, 1989.

[11] CM Gosselin and Jorge Angeles. The optimum kinematic design of a spherical three-degree-of-freedom parallel manipulator. *Journal of mechanisms, transmissions, and automation in design*, 111(2):202–207, 1989.

[12] CM Gosselin, Jaouad Sefrioui, and Marc J Richard. On the direct kinematics of spherical three-degree-of-freedom parallel manipulators of general architecture. *Journal of Mechanical Design*, 116(2):594–598, 1994.

[13] Timothy F Havel. Some examples of the use of distances as coordinates for euclidean geometry. *Journal of Symbolic Computation*, 11(5-6):579–593, 1991.

[14] Yu Lei Hou, Xin Zhe Hu, and Da Xing Zeng. Kinematics analysis of a 3-rrr spherical parallel mechanism with coaxial input shafts. In *Applied Mechanics and Materials*, volume 404, pages 237–243. Trans Tech Publ, 2013.

[15] Yu Lei Hou, Xin Zhe Hu, and Da Xing Zeng. Kinematics analysis of a 3-rrr spherical parallel mechanism with coaxial input shafts. In *Applied Mechanics and Materials*, volume 404, pages 237–243. Trans Tech Publ, 2013.

[16] https://it.rs online.com/web/. *RS Component*.

[17] https://www.isquared.eu.com/en/uprint serie/. *uPrint SE 3D Serie*.

[18] https://www.skf.com/it/products/index.html. *SKF*.

[19] https://www.stratasys.com/es/materials/search/absplus. *ABSplusP430 - FDM 3D Printing Process*.

[20] http://www.3dprinterscanada.com/fdm-design-series-dimension-1200es-3d printer.php. *Dimension 1200es 3D Printer*.

[21] http://www.robotis.us/dynamixel/. *Robotis Dynamixel*.

[22] KH Hunt. Structural kinematics of in-parallel-actuated robot-arms. *Journal of Mechanisms, Transmissions, and Automation in Design*, 105(4):705–712, 1983.

[23] Julius Klein, Steve Spencer, James Allington, James E Bobrow, and David J Reinkensmeyer. Optimization of a parallel shoulder mechanism to achieve a high-force, low-mass, robotic-arm exoskeleton. *IEEE Transactions on Robotics*, 26(4):710–715, 2010.

[24] Temei Li and Shahram Payandeh. Design of spherical parallel mechanisms for application to laparoscopic surgery. *Robotica*, 20(2):133–138, 2002.

[25] Jean-Pierre Merlet. *Parallel robots*, volume 128. Springer Science & Business Media, 2006.

[26] Jean-Pierre Merlet. *Parallel robots*, volume 128. Springer Science & Business Media, 2006.

[27] Josep M Porta, Federico Thomas, Lluís Ros, and Carme Torras. A branch-and-prune algorithm for solving systems of distance constraints. Institute of Electrical and Electronics Engineers, 2003.

[28] Nicolás Rojas and Federico Thomas. The closure condition of the double banana and its application to robot position analysis. 2013.

[29] Aleix Rull and Federico Thomas. On generalized euler angles. In *New Trends in Mechanism and Machine Science*, pages 61–68. Springer, 2015.

[30] Federico Thomas and Josep M Porta. Closure polynomials for strips of tetrahedra. In *Advances in Robot Kinematics 2016*, pages 303–312. Springer, 2018.

[31] Federico Thomas and Lluís Ros. Revisiting trilateration for robot localization. *IEEE Transactions on robotics*, 21(1):93–101, 2005.

[32] Guanglei Wu. Multiobjective optimum design of a 3-rrr spherical parallel manipulator with kinematic and dynamic dexterities. 2012.