POLITECNICO DI TORINO
Department of Electronics, Information and Bioengineering
Master of Science in Biomedical Engineering

Cartesian Control of a 3DoF Upper Limb Prosthetic Device

Supervisor: Prof. Carlo FERRARESI
Co-supervisor: Ing. Nicolò BOCCARDO
Co-supervisor: Ing. Federico TESSARI

Candidate:
Pietro DI BELLO
Identification number: 267445

Academic Year 2020-2021

Ai miei genitori, che ringrazio e mai mi stancherò di ringraziare per tutto ciò che hanno sempre fatto e sono per me.
A voi dedico il compimento di questo percorso. Siete i primi fautori dell'uomo che son diventato e mi appresto a diventare, avete costruito per me una vita che migliore non potevo desiderare. Cercherò sempre di rendervi fieri di me, vi voglio bene.

Pietro

Acknowledgements

Ringrazio il professor Carlo Ferraresi per la disponibilità nel voler essere mio relatore, per la pacatezza e per la fiducia e libertà datemi nel lavoro.

Ringrazio Nicolò Boccardo per la possibilità datami di lavorare nel meraviglioso mondo del dipartimento Rehab dell'IIT di Genova, per i consigli, l'aver creduto in me e per avermi mostrato cosa vuol dire essere leader di un gruppo con carisma, pragmaticità e la giuasta dose di umorismo.

Ringrazio Federico Tessari per l'aiuto, l'infinita pazienza e disponibilità dimostratimi e per la ferma fiducia che ha sempre avuto in me e nelle mie capacità, più di quanta ne avessi io effettivamente.

Ringrazio Dario Di Domenico, amico fraterno e in questa esperienza anche collega, per esser sempre stato di supporto, all'inizio e durante tutti i mesi di tesi.

Ringrazio Riccardo per l'amicizia, il clima ridente e l'entusiasmo che ha sempre portato e trasmesso.

Ringrazio Indya, animo davvero simile al mio, per l'affetto, il supporto e per la fantastica collega che sarà nell'imminente futuro.

Ringrazio Laura per avermi ricordato l'importanza della "leggerezza" e del perseguire la serenità.

Un ringraziamento va anche a tutte le splendide persone che questa esperienza mi ha dato la possibilità di conoscere e che mi hanno accompagnato nei mesi di tesi, colleghi ma prima di tutto amici.

Infine, il più affettuoso grazie va ad Elena per aver sempre creduto in me, aver gioito dei miei successi ed esser stata prima sostenitrice nei momenti più difficili, esser stata forte quando io non lo sono stato e, nonostante tutto, avermi riservato i migliori pensieri, sentimenti ed intenti. La ringrazio, nella speranza ed obiettivo di riuscire a ricambiare tutto l'amore, tutto il supporto, tutto il bene... e che sia sempre fiera di me.

Contents

List of Figures

List of Tables

Abbreviations

ADLs: Activities of Daily Living
ANFIS: Adaptive Neuro-Fuzzy Inference System
ANN: Artificial Neural Network
BMS: Battery Management System
DAG: Dynamic Adaptive Grasp
DoF: Degree/Degrees of Freedom
EE: Elbow Extension
EF: Elbow Flexion
EFE: Elbow FlexionExtension
EMG: Electromyography
FIS: Fuzzy Inference System
IIT: Istituto Italiano di Tecnologia
INAIL: Istituto Nazionale per l'Assicurazione contro gli Infortuni sul Lavoro
OT: Distance between Origin and Target
PID: Proportional Integral Derivative
PR: Precision
RMS: Root Mean Square
RMSE: Root Mean Square Error
ROM: Range Of Motion SCMM: Scheda Controllo Motore Mano
SCMG: Scheda Controllo Motore Gomito
SCMS: Scheda Controllo Motore Spalla
SE: Shoulder Extension
SF: Shoulder Flexion
SFE: Shoulder FlexionExtension
sEMG: surface Electromyography
TL: Total Length
TR: Training Set
VR: Virtual Reality
WE: Wrist Extension
WF: Wrist Flexion
WFE: Wrist FlexionExtension
WP: Wrist Pronation
WS: Wrist Supination

Abstract

Background

Currently, most upper limb prostheses are moved by direct control of every different joints forming the polyarticulate arm [29]. This strategy has the inherent limitation that it is not intuitive to control one joint at a time. The problem is more apparent when more joints need to be controlled sequentially. The state of the art offers numerous prosthetic solutions for the upper limb which differ in their constituent elements and operating principles [37]. Throughout history, advances in prosthetics have always sought to follow the needs of amputees, which appear to be more or less the same from the outset and to which we try to respond as best we can, supported by scientific and technical progress [41]. There is a long way to go, as indicated by the extensive literature on both types of prosthesis and control strategies, specifically for myoelectric prostheses. An element to be considered immediately is the declination of robotics towards prosthetics. It therefore appears that the basic notions in the design of prostheses and control systems are to be found in the basic principles of this field. A prosthesis is nothing more than a serial manipulator, actuated by an energy source that can be electrical or mechanical and must be controlled to perform a given task [13]. Known notions of kinematic theory, quaternion algebra, have been helpful in pursuing the objective.

Materials and methods

The thesis is carried out at the Rehab Technologies Laboratory of Istituto Italiano di Tecnologia (IIT) in Genova. The main objective of this thesis was to develop an inverse kinematic control to move the Hannes arm in a real and virtual environment. The control system is able to convert the movements of the end-effector, synthesized via 3D joystick or keyboard, to joint angles of the upper limb robotic arm, measured by Encoders or IMU sensors, remotely. The 3D-position of the end-effector (Hannes hand) must be converted into reference signals for the joint angles of the robotic system. The validation of the control method would be performed comparing the estimated joint angles via inverse kinematics with the joint angles measured by the IMUs and Encoders. The final result will be the control of the entire Hannes arm on a Cartesian space using the 3D joystick to synthesize the reference of end-effector and convert them in prosthetic movements.

Three different approaches have been tried to achieve the goal. The three different algorithms find the roots of their operating principles in different fields. From the use of

high-level applications such as the Unity virtual reality environment, through Artificial Neural Networks, to a lower-level and more analytical direction with inverse kinematics calculations and a recursive approach. The different approaches are evaluated individually and compared to see if and which one will lead to the goal. The first, geometric, very intuitive method was abandoned before moving on to peer testing because of the intrinsic limits it had. Extensive evaluation and study was carried out on the ANFIS model in order to obtain the functions with the best properties and then to compare them with the recursive approach algorithm, of which there is a double version, the standard and the optimised one. The attempt to optimise the recursive approach arose from the intrinsic limitations of this method, which were known in advance, but did not prevent it from being used successfully. This is true to such an extent that the recursive algorithm was used to control the Hannes prosthesis for the execution of two trajectories chosen as the most useful and elementary in the hypothetical future application of the method for patient use. The setup used was a prototype comprising 3 joints: shoulder, elbow and wrist, all in flexion-extension, therefore with the shoulder joint in addition, compared to the wider current configuration of the Hannes arm prosthetic device [24].

Results

The results concern a qualitative assessment of all three approaches, singularly observed and compared. One of the three was discarded before moving on to the experimental part, effectively limiting the testing phase to two of them. Both the ANFIS method and the recursive approach were shown to give a positive answer to the problem with different degrees of precision but acceptable accuracy overall. Both maintain some criticalities. The recursive approach in its optimised form also gave a positive response in the practical attempt to control Hannes arm prepared in a prototype version.

Conclusions

As the results show, both of the most widely studied approaches have potential and are applicable for achieving the thesis goal. The initial demand was not fully met, it was not possible to integrate IMUs into the process and the robustness of the algorithm chosen as the best is still to be improved. Even if the actual Hannes arm does not have a definitive 3 DoF configuration, the obtained results - in terms of cartesian control of the system - showed satisfactory performances. A hypothesis for alternative control strategies has been marked out, the value of which will be investigated in the future.

Keywords

Inverse Kinematics, Cartesian Control, Upper Limb Prosthesis, Artificial Neural Networks, Recursive Approach, Virtual Reality, Hannes Hand.

Introduction

A pivotal element of human evolution is dexterity: a capacity able to produce a direct connection between the brain and the external word allowing an interaction with them and manipulate everything for a practical doing. Hands are an afferent and efferent interface between the human being and the world around him. The loss of an upper limb or part of it produces a consequent impairment that affect the dexterity as long as the physiological disorder ought to the acceptance of the new body state decreasing, consequently, the quality of daily life.

Therefore, from the beginning, man has tried to compensate the loss, using anesthetic prosthesis and, with the progress of technologies, they try to improve the effectiveness and efficiency to perform dexterity tasks in the activities of daily living. Along with the technology evolution and electronic improvements, prostheses became complex mechatronic system need to be controlled by means of human intentions [13]. Despite the most advanced upper limb prostheses integrate several active degree of freedom controlled by electrical signals of voluntary contraction, they still do not achieve the complexity, dexterity, and adaptability of the human arm, making them difficult to be integrated into the body scheme to promote the so called embodiment. Moreover, the insufficient efficacy of prosthetic devices, the low quality of their integration into the body scheme, the lack of ownership and, consequently, the absence of embodiment of this systems are the main reason for high prosthesis abandonment rate. [24] A fundamental problem in the realization of a prosthesis is to create an object able to satisfy the requirements of users, in particular it must perform complex tasks without a huge mental effort. Balancing these three characteristics in an optimal way is the goal.

The amount of amputations usually has a prevalent percentage at the level of transradial or transhumeral, while the disarticulation of the shoulder is not so frequent. The main problem of shoulder prostheses is their incumbrance, weight and size. In fact, they need to replicate numerous movements lost but the mechanical and the control is complicated and produce a huge problem for the user acceptance. Shoulder disarticulation prosthesis are difficult to control. The final device consists in a wrist in prono supination, a wrist in flexion extension, an elbow in flexion extension and the shoulder that need to be controlled by the end user. Currently, body-power or myoelectric signals can only control one function at a time in almost all cases [22].

The master thesis is conducted at the Rehab-Technologies Laboratory of Istituto Italiano di Tecnologia (IIT) of Genova.

The thesis is organized as follows:

1. The part 1 summarizes the state of the art of upper arm prosthesis and it outlines the most widespread control strategies. There is also an additional chapter related

to the kinematic theory and Quaternion theory;

2. The part 2 defines the materials and methods used for the experimental study. A broad description of the Hannes system is included along with a discussion of the three developed and studied algorithms of this study;

3. The part 3 reports the results of the evaluation test on the algorithms with an eye on the single one and then comparing them. Also results of the trial on the physical setup is reported;

4. The part 4 discusses the study results and the techniques implemented to have reach the goal comparing the expectations to the effective incomes;

5. The part 5 outlines the conclusions on the research work of this thesis and it introduces possible future developments.

Chapter 1

Background

1.1 Brief history

Over the centuries, prostheses have undergone significant improvements, a long and complex history from their primitive origins until today's sophisticated ones, to exciting visions of the future. As in any research field, some ideas and inventions have been expanded while others have fallen by the wayside or become obsolete. All of this has been moved by the attempt of man to fulfill an aesthetic need or to recover functionality after a mutilation.

(a) *Igneous big toe prosthetic, Egypt 1000-600 B.C.*

(b) *PROPRIO FOOT by Ossur, adaptive microprocessor controlled ankle*

Figure 1.1: *Foot prosthesis, from the beginning to nowadays*

According to our sources [41], the first historical artifact date back to 3000 years ago, represents an igneous prosthesis used by an Egyptian noblewoman to replace the missing big toe of her right foot. Even then, the device had an aesthetic but also a practical purpose, as it was essential for wearing the classic Egyptian footwear of the time. From that wooden component, we have come to the present day with a multitude of different prosthetic solutions, suited to the many cases of amputation and their consequent needs. The more time goes by, the more the process of integrating the device with the human being is aimed at effectiveness, efficiency in restoring lost functions, but simultaneously with embodiment and comfort. Remaining in the modern age, looking at one of the first upper limb prosthesis (1.2a) and comparing with one

which intends to overcome the limitation of the state of the art (1.2b), it's clear that a giant step has been made. Anyway, there is still a long way to go in this respect, the current upper limit is technology but this is bound to improve with time thanks to the effort and work of researchers and companies involved in the field.

(a) *Eiserne Hand, passive hand prosthesis XVI c.*

(b) *Hannes arm and cosmetic gloves*

Figure 1.2: *Evolution of prosthetic arm*

1.2 State of the art

Outside the world of prosthetics, robotics is finding in the last twenty years [18] more and more expression and expansion, with continuous improvements on all fronts. Since the human being can be modelled as a set of kinematic chains, robotic manipulators and arms are one of the many fields that can be linked to prosthetics. Prostheses, which are increasingly proximal, advanced and complete in their functionality, are kinematic chains, specific declinations of mechanical actuators. The challenge solution of the kinematic and dynamic problems to be solved in order to allow the user to control the prosthesis must be extremely accurate and respect a calculation rate compatible with the real time use of the device. As a result, in some time-critical applications, the capacity to identify an accurate solution for a human model at a suitable rate might still be a bottleneck [35]. Solving the kinematics for a human model might be challenging compared to industrial manipulators since people can be depicted as highly articulate kinematic chains. Human kinematics is redundant, has a high number of degrees of freedom and should also account for musculoskeletal restrictions to provide realistic motion. Furthermore, a moving human in space is a floating base system, implying that the configuration space is based on a differentiable manifold.[42]

1.2.1 Upper limb prosthesis

The upper limb, in particular the hand, is the most important part of the body to explore and interact with the external world, performing a wide range of tasks in a daily life scenario as well as work and sport. The primary functions of a hand are grasping and manipulation. However, considering its proprioceptive sensitivity, it can be compared to a true sense organ capable of preventing injury, defining object contours, and sensing temperature, as well as being one of the most refined effectors of psycho-emotional expression, along with the rest of the upper limb. In the field of body

language, of course, it is an element of the gestures [37]. It is certain that the loss of an upper limb will have physical and psychological consequences. It has a direct impact on dexterity and motility, making it difficult for a person to walk or balance correctly. The body's biomechanics changes in an attempt to compensate for the lost limb. Excessive usage of the rest of the body, as well as inappropriate body postures, result in significant weariness [36]. Furthermore, phantom-limb pain is a common occurrence [12]. Losing a limb has been shown to have a significant impact on a person's sense of body image and, as a result, self-image, affecting life satisfaction, social life, and the nature of social interactions [4].

There is no evidence or study that might say that upper limb loss is more disabling than the loss of a lower limb but it's certain that abandonment rate is higher for upper limb prostheses. According to some studies [14] abandonment of lower limb prosthesis is infrequent, from 4% to 11%. On the other hand, Sugawara et al. estimated an abandonment rate of 33.87%, namely one user out of three decides to renounce on the prosthesis use [39]. Contrary to this, according to several studies completed over the last two decades, substantial rejection rates of all types of upper limb prosthetic devices across a wide range of users continue to be reported. Rejection rates for myoelectric devices range from 25% to more than 50%, and rejection rates for body-powered devices might reach 35%, depending on the research group [43]. A lower limb prosthesis is always used to return to walking when the stump allows it, however many upper limb amputees do not utilize active prostheses and compensate by overworking the remaining limb. This is also true in the event of a double amputation at birth, when the subject begins to use his feet for tasks that would typically be done with his hands, whereas a subject without lower limbs begins to utilize prosthesis.
Certainly, walking is fundamental for the correct maintenance of the physiological processes of the organism in the long term and has a substantial effect on the fruition of countless aspects of life [26]. So the lower rates of abandonment for lower limb prostheses are to be explained by the essentiality of the movement which they restore. However, in my opinion, it also should be mentioned that nowadays lower limb prostheses respond quite well to the amputee's needs on several fronts: restoring movement but also comfortable conditions of use of the device, practicality and intuitiveness. Therefore, on my view, the problem of the high rate of abandonment of upper limb prostheses lies not only in a reduced need to remedy the impairment compared to lower limb amputation, but also and above all in the benefit/cost ratio, perhaps still too low in terms of motor skills regained in the face of discomfort or having to perform movements through controls which are not very intuitive and not immediate.
Upper limb amputation has an incidence of 3.9 per 100,000, with fingers being the most affected site (3.2 per 100,000). Trauma is the main cause of amputation, with a clear male predominance, followed by neoplasms, vascular diseases, infectious diseases and tumours. Congenital malformations, divided into longitudinal and transverse, include Ameliae, which consist of the absence of one or more limbs due to the stunting of embryonic development [37].

The definition of upper limb prosthesis, found in the norm, is: "orthopaedic aid that compensate or substitute, although partially the missing limb both on a functional

and aesthetic aspect (...) An upper limb prosthesis is a combination of compatible components, usually produced by a single producer and commercially available. The components might be integrated with any other component individually fabricated, to produce a range of different upper limb prosthesis" [4].

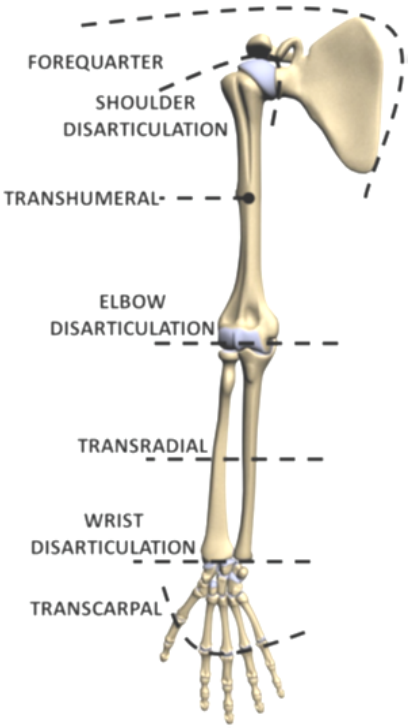


Figure 1.3: *Levels of upper limb amputation [7]*

The norm provides two different methods of classification:

1. Depending on the level of amputation.
2. Depending on the construction features, the functional characteristics and the method of control.

"The levels of upper limb loss can be classified as transcarpal, wrist disarticulation, transradial, elbow disarticulation, transhumeral, shoulder disarticulation and forequarter" [13] Fig.1.3
As a result, the second classification separates the prosthetics world into two major groups Fig. 1.4. The ability to generate force distinguishes active prostheses from passive prostheses and this force can be used to accomplish grip tasks in active prosthetics, making them functional. Passive prosthesis, on the other hand, are unable to move because to a lack of active power. A more detailed explanation of the classification is based on technological characteristics.

The aim of active prosthesis is to replace and restore the functionality and key movements of the lost limb. The method used to apply the external force generated by the gripping mechanism differs between active prostheses.
There are three types of active prosthesis [4]:

1. **Body-powered/intra-corporeal energy prostheses**
   The residual muscles of the stump, as well as muscles from other regions of the

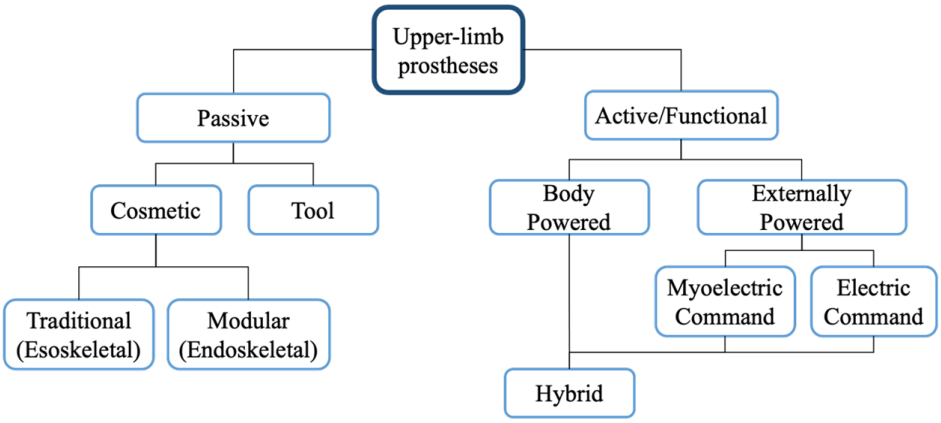


Figure 1.4: *Classification of upper-limb prostheses depending by their functionalities*

body, are used in body-powered prostheses Fig.1.5. These prostheses are able to move thanks to the mechanical energy generated by cables that are actuated by braces. These kinematic prostheses have reduced functionality and comfort, allowing only few movements. They consume more phisical energy and can only generate a limited amount of prehensile strength. Nonetheless, their relative lightness, reliability, robustness, mobility in any context, and sensory input provided by cables and bracing make them popular. In fact cable-driven systems account for around half of the present market for upper limb prosthesis [30].

Figure 1.5: *Body powered prosthesis with two different terminal devices*

2. **Externally powered/Extra-corporeal energy prostheses**
   The movements of externally powered prostheses, also known as electromechanical prostheses, are operated by electrical motors fed by batteries, whose rotational verse dictates the movement's verse. They can increase grip strength while reducing electrical energy consumption and improving comfort, albeit their weight and design complexity may be significant.

   According to the control method, they are defined as:

   (a) *Myoelectric prostheses*
   Precise groups of muscles are voluntarily activated in an isolate way by the subject. Surface electromyographic (EMG) signals, generated by isometric

Figure 1.6: *Externally powered myoelectric prosthesis*

contraction of the residual muscles of the limb and collected through surface electrodes, actuate the control.

(b) *Electronic prostheses*
These prostheses are typically employed when there are clear-cut bone zones that can trigger pressure sensors. Specific switches or slider-type inputs activate the control method.

3. **Hybrid prostheses**
They are designed for trans-humeral amputees and combine the externally powered and kinematic prosthesis, usually consisting of a myoelectrically controlled hand and a body-powered elbow controlled by cables and braces.

Figure 1.7: *Hybrid prosthesis: a body powered elbow with a myoelectric hand*

1.2.2 Control strategies

The EMG signals generated in skeletal muscles, which reflect the user's intention, are used to control the myoelectric prosthetic hand. Various decoding scheme, as proportional control, on–off control, direct myoelectric control, pattern recognition, and postural control, are utilized to actuate the prosthetic hand using the EMG signals according with the user intention [46].

The first and most widespread method of control, mostly due to its robustness is the on-off control [43], which is based on the alternating or simultaneous contraction of antagonistic muscle groups. Depending on the proximity of the amputation, mutually exclusive movements of each joint replaced by the prosthesis (flexion/extension, pronation/supination, opening/closing) are activated one at a time by two electrodes placed in two loci on the stump which detect one or both contractions of the residual muscle and activate the associated movement. By the co-contraction of the residual muscles of both mentioned two loci, it is possible to switch from a joint to another and select the one to control. The single contraction activate one of the mutually exclusive movement of the joint (exemple: flession). This method suffers from being counter intuitive and requires the amputee to learn a new control scheme, the mental effort required is considerable at least in the first instance. A lot of training is needed to reach a good level of dexterity.
This control technique is clearly not natural due to the switching phase to select the joint controlled that introduce a time delay between user intention and prosthesis action. The usual control has had problems in expanding the available DoF in the prosthetic device due to the difficulties in known the actual joint controlled and the time consuming during the switching between joints. The implanted myoelectric sensor control approach has the same features as sEMG control [33], but it is more reliable since the signals received through the needles are less noisy. Moreover an advantage is having a stable implant, anchored to the nerve, which solves the problem of electrode shifting which is one of the main causes of degradation of control, also in the case of pattern recognition.

"Proportional control is exhibited by a prosthesis system if and only if the user can control at least one mechanical output quantity of the prosthesis (e.g., force, velocity, position, or any function thereof) within a finite, useful, and essentially continuous interval by varying his/her control input within a corresponding continuous interval."

A system in which the electromyogram (EMG) from the flexors and extensors of the user's forearm is monitored, amplified, filtered, and smoothed by two active electrodes is a simple example of proportional myoelectric control. This gives EMG amplitude estimations to send to a hand controller. The controller sets a voltage provided to the motor that is proportionate to the contraction intensity after applying thresholds to reduce ambiguity at low contraction levels [17]. Several commercial prosthesis makers provide this feature.

Direct myoelectric control share features with proportional control and needs independent EMG sites to allow control of single finger movements. However, crosstalk in EMG signals is a barrier in achieving independent control of hand. As already said this might be partially solved with implant-able myoelectric sensor wich give intramuscular EMG signals [28].

One promising solution that is increasingly being developed because of its excellent results and great future potential is Targeted Muscle Reinnervation (TMR). This consists of transferring nerve of the residual part into healthy muscle to be used as amplifier (usually chest) In fact, even after amputation, the nerves remain active and able to transport information from the brain in an efferent manner. "The innervated muscles then serve as biological amplifiers of the amputated nerve motor signals. By transferring multiple nerves, TMR myoelectric signals allow intuitive, simultaneous

control of multiple joints in an advanced prosthesis" [5]. After surgery, the nerves grow into the new target muscles and can induce their contraction. As a result, the amputee moves selected muscles, but thinks he is moving the limb that is no longer present, the so called phantom limb. These chosen muscles are usually residual muscles of the amputated limb or chest, which are big muscles with a few number of movements. The signals are then picked up by EMG sensors and used to activate the prosthesis. The big step forward is that with this technique it is possible to allow the subject to perform a given movement by performing the same contraction as he/she would perform with the limb no longer present. In addition, TMR allows additional control sites to be generated, so that the subject no longer has to be limited to two as in the on-off case.

Complex control strategies take into account a huge number of information useful to decode user intention in a more reliable way, so the control can be more intuitive than the dual-site. Feature extraction and feature classification of segmented data in signal processing are commonly used in pattern recognition-based myoelectric control to provide commands to the motor controller. Depending on the amount of features, some signal processing may entail feature reduction or feature selection (FS) between extraction and classification. To determine the information content of the MES, several characteristics in time, frequency, and time–frequency are extracted in general. Pattern-based recognition for myoelectric control involves a reliable approach for extracting information from retrieved features [15].
This approach requires the application of several electrodes, from 6 to 10. In more advanced control strategies, a specific sEMG pattern is mapped and assigned to a specific movement. The classifier then associates a certain number and type of muscle signals with the corresponding movement.The fundamental concept behind this method is to examine these patterns and develop a model that can determine if the replicated signals are comparable to those collected during the training phase using a likelihood function. Because the model accuracy is directly connected to the input data used to train the algorithm, the training phase is crucial. To allow the classifier to discriminate between different gestures, it is critical to supply meaningful signals with distinct patterns for each intended movement[10].

1.3 Kinematic theory

A mechanical structure is made up of a set of bodies, called links, which are assumed to be rigid, and joints. Robots are driven by means of actuators, that could be placed in all joints (full actuated systems) or in some of that (under actuated systems), which determine their configuration. The configuration of a joint is defined as the angle for rotational joints joints and the elongation for prismatic joints. The knowledge of the configuration of a joint is made possible by special sensors (angular sensors such as encoders or elongation sensors). A distinction is therefore made between actuated and non-actuated, sensorised and non-sensorised joints. An actuated and sensorised coupling is defined as active. A non-actuated and non-sensorised coupling is defined as passive. A multibody system can be organised in a structure called a kinematic chain, which can belong to two different categories: serial or parallel.

A robot can be described analytically through its kinematic model. A serial kinematic robot can be defined as a series of reference frames, located at actuated joints

(DoF), in relative motion to each other. The analytical description concerns the relationships between the positions of the joints (angles and/or elongations) and the configuration or positioning of a particular member of the structure, typically the end effector, or the body bearing the end organ.
In serial manipulators, the end effector is placed at the end of the chain and at the other end is the base, which is usually integral with the environment. It is precisely with respect to the environment that the end effector is usually described, using a homogeneous matrix. The kinematic description of a robot, or kinematic problem, can be approached in two ways, depending on the known elements in one's possession and the unknown ones. The two possibilities are:

- **Direct kinematic problem**: It concerns the determination of the end effector pose, when the configurations of the sensed joints (q_s) are known. Q_s is the set of values that can be assumed by the configurations of the sensed joints. Mathematically, it is a question of obtaining an explicit expression of the direct kinematic map.
- **Inverse kinematic problem**: It consists in determining the configurations to be assumed by the actuated joints in order to make the end effector position an assigned one.

Some useful definitions:

- **Joint space** (or configuration space) is the space in which the vector q of joint variables is defined. Its dimension is indicated by n.
- **Operational space** (or Cartesian space) is the space in which the vector $x = [p, \Phi]^T$ is defined. Its dimension is indicated with m. p is the vector of Cartesian coordinates of the position of the end effector. In general it has dimension 3×1 (coordinates x, y, z). Φ is the vector representing the orientation of the end effector. In general it has dimension 3×1. The actual dimensions of the vectors p and θ depend on the workspace and the task to be performed by the robot.
- **Workspace**: within the operational space, to evaluate the performance of a robot, the workspace is defined as the region including all the possible reachable position of the end effector resulting from all the possible joint configuration depending on the DoF of the kinematic chain. **Reachable workspace**: region that the tool backhoe origin can reach in at least one orientation. **Right-hand working space**: region that the origin of the tool reference frame can reach with different orientations. It is a subset of the first.

A serial arm is intended as an open kinematic chain. The procedure can be summarised as follows:

1. each member is associated with a reference frame integral with it, which describes its configuration;
2. by means of homogeneous transformations, the position of an orientation axes tern with respect to the previous one is described starting from the base orientation axes tern to the end effector one;

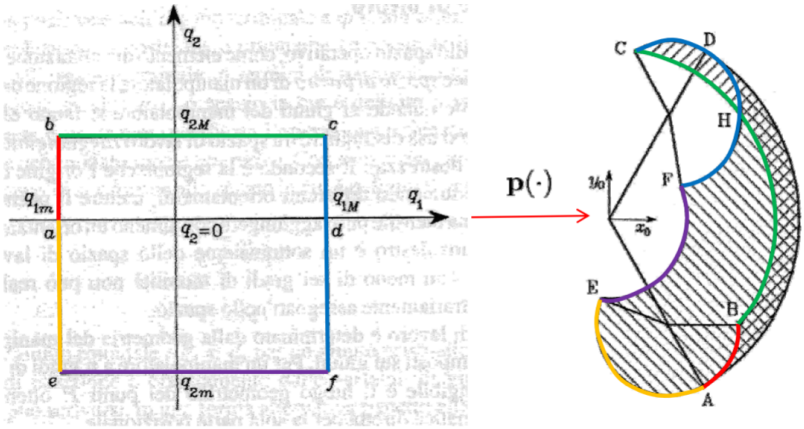


Figure 1.8: *Workspace of a 3DoF planar arm*

3. this description represents the posture of the end effector with respect to the base.

The direct kinematic model is more readily understood and easier to resolve. There are general and automatic techniques for calculating the space of transformations $A(q)$ to arrive at the configuration of the end effector and the solution is univocal.

In the inverse kinematic model, the aim is to determine the joint variables once the position and orientation of the end effector have been assigned. There is no general technique which, if applied systematically, provides a solution. The solution sought may not be unique. One can have:

- no solution (one is required to be outside the workspace);
- a finite set of solutions (one or more);
- infinite solutions.

Closed-form and non-numerical solutions are sought for computational reasons and because expressing solutions in analytical form allows one to select a particular solution (if one has more than one solution). To obtain a solution in closed form to the problem of inverse kinematics, there are essentially two techniques:

- **Algebric**, which consists of manipulations of the kinematic equations until we obtain a set of equations that allow us to invert the equations.
- **Geometric**, which is based, when and if possible, on geometric considerations, depending on the structure of the manipulator, which help in the solution.

Redundancy

A manipulator is said to be redundant from a kinematic point of view when it has more degrees of freedom than the number of variables required to characterise a given task. In terms of joint and operational spaces, a manipulator is intrinsically redundant if $m < n$. Redundancy is however a concept related to the type of task to be performed. Even if $m = n$ a manipulator can be functionally redundant if only h components of

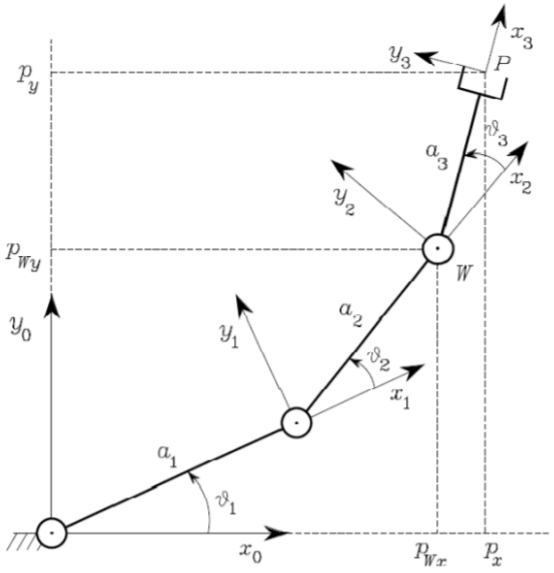


Figure 1.9: *3DoF planar arm*

the operating space are involved. For example, observing Fig. 1.9, for 3-arm planar $m = 3, n = 3$. If the orientation of the end organ is also affected $h = m = n = 3$, the manipulator is not redundant. If orientation is not involved then $h = 2; m = n = 3$, thus leaving one degree of redundancy. Since in $\mathbb{R}^{\mathbb{H}}$ space a maximum of 6 Dof are possible (sum of 3 translational DoF and 3 rotational DoF), $m = 6$ at most. Consequently, a robot with $n = 6$ is non-redundant if $h = 6$ because $h = n = m, r = n - m = 0$, e.g.the human arm has 7 DoFs (1 time redundant).

1.4 Quaternions

In mathematics, quaternions are extensions of complex numbers, they satisfy all the properties of fields, such as real or complex numbers, except for the commutative property of the product [6]. Complex numbers form a 2-dimensional space, i.e. a plane. Quaternions contain the complex numbers and form a real vector space of dimension 4. The two properties of body and vector space give quaternions a non-commutative division algebra structure. Quaternions are frequently employed in theoretical physics and more applicable disciplines such as 3D computer graphics and robotics (to find the spatial position of multi-joint mechanical arms) because they have a significant applicability in the modeling of spatial rotations.

Quaternions were formalised by Irish mathematician William Rowan Hamilton in 1843. He needed a way to extend complex numbers across a larger number of spatial dimensions. After failing to find a three-dimensional extension, he devised a four-dimensional one: quaternions. In a broader sense, Hamilton is credited with inventing the vector product and scalar product in vector spaces. A quaternion is an ordered quadruple of real numbers, with the first coordinate being the 'scalar' component and the remaining three being the 'vector' part, according to Hamilton. The scalar portion of the product is the scalar product of the vector component modified by a sign, and

the vector part of the product is the vector product when two quaternions with zero scalar part are multiplied.

Quaternions are most commonly used to represent rotations and directions in three-dimensional space today. As a result, 3D computer graphics, control theory, signal processing, attitude control, physics, and astrodynamics may all benefit from their usage [38].

1.4.1 Definition

The simplest way to express a quaternion (q) is its vectorial form:

$$q = t + u \cdot i + vj + wk \tag{1.1}$$

The resemblance to the structure of complex numbers is clear. The imaginary dimension of the quaternion space is increased by adding two "components" j and k. Quaternions are defined by the set-theoretic form, which considers them to be extensions of complex numbers. As a result, the set $\mathbb{H}$ can be defined as follows:

$$\mathbb{H} = \{q = a + bi + cj + dk \mid a, b, c, d \in \mathbb{R},\ i^2 = j^2 = k^2 = ijk = -1\} \tag{1.2}$$

1.4.2 Basic Opertions

In this set we can define two operations, sum and product, which give H, as already mentioned, an algebraic structure of vector space of dimension 4: $\dim(\mathrm{H}) = 4$, having base: $\{1, i, j, k\}$. Considering two quaternions $q_1 = a_1 + b_1 i + c_1 j + d_1 k$ and $q_2 = a_2 + b_2 i + c_2 j + d_2 k$, the following elementary operations apply:

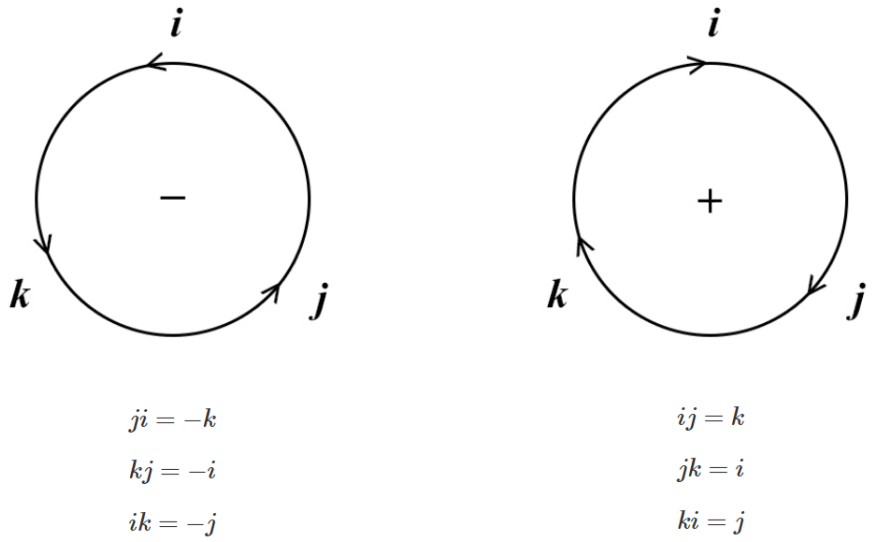
$$\begin{aligned} q_1 + q_2 &= (a_1 + b_1 i + c_1 j + d_1 k) + (a_2 + b_2 i + c_2 j + d_2 k) \\ &= (a_1 + a_2) + (b_1 + b_2)i + (c_1 + c_2)j + (d_1 + d_2)k \end{aligned} \tag{1.3}$$

$$\begin{aligned} q_1 - q_2 &= (a_1 + b_1 i + c_1 j + d_1 k) - (a_2 + b_2 i + c_2 j + d_2 k) \\ &= (a_1 - a_2) + (b_1 - b_2)i + (c_1 - c_2)j + (d_1 - d_2)k \end{aligned} \tag{1.4}$$

$$i^2 = j^2 = k^2 = ijk = -1 \tag{1.5}$$

The quaternions form a non-commutative entity. They satisfy all the usual properties of fields, such as real numbers, complex numbers, except the commutative property of the product. The product between quaternions is therefore anti-commutative with reference to the following figure 1.10

$$\begin{aligned} q_1 q_2 &= (a_1 + b_1 i + c_1 j + d_1 k)(a_2 + b_2 i + c_2 j + d_2 k) \\ &= (a_1 a_2 - b_1 b_2 - c_1 c_2 - d_1 d_2) + (a_1 b_2 + b_1 a_2 + c_1 d_2 + c_2 + d_1)i \\ &+ (a_1 c_2 + c_1 a_2 + d_1 b_2 - b_1 d_2)j + (a_1 d_2 + d_1 a_2 + b_1 c_2 + c_1 b_2)k \end{aligned} \tag{1.6}$$



$$ji = -k \qquad ij = k$$
$$kj = -i \qquad jk = i$$
$$ik = -j \qquad ki = j$$

Figure 1.10: *Product between quaternion components*

$$q^{-1} = \frac{q^*}{|q^2|} \rightarrow \; qq^{-1} = q\frac{q^*}{|q|^2} = \frac{|q|^2}{|q|^2} = 1 \tag{1.7}$$

One property of quaternions is that they can be expressed in 'b-complex' form, i.e. as the sum of two complex numbers:

$$q = (a + bi) + (c + di)j \tag{1.8}$$

ascribed in a compact form as:

$$\mathbb{H} = \mathbb{C} \oplus \mathbb{C}j \tag{1.9}$$

1.4.3 Conjugate and Norm

It is defined as a conjugate quaternion of q and it's written q^* the following element of H:

$$q^* = a - bi - cj - dk \tag{1.10}$$

The following properties apply to the conjugate:

$$(q_1 q_2)^* = q_1^* q_2^* \tag{1.11}$$
$$(q_1 + q_2)^* = q_1^* + q_2^* \tag{1.12}$$
$$(q^*)^* = q \tag{1.13}$$

For each quaternion $q \in \mathbb{R}$ the norm is defined by the following expression:

$$\|q\| = \sqrt{qq^*} = \sqrt{a^2 + b^2 + c^2 + d^2} \in \mathbb{R} \tag{1.14}$$

The set $q \in \mathbb{H}$ of the quaternions is a vector space on $\mathbb{R}$

1.4.4 Matrix Representation

Quaternions have multiple representations, it is possible to express the quaternion in the so-called matrix form. The following complex matrices are associated with the quaternion $q = a + bi + cj + dk$

$$1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, I = \begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix}, J = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, K = \begin{pmatrix} 0 & i \\ i & 0 \end{pmatrix} \tag{1.15}$$

1.4.5 Rotations

The most important property of a quaternion is that it represents a rotation in $\mathbb{R}^3$. Any three-dimensional rotation can be represented by the so-called axis-angle notation $R = e^{\theta\hat{u}}$. If we want to represent this rotation as a quaternion, we can write:

$$q = e^{\theta\hat{u}} = \begin{bmatrix} cos(\frac{\theta}{2}) \\ \hat{u}sin(\frac{\theta}{2}) \end{bmatrix} \tag{1.16}$$

where θ is the angle of rotation and $\hat{u}$ a three-dimensional verse. In this example, the rotation of an angle θ around $\hat{u}$ has been described.
A quaternion $q = q_1 + q_2 i + q_3 j + q_4 k$ can be transformed and/or converted into a real rotation matrix [38]:

$$\begin{pmatrix} q_1^2 + q_2^2 - q_3^3 - q_4^4 & 2q_2q_3 - 2q_1q_4 & 2q_2q_4 + 2q_1q_3 \\ 2q_2q_3 + 2q_1q_4 & q_1^2 - q_2^2 + q_3^2 - q_4^2 & 2q_3q_4 + 2q_1q_2 \\ 2q_2q_4 - 2q_1q_3 & 2q_3q_4 + 2q_1q_2 & q_1^2 - q_2^2 - q_3^2 + q_4^2 \end{pmatrix} \tag{1.17}$$

Chapter 2

Materials and methods

This section describes all the materials and methods used for the development of the project. The chapter is focused on the is on the description of the activities carried out and the used hardware and software. While respecting all commercial and proprietary secrets, the Hannes system is described in its current configuration with a view to its future developments. The Hannes system consists of the Hannes hand [24], a prosthetic device that can perform finger movements (grasping), the two DoFs of the wrist (wrist pronation/supination - WPS, wrist flexion/extension - WFE) and the DoF of the elbow (elbow flexion/extension - EFE). Each DoF is driven by a different electric motor. Hannes hand is a prosthetic device realised by the collaboration between the IIT Rehab Technologies department and the INAIL Prosthesis Centre of Vigorso di Budrio.

In order to allow a better reading and understanding the part 2 is divided into several paragraphs. Section 2.1 describes the Hannes prosthesis concerning the mechanical design, something about the electronic design and the software including its version in virtual reality. The study protocol is outlined in sec. 2.2 explaining the evaluation made for every method individually and compared with the others. In sec.2.3 the description of all the algorithms used and developed in the study.

2.1 Hannes arm

The Hannes prosthesis, the result of a long-standing collaboration between IIT and INAIL, is a polyarticulated hand capable of restoring 90% of the functionality lost to transradial amputees [24]. An active elbow has recently been developed for transhumeral patients and the shoulder is currently under development. The latter is one of the team's challenges, in order to include shoulder disarticulation amputees among the possible users of this device.
The prosthetic design was guided by the study of anthropometry, as well as structural and aesthetic characteristics, in order to make the prosthesis perceived as an integral part of the body for the user. The prosthesis has key biometric properties that make it uniquely similar to a human hand.
In figure 2.1 it is possible to observe the result of the iterative design process with the previous prototypes in the background and the last model in the foreground, decidedly the most accurate for anthropometric shape. As shown in Fig.2.2a, Hannes systems used in the study consists of 4 main interacting physical components:

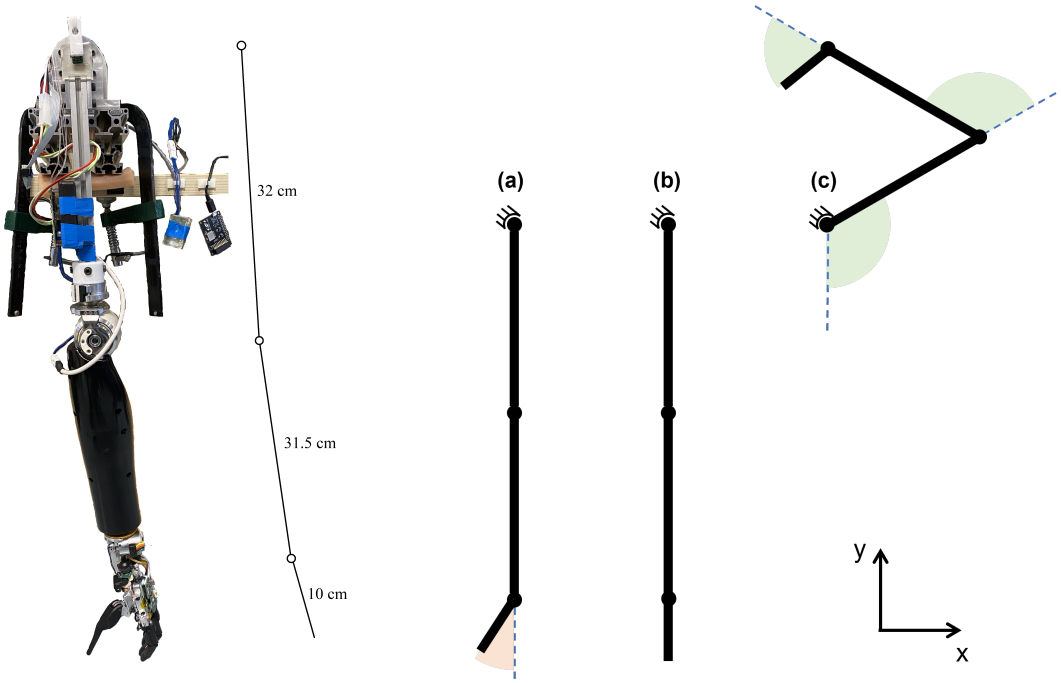
Figure 2.1: *Evolution of Hannes hand*

1. Hannes hand;
2. An active flexion-extension (F/E) wrist module and an active pronation-supination (P/S) wrist module;
3. A housing resembling the anthropometric shape of the forearm, containing the battery and the control electronics with an active flexion-extension elbow module attached;
4. A component with a length proportional to the anthropometric model to replicate a humerus, attached to motor with a brake with to reproduce the flexion-extension movement of the shoulder in the experimental phase.

2.1.1 Hardware

Hand

The differential mechanism behind the Dynamic Adaptive Grasp (DAG) system Fig. 2.3, which provides the prosthesis its unique capacity to adjust to the form of the item and to any sort of external force, is a crucial feature of Hannes' hand. The conducting wire is driven by the DC motor (Faulhaber CR 2642 connected with a 19:1 planetary gearbox), which transmits the force to the cable-based mechanism located in the palm. The lead wire runs from the motor shaft to the thumb, passing via two differential elements placed on linear guides, each of which has a special bushing bearing and two rails for the bushing to travel along. Each bushing has two idler pulleys: one holds the follower wire, while the other drives two neighboring fingers.
To replicate distinct grasping actions, the thumb can take three alternative positions: pinch, power, and lateral configurations. In one of the foregoing arrangements, the thumb is manually moved. The lateral posture is appropriate for handling thin objects like credit cards, the power position is appropriate for gripping and hauling large goods, and the pinch position is appropriate for holding tiny objects with the thumb and index finger.



(a) *Physical setup used for experiment with the Bluetooth Dongle for communication*

(b) *Simple scheme of arm configuration: a) minimum ROM - b) rest ROM - c) maximum ROM pose*

Figure 2.2: *Hannes arm*

Wrist

The active FE wrist is composed of a BLDC motor integrated with a custom 240:1 gearbox. It is designed to have the same rotation axis, size, and speed and torque of the healthy limb. Another important achievement is to have a non back-driveable mechanism to support high static loads. The electric motor for the pronation-supination movement is located in the socket next to the connection interface. The design allows the patient to easily attach and detach the hand prosthesis and FE wrist actuator while maintaining a secure electrical connection between the two portions of the prosthesis system while in operation. The motor deputed to the FE is part of body of the hand while the PS motor is proximally located in the forearm as in the human physiology.

Elbow

Hannes' myoelectric elbow module has an active component for lifting objects controlled by the user, as well as a passive gravity compensation device and a user-controllable clutch that allows the elbow prosthesis to enter a passive free-swing mode with a natural range of motion for walking. For active movement and lifting of prosthetic weights, Hannes' elbow features a non-backdriveable, low-noise powertrain. This technology allows the device's battery life to be extended by lowering the motor's required current consumption, as well as reducing the device's total size by lowering the powertrain torque needs.

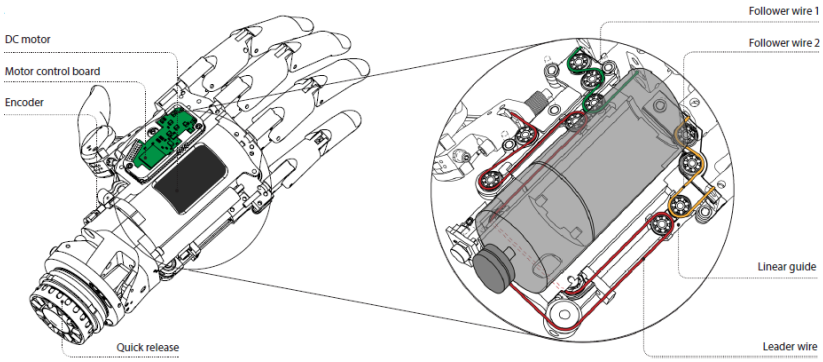


Figure 2.3: *Transmission mechanism based on DAG system*

Shoulder Prototype

The Hannes Arm doesn't natively include a shoulder module yet so a prototype joint was developed. It consist of a hollow shaft resembling the anatomical dimension of a human humerus attached to a BLDC motor that actuates the shoulder FE.

Range of Motion

The Hannes arm setup used in the study was set to have certain ROM for each joint. As rest position, consider the polyarticulate stretched downwards as a single arm stretched across the body in a resting position. This should be taken into account principally for the zero position of the shoulder joint. The rest of each joint has been set as the angular position so that the two links, the one preceding and the one following the joint, lie on the same line. The result of this choice leads to a configuration whereby all joints in the rest position lead to the arm being straight and extended along the vertical. The zero position for the shoulder and elbow joints corresponds to the rest one, while the zero position for Wrist FE joint corresponds to a fully flexed wrist. Analysing the individual joints, they have the following ROMs:

- Shoulder: $0° : 120°$
- Elbow: $0° : 120°$
- Wrist FE: $-33° : 54°$
- Wrist PS: $0° : 360°$

Electronic architecture

The Hannes system's electronic architecture consists of:

- Scheda Controllo Motore Mano (SCMM): consisting of a DC motor driver to actuate the grasping movement, a position encoder, a microcontroller for communication and an IMU to sense the orientation of the hand in space. It also includes:
  - Scheda Controllo Motore Wrist FE (SCMWFE): The electronic components consist of a BLDC motor drive, a position encoder and current sensor;

  - Scheda Controllo Motore Wrist PS (SCMWPS): The electronic components consist of a DC motor drive.

- Scheda Controllo Motore Gomito (SCMG): consisting of a BLDC motor driver, a position encoder, a microcontroller for communication and an IMU to sense the orientation of the forearm in space;
- Scheda Controllo Motore Spalla (SCMS): consisting of a BLDC motor driver, a position encoder, a microcontroller for communication and an IMU to sense the orientation of the arm in space;
- The EMG-Master (EMGM): This electronic board houses the firmware that converts the electromyographic signals or angle values to be attained for each joint into the class of movement necessary to accomplish the kinematic chain's intended configuration. The EMGM feeds the SCM boards the synthesised position reference, which in turn generate the PWM needed to control the electric motors;
- Battery Management System (BMS): is a dedicated electronic board that monitors the battery state and ensures that the system operates safely. With a voltage of 12V, the rechargeable battery pack can power the full Hannes system through two independent cables (VSYS and GROUND).

These major components communicate through the CAN bus. For position-controlled joints, the Hannes arms requires normalised reference inputs in position that are mapped to their respective ROM. For the Wrist PS due to the mechanical implementation of the joint the reference input is interpreted as a speed reference. So the required inputs from each board is a number between 0 and 100 interpreted as percentage o the full-scale ROM previously defined. 2.1.1 External information is delivered to the EMGM via Bluetooth, as shown in figure 2.2a. The three IMU sensors Fig.**??** also located on each of the three SCM boards, giving information on the relative orientation of the three joints via their respective quaternions (chapter 1.4).

Figure 2.4: *IMU sensor*

2.1.2 Software: Virtual reality

The software applications Blender and Unity are used to create virtual reality. Virtual reality was employed in the thesis to build the control algorithm and to provide visible feedback on the control algorithm's outcomes, as well as to move the virtual version of the prosthetic polyarticulate. VR might be employed as a training or rehabilitation aid for amputees in the future. By operating the virtual Hannes first, and later the actual device, the patient may learn to move the prosthesis. Furthermore, the definition

of relative mobility between the skeleton linkages allows the object's pieces to move around the specified joint, such as the wrist or elbow. It is possible to provide an initial and ultimate attitude (position and orientation) to each joint in the Blender 3D environment, resulting in a movement trajectory between the two poses. It is feasible to create the required motion, such as wrist, elbow, or shoulder flexion/extension, by combining all of the independently specified trajectories. To allow virtual reality control through Bluetooth® connection with the EMG-Master, the body motions are described in Blender, while the animation is handled in Unity.

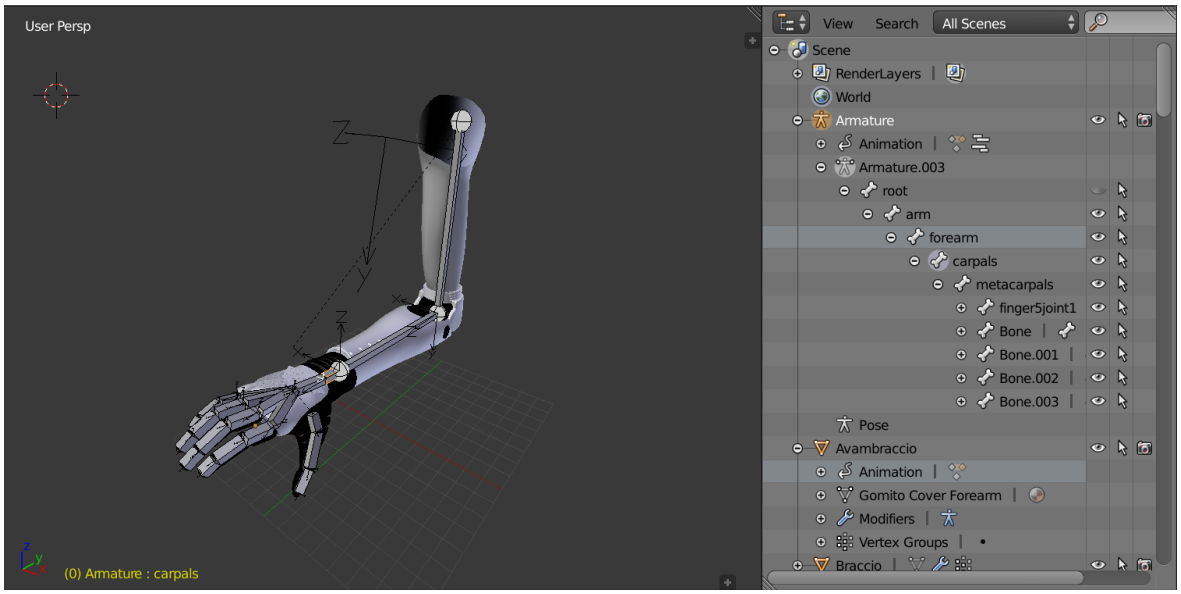
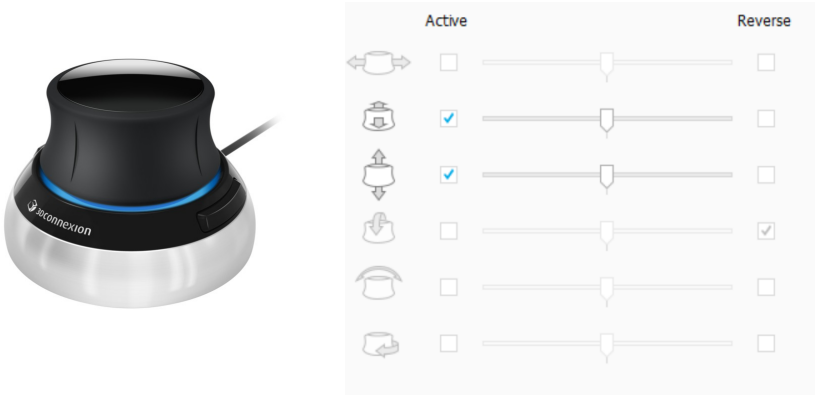


Figure 2.5: *Hannes arm in Blender virtual environment*

Blender [3] is a free open-source 3D computer graphics program which can be used for virtual reality, visual effects, interactive 3D applications, and computer games. In the context of this thesis Blender was used to create a humerus and shoulder adjoint component [3], provisionally up to the CAD transposition of what is still a project. It was also utilized to add and improve the model's repeatable motions. The Hannes system 2.5 is created using a simpler structure (called a skeleton) made up of links and joints. Furthermore, the Vertex Paint tool, which may group specific portions of the body and define all deformation regions, is used to define all deformation areas of the body.Furthermore, the Vertex Paint tool, which can group particular portions of the object and have them move concurrently with a given deformation parameter, is used to specify all deformation areas of the body. The Blender model is then imported into the Unity environment, which allows the physical item to be animated. Unity is a cross-platform game engine for making virtual reality and augmented reality games in 3D and 2D [16]. The Animator is applied to the virtual prosthesis in Unity and is defined with five separate layers: the first is related to hand closing/opening, the second is related to wrist pronation/supination, the third is related to wrist flexion/extension, the fourth is related to elbow flexion/extension, and the fifth is related to shoulder flexion/extension. Only the three layers relating to the three joints' flexion/extension motions were used in the thesis. prono-supination of the wrist was examined only at first but then abandoned. The animator's three layers can be active at the same time

to produce a sequence of motions that must belong to separate joints.



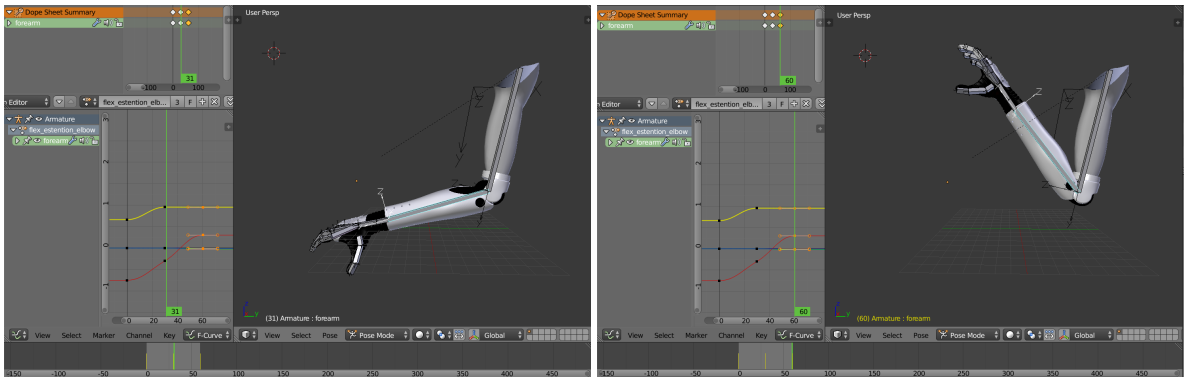
(a) *3DConnexion SpaceMouse®Compact* **(b)** *SpaceMouse enabled DoF*

Figure 2.6: *3D mouse used for prosthesis control in Virtual Reality*

Each layer therefore has its own corresponding animation. In order to translate the activation of the animations into movement of Hannes Arm, what is done is to give to Animator, as input, numbers between 0 and 1, corresponding, instant by instant, to the desired percentage of each animation, that is the desired percentage of each movement of each joint. All of this considering the minimum value of the ROM of the joint as zero percentage and the maximum as 100. The three joints of virtual Hannes Arm in the Unity world, based on the animations created in Blender, have the following ROMs:

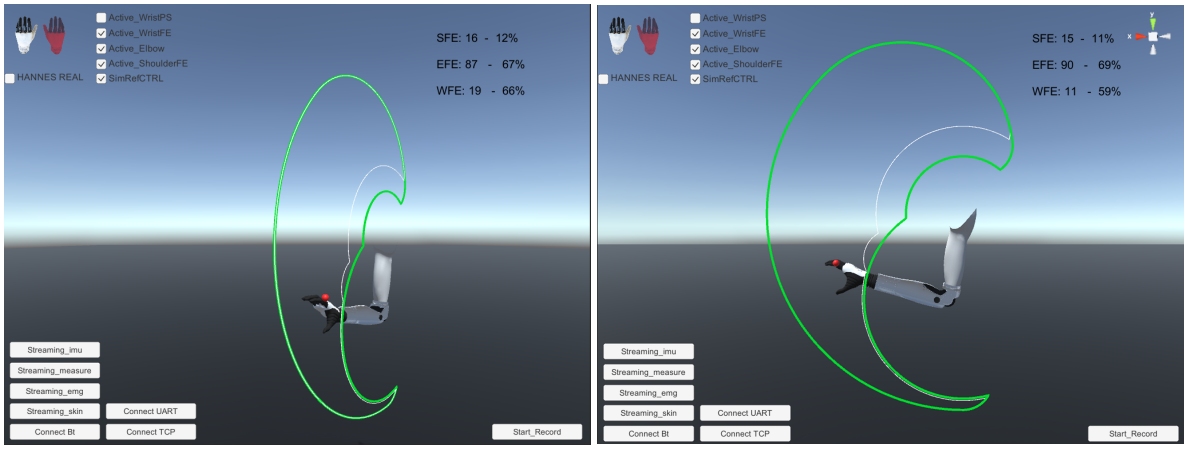
- Shoulder: $0° : 135°$
- Elbow: $0° : 130°$
- Wrist: $-60° : 60°$

Some C# scripts have been built to manage the Hannes virtual system using input commands provided by the keyboard directional arrows or Mouse3D Fig.2.6a. The class number is created at the firmware level and is the result of the classifier model. Each layer is linked to a numbered motion class.



(a) *50% of EFE trajectory* **(b)** *100% of EFE trajectory*

Figure 2.7: *Frames of elbow flexion/extension movement in Blender*



(a) *perspective view* **(b)** *side view*

Figure 2.8: *virtual Hannes Arm un Unity environment*

2.2 Study Protocol

The aim of this thesis project is to develop a control system that allows the *Hannes arm* to be controlled in Cartesian space by acting directly on the end effector. Specifically, the aim is to idealise the end effector as a point in Cartesian space, ideally corresponding to a point on the hand, approximately on the knuckle of the middle finger, to be moved in a plane parallel to the sagittal plane. In Unity plane XY was chosen as seeable in Fig. 2.8b. The result is a movement in two dimensions that can be controlled with 4 inputs, from the keyboard using the arrow keys and associating each one with a direction in space ($\uparrow$ for up movement, $\downarrow$ for down movement, $\leftarrow$ for backwards, $\rightarrow$ for forwards) or, alternatively, by control with a joystick. The SpaceMouse®Compact from 3DConnexion was chosen for this purpose Fig. 2.6a.
With this second method, it was possible to maintain a correspondence between mouse movement and generated input; in fact, the 3D mouse has 12 DoFs, of which only the two required were used Fig.2.6b. Then the movement in Cartesian space of the end effector must be translated into the achievement of a suitable configuration of the robotic arm, all to be realised in real time.

The first step consisted in studying the Unity platform and the asset already partially developed previously, the CAD and the communication protocols to be used. Three ideas followed one another to achieve the objective, two calculation algorithms implemented through scripts on Unity and the use of the ANFIS method on Matlab with the subsequent attempt to transpose it into the virtual environment of Unity. Depending on the results given by these three algorithms, one of them was chosen for its evident better features and high potential for future developments.

Before describing the three ideas for the control system, it is necessary to take a look at the type of output required for the algorithm to serve as an instruction for the movement of each motor of the three joints of interest. Once the ROMs of each joint are known, the reference angle value to which the joint is to be brought must be input as a percentage value of the range of motion, i.e. of the entire executable span. For example on a ROM like $[0° : 120°]$ the reference angle $90°$ is sent as 70 out of 100.

Except for the first experimented sec. 2.3.1, the ANFIS and Recursive algorithms

sec.2.3.2 and 2.3.3 have been tested in Matlab environment in order to evaluate the properties, compare them and be able to exercise a choice of the best one evaluating the theoretical base and the performances of the single ones on an analytical basis. The analytical comparison between ANFIS method and recursive approach (in its standard and optimized version) was performed by generating three trajectories of 100 points each. These trajectories reproduce two basic movements (vertical trajectory, e.g., to lift an object, and horizontal trajectory, e.g., to direct an object forward or bring an object to the mouth) and a more complex trajectory, an arc of circumference. Each trajectory, which is nothing more than a set of coordinates of an end effector, was fed to the three algorithms that for each one have calculated, with differing degrees of precision, the respective angles for each joint so that the chosen kinematic chain could perform the different trajectories. Once the angle values relative to each of the points were obtained, through direct kinematics calculations the positions of the end effector forming the trajectories predicted by each different algorithm were calculated. The angle data for each joint and the position (x, y) of the end effector were then collected. In addition calculated parameters to evaluate the effectiveness of the algorithms. As

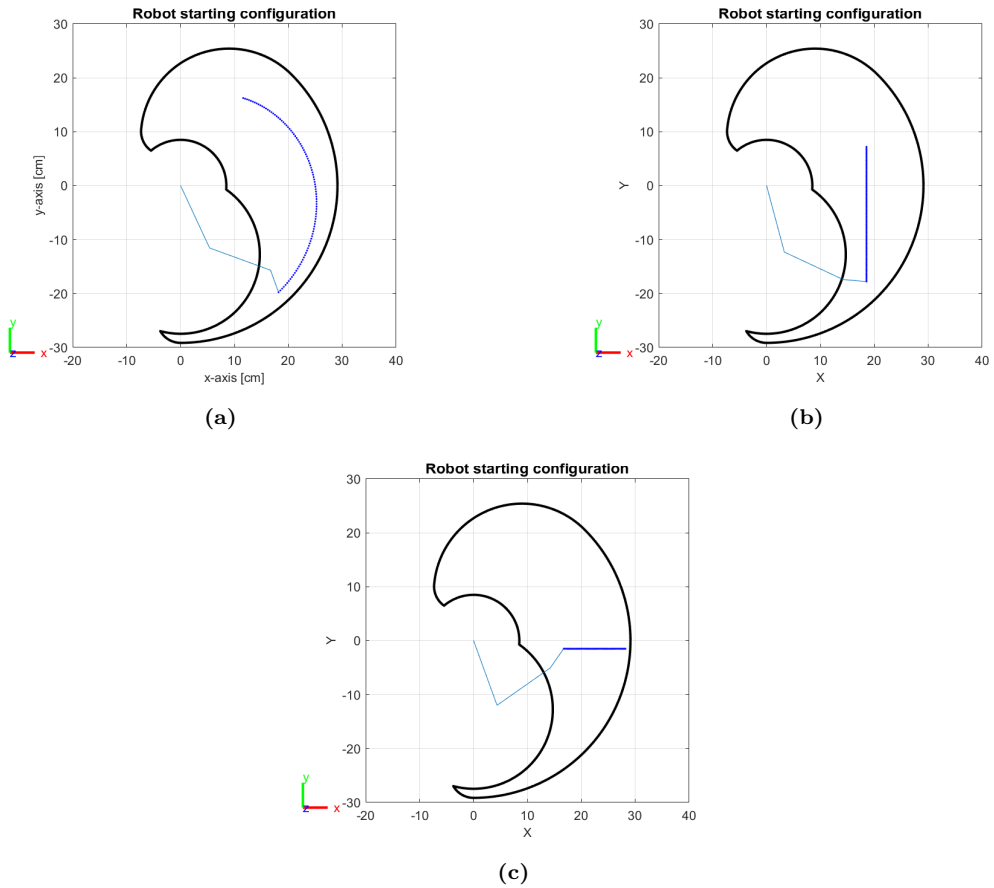


Figure 2.9: *Tested trajectories and respective starting configuration of the robot*

far as practical applicability is concerned, the evaluations on the various methods have been done working in Unity environment because the future direction of the project is to make full use of virtual reality and explore its potential for control and training with Hannes arm. In this sense, the efforts were aimed at obtaining the three angle

values, one for each joint, derived from inverse kinematics calculations performed on the coordinates of an end effector controlled in virtual environment, to obtain the right configuration of the joints and the consequent laying of Hannes arm.

Once the most valid and promising algorithm was chosen, for results obtained and applicability to virtual reality, a test was performed with the real setup described previously (reference to the chapter) to observe its behavior and a first example of application. In this case the three values calculated were the inputs sent to each motor of each different joint to modify the configuration in real time. A qualitative evaluation of this phase was performed on the trajectory results obtained since the purpose of this thesis is not concerned with.

The test consisted in executing in a virtual environment two trajectories, first a vertical and then a horizontal one similar to those with which the algorithms are tested on matlab, on which the chosen algorithm (recursive approach with optimization) calculated in real time the inverse kinematics obtaining the three angle values, one for each joint, to be sent as a reference to the physical set up that performed the movement accordingly.

2.3 Algorithms

Three algorithms have been tested since the start of the study:

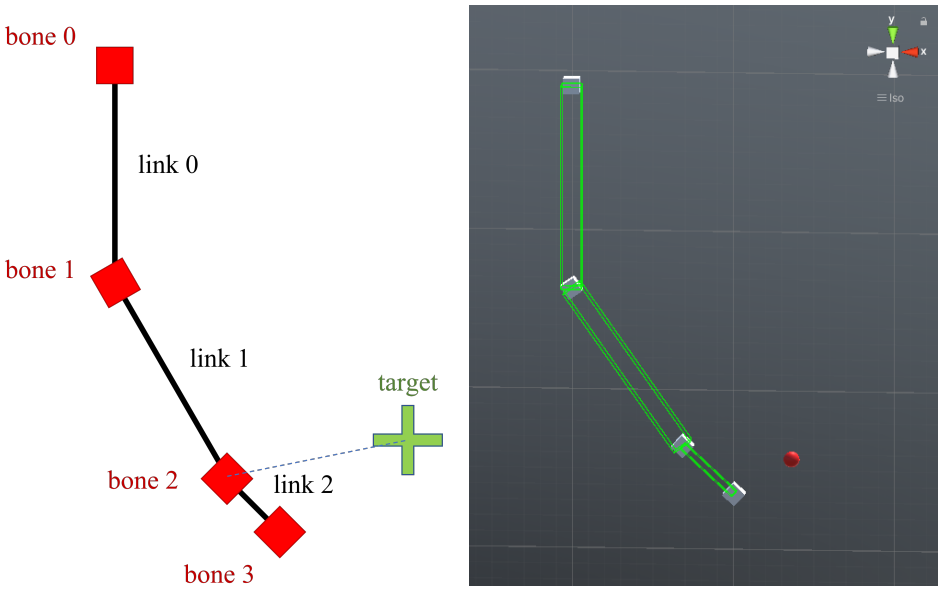
1. Fast IK
2. Recursive approach
3. ANFIS

The first one, an extremely intuitive approach to solving the kinematic problem, was soon abandoned because it was partial and crude, with little potential for development, to the extent that it was not necessary to carry out quantitative tests. The second and third solutions alternated several times in being the preferred candidate. Both with strengths and weaknesses showed great potential for solving the question this thesis project attempts to answer. All of them are listed below with conceptual and operational descriptions.

2.3.1 Fast IK fabric

This algorithm is based on a reinterpretation of an inverse kinematics algorithm available on Unity's asset store [11] rearranged for the case study. An armature is created on blender, composed of a certain number of bones equal to the number of joints, plus one. The goal is that, in Unity, the armature follows a target (which is actually acted on with keyboard inputs or 3D mouse), pointing to the target and if this enters the sphere of radius equal to the length of the arm it bends so that the position of its end effector coincides with the target. Substantially, inverse kinematics.
Some known parameters are needed to develop the algorithm. One of them is the total length of the kinematic chain (TL). It is also necessary to know whether the distance between target and origin (OT) is greater or less than the total length of the chain. The origin is defined as the base of the kinematic chain, i.e. its most proximal joint

that is integral in position with the environment. In the case where $OT > TL$ it is very simple because the vector connecting the base to the target is calculated and the chain is arranged along the direction of the vector. The direction is obtained by subtracting the positions of the two points (base bone and target) and normalising.
Another necessary parameter to set is the length of the chain intended as the number of bones composing the chain. Other parameters are: the number of iterations of the algorithm, the distance (*Delta*) between target and end effector beyond which the solver stops because it has already reached the target with a satisfactory degree of approximation. In our case the chain is composed of 4 bones and 3 links. A loop is



(a) *simple scheme*

(b) *Kinematic chain resembling an arm in Unity environment*

Figure 2.10: *Kinematic chain schemes for FAST IK method*

used to generate a virtual skeleton that makes it possible to display the relationship between one bone and another according to the set length 2.10b. The 4 bones are arranged at an initial distance which reproduces the anthropometric measurements and the average distance between shoulder-elbow-wrist-knuckle.
As mentioned above, for a chain of 3 links, as in our case, 4 bones are needed, numbered from 0 to 3, going from the most proximal to the most distal. Therefore, translating to a human arm, bone 4 corresponds to the knuckle, bone 3 to the wrist, bone 2 to the elbow and bone 0 to the shoulder Fig. 2.10. With a loop, each bone is associated with a length equal to the distance between itself and the next bone, it follows that the last bone has zero length. The total length is equal to the sum of all the lengths of every bone.

The algorithm is based on an iterative process, a loop whereby for each bone the position is detected, mathematics is performed on it and it is reset after the calculations made, starting from the most distal segment to the most proximal and vice versa. It does not act directly on the bone, but calculations are made on its position, which is then set to the new calculated value. The first step is to find out if the target distance is greater than the total length. If so, the new position of each bone is calculated,

starting with the most proximal. The n^{th} bone is placed at a distance equal to the length of the previous bone from the previous bone along the direction connecting the target to the bone 0.
In the case where $TL < OT$ the procedure is different. The process is divided into two phases, firstly a backward phase and then a forward phase, repeated iteratively until the chosen number of iterations or until the Delta parameter is reached.

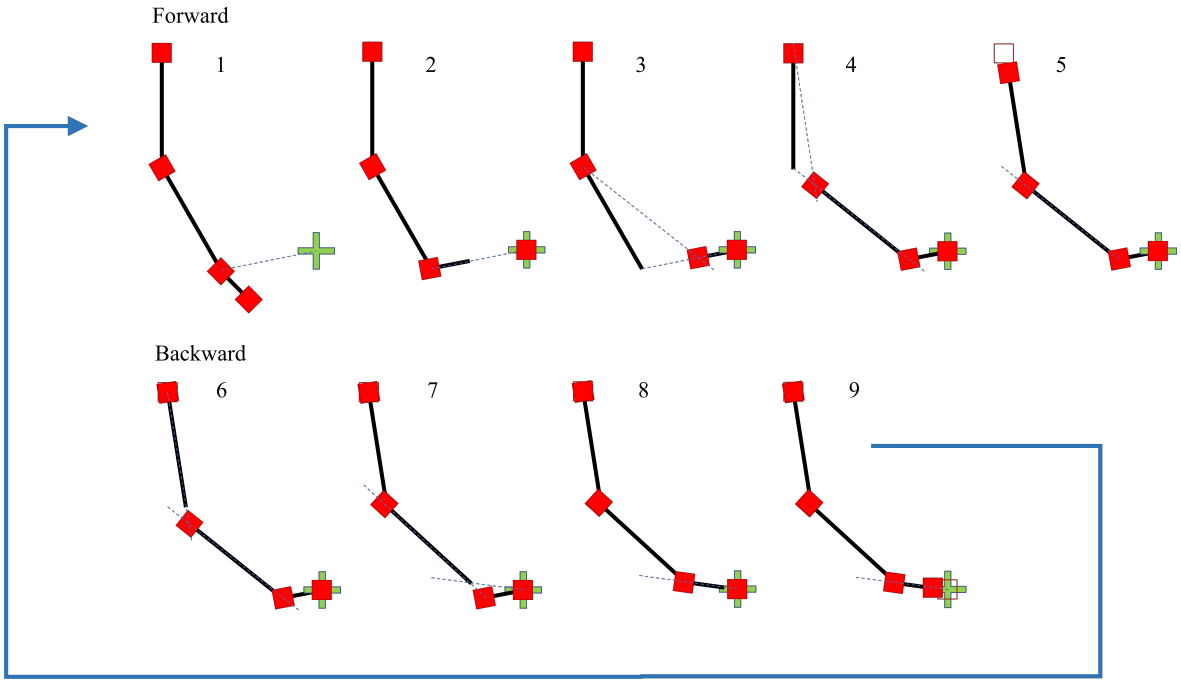


Figure 2.11: *Backward and forward phases of FaST IK algorithm*

Backward phase: Given the target position, this will be the new position ($p_{0_{(i+1)}}$) of the most distal zero-length bone, bone 4, the knuckle. Then the direction starting from the position of bone 3 is calculated and bone 3 is placed along that direction at a distance equal to the length of bone 3. This is the new position of bone 3. One proceeds up to bone 0 always considering for the calculation of the vector connecting bone $n-1$ to bone n, the position of bone n resulting from the calculation at the previous step $p_{n_{(i+1)}}$. This leads to a bending of the arm towards the target but there is the problem that, stopping here, the base of the arm is displaced from its starting position and this must not happen. To overcome this, the same procedure is repeated but in reverse, from the base to the target, from bone 0 to bone 3, the forward phase.

Forward phase: place bone 0 in its starting position (new p_{0_i}) and orientate it according to the direction that connects the point where it is with the new position of bone 1 (now $p_{1_{(i-1)}}$ and place bone 1 at a distance equal to bone 0 length. Then, in a reverse way than before, the position of bone n is defined by the direction that connects bone $n-1$ in its new correct position ($p_{(n-1)_i}$) with the position of bone n resulting from the previous calculation ($p_{n_{(i-1)}}$) that is updated with the method described for the case of bone 0 and 1.

This two-phases process must be repeated a number of times to correctly estimate the proper position. This parameter has been chosen as 10 but can be modified. The backward part is more complex because you start with the highest index bone, the most distal and decrease the index until you get to the most proximal. The forward part is easier because it is not set on the target but on the base, which is fixed; the code is the same but in the opposite direction, from the most proximal to the most distal Fig.2.11.

With these lines the model bends following an inverse kinematics without any constrain on the DoF of the bones, moving in space the target the chain bends without any criterion inherent to ROM of the joints that are vaguely similar to those of a human arm. In order to overcome this problem, a pole can be inserted, i.e. a body towards which the chain can be extended when bending. Apparently this may not be necessary in this case because the movement of the target in a plane naturally rotates all the bones around its axis perpendicular to the plane on which the motion of the target lies. The absence of a z-component in the movement of the target reduces the whole thing to a 2D problem. The logic of the "pole" in this algorithm consists in making sure that, given a body defined as a pole, movable in space, the bones maintain a minimised distance from the pole. As far as rotations are concerned, the bone corresponding to the end effector will have the same rotation as the target if it should rotate as well as translate (in the experimented case it was limited to translating), the other bones will have a rotation consequent to the position of the bone following them. In this the model replicates the functioning of a human limb. (For the algorithm see Appx.A.1)

Once the algorithm is ready to move the kinematic chain the next step was getting, during the real time simulation, the istantaneous z-euler angle of every joint. theese three angles correspond to θ_1 , θ_2 and θ_3. The three values were converted in a percentage of the relative joint ROM so they could be fit the mathematical configuration (2.1.1) to be interpretated by the animator wich shows, instant-by-instant, the three animations combined to forming the real time changing configuration of Virtual Hannes Arm that moover following the target movement

2.3.2 Recursive approach

In the calculation of direct kinematics, the solution of the problem is unambiguous. Given a kinematic chain with n joints and n DoF, knowing the angular position of each joint gives the position of the end effector sec.1.3. The direct kinematic problem is even easier in the case under study, a manipulator that can be modelled as a planar 3DoF manipulator Fig. 1.9 .

For the inverse kinemtic problem the situation changes. In fact, in order to pass from the position/orientation of the end effector to the joint angle values, there is no technique which, when applied, gives a unique solution.
The solution obtained is not unique, it is possible to have:

- No solution if you start from a point outside the working space;
- A finite set of solutions;
- Infinite solutions

In addition, since the manipulator is redundant (with redundancy $r = m - n = 1$ with $m = 2$ and $n = 3$) it is not possible to invert the Jacobian to go from the formulation $\Delta q = J\Delta p$ to $\Delta p = J^{-1}\Delta q$. In the non-redundant case the Jacobian matrix is a 2×2 and therefore invertible. In the present case, the Jacobian is 2×3.
The pseudo-inverse method was used to solve this problem:

$$\Delta q = (JJ^T)^{-1}J^T\Delta p = J^\sharp\Delta p \tag{2.1}$$

This method allows a formulation of the type: $\Delta q = J^\sharp\Delta p$ where Δq is 3×1, $J^\sharp$ is 3×2 and Δp is 2×1. It allows to obtain through easy calculations the angle values, starting from a known initial configuration of the robot, depending on the varying position of the end effector.

The advantage of this method is the very low computational weight and that it follows the least squares method, so that when passing from a configuration C_i to one C_{i+1} the joints will change their angular co-ordinate by the minimum necessary to perform the movement. This characteristic satisfies the problem of infinite solutions, in fact, after imposing an initial configuration known at the beginning of the process, from then on, the kinematic chain moves in relation to the end effector according to the calculations and we have the certainty that, among the infinite solutions to pass from the configuration C_i to C_{i+1}, the algorithm will give an output of only one. The starting point of the algorithm is a known configuration to be imposed on the kinematic chain so that the initial position of the end effector can be calculated by direct kinematics. In the development phase the initial configuration is chosen arbitrarily with the care that this falls within the working space. In the future, when the algorithm would have practical application with the prosthesis, the initial configuration will be taken as input by reading the angular values provided by the encoders or by the IMU at the moment of switching on the prosthesis.

From theory (sec. 1.3) it is known that the Jacobian formula is:

$$J = \begin{bmatrix} \frac{\partial x_p}{\partial \theta_1} & \frac{\partial x_p}{\partial \theta_2} & \frac{\partial x_p}{\partial \theta_3} \\ \frac{\partial y_p}{\partial \theta_1} & \frac{\partial y_p}{\partial \theta_2} & \frac{\partial y_p}{\partial \theta_3} \end{bmatrix} \tag{2.2}$$

Specifically in this case:

$$\mathrm{J} = \begin{bmatrix} -l_1\sin\theta_1 - l_2\sin(\theta_1+\theta_2) - l_3\sin(\theta_1+\theta_2+\theta_3) & -l_2\sin(\theta_1+\theta_2) - l_3\sin(\theta_1+\theta_2+\theta_3) & -l_3\sin(\theta_1+\theta_2+\theta_3) \\ l_1\cos\theta_1 + l_2\cos(\theta_1+\theta_2) + l_3\cos(\theta_1+\theta_2+\theta_3) & +l_2\cos(\theta_1+\theta_2) + l_3\cos(\theta_1+\theta_2+\theta_3) & +l_3\cos(\theta_1+\theta_2+\theta_3) \end{bmatrix} \tag{2.3}$$

At this point, the pseudoinverse $J^\sharp = (JJ^T)^{-1}J^T$ is calculated . Once the pseudoinverse is obtained, in order to obtain the vector Δq, containing the angular increments of the joints, it is necessary to know the increments of x and y coordinates.
As well known, the algorithm was developed in the Unity environment, where everything is controlled via $C\sharp$ scripts sec.A.2. Virtual Hannes is positioned in virtual space. The shoulder joint is not prone to translational but only rotational motion and is considered as the reference for the movement of the end effector. The end effector, in the virtual environment, is nothing more than an object (red dot in the Fig. 2.8a) whose position is known, iteration by iteration. Once the initial configuration has been set, the object is moved by the user using the keyboard or 3Dmouse. The successive positions of the end effector are memorised and the distance travelled between one moment

and the next is calculated, thus obtaining the Δx_p and Δy_p required to calculate the angle values, which are then updated. The problem is apparently solved because with the position of the end effector as input, the angle values at each joint are obtained as output.

This method has two main limitations:

- It has no control over whether the end effector remains within the working space;
- Even when the end effector would be within the workspace, it is not possible to predict the joint configuration and constrain it within the ROMs of the Hannes arm joints sec. 2.1.2.

In addition there would be non-conservativity but this is not a major limitation for the objective of the study and possible future application to a prosthetic poly-articulate.

To solve the first limitation, the idea is to create a zone outside which the end effector cannot go. On the basis of the ROMs of the three joints the whole working space was calculated, then the curves delimiting the boundaries of the working space were obtained. The resulting curve Fig.2.8b was inserted in the plane parallel to the sagittal plane in which the end effector is moved. The curve is pruned coherently with the position of the shoulder, defined before as the origin of the reference system. By means of a Unity function it is possible to know whether the end effector is inside or outside the curve. By setting flags, the movement of the end effector is allowed as long as it is inside the curve. When the end effector leaves the curve, it is repositioned to the last position inside the curve, stored in a vector containing the last 5 positions of the end effector. In fact, this acts as a positional control for the end effector that does not move from the shoulder beyond the allowed limits.
Even remaining within the working space, the Jacobian pseudo-inverse does not allow the single joint configuration to be controlled. In order to meet this need, the optimisation of the recursive method was implemented.

From the literature study [44][20][27][32] the optimised form was formulated:

$$\Delta q = J^{\sharp}\Delta p - \alpha(I - J^{\sharp}J)\nabla H \tag{2.4}$$

The α coefficient varies between 0 and 0.5 and allows to modulate how much to make the optimisation weigh, how much to make it affect the result. The core of the additional component is the ∇H, whose formulation is expressed as follows:

$$H_{(\theta)} = \sum_{i=1}^{n} \frac{(\bar{\theta}_{max} - \bar{\theta}_{min})^2}{4(\bar{\theta}_{max} - \theta_i)(\theta_i - \bar{\theta}_{min})} \tag{2.5}$$

$$\nabla H_{(\theta)} = \left[\frac{\partial H}{\partial \theta_1}, \frac{\partial H}{\partial \theta_2}, \ldots, \frac{\partial H}{\partial \theta_n}\right]^T \tag{2.6}$$

where θ_{max} and θ_{min} are the upper and lower limits of the ROM of the individual joint. The introduction of this factor ensures that, in the calculation of the solution, the limits are avoided by redistributing the weights of the three joints movement if one of them approaches to a limit configuration.

Compared to the recursive approach, the optimisation consists in adding a corrective factor to the previous calculation of angle increment Eq. 2.4.

2.3.3 ANFIS

Introduction to ANN

ANN (Artificial Neural Network) are usefull for the analysis of different systems parameters. These kind of networks try to mimic the human reasoning and are inspired to the human biological nervous system [36]. Human's brain can recognize and elaborate data in an easy way, making it possible to classify objects by analysing, for istance, the amount of visual information available. The human brain performs task by exploiting the potentiality of nerve cells called neurons that works together by exchanging information. Artificial network' aim is to resemble this kind of structure so to make it easier to automate processes involving difficult tasks and that requires the implementation of an intelligence [2]. ANN structure is organized in multiple layers, each layer is composed by processing units called neurons. The neurons are the core of the information-processing learning algorithm, they collect the input data and provide to compute the output. The networks can have different characteristics settable through some parameters: the type of functions used to evaluate the performances, the applied rules, the type of computation. All of these are different aspects of the mathematical model of an ANN [2]. The network also needs a training phase that is done by using known input-output data pairs, after the training it can be used to estimate and evaluate unknown parameters.

ANNs may be used to analyse the parameters of many systems; these networks are based on the biological nervous system of humans [2] and attempt to emulate human thinking. The human brain has the ability to detect and construct data in a simple manner that allows it to categorise things by analysing, for example, the quantity of visual information available. The human brain accomplishes tasks by utilising the potential of nerve cells known as neurons, which communicate and exchange information.

Introduction to Fuzzy logic

ANNs can learn from a training data set automatically and discover a fair approximation to link a given input to a projected output [8]. The Fuzzy Inference System is an adaptable intelligent decision-making system (FIS). Fuzzy logic is a type of logic that attempts to replicate the human ability to ascribe a level of truth to propositions. Fuzzy logic proposes to make judgments in an environment of uncertainty and imprecision, thanks to its brief structure, which allows it to capture the imprecise patterns of thinking that are typical of humans [34]. If-then rules are used in fuzzy logic to establish the input-output relationship. This logic allows to assign a value to the assertion to be evaluated, different levels of truth that ranges from 0 and 1.

If-then rules are the heart of the FIS, and they can take many forms. A membership function is a graph that shows how to translate a point in the input space to a membership value between 0 and 1. The FIS structure is made up of several blocks, and a schematic 2.12 is provided to explain how the input is elaborated to produce the final output. Each block of the FIS serves a specific purpose:

- The database identifies the membership functions to be used in the fuzzy rules;
- The rule base contains a certain number of fuzzy if-then rules;

- The decision-making unit performs the operations of inference on the rules;
- The fuzzification interface computes the degrees of match between the crisp inputs and linguistic values;
- The defuzzification interface elaborates and aggregates the qualified consequents of the inference fuzzy results to obtain the crisp output;

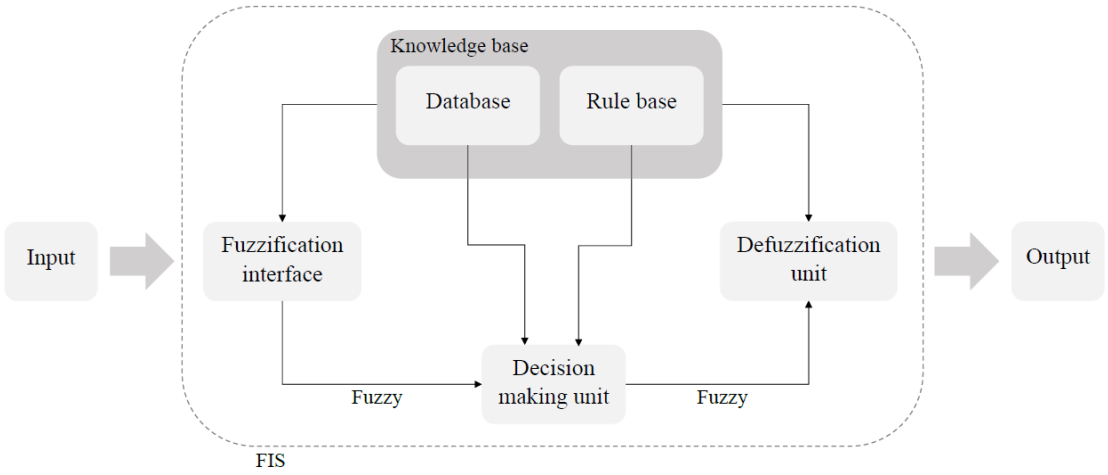


Figure 2.12: *Fuzzy inference system block diagram [34]*

Adaptive-Network-Based Fuzzy Inference System

By merging ANN and FIS techniques and taking use of both, adaptive neuro-fuzzy inference systems (ANFIS) are created. ANFIS is a multilayer feedforward network [34], in which each of the network's nodes performs a separate action on the input. To forecast the output with the least amount of error, ANFIS employs fuzzy if-then rules, which are implemented inside a neural network-like structure, as well as a learning algorithm. The ANFIS learning method is divided into two phases: the first is termed off-line learning and is a mechanism; this phase is a forward pass with least square error evaluation; and the second is a gradient descent approach that uses mathematical techniques such as back propagation [8][9]. To obtain information on the data set, a fuzzy based approach is applied. The ideal parameters for the membership functions may be identified using neuro-adaptive learning approaches to acquire the best FIS tracking for the supplied inputoutput data. A network structure like the ANN [40] can be used to understand the resulting input-output map. Following sufficient training, the layered structure is capable of mapping the system's inputs to the projected outputs. This predictive system may be utilised in a variety of applications (control systems, image processing, decision making), but we're particularly interested in using it to solve the inverse kinematic issue.

ANFIS application for inverse kinematic

ANFIS may be used to analyze robotic kinematic chains and produce the appropriate joint configurations to achieve the manipulator's intended behaviour [1]. In terms of the

significant amount of calculation required, neural networks can be used to overcome the fundamental limits of analytic and geometric solutions [23]. To produce an accurate and repeatable estimated solution, the predictive techniques may be employed to discover an approximation mapping function that can link the posture of the end-effector with the joints configuration [19].

So, after a first attempt with the recursive approach, the ANFIS method has been tested. Carried out in the MatLab environment, the procedure consists of the following steps. Initial parameters were set for the creation of the dataset and the training of the networks. These parameters include:

- three lengths L1, L2 and L3 of the three links of the 2D robot reproducing Hannes Arm, set equal to the lengths taken from the virtual model of Unity;
- the ROMs of the three links equal to those of Hannes Arm in the Unity environment; [Reference earlier]
- vector containing three step options with which to increase the angles of the three joints for the creation of the dataset;
- vector containing the initial FIS values for training;
- vector containing the number of epochs for training;

```
step=[10 5 2];
infis=[8];
epoch=[10 50 100 200 300];
%dimensioni hannes virtuale
l1=12.74508; %arm length
l2=12.04653; %forearm length
l3=4.355566; %hand length
```

Once the initial parameters have been set, the creation of the dataset.Through a loop a data pool is made (this procedure has been performed three times, one for each different step of variation of the angles). All the possible configurations are calculated by varying, according to the step, the angles of the three joints between the limits of the respective ROMs 2.1.1. Then, by means of direct kinematics calculations, the x and y positions attainable by the end effector are calculated, given those joint configurations. Then three three-dimensional matrices are formed containing each possible x and y coupling and its associated θ_1, θ_2 or θ_3 value.

Once the dataset is created, the networks are trained to obtain 3 different vectors containing an ANFIS network for each of the three joints. Consequently, having 7 possible values of initial fis, 6 possible values of number of epochs and 3 steps of angle, leading to 3 datasets, from the training come out 126 tris (one for each joint) of ANFIS functions. This means 126 cases resulting from different combinations of factors (dataset, initial fis and number of epochs) for the calculation of the ANFIS networks.

Next, the trajectory on which to test the networks is created. A curved trajectory of 100pt was chosen, calculated using direct kinematics, which is therefore certain to be

within the working space. Then, for each of the 126 combinations, inverse kinematics is performed starting from the known X and Y values of the trajectory and feeding them to the ANFIS networks to calculate θ_1, θ_2 and θ_3 values and the three resulting angular trajectories. Given the estimated angles, saved as vectors, direct kinematics is performed again to display the trajectory composed of the estimated end effector positions. The two trajectories are compared by calculating 15 parameters chosen to evaluate the goodness of fit of the i^{th} ANFIS network. These parameters are:

1. Maximum end effector x position error
2. Minimum end effector x position error
3. Maximum end effector y position error
4. Minimum end effector y position error
5. Maximum θ_1 error
6. Minimum θ_1 error
7. Maximum θ_2 error
8. Minimum θ_2 error
9. Maximum θ_3 error
10. Minimum θ_3 error
11. RMSE of x
12. RMSE of y
13. RMSE of θ_1
14. RMSE of θ_2
15. RMSE of θ_3

Observing the comparison of the evaluation quality parameters of all the different ANFIS function a choice has been made of the best and worst ANFIS function above all sec. 3.2

In order to provide an instant-by-instant visualisation of the configuration of the arm whose end effector was executing the calculated trajectory, a 2D robot was created with the same proportions as the virtual Hannes Arm. The robot was made to traverse the trajectory by imposing, instant by instant, the configuration of the three joints. This was done to observe that the angles contributing to the formation of the trajectory of the end effector were also consistent with a human like movement.

Once the algorithm was developed in the MatLab environment for trajectory estimation, the next step was an attempt to bring it into the virtual environment of Unity. In Unity, unlike MatLab, there are no open source libraries or functions that allow one to build ANFIS networks, and creating them from scratch immediately seemed a major obstacle. During the attempts in this direction, the continuation of the study of the recursive approach 3.2 led to go back to that path.

Chapter 3

Results

The results of the experiments described in section 2.2 are reported in this chapter. The three algorithms described were evaluated individually and then compared.

3.1 Fast IK

There are no analytical results of the first (Fast IK) because the method was discarded almost immediately for obvious limitations related to the impossibility of setting costrains of any kind and because the consequent movement strategy was profoundly different from a human like. As can be seen in Fig. 2.11 the first segment which moves in reaching a target is the most distal one. The most distal joint is the one that performs greater angular excursions. In human movements, considering the three joints (shoulder, elbow and wrist) the two more proximal joints are the ones to move more in the cinematic of the arm [31][45].

3.2 ANFIS

The ANFIS method required a preprocessing phase of tens of hours of calculation to study all the combinations of parameters (see sec. 2.3.3) in order to extract the best ANFIS functions. These parameters can be viewed in Tab. from B.1 to B.6. Are shown all parameters combinations for each ANFIS network tested and the relative results. An overall assessment was made of all 126 ANFIS rules and the calculated parameters, as said in sec. 2.3.3, for the comparison between them, some more significant parameters were chosen, the trends of which can be seen when the three characteristics of each anfis function (angle step, initial fis and number of epochs) vary. The following figures show the trends of maximum error of the x-coordinate, maximum error of the y-coordinate, RMSE of the x-coordinate and RMSE of the y-coordinate of the predicted trajectory with respect to the desired one. As the best ANFIS network from the performance point of view, number 117 was identified, which was then used to be compared to the recursive approach. In the following figure is shown the comparison between trajectories predicted by the ANFIS rules with best and worst performances upon the calculated ones. In the following tables are shown the quality parameters to evaluate the quality of the ANFIS rules appeard as the best and worst one. Parameters relative to three

trajectories predicted by the ANFIS rules.

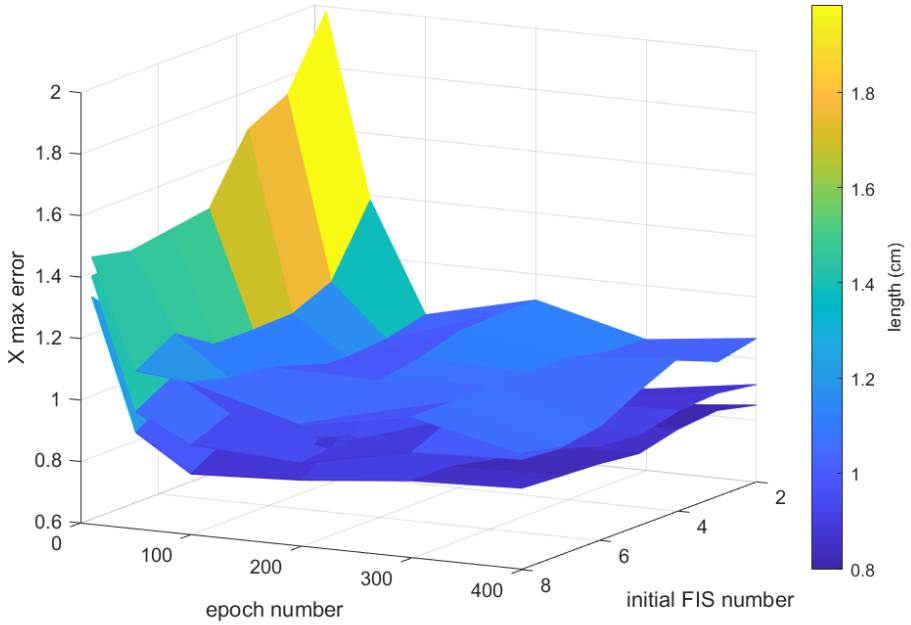


Figure 3.1: *Max error on X coordinate in ANFIS evaluation*

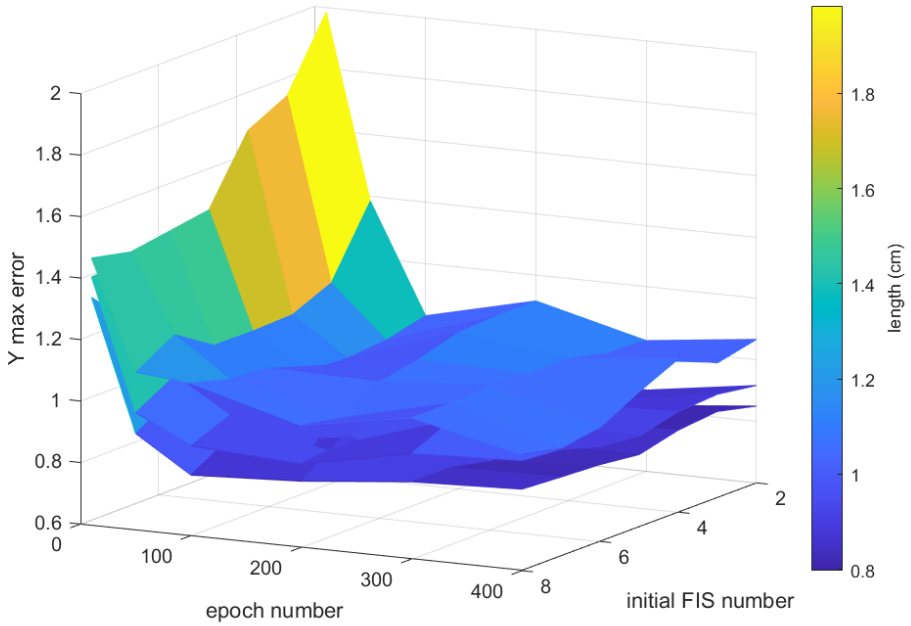


Figure 3.2: *Max error on Y coordinate in ANFIS evaluation*

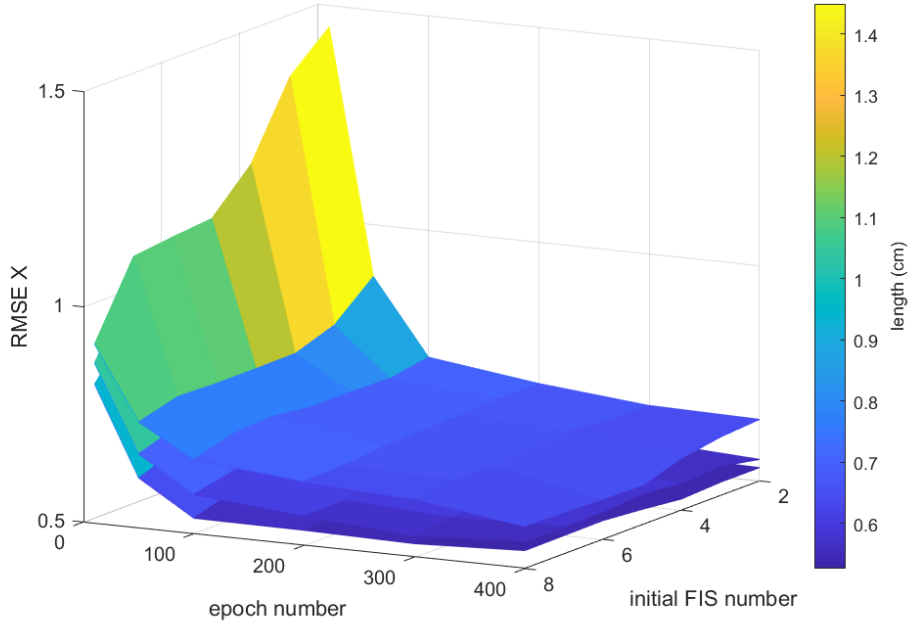


Figure 3.3: *RMSE on X coordinate in ANFIS evaluation*

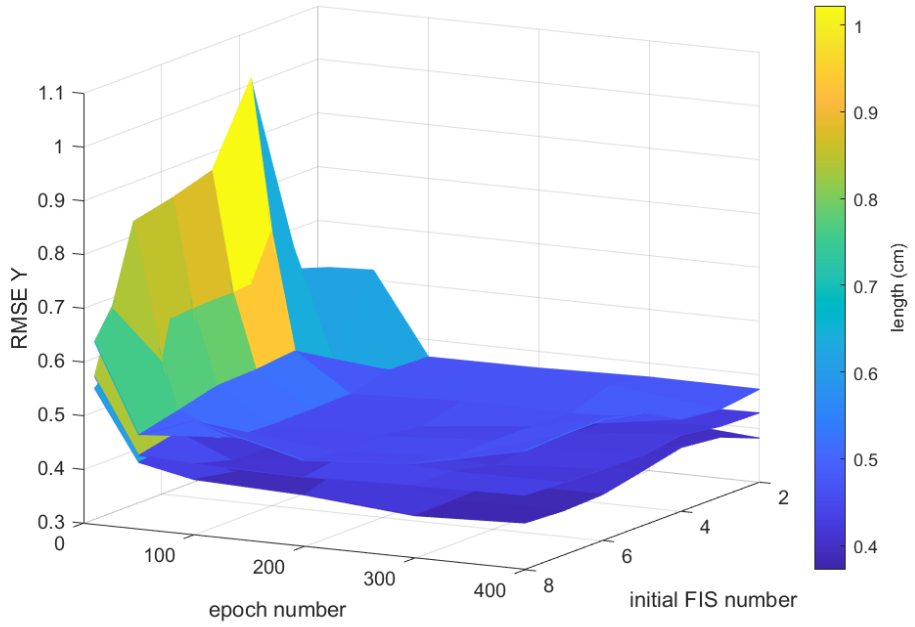


Figure 3.4: *RMSE on Y coordinate in ANFIS evaluation*

3.3 Comparison between ANFIS and recursive

In the following figures it is possible to observe the results obtained by the three methods (ANFIS, Recursive and Optimized Recursive) in the attempt to reproduce the three trajectories given as input through the calculation of the inverse kinematics,

	RMSE x [cm]	RMSE y [cm]	MaxError x [cm]	MaxError y [cm]
ANFIS best	0.552	0.404	0.856	0.636
ANFIS worst	1.432	0.582	1.968	1.173

Table 3.1: *Quality parameters in ANFIS comparison - curve case*

	RMSE x [cm]	RMSE y [cm]	MaxError x [cm]	MaxError y [cm]
ANFIS best	0.502	0.351	0.649	0.616
ANFIS worst	0.909	0.649	1.507	0.840

Table 3.2: *Quality parameters in ANFIS comparison - vertical case*

	RMSE x [cm]	RMSE y [cm]	MaxError x [cm]	MaxError y [cm]
ANFIS best	0.594	0.307	0.948	0.515
ANFIS worst	0.776	0.545	1.428	1.112

Table 3.3: *Quality parameters in ANFIS comparison - horizontal case*

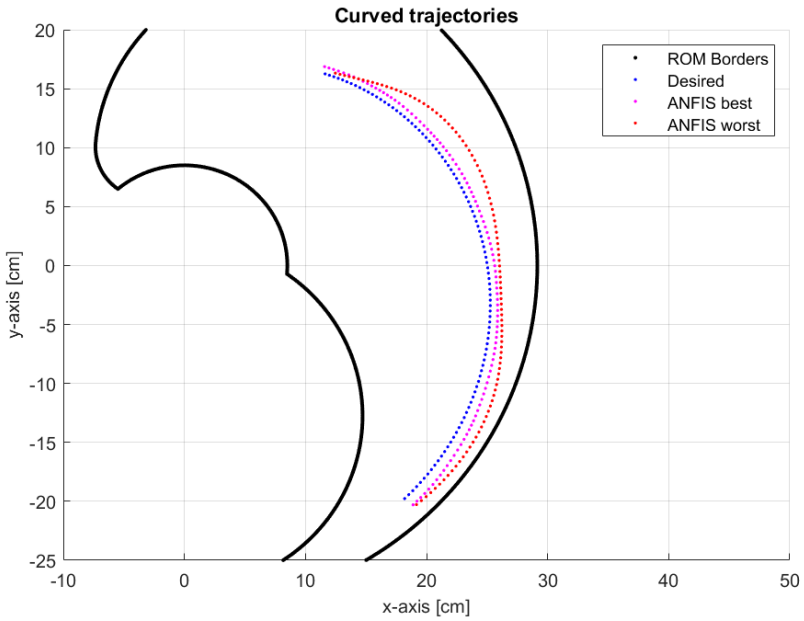


Figure 3.5: *Comparison between best and worst ANFIS function - curve case*

obtaining the three joint angles and subsequent calculation of the position of the end effector of the kinematic chain schematized Fig.2.9. In Fig.3.8 we observe the results for the curved trajectory. The image on the top represents a comparison between different algorithm predicted trajectories, more easily observable in Fig.3.11. The following 9 images show the trends of the angles of the three joints, in the three different algorithm cases, which resulted in the tracking of the predicted trajectories.

In Fig.3.9 we observe the results for the curved trajectory. The image on the top represents a comparison between different algorithm predicted trajectories, more easily observable in Fig. 3.12. The following 9 images show the trends of the angles of the three joints, in the three different algorithm cases, which resulted in the tracking of

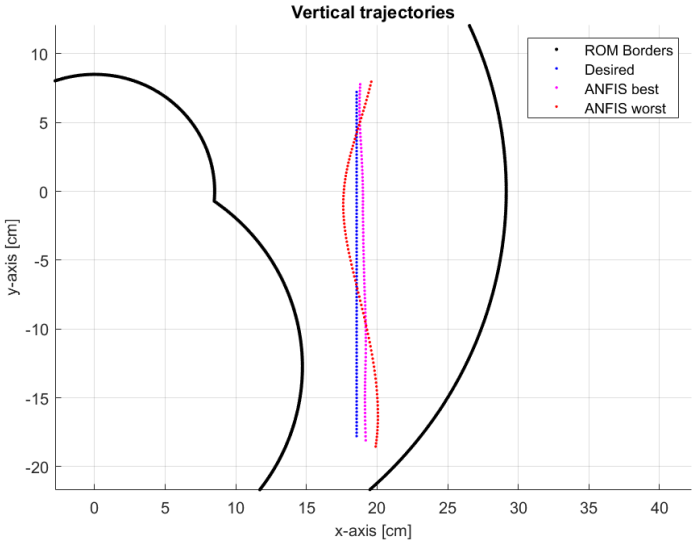


Figure 3.6: *Comparison between best and worst ANFIS function - vertical case*

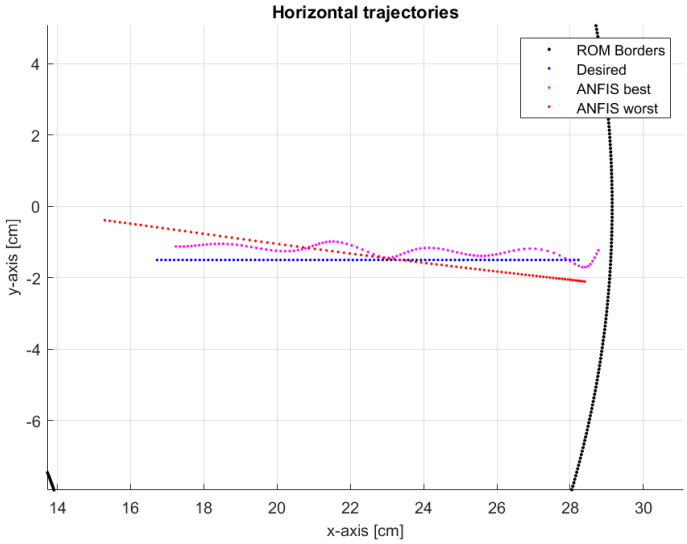


Figure 3.7: *Comparison between best and worst ANFIS function - horizontal case*

the predicted trajectories.

In Fig.3.10 we observe the results for the curved trajectory. The image on the top represents a comparison between different algorithm predicted trajectories, more easily observable in Fig.3.13. The following 9 images show the trends of the angles of the three joints, in the three different algorithm cases, which resulted in the tracking of the predicted trajectories.

Tables 3.4, 3.5, 3.6 lists the parameters calculated to evaluate the characteristics of the different methods. The parameters chosen are RMSE of the x-coordinate, RMSE of the y-coordinate, span of the shoulder, elbow and wrist angle in flexion-extension.

	RMSE x [cm]	RMSE y [cm]	θ_1 span [deg]	θ_2 span [deg]	θ_3 span [deg]
ANFIS	0.552	0.404	69.734	54.297	5.414
Rec	0.223	0.050	67.324	57.986	11.387
Rec Opt	0.479	0.146	65.373	50.773	50.486

Table 3.4: *Quality parameters in algorithms comparison - case curve*

	RMSE x [cm]	RMSE y [cm]	θ_1 span [deg]	θ_2 span [deg]	θ_3 span [deg]
ANFIS	0.502	0.351	46.228	48.894	1.463
Rec	0.113	0.005	46.116	41.821	14.636
Rec Opt	0.067	0.078	44.807	45.045	16.112

Table 3.5: *Quality parameters - case vertical*

	RMSE x [cm]	RMSE y [cm]	θ_1 span [deg]	θ_2 span [deg]	θ_3 span [deg]
ANFIS	0.594	0.307	56.238	89.313	14.241
Rec	0.025	0.012	50.373	74.562	23.633
Rec Opt	0.024	0.015	50.440	74.185	25.476

Table 3.6: *Quality parameters - case horizontal*

3.4 Experiment with physical setup

Below are shown the trends of the trajectories imposed by the controlled end effector in the virtual environment and that performed in the real world by the physical setup. Figure 3.9 concerns the results for the vertical trajectory. On the left side of the figure are the two trajectories, the one resulting from the three angles from the inverse kinematics calculations with the optimised recursive method imposed as input and the one resulting from the three angle values read by the encoders placed on each Hannes arm joint. The encoder values represent the real configuration reached by the Hannes arm during the test

Under the xy trajectory are presented the angular trajectory, one for each joint, with the comparison between the reference sent to the joint and the value read from the encoder explaining how much the motor accomplish the task imposed by the reference. Figure 3.10 concerns the results for the vertical trajectory. At the top of the figure are the two trajectories, the one resulting from the three angles from the inverse kinematics calculations with the optimised recursive method imposed as input and the one resulting from the three angle values read by the encoders placed on each Hannes arm joint, also in this case the orange trajectory is the one executed by deh physical setup in dependence of the angle reached by each joint. On the right of the xy trajectory, also in this case, are presented the angular trajectories, one for each joint, with the comparison between the reference sent to the joint and the value read from the encoder explaining how much the motor accomplish the task imposed by the reference.

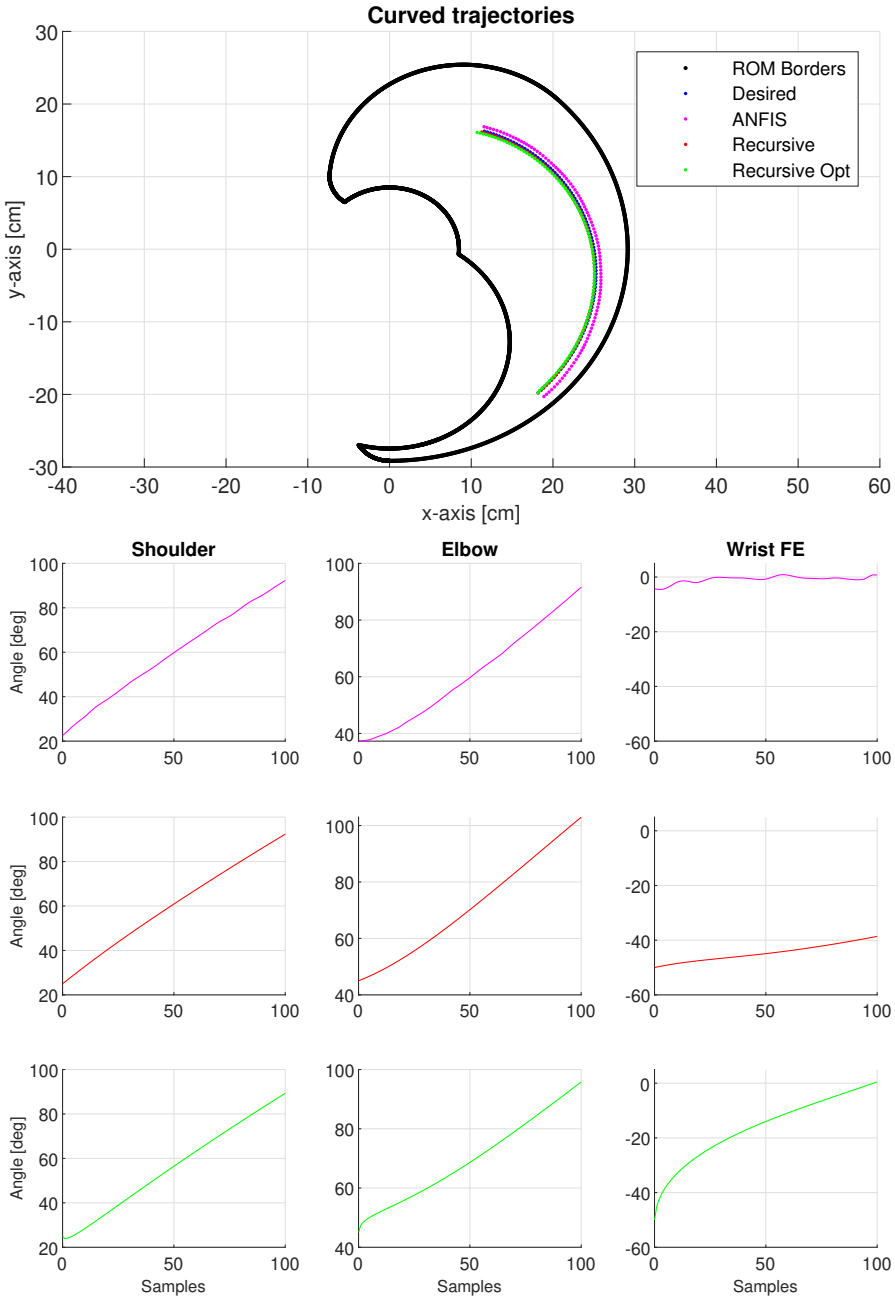


Figure 3.8: *Results for curve trajectory*

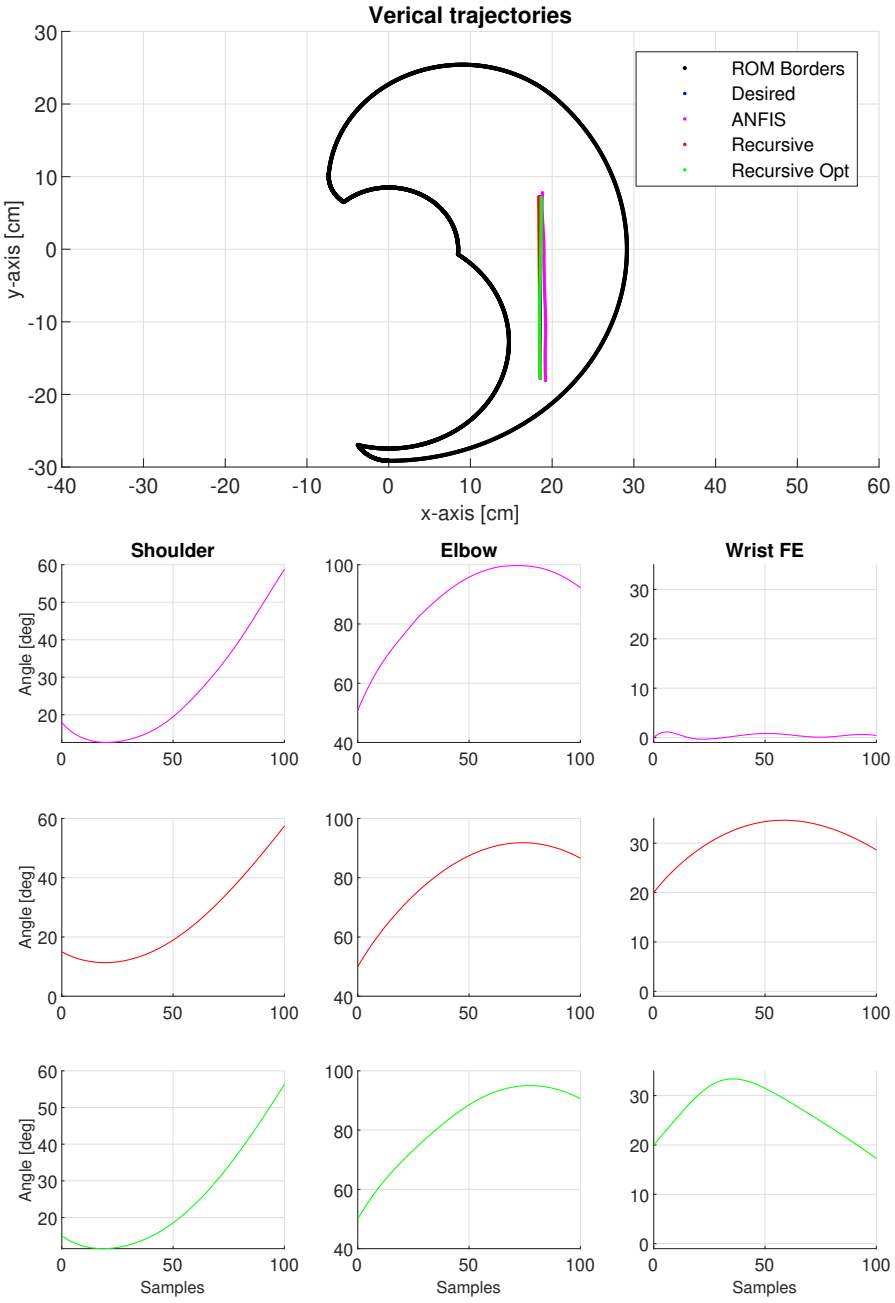


Figure 3.9: *Results for vertical trajectory*

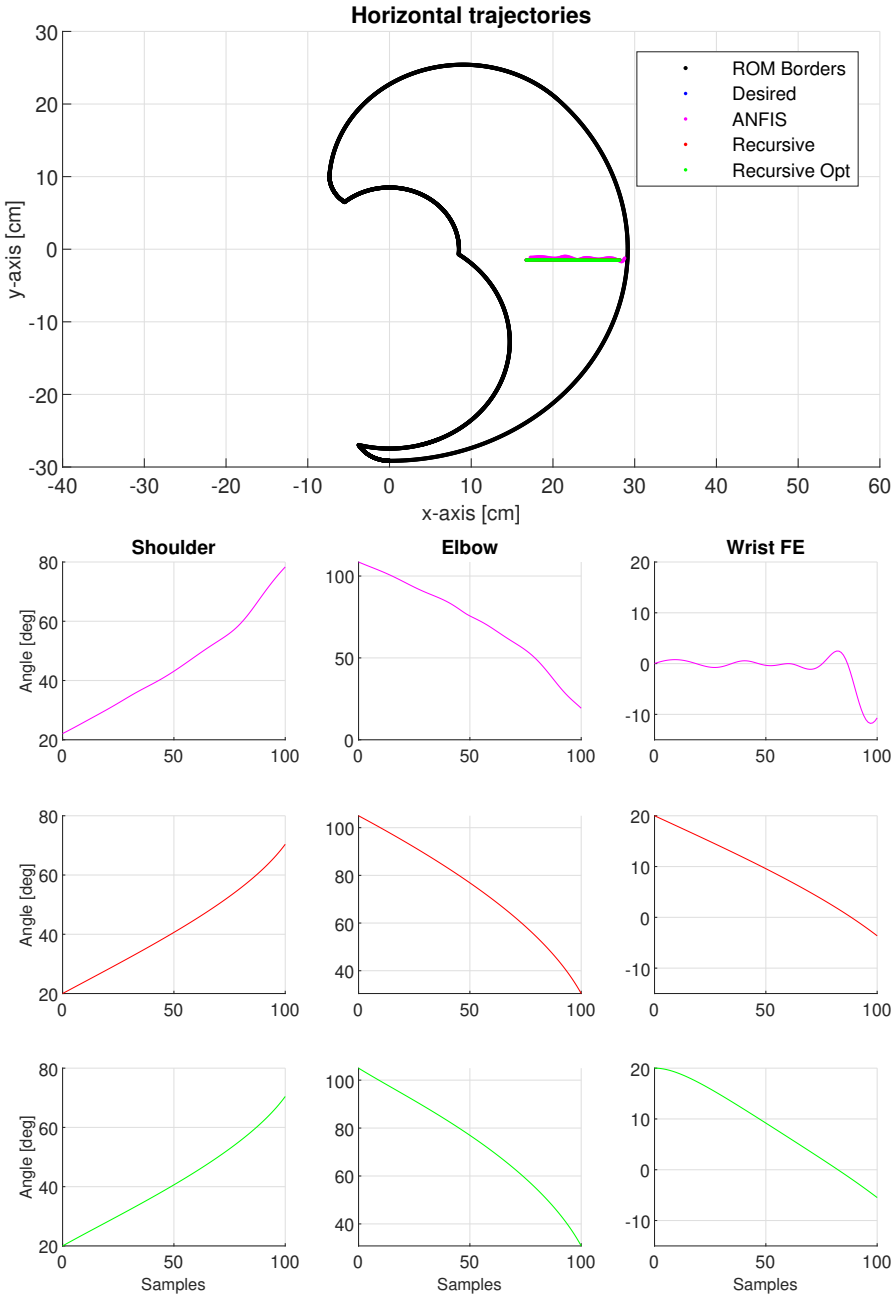


Figure 3.10: *Results for horizontal trajectory*

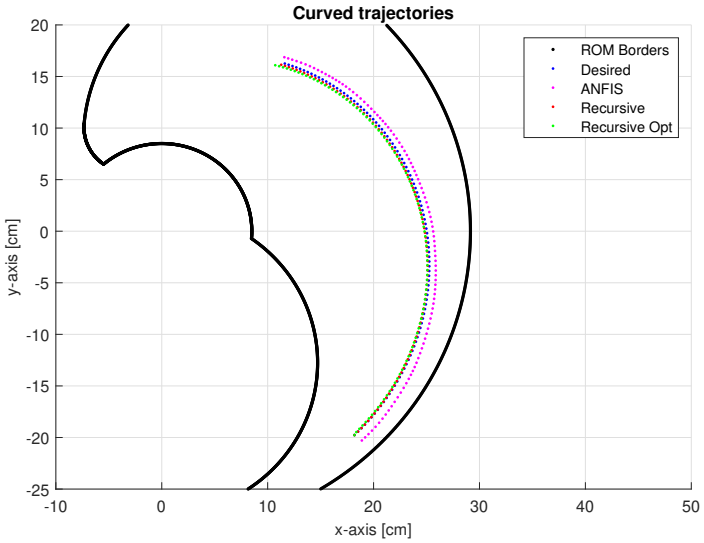


Figure 3.11: *Focus on desired and calculated trajectories - curve case*

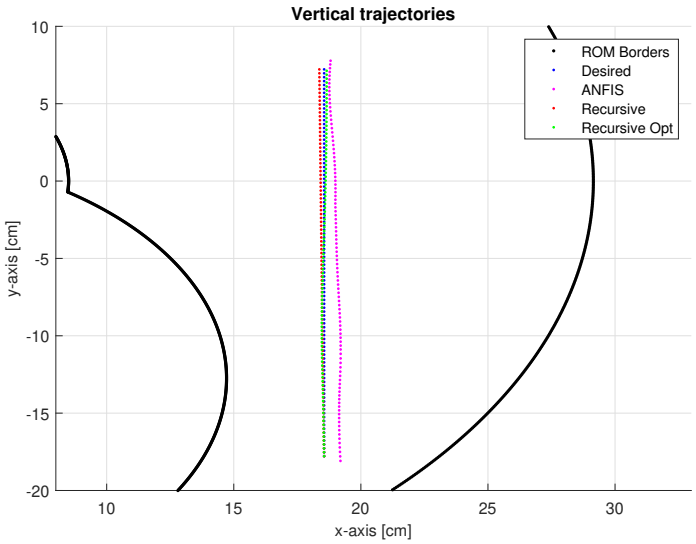


Figure 3.12: *Focus on desired and calculated trajectories - vertical case*

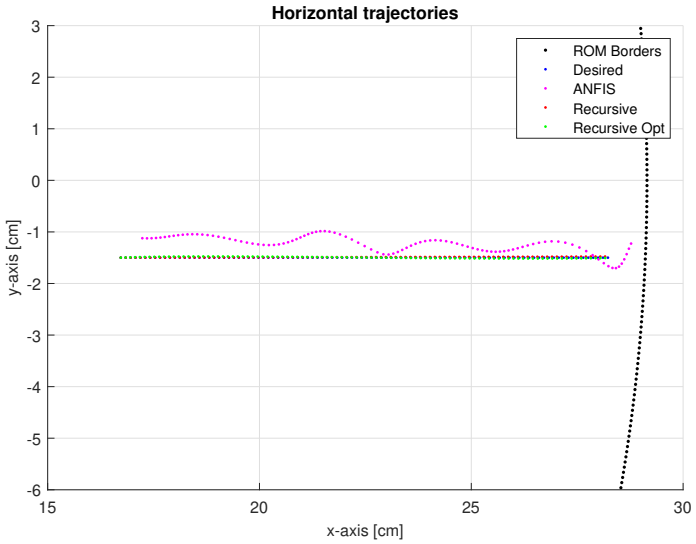


Figure 3.13: *Focus on desired and calculated trajectories - horizontal case*

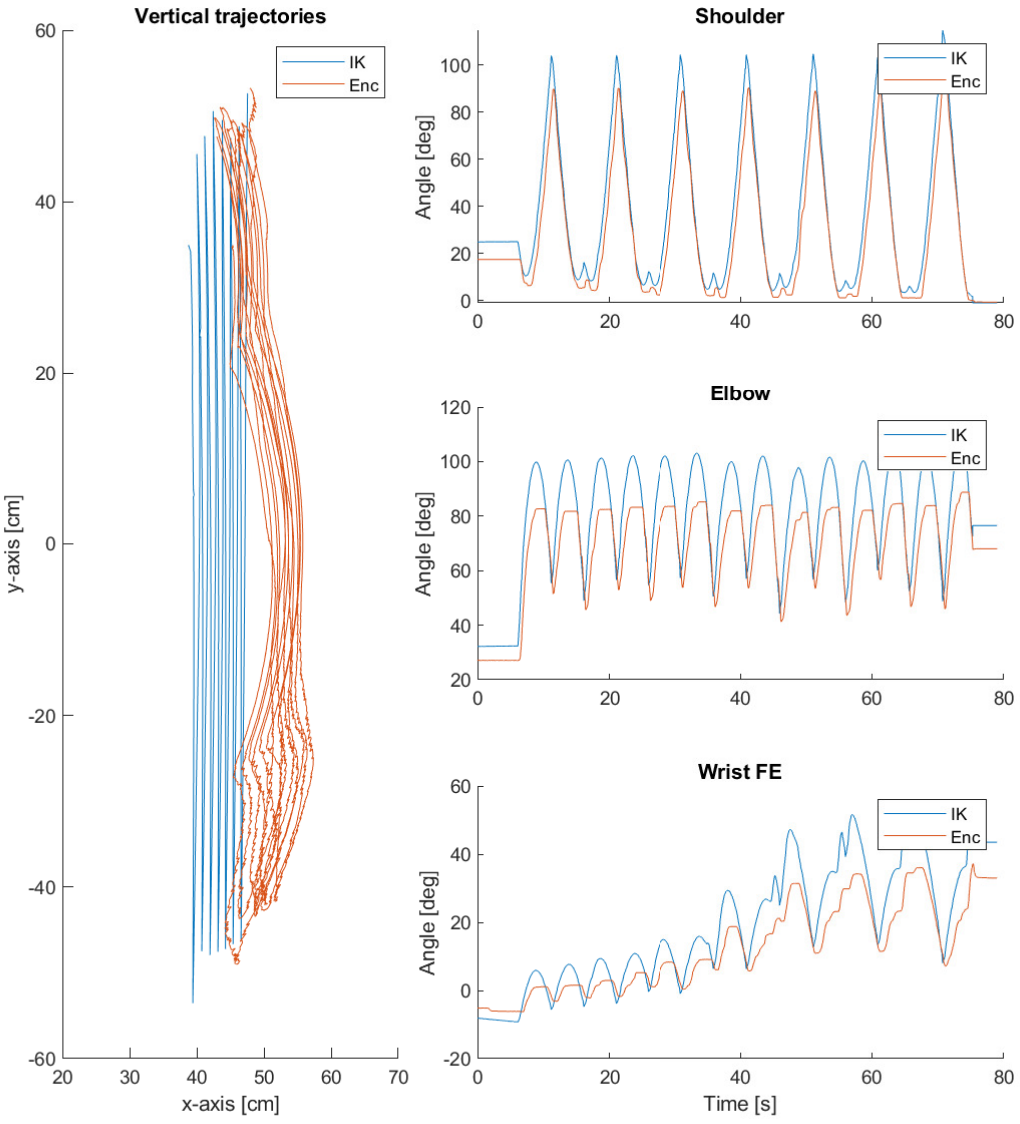


Figure 3.14: *Fuzzy inference system block diagram [34]*

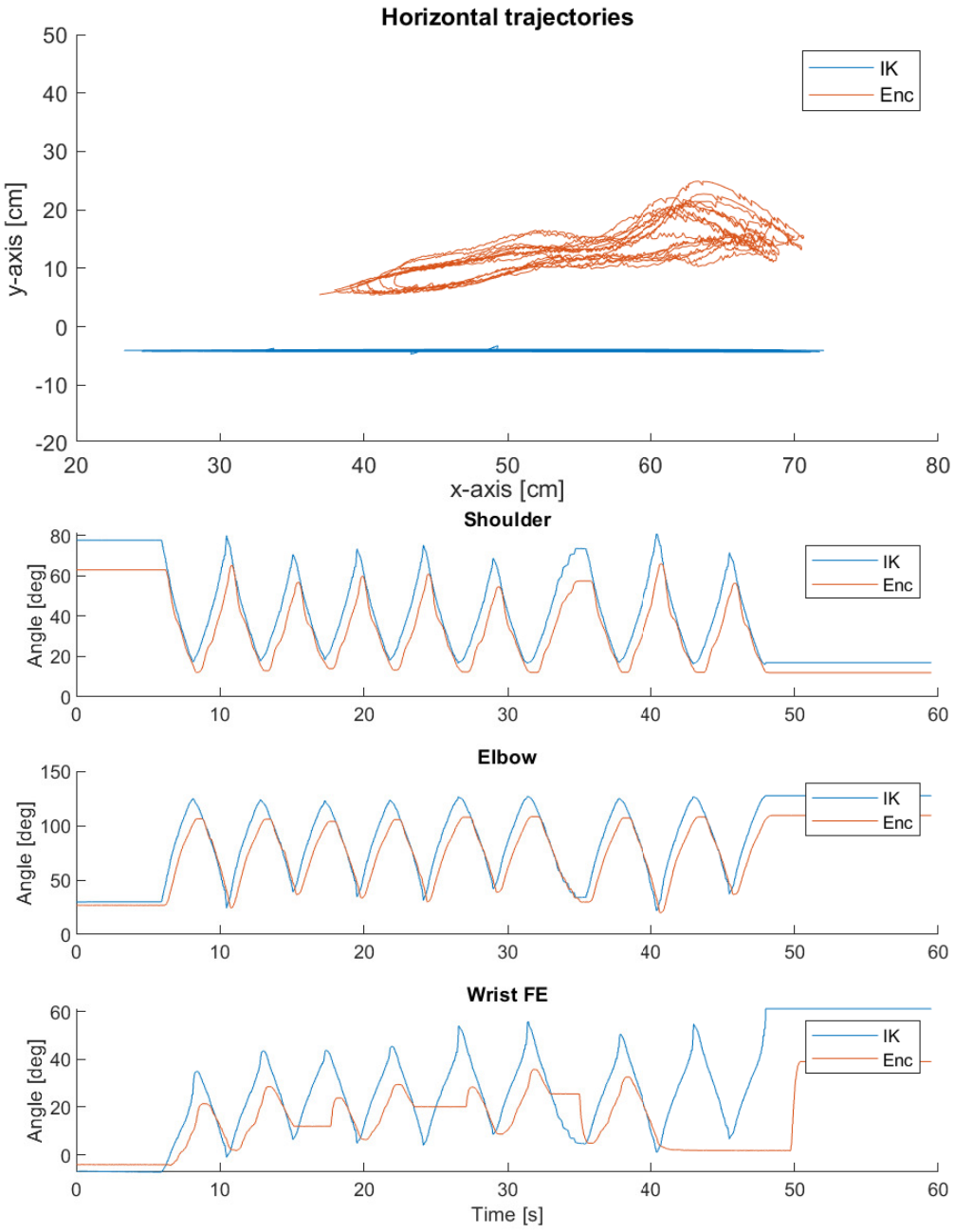


Figure 3.15: *Fuzzy inference system block diagram [34]*

Chapter 4

Discussion

As already mentioned, the first algorithm tested was the FAST IK, sec.2.3.1. The name given is not a coincidence since, at the beginning of the study, the approach was very intuitive, looking for a method that would allow bodies to be moved easily in the Unity virtual environment, paying less attention to the analytical part of the problem. The convenience of a high level environment, like Unity virtual reality, was inherently a limitation in the development of the first method because the high level, while allowing immediate use and graphical rendering, did not permit to delve into the analytical part of the kinematics and define the calculations. Having a profound knowledge and control above the math behind the algorithm is the real need for enriching it with specificity and robustness.
FAST IK is a geometric approach based on the characteristics and possibilities of manipulation and use of virtual objects in Unity. The abandonment of this method is also probably due to the lack of familiarity with the Unity world during the first weeks of work. In fact, we turned towards analytical solutions, whose basic principles were better known, such as the kinematic theory of serial manipulators [25][21], for the reasons just explained but also to work in a more known field; this due to the notion background coming from university study. A window remains open for higher level approaches to kinematics, since future developments of the project will make use of Unity Assets for modelling serial arms and their kinematics, born to be used for the development of video game material but with a good margin of applicability for the goal of study.

For other approaches, there is certainly more data to discuss. Regarding the evaluation of the 126 triplets of ANFIS functions created (App.B), it is indicative to observe the comparison between the one considered worst and the one chosen as best. In all three cases of trajectory studied (Tab. 3.1, 3.2, 3.3) the position error of x and y coordinates is approximately double for the worst case compared to the best case. It can be seen that in the horizontal trajectory there is less precision over the y-coordinate while for the vertical trajectory there is less precision over the x-coordinate among the various positions of the end effector. Also for the RMSE values there is a substantial difference between the best and the worst case. The best solution returns quite good results, which are then compared with those of the other algorithms, on the same trajectories.

As mentioned above, the choice of the best ANFIS function was made by assessing quality parameters. The trend of these values, as the three parameters describing the characteristics of the ANFIS network (size of the dataset, Initial FIS and number of epochs of the train), change and can be observed in the figures Fig.3.1, 3.2, 3.3, 3.4. It can be seen that as the number of initial FIS, the number of epochs and the number of elements in the dataset increase as the parameters quality improves. In every graph, each surface shows the values of the parameter under consideration, relative to the train on a different dataset. It can be seen that for the same number of INFIS and number of epochs, a larger dataset corresponds to better parameters (lower maximum errors on x and y and lower RMSE values). The same is true for the other two parameters, but note that the most influential is the number of epochs in the first phase. In the transition from a few tens to a little more than 100 epochs the improvement is substantial, above 150 epochs, with the same dataset and INFIS the parameters continue their trend but slowly. Considering the INFIS number parameteer, it can be said that after 5 INFIS there is no longer a substantial progressive trend in the quality of the ANFIS functions.

With reference to the results that the different algorithms given in the estimation of the three trajectories studied, it can be said that the recursive approach and the optimized recursive give the best results in terms of accuracy. The optimization introduces a worsening in this sense but it is a datum coherent with the expectations because as explained in the section sec.2.3.2 its objective was to comply with the problem of the recursive approach of not allowing the control on the single joint and even less to set a priori a range within which the solution of the inverse kinematics must give the values of each angle. This therefore results in a slight inaccuracy with respect to the pure recursive method, an acceptable inaccuracy given the introduced benefit of being able to mediate the joint motion weights in such a way as to remain far from the limit configurations (ROM edges). It can be seen that (Tab. 3.1, 3.2, 3.3) ANFIS and Recursive approach present angle spans for each joint more consistent with a human like motion, with the shoulder and elbow joints much more involved than the wrist in reaching movements [31]. The results obtained are consistent with expectations confirming the validity of the approaches tested for the calculation of the inverse kinematics of the prosthetic serial polyarticulate for the purpose of a Cartesian control. All this at least in theory, in virtual reality with a whole series of approximations and ideals set as for example the reduction of the problem to purely kinematic and not dynamic, as differently is in the real case of control of a robot.

The next step is the evaluation of the applicability of such algorithms to the real case. With regard to the analysis of the trajectories of the end effector, resulting from giving input angles to the joints, Fig. 3.14 and 3.15 both vertical and horizontal, it appears that there are discrepancies between the ideal trajectory of an end effector of a ideal kinematic chain that would follow the imposed angular trajectories and the one actually followed. This according to the angular measurements of the encoders that recorded the actual angular configurations reached by the joints. These discrepancies, however, are reasonable and more than acceptable considering the fact that we are talking about a prosthetic robot (a prototype in this setup Fig.2.2a) that has lengths

in the order of tens of centimeters chained together through the active joints in a polyarticulate of more than 80 cm. So a discrepancy of about 10cm is more than acceptable. One more reason for this acceptability is the fact that, analyzing the plots of the angular trajectories, it is possible to notice that there are propagation errors due to intrinsic errors in the control of the single joint; because everything depends on how the dynamics of the single joint is at a mechanical level and how the control makes sure that the reference is followed on each single joint (mechatronic side). In fact, analyzing the angular trajectories of the vertical study case Fig. 3.14, it emerges that for the shoulder the reference (IK) and the measurement (Encoder) are consistent. Slightly inferior thing happens for the elbow in terms of performance goodness of the reference control and the Proportional-Integrative-Derivative (PID) that runs underneath, because in the elbow it's observable how there is a saturation, probably due to the poles and zeros of the system that make it unable to deliver all the force necessary to follow the trajectory imposed by the inverse kinematics. In fact, we can see a saturation around 80-90 degrees, angle at maximum excursion in terms of torque, the most unfavorable configuration in terms of leverage considering that the elbow also perceives the weight of the hand. The elbow, as it is designed, has adjusters for gravity compensation that are not perfectly optimised for the case studied. This can justify why the engine can not accomplish the performance for which the reference is followed. Another relevant aspect is considering that the battery pack with which the system was powered was the standard battery pack for Hannes arm, optimized for wrist and hand control and here instead used to move also shoulder and elbow. So even if mechanically or in an ideal condition the motor could have followed the trajectories, with a battery pack not performing at its maximum, it cannot deliver the necessary torque. Regarding the wrist in flex-extension, in the vertical trajectory Fig. 3.14 the profile is not followed in an optimal way but it seems to be respected enough even if with the same evaluation criterion as before. Battery in trouble and unfavorable leverage.In the vertical case it can be noted that even the trajectory given as input is not perfectly vertical and repeated identical to itself although the input given in the trajectory calculation phase was. This aspect can be traced back to the problem of the non-conservativeness of the control algorithm sec.A.2, which is not a problem in the field of prosthetic control in which the robot (the prosthesis) is not asked to replicate trajectories precise to the millimetre but rather to be easily usable for the patient whose feedback is the visual one on the basis of which it controls the prosthetic limb. It is not important that a given point in space is reached by the end effector in exactly the same way every time. there is a certain tolerance, and in any case any such constrain is at a much later stage in the development of the control system.

In the horizontal trajectory Fig. 3.15, it can be seen that the situation is very similar to the one above with regard to the shoulder and elbow, with the added advantage that in the horizontal movement the joints have the convenience of not having to move against gravity. It can be noticed how the measured angle values (Encoder) of elbow and shoulder follow well those given as input (IK). The wrist joint in FE is a different matter. This joint is always in an unfavourable configuration since it has to go against gravity in the flexion phase. As said before, a non-optimal PID setting and energy distributed over three joints in parallel with a reduced input at the wrist, probably means that, with the unfavourable kinematic mechanism at the reduction level, this

has such a low efficiency that it sometimes fails to follow the reference as it appears in the final iteration of the trial movement. All this leads us to say that the trajectories are followed with sufficient confidence. We are talking about an error that even in the horizontal trijectory is around ten centimetres, which on Hannes arm's ROM is more than reasonable.

Chapter 5

Conclusions

The main objective of this thesis project is to develop a Cartesian control system for the Hannes upper limb prosthesis by controlling three degrees of freedom. The prosthetic device considered is Hannes, a polyarticulated hand currently designed to restore over 90% of lost functionality in people with transradial amputations. The device has already been improved adding first an active elbow joint for transhumeral amputees, which is currently in the trial phase, and secondly an active shoulder flexion joint. The aim of the developed control strategy is to be used for the complete full-arm Hannes configuration, including the three joints (shoulder, elbow, wrist), and it is intended for use by subjects who have undergone disarticulation of the shoulder. Three methods have been tried out, different in terms of mathematical approach, intuitiveness, analytical modelling, level of action (low/high). These three methods are the Fast Inverse Kinematic, the ANFIS method and the recursive approach in its standard and optimised version. All methods showed good potential. The first one seemed too generic and difficult to develop due to lack of expertise in the virtual world of Unity but, in the continuation of the study, space will be given to the possibility of focusing the work by exploiting the potential of this virtual reality world. The two more analytical approaches found more space in the testing phase and showed their strengths and limitations.

The ANFIS method is the representation of machine learning in this study. In the face of a very long preprocessing it offers the possibility of creating known functions, robust and on whose performance one can easily investigate. It allows the ROMs of the joints and all the necessary boundary conditions to be set a priori, thus ensuring that the robot, i.e. the prosthetic arm, remains inside the workspace. The first limitation of this approach is that the results obtained by the ANFIS method were found to be slightly less accurate than the recursive method. Moreover, it turned out to be difficult to transpose in the Unity universe. This second limitation results from the choice to orient the development efforts - concerning Hannes control systems and future implementation - in a direction including Unity virtual reality.

The recursive method is the approach that has been preferred and brought to the practical level for a first attempt to control the prosthesis. The reasons for this choice are to be found in the fact that it combined - better than the others - adjustability, ease of use, a simple transposition to the Unity world and needed a very clear theoretical basis of easy interpretation. It can be said that it is the method that best combines

the advantage of easy analytical methods for calculation of the three angles parameters needed for the prosthesis control with the use of a high-level environment such as Unity.

Obviously, the algorithm still has some criticalities linked to the non-conservativeness and imperfect robustness of the implementation of the constraints. In addition, it was not possible to complete the purpose of integrating IMUs into the process. One of the objectives was to ensure that the IMUs acted as a validator of the algorithm in the development phase and were subsequently an integral part of the system so that it could independently receive the angle values for the initial configuration, whatever the position of the prosthesis at the ignition was. The aim is to overcome these shortcomings in future developments of the study.

In any case, as indicated by the collected results, the total overview of the study is positive as it was possible, with a sufficient degree of precision, to control the Hannes prototype in the execution of movements such as linear trajectories. Due to the few control sites available, in fact, these trajectories are the essential and most common ones that an amputee with that degree of proximity can perform in the attempt to execute the primary tasks for the recovery of lost motor functions.

Appendix A

Code Scripts

A.1 FAST IK FABRIC

```
1  using System.Collections;
2  using System.Collections.Generic;
3  #if UNITY_EDITOR
4  using UnityEditor;
5  #endif
6  using UnityEngine;
7  using System;
8  using SpaceNavigatorDriver;
9
10 namespace DitzelGames.FastIK
11 {
12     /// Fabrik IK Solver
13     public class FastIKFabric : MonoBehaviour
14     {
15         // Chain length of bones
16         public int ChainLength = 3;
17         // Target the chain should bent to
18         // </summary>
19         public Transform Target;
20         public Transform Pole;
21         // Solver iterations per update
22         [Header("Solver Parameters")]
23         public int Iterations = 10;
24         // Distance when the solver stops
25         public float Delta = 0.05f;
26         // Strength of going back to the start position.
27         [Range(0, 1)]
28         public float SnapBackStrength = 1f;
29
30         protected float[] BonesLength; //Target to Origin
31         protected float TotalLength;
32         protected Transform[] Bones;    //array contenente i bones
33         protected Vector3[] Positions;  //array contenente le ...
               posizioni dei bones
34         protected Vector3[] StartDirectionSucc;
35         protected Quaternion[] StartRotationBone;
```

```
36          protected Quaternion StartRotationTarget;
37          protected Transform Root;
38
39          public Transform shoulder, elbow, wrist;
40
41          public static float SFEangle, EFEangle, WFEangle, PSangle;
42          //public float wristcontrolvalue=0;
43          void Init()
44          {
45              //initial array
46              Bones = new Transform[ChainLength + 1];
47              Positions = new Vector3[ChainLength + 1];
48              BonesLength = new float[ChainLength];
49              StartDirectionSucc = new Vector3[ChainLength + 1];
50              StartRotationBone = new Quaternion[ChainLength + 1];
51
52              //find root
53              Root = transform;
54              for (var i = 0; i ≤ ChainLength; i++)
55              {
56                  if (Root == null)
57                  \\throw new UnityException("The chain value is ...
                        longer than the ancestor chain!");
58                  Root = Root.parent;
59              }
60              //init target
61              if (Target == null)
62              {
63                  Target = new GameObject(gameObject.name + " ...
                        Target").transform;
64                  SetPositionRootSpace(Target, ...
                        GetPositionRootSpace(transform));
65              }
66              StartRotationTarget = GetRotationRootSpace(Target);
67
68              var current = transform;
69              TotalLength = 0;
70              for (var i = Bones.Length - 1; i ≥ 0; i--)
71              {
72                  Bones[i] = current;
73                  StartRotationBone[i] = ...
                        GetRotationRootSpace(current);
74
75                  if (i == Bones.Length - 1)
76                  {
77                      //leaf
78                      StartDirectionSucc[i] = ...
                            GetPositionRootSpace(Target) - ...
                            GetPositionRootSpace(current);
79                  }
80                  else
81                  {
82                      //mid bone
83                      StartDirectionSucc[i] = ...
                            GetPositionRootSpace(Bones[i + 1]) - ...
                            GetPositionRootSpace(current);
```

```
84                      BonesLength[i] = ...
                            StartDirectionSucc[i].magnitude;
85                      TotalLength += BonesLength[i];
86                  }
87                  current = current.parent;
88              }
89          }
90          // Start is called before the first frame update
91          void Awake()
92          {
93              Init();
94          }
95          // Update is called once per frame
96          void LateUpdate()
97          {
98              if ((Mover.KeyROM == false))
99              {
100                 ResolveIK();
101             }
102             EFEangcalc();
103             WFEangcalc();
104             PSangcalc();
105             SFEangcalc();
106         }
107         private void ResolveIK()
108         {
109             if (Target == null)
110                 return;
111             if (BonesLength.Length != ChainLength)
112                 Init();
113
114             //  root
115             //  (bone0) (bonelen 0) (bone1) (bonelen 1) (bone2)...
116             //   x--------------------x--------------------x---...
117
118             for (int i = 0; i < Bones.Length; i++)
119                 Positions[i] = GetPositionRootSpace(Bones[i]);
120
121             var targetPosition = GetPositionRootSpace(Target); ...
                    //posizione del target rispetto al bone radice ...
                    (spalla)
122             var targetRotation = GetRotationRootSpace(Target); ...
                    //orientazione del target rispetto al bone ...
                    radice (spalla)
123
124             //is the target reachable?
125             if ((targetPosition - ...
                    GetPositionRootSpace(Bones[0])).sqrMagnitude ≥ ...
                    TotalLength * TotalLength)
126             {
127                 //just strech it
128                 var direction = (targetPosition - ...
                        Positions[0]).normalized;
129                 //set everything after root
130                 for (int i = 1; i < Positions.Length; i++)
131                     Positions[i] = Positions[i - 1] + direction ...
```

```
                    * BonesLength[i - 1];
132             }
133             else
134             {
135                 for (int i = 0; i < Positions.Length - 1; i++)
136                     Positions[i + 1] = Vector3.Lerp(Positions[i ...
                        + 1], Positions[i] + ...
                        StartDirectionSucc[i], SnapBackStrength);
137                 for (int iteration = 0; iteration < Iterations; ...
                    iteration++)
138                 {
139                     //back
140                     for (int i = Positions.Length - 1; i > 0; i--)
141                     {
142                         if (i == Positions.Length - 1)
143                             Positions[i] = targetPosition; ...
                                //attach the end effector to the ...
                                target
144                         else
145                             Positions[i] = Positions[i + 1] + ...
                                (Positions[i] - Positions[i + ...
                                1]).normalized * BonesLength[i]; ...
                                //set in line on distance
146                     }
147                     //forward
148                     for (int i = 1; i < Positions.Length; i++)
149                     {
150                         Positions[i] = Positions[i - 1] + ...
                            (Positions[i] - Positions[i - ...
                            1]).normalized * BonesLength[i - 1];
151                     }
152                     //close enough?
153                     if ((Positions[Positions.Length - 1] - ...
                        targetPosition).sqrMagnitude < Delta * ...
                        Delta)
154                         break;
155                 }
156             }
157 
158             //set position & rotation
159             for (int i = 0; i < Positions.Length; i++)
160             {
161                 if (i == Positions.Length - 1)
162                     SetRotationRootSpace(Bones[i], ...
                        Quaternion.Inverse(targetRotation) * ...
                        StartRotationTarget * ...
                        Quaternion.Inverse(StartRotationBone[i]));
163                 else
164                     SetRotationRootSpace(Bones[i], ...
                        Quaternion.FromToRotation(StartDirectionSucc[i], ...
                        Positions[i + 1] - Positions[i]) * ...
                        Quaternion.Inverse(StartRotationBone[i]));
165                 SetPositionRootSpace(Bones[i], Positions[i]);
166             }
167         }
168 
```

```
169         private Vector3 GetPositionRootSpace(Transform current)
170         {
171             if (Root == null)
172                 return current.position;
173             else
174                 return Quaternion.Inverse(Root.rotation) * ...
                        (current.position - Root.position);
175         }
176         private void SetPositionRootSpace(Transform current, ...
                Vector3 position)
177         {
178             if (Root == null)
179                 current.position = position;
180             else
181                 current.position = Root.rotation * position + ...
                        Root.position;
182         }
183 
184         private Quaternion GetRotationRootSpace(Transform current)
185         {
186             //inverse(after) * before => rot: before -> after
187             if (Root == null)
188                 return current.rotation;
189             else
190                 return Quaternion.Inverse(current.rotation) * ...
                        Root.rotation;
191         }
192 
193         private void SetRotationRootSpace(Transform current, ...
                Quaternion rotation)
194         {
195             if (Root == null)
196                 current.rotation = rotation;
197             else
198                 current.rotation = Root.rotation * rotation;
199         }
200         void OnDrawGizmos()
201         {
202 #if UNITY_EDITOR
203             var current = this.transform;
204             for (int i = 0; i < ChainLength && current != null ...
                    && current.parent != null; i++)
205             {
206                 var scale = Vector3.Distance(current.position, ...
                        current.parent.position) * 0.1f;
207                 Handles.matrix = ...
                        Matrix4x4.TRS(current.position, ...
                        Quaternion.FromToRotation(Vector3.up, ...
                        current.parent.position - current.position), ...
                        new Vector3(scale, ...
                        Vector3.Distance(current.parent.position, ...
                        current.position), scale));
208                 Handles.color = Color.green;
209                 Handles.DrawWireCube(Vector3.up * 0.5f, ...
                        Vector3.one);
210                 current = current.parent;
```

```
211             }
212 #endif
213         }
214         public float EFEangcalc()
215         {
216             float unityEFEangle = elbow.localEulerAngles.x;
217 
218             if ((0f ≤ unityEFEangle) && (unityEFEangle ≤ 90f))
219             {
220                 EFEangle = 90f - unityEFEangle;
221             }
222             else if ((315f ≤ unityEFEangle) && (unityEFEangle ≤ ...
                    360f))
223             {
224                 EFEangle = 450f - unityEFEangle;
225             }
226             else
227             {
228                 Mover.AngleFlag = true;
229             }
230             return EFEangle;
231         }
232         public float WFEangcalc()
233         {
234             float unityWFEangle = wrist.localEulerAngles.x;
235                 if ((300f ≤ unityWFEangle) && (unityWFEangle ≤ ...
                        360f))
236                 {
237                     WFEangle = 360f - unityWFEangle;
238                 }
239                 else if ((0f ≤ unityWFEangle) && (unityWFEangle ...
                        ≤ 60f))
240                 {
241                     WFEangle = -unityWFEangle;
242                 }
243                 else
244                 {
245                     Mover.AngleFlag = true;
246                 }
247             return WFEangle;
248         }
249         float PSangcalc()
250         {
251             float unityPSangle = ...
                    rollicchio.transform.eulerAngles.x;
252             PSangle = unityPSangle;
253             return PSangle;
254         }
255         float SFEangcalc()
256         {
257             float unitySFEangle = shoulder.localEulerAngles.x;
258 
259             if ((0f ≤ unitySFEangle) && (unitySFEangle ≤ 5f)) ...
                    //tolleranza di 5 gradi ma dovrebbe bloccarsi a 0
260             {
261                 SFEangle = -unitySFEangle;
```

```
262                 }
263                 else if ((225f ≤ unitySFEangle) && (unitySFEangle ≤ ...
                        360f))
264                 {
265                     SFEangle = 360f - unitySFEangle;
266                 }
267                 else
268                 {
269                     Mover.AngleFlag = true;
270                 }
271                 return SFEangle;
272             }
273         }
274 }
```

A.2 Recursive Approach

```
1  public class JacobianMethod : MonoBehaviour
2  {
3  private int e = 0;        //serve per traiettoria
4  private string CurrentDirectory;
5  private string FileSession;
6  private string FilePath;
7  private string FileIK = "IK.txt";
8  static StreamWriter writeIK;
9  static string specifier = "N7";
10
11 //i 4 joint formanti il ghost, ovvero la catena cinematica ...
       semplificata
12 public Transform shoulder, elbow, wrist, knuckle;
13
14 //parametri per pinverse
15 public static float L1, L2, L3;
16 public static double alfa1, alfa2, alfa3;
17 public static float th1, th2, th3;
18
19 public float th1_H_MAX, th1_H_MIN, th2_H_MAX, th2_H_MIN, ...
       th3_H_MAX, th3_H_MIN;
20 public static float th1_H_MAXs, th1_H_MINs, th2_H_MAXs, ...
       th2_H_MINs, th3_H_MAXs, th3_H_MINs;
21 public static double t1max, t1min, t2max, t2min, t3max, t3min;
22
23 public float th1maxOpt, th1minOpt, th2maxOpt, th2minOpt, ...
       th3maxOpt, th3minOpt;
24 public float th1_H_maxOpt, th1_H_minOpt, th2_H_maxOpt, ...
       th2_H_minOpt, th3_H_maxOpt, th3_H_minOpt;
25
26 double[,] jac;
27 double[,] jactras;
28 double[,] jacperjactras;
29 double[,] invdijacperjactras;
30 double[,] pinv;
```

```
31 double[,] id;
32 double[,] pinvperjac;
33 double[,] idmenopinvperjac;
34 double[,] gradH;
35 double[,] gradHperidmenopinvperjac;
36
37 double sinth1, costh1;
38 double th1plusth2, sinth1plusth2, costh1plusth2;
39 double th1plusth2plusth3, sinth1plusth2plusth3, ...
       costh1plusth2plusth3;
40
41 public double ALFA;
42 private double Δtime = 0.02;          //ATTENZIONE A QUESTO ...
       PARAMETRO
43
44 double Xinc=0, Yinc=0;
45
46 // Start is called before the first frame update
47 void Awake()
48 {
49
50     //parametri per pinverse
51     L1 = Vector3.Distance(shoulder.position, elbow.position);
52     L2 = Vector3.Distance(elbow.position, wrist.position);
53     L3 = Vector3.Distance(wrist.position, knuckle.position);
54
55     //nella realt  saranno forniti dagli IMU
56     alfa1 = -75.0;// shoulder.localEulerAngles.x;
57     alfa2 = 90.0;// elbow.localEulerAngles.x;
58     alfa3 = 10.0;// wrist.localEulerAngles.x;
59
60     jac = new double[2, 3];
61     jactras = new double[3, 2];
62     jacperjactras = new double[2, 2];
63     invdijacperjactras = new double[2, 2];
64     pinv = new double[3, 2];
65     id = new double[3, 3];
66     pinvperjac = new double[3, 3];
67     idmenopinvperjac = new double[3, 3];
68     gradH = new double[3, 1];
69     gradHperidmenopinvperjac = new double[3, 1];
70
71     id[0, 0] = (double) 1;
72     id[0, 1] = (double) 0;
73     id[0, 2] = (double) 0;
74     id[1, 0] = (double) 0;
75     id[1, 1] = (double) 1;
76     id[1, 2] = (double) 0;
77     id[2, 0] = (double) 0;
78     id[2, 1] = (double) 0;
79     id[2, 2] = (double) 1;
80
81     th1 = (float)(Math.PI * alfa1 / 180.0);
82     th2 = (float)(Math.PI * alfa2 / 180.0);
83     th3 = (float)(Math.PI * alfa3 / 180.0);
84
```

```
85      t1max = 45*Math.PI/180.0;
86      //t1avg = -22.5 * Math.PI / 180.0;
87      t1min = -90 * Math.PI / 180.0;
88      t2max = 130 * Math.PI / 180.0;
89      //t2avg = (130 / 2) * Math.PI / 180.0;
90      t2min = 0 * Math.PI / 180.0;
91      t3max = 60 * Math.PI / 180.0;
92      //t3avg = 0 * Math.PI / 180.0;
93      t3min = -60 * Math.PI / 180.0;
94
95
96
97      ALFA = 0.05;
98  }
99
100 // Update is called once per frame
101 void Update()
102 {
103     th1_H_MAXs = th1_H_MAX;
104     th1_H_MINs = th1_H_MIN;
105     th2_H_MAXs = th2_H_MAX;
106     th2_H_MINs = th2_H_MIN;
107     th3_H_MAXs = th3_H_MAX;
108     th3_H_MINs = th3_H_MIN;
109     //if (e ≤ 99)
110     {
111         //Debug.Log("L1 is:" + L1);
112         //Debug.Log("L2 is:" + L2);
113         //Debug.Log("L3 is:" + L3);
114
115         //Debug.Log("th2 is:" + th2 * 180.0 / Math.PI);
116         //Debug.Log("th3 is:" + th3 * 180.0 / Math.PI);
117
118         Xinc = (Mover.Targetposition[1].x - ...
                Mover.Targetposition[0].x); //incremento di X ...
                istante per istante                //CHE SIA LA ...
                POSIZIONE GIUSTA PER GLI INCREMENTI?
119         Yinc = (Mover.Targetposition[1].y - ...
                Mover.Targetposition[0].y); //incremento di Y ...
                istante per istante
120
121         //CONTI MATRICI
122         //MATRICI
123         sinth1 = Math.Sin(th1);
124         costh1 = Math.Cos(th1);
125         th1plusth2 = th1 + th2;
126         sinth1plusth2 = Math.Sin(th1plusth2);
127         costh1plusth2 = Math.Cos(th1plusth2);
128         th1plusth2plusth3 = th1 + th2 + th3;
129         sinth1plusth2plusth3 = Math.Sin(th1plusth2plusth3);
130         costh1plusth2plusth3 = Math.Cos(th1plusth2plusth3);
131
132         //calcolo delle rotation dei tre giunti
133         jac[0, 0] = (double)(-L1 * sinth1 - L2 * sinth1plusth2 ...
                - L3 * sinth1plusth2plusth3);
134         jac[0, 1] = (double)(-L2 * sinth1plusth2 - L3 * ...
```

```
            sinth1plusth2plusth3);
135         jac[0, 2] = (double)(-L3 * sinth1plusth2plusth3);
136
137         jac[1, 0] = (double)(L1 * costh1 + L2 * costh1plusth2 + ...
            L3 * costh1plusth2plusth3);
138         jac[1, 1] = (double)(L2 * costh1plusth2 + L3 * ...
            costh1plusth2plusth3);
139         jac[1, 2] = (double)(L3 * costh1plusth2plusth3);
140
141         jactras[0, 0] = jac[0, 0];
142         jactras[0, 1] = jac[1, 0];
143         jactras[1, 0] = jac[0, 1];
144         jactras[1, 1] = jac[1, 1];
145         jactras[2, 0] = jac[0, 2];
146         jactras[2, 1] = jac[1, 2];
147
148         jacperjactras[0, 0] = jac[0, 0] * jactras[0, 0] + ...
            jac[0, 1] * jactras[1, 0] + jac[0, 2] * jactras[2, 0];
149         jacperjactras[0, 1] = jac[0, 0] * jactras[0, 1] + ...
            jac[0, 1] * jactras[1, 1] + jac[0, 2] * jactras[2, 1];
150         jacperjactras[1, 0] = jac[1, 0] * jactras[0, 0] + ...
            jac[1, 1] * jactras[1, 0] + jac[1, 2] * jactras[2, 0];
151         jacperjactras[1, 1] = jac[1, 0] * jactras[0, 1] + ...
            jac[1, 1] * jactras[1, 1] + jac[1, 2] * jactras[2, 1];
152
153         double determinante = 1 / (jacperjactras[0, 0] * ...
            jacperjactras[1, 1] - jacperjactras[0, 1] * ...
            jacperjactras[1, 0]);
154
155         invdijacperjactras[0, 0] = determinante * ...
            jacperjactras[1, 1];
156         invdijacperjactras[0, 1] = determinante * ...
            -jacperjactras[0, 1];
157         invdijacperjactras[1, 0] = determinante * ...
            -jacperjactras[1, 0];
158         invdijacperjactras[1, 1] = determinante * ...
            jacperjactras[0, 0];
159
160         pinv[0, 0] = jactras[0, 0] * invdijacperjactras[0, 0] + ...
            jactras[0, 1] * invdijacperjactras[1, 0];
161         pinv[1, 0] = jactras[1, 0] * invdijacperjactras[0, 0] + ...
            jactras[1, 1] * invdijacperjactras[1, 0];
162         pinv[0, 1] = jactras[0, 0] * invdijacperjactras[0, 1] + ...
            jactras[0, 1] * invdijacperjactras[1, 1];
163         pinv[1, 1] = jactras[1, 0] * invdijacperjactras[0, 1] + ...
            jactras[1, 1] * invdijacperjactras[1, 1];
164         pinv[2, 0] = jactras[2, 0] * invdijacperjactras[0, 0] + ...
            jactras[2, 1] * invdijacperjactras[1, 0];
165         pinv[2, 1] = jactras[2, 0] * invdijacperjactras[0, 1] + ...
            jactras[2, 1] * invdijacperjactras[1, 1];
166
167         pinvperjac[0, 0] = pinv[0, 0] * jac[0, 0] + pinv[0, 1] ...
            * jac[1, 0];
168         pinvperjac[0, 1] = pinv[0, 0] * jac[0, 1] + pinv[0, 1] ...
            * jac[1, 1];
169         pinvperjac[0, 2] = pinv[0, 0] * jac[0, 2] + pinv[0, 1] ...
```

```
            * jac[1, 2];
170         pinvperjac[1, 0] = pinv[1, 0] * jac[0, 0] + pinv[1, 1] ...
            * jac[1, 0];
171         pinvperjac[1, 1] = pinv[1, 0] * jac[0, 1] + pinv[1, 1] ...
            * jac[1, 1];
172         pinvperjac[1, 2] = pinv[1, 0] * jac[0, 2] + pinv[1, 1] ...
            * jac[1, 2];
173         pinvperjac[2, 0] = pinv[2, 0] * jac[0, 0] + pinv[2, 1] ...
            * jac[1, 0];
174         pinvperjac[2, 1] = pinv[2, 0] * jac[0, 1] + pinv[2, 1] ...
            * jac[1, 1];
175         pinvperjac[2, 2] = pinv[2, 0] * jac[0, 2] + pinv[2, 1] ...
            * jac[1, 2];
176
177         idmenopinvperjac[0, 0] = id[0, 0] - pinvperjac[0, 0];
178         idmenopinvperjac[0, 1] = id[0, 1] - pinvperjac[0, 1];
179         idmenopinvperjac[0, 2] = id[0, 2] - pinvperjac[0, 2];
180         idmenopinvperjac[1, 0] = id[1, 0] - pinvperjac[1, 0];
181         idmenopinvperjac[1, 1] = id[1, 1] - pinvperjac[1, 1];
182         idmenopinvperjac[1, 2] = id[1, 2] - pinvperjac[1, 2];
183         idmenopinvperjac[2, 0] = id[2, 0] - pinvperjac[2, 0];
184         idmenopinvperjac[2, 1] = id[2, 1] - pinvperjac[2, 1];
185         idmenopinvperjac[2, 2] = id[2, 2] - pinvperjac[2, 2];
186
187
188         //HANNES SREAL
189         if (CartesianMotion.HANNES_REALstatus==true)
190         {
191             gradH[0, 0] = ((th1_H_MAX - th1_H_MIN) * (th1_H_MAX ...
                - th1_H_MIN)) * (2 * th1 - th1_H_MAX - ...
                th1_H_MIN) / (4 * ((th1_H_MAX - th1) * ...
                (th1_H_MAX - th1) * (th1 - th1_H_MIN) * (th1 - ...
                th1_H_MIN)));
192             gradH[1, 0] = ((th2_H_MAX - th2_H_MIN) * (th2_H_MAX ...
                - th2_H_MIN)) * (2 * th2 - th2_H_MAX - ...
                th2_H_MIN) / (4 * ((th2_H_MAX - th2) * ...
                (th2_H_MAX - th2) * (th2 - th2_H_MIN) * (th2 - ...
                th2_H_MIN)));
193             gradH[2, 0] = ((th3_H_MAX - th3_H_MIN) * (th3_H_MAX ...
                - th3_H_MIN)) * (2 * th3 - th3_H_MAX - ...
                th3_H_MIN) / (4 * ((th3_H_MAX - th3) * ...
                (th3_H_MAX - th3) * (th3 - th3_H_MIN) * (th3 - ...
                th3_H_MIN)));
194
195             gradHperidmenopinvperjac[0, 0] = ...
                idmenopinvperjac[0, 0] * gradH[0, 0] + ...
                idmenopinvperjac[0, 1] * gradH[1, 0] + ...
                idmenopinvperjac[0, 2] * gradH[2, 0];
196             gradHperidmenopinvperjac[1, 0] = ...
                idmenopinvperjac[1, 0] * gradH[0, 0] + ...
                idmenopinvperjac[1, 1] * gradH[1, 0] + ...
                idmenopinvperjac[1, 2] * gradH[2, 0];
197             gradHperidmenopinvperjac[2, 0] = ...
                idmenopinvperjac[2, 0] * gradH[0, 0] + ...
                idmenopinvperjac[2, 1] * gradH[1, 0] + ...
                idmenopinvperjac[2, 2] * gradH[2, 0];
```

```
198
199             if ((th1 ≥ (th1_H_MAX - 0.5f) * Math.PI / 180f) | ...
                    (th1 ≤ (th1_H_MIN + 0.5f) * Math.PI / 180) | ...
                    (th2 ≥ (th2_H_MAX + 0.5) * Math.PI / 180) | (th2 ...
                    ≤ (th2_H_MIN + 0.5) * Math.PI / 180) | (th3 ≥ ...
                    (th3_H_MAX - 0.5) * Math.PI / 180) | (th3 ≤ ...
                    (th3_H_MIN + 0.5) * Math.PI / 180))
200             {
201                 Debug.Log("Hannes real is OUT of rom");
202                 Mover.AngleFlag = true;
203                 transform.position = Mover.Targetposition[0];
204                 Mover.Targetposition[1] = Mover.Targetposition[0];
205                 Mover.Targetposition[2] = Mover.Targetposition[0];
206                 Mover.Targetposition[3] = Mover.Targetposition[0];
207                 Mover.Targetposition[4] = Mover.Targetposition[0];
208             }
209             else
210             {
211                 Debug.Log("Hannes real is IN rom");
212
213                 Xinc = (Mover.Targetposition[1].x - ...
                        Mover.Targetposition[0].x)/10; //incremento ...
                        di X istante per istante                //CHE ...
                        SIA LA POSIZIONE GIUSTA PER GLI INCREMENTI?
214                 Yinc = (Mover.Targetposition[1].y - ...
                        Mover.Targetposition[0].y)/10; //incremento ...
                        di Y istante per istante
215
216                 if ((th1 ≥ (th1_H_maxOpt * Math.PI / 180)) | ...
                        (th1 ≤ (th1_H_minOpt * Math.PI / 180)) | ...
                        (th2 ≥ (th2_H_maxOpt * Math.PI / 180)) | ...
                        (th2 ≤ (th2_H_minOpt * Math.PI / 180))) //| ...
                        (th3 ≤ th3_H_minOpt * Math.PI / 180) | (th3 ≥...
                         th3_H_maxOpt * Math.PI / 180))
217                 {
218                     //con ottimizzazione
219                     th1 = (float)((th1 + (pinv[0, 0] * Xinc + ...
                            pinv[0, 1] * Yinc)) - ∆time * (ALFA * ...
                            gradHperidmenopinvperjac[0, 0]));
220                     th2 = (float)((th2 + (pinv[1, 0] * Xinc + ...
                            pinv[1, 1] * Yinc)) - ∆time * (ALFA * ...
                            gradHperidmenopinvperjac[1, 0]));
221                     th3 = (float)((th3 + (pinv[2, 0] * Xinc + ...
                            pinv[2, 1] * Yinc)) - ∆time * (ALFA * ...
                            gradHperidmenopinvperjac[2, 0]));
222                                     }
223                 else
224                 {
225                     //senza ottimizzazione
226                     th1 = (float)(th1 + (pinv[0, 0] * Xinc + ...
                            pinv[0, 1] * Yinc));
227                     th2 = (float)(th2 + (pinv[1, 0] * Xinc + ...
                            pinv[1, 1] * Yinc));
228                     th3 = (float)(th3 + (pinv[2, 0] * Xinc + ...
                            pinv[2, 1] * Yinc));
229                 }
```

```
230             }
231         }
232         else if (CartesianMotion.HANNES_REALstatus == false)
233         {
234 
235             gradH[0, 0] = ((t1max - t1min) * (t1max - t1min)) * ...
                (2 * th1 - t1max - t1min) / (4 * ((t1max - th1) ...
                * (t1max - th1) * (th1 - t1min) * (th1 - t1min)));
236             gradH[1, 0] = ((t2max - t2min) * (t2max - t2min)) * ...
                (2 * th2 - t2max - t2min) / (4 * ((t2max - th2) ...
                * (t2max - th2) * (th2 - t2min) * (th2 - t2min)));
237             gradH[2, 0] = ((t3max - t3min) * (t3max - t3min)) * ...
                (2 * th3 - t3max - t3min) / (4 * ((t3max - th3) ...
                * (t3max - th3) * (th3 - t3min) * (th3 - t3min)));
238 
239             gradHperidmenopinvperjac[0, 0] = ...
                idmenopinvperjac[0, 0] * gradH[0, 0] + ...
                idmenopinvperjac[0, 1] * gradH[1, 0] + ...
                idmenopinvperjac[0, 2] * gradH[2, 0];
240             gradHperidmenopinvperjac[1, 0] = ...
                idmenopinvperjac[1, 0] * gradH[0, 0] + ...
                idmenopinvperjac[1, 1] * gradH[1, 0] + ...
                idmenopinvperjac[1, 2] * gradH[2, 0];
241             gradHperidmenopinvperjac[2, 0] = ...
                idmenopinvperjac[2, 0] * gradH[0, 0] + ...
                idmenopinvperjac[2, 1] * gradH[1, 0] + ...
                idmenopinvperjac[2, 2] * gradH[2, 0];
242 
243             //if ((th2 ≥ (t2max - 0.5*Math.PI/180)) | (th2 ≤ ...
                (t2min + 0.5 * Math.PI / 180)) | (th3 ≤ (t3min + ...
                0.5 * Math.PI / 180)) | (th3 ≥ (t3max - 0.5 * ...
                Math.PI / 180)) | (th1 ≥ (t1max - 0.5 * Math.PI ...
                / 180)) | (th1 ≤ (t1min + 0.5 * Math.PI / 180)))
244             if ((th2 ≥ (119f * Math.PI / 180)) | (th2 ≤ (1f * ...
                Math.PI / 180)) | (th3 ≤ (-54f * Math.PI / 180)) ...
                | (th3 ≥ (29 * Math.PI / 180)) | (th1 ≥ (29f * ...
                Math.PI / 180)) | (th1 ≤ (-89f * Math.PI / 180)))
245             {
246                 Debug.Log("VIRTUAL is OUT of rom");
247                 Mover.AngleFlag = true;
248 
249                 transform.position = Mover.Targetposition[0];
250                 Mover.Targetposition[1] = Mover.Targetposition[0];
251                 Mover.Targetposition[2] = Mover.Targetposition[0];
252                 Mover.Targetposition[3] = Mover.Targetposition[0];
253                 Mover.Targetposition[4] = Mover.Targetposition[0];
254             }
255             else
256             {
257                 //Debug.Log("VIRTUAL is IN of rom");
258 
259                 Xinc = (Mover.Targetposition[1].x - ...
                    Mover.Targetposition[0].x); //incremento di ...
                    X istante per istante                //CHE ...
                    SIA LA POSIZIONE GIUSTA PER GLI INCREMENTI?
260                 Yinc = (Mover.Targetposition[1].y - ...
```

```
                    Mover.Targetposition[0].y); //incremento di ...
                    Y istante per istante
261
262                 if ((th1 ≥ (th1maxOpt * Math.PI / 180)) | (th1 ≤...
                     (th1minOpt * Math.PI / 180)) | (th2 ≥ ...
                    (th2maxOpt * Math.PI / 180)) | (th2 ≤ ...
                    (th2minOpt * Math.PI / 180))) //| (th3 ≤ ...
                    th3minOpt * Math.PI / 180) | (th3 ≥ ...
                    th3maxOpt * Math.PI / 180))
263                 {
264                     //con ottimizzazione
265                     th1 = (float)((th1 + (pinv[0, 0] * Xinc + ...
                        pinv[0, 1] * Yinc)) - ∆time * (ALFA * ...
                        gradHperidmenopinvperjac[0, 0]));
266                     th2 = (float)((th2 + (pinv[1, 0] * Xinc + ...
                        pinv[1, 1] * Yinc)) - ∆time * (ALFA * ...
                        gradHperidmenopinvperjac[1, 0]));
267                     th3 = (float)((th3 + (pinv[2, 0] * Xinc + ...
                        pinv[2, 1] * Yinc)) - ∆time * (ALFA * ...
                        gradHperidmenopinvperjac[2, 0]));
268                 }
269                 else
270                 {
271                     //senza ottimizzazione
272                     th1 = (float)(th1 + (pinv[0, 0] * Xinc + ...
                        pinv[0, 1] * Yinc));
273                     th2 = (float)(th2 + (pinv[1, 0] * Xinc + ...
                        pinv[1, 1] * Yinc));
274                     th3 = (float)(th3 + (pinv[2, 0] * Xinc + ...
                        pinv[2, 1] * Yinc));
275                 }
276             }
277
278         }
279
280     }
281     e = e+1;
282     //Debug.Log("\nth1 is:" + th1 * 180.0 / Math.PI + "\nth2 ...
        is:" + th2 * 180.0 / Math.PI + "\nth3 is:" + th3 * 180.0 ...
        / Math.PI);
283
284 }
285 }
```

A.3 ANFIS

```
1 % SCRIPT TO CREATE SEVERAL ANFIS NETWORKS
2 step=[5]; %angles step [deg]
3 infis=[8];
4 epoch=[10 50 100 200 300];
5
```

```
totnum=length(infis)*length(epoch)*length(step);

% Index definition
i=1; % index for the resolution values vector
j=1; % index for InitialFIS vector
k=1; % index for EpochNumber vector
m=1; % index to fill the anfis vector structure
n=1;
counter = 1;
T = 0;

%virtual hannes dimension
l1=12.74508;
l2=12.04653;
l3=4.355566;

theta1 = -90:step(i):45; % all possible theta1 values
theta2 = 0:step(i):130; % all possible theta2 values
theta3 = -60:step(i):60; % all possible theta3 values

[THETA1,THETA2,THETA3] = ...
   meshgrid(theta1,theta2,theta3); % generate a grid ...
   of theta1, theta2 and theta3 values

X = l1*cosd(THETA1)+l2*cosd(THETA1+THETA2)+...
l3*cosd(THETA1+THETA2+THETA3); % compute x coordinates
Y = l1*sind(THETA1)+l2*sind(THETA1+THETA2)+...
l3*sind(THETA1+THETA2+THETA3) ; % compute y coordinates

data1 = [X(:) Y(:) THETA1(:)]; % create x-y-theta1 ...
   dataset
data2 = [X(:) Y(:) THETA2(:)]; % create x-y-theta2 ...
   dataset
data3 = [X(:) Y(:) THETA3(:)]; % create x-y-theta3 ...
   dataset

%plot del ROM
plot(X(:),Y(:),'.','Color',[50/256,205/256,50/256]);
axis equal;
grid on;
hold on
xlabel('X','fontsize',10)
ylabel('Y','fontsize',10)
title('X-Y coordinates for all theta1, theta2 and ...
   theta3 combinations - step 10 deg','fontsize',12)
axis([-20 40 -30 30]);
```

```
46
47 %% Training delle ANFIS
48 for j=1:length(infis)
49     for k=1:length(epoch)
50         tic
51         opt = anfisOptions;
52         opt.InitialFIS =infis(j) ;
53         opt.EpochNumber = epoch(k);
54         opt.DisplayANFISInformation = 0;
55         opt.DisplayErrorValues = 0;
56         opt.DisplayStepSize = 0;
57         opt.DisplayFinalResults = 1;
58         %opt.OptimizationMethod??
59
60         disp('--> Training first ANFIS network.')
61         anfisvector1{m}=anfis(data1,opt);
62         disp('--> Training second ANFIS network.')
63         anfisvector2{m}=anfis(data2,opt);
64         disp('--> Training third ANFIS network.')
65         anfisvector3{m}=anfis(data3,opt);
66
67         Line = '/n THE GROUP OF THREE nr %4.2d of ...
              %8.3d IS DONE /n';
68         fprintf(Line,m,totnum)
69         if counter ~= 1
70             T_LEFT = (totnum - counter)*T;
71             fprintf('\t>> Extimated time left: %d ...
                  secs<<\n',T_LEFT);
72         end
73         m=m+1;
74         T = (T+toc)/counter;
75         counter = counter + 1;
76     end
77 end
78
79 save  anfisvector1;
80 save  anfisvector2;
81 save  anfisvector3;
82
83 %% TRAJECTORY DEFINITION
84 theta1_tj = linspace(-50,0,100);
85 theta2_tj = linspace(45,90,100);
86 theta3_tj = linspace(-60,15,100);
87 xtg= l1*cosd(theta1_tj)+l2*cosd(theta1_tj+theta2_tj)+...
88 l3*cosd(theta1_tj+theta2_tj+theta3_tj); %x coord
89 ytg = l1*sind(theta1_tj)+l2*sind(theta1_tj+theta2_tj)+...
```

```
l3*sind(theta1_tj+theta2_tj+theta3_tj) ; %y coord
XYtg = [xtg(:) ytg(:)]

plot(xtg,ytg,'.b','MarkerSize',15), grid on
Xd = xtg(:);
Yd = ytg(:);

ErrorMatrix = zeros(length(anfisvector1),16);

for n=1:length(anfisvector1)

    anfis1=anfisvector1{n};
    anfis2=anfisvector2{n};
    anfis3=anfisvector3{n};

    THETA1P = evalfis(anfis1,XYtg); % theta1 predicted
    THETA2P = evalfis(anfis2,XYtg); % theta2 predicted
    THETA3P = evalfis(anfis3,XYtg); % theta3 predicted

    Xp = l1*cosd(THETA1P)+l2*cosd(THETA1P+THETA2P)+...
    l3*cosd(THETA1P+THETA2P+THETA3P);
    Yp = l1*sind(THETA1P)+l2*sind(THETA1P+THETA2P)+...
    l3*sind(THETA1P+THETA2P+THETA3P) ;

    MaxPositionErrorX=max(abs(Xd-Xp));
    MaxPositionErrorY=max(abs(Yd-Yp));
    MinPositionErrorX=min(abs(Xd-Xp));
    MinPositionErrorY=min(abs(Yd-Yp));
    MaxTheta1error=max(abs(theta1_tj-THETA1P'));
    MaxTheta2error=max(abs(theta2_tj-THETA2P'));
    MaxTheta3error=max(abs(theta3_tj-THETA3P'));
    MinTheta1error=min(abs(theta1_tj-THETA1P'));
    MinTheta2error=min(abs(theta2_tj-THETA2P'));
    MinTheta3error=min(abs(theta3_tj-THETA3P'));
    XRMSE = sqrt(mean((Xd-Xp).^2));
    YRMSE = sqrt(mean((Yd-Yp).^2));
    Theta1RMSE = sqrt(mean((theta1_tj-THETA1P').^2));
    Theta2RMSE = sqrt(mean((theta2_tj-THETA2P').^2));
    Theta3RMSE = sqrt(mean((theta3_tj-THETA3P').^2));

    info=[n MaxPositionErrorX MinPositionErrorX...
            MaxPositionErrorY MinPositionErrorY...
            MaxTheta1error MinTheta1error...
            MaxTheta2error MinTheta2error...
            MaxTheta3error MinTheta3error...
            XRMSE YRMSE...
```

```
136             Theta1RMSE Theta2RMSE Theta3RMSE];
137     ErrorMatrix(n,:)=info;
138 end
139 hold on
140 plot(Xp(:),Yp(:),'.c','MarkerSize',15), grid on
141
142 ANFISn=ErrorMatrix(:,1);
143 MaxPositionErrorX= ErrorMatrix(:,2) ;
144 MinPositionErrorX=ErrorMatrix(:,3);
145 MaxPositionErrorY=ErrorMatrix(:,4);
146 MinPositionErrorY=ErrorMatrix(:,5);
147 MaxTheta1error=ErrorMatrix(:,6);
148 MinTheta1error=ErrorMatrix(:,7);
149 MaxTheta2error=ErrorMatrix(:,8);
150 MinTheta2error=ErrorMatrix(:,9);
151 MaxTheta3error=ErrorMatrix(:,10);
152 MinTheta3error=ErrorMatrix(:,11);
153 XRMSE = ErrorMatrix(:,12);
154 YRMSE =ErrorMatrix(:,13) ;
155 Theta1RMSE =ErrorMatrix(:,14) ;
156 Theta2RMSE =ErrorMatrix(:,15);
157 Theta3RMSE =ErrorMatrix(:,16);
158
159 T=table(ANFISn, MaxPositionErrorX, MinPositionErrorX,...
160                 MaxPositionErrorY, MinPositionErrorY,...
161                 MaxTheta1error, MinTheta1error,...
162                 MaxTheta2error, MinTheta2error,...
163                 MaxTheta3error, MinTheta3error,...
164                 XRMSE, YRMSE,...
165                 Theta1RMSE, Theta2RMSE,Theta3RMSE)
166 writetable(T,'ANFISerrorComparison.txt','Delimiter','\t');
167 type ANFISerrorComparison.txt
168
169 xlswrite('ErrorMatrixProvan5deg.xlsx',ErrorMatrix);
170
171 %% definizione del robot per visualizzare esiti
172 robot = ...
      rigidBodyTree('DataFormat','column','MaxNumBodies',3);
173
174     L1=12.74508;
175     L2=12.04653;
176     L3=4.355566;
177
178     %spalla
179 body = rigidBody('link1');
180 joint = rigidBodyJoint('joint1', 'revolute');
```

```
181 setFixedTransform(joint,trvec2tform([0 0 0]));
182 joint.JointAxis = [0 0 1];
183 body.Joint = joint;
184 addBody(robot, body, 'base');
185     %gomito
186 body = rigidBody('link2');
187 joint = rigidBodyJoint('joint2','revolute');
188 setFixedTransform(joint, trvec2tform([L1,0,0]));
189 joint.JointAxis = [0 0 1];
190 body.Joint = joint;
191 addBody(robot, body, 'link1');
192     %polso
193 body = rigidBody('link3');
194 joint = rigidBodyJoint('joint3','revolute');
195 setFixedTransform(joint, trvec2tform([L2,0,0]));
196 joint.JointAxis = [0 0 1];
197 body.Joint = joint;
198 addBody(robot, body, 'link2');
199     %end effector
200 body = rigidBody('tool');
201 joint = rigidBodyJoint('fix1','fixed');
202 setFixedTransform(joint, trvec2tform([L3, 0, 0]));
203 body.Joint = joint;
204 addBody(robot, body, 'link3');
205
206 showdetails(robot);
207 show(robot);
208 view(2)
209 ax = gca;
210 ax.Projection = 'orthographic';
211
212 %% cinematica inversa
213 syms t1;
214 syms t2;
215 syms t3;
216
217 qs(:,1)=THETA1P;
218 qs(:,2)=THETA2P;
219 qs(:,3)=THETA3P;
220 qs=deg2rad(qs);
221 t = (0:0.01:10); % Time
222 count = length(t);
223
224 framesPerSecond = 100000;
225 r = rateControl(framesPerSecond);
226 for i = 1:count
```

```
227       show(robot,qs(i,:)','PreservePlot',false);
228       view(2)
229       ax = gca;
230       ax.Projection = 'orthographic';
231       drawnow
232       hold on
233       waitfor(r);
234       disp('--> point')
235   end
```

Appendix B

ANFIS evaluation parameters

Step [deg]	Infis	n epoch	Th1RMSE	Th2RMSE	Th3RMSE	MaxTh1Err [cm]	MinTh1Err [cm]	MaxTh2Err [deg]	MinTh2Err [deg]	MaxTh3Err [deg]	MinTh3Err [deg]
10	2	10	5.409	13.705	27.441	9.992	0.034	18.539	0.019	54.972	0.292
10	2	50	5.098	13.192	27.578	9.337	0.056	17.073	0.181	55.332	0.270
10	2	100	3.527	12.290	28.273	5.532	0.082	16.589	0.060	55.595	0.278
10	2	200	3.463	11.198	28.400	5.359	0.076	15.447	0.158	55.511	0.272
10	2	300	3.448	11.114	28.405	5.330	0.069	15.329	0.064	55.516	0.275
10	2	400	3.443	11.040	28.410	5.320	0.062	15.219	0.085	55.521	0.280
10	3	10	3.923	10.459	30.104	6.129	0.095	17.736	1.929	56.403	0.400
10	3	50	3.661	10.041	29.831	5.656	0.120	16.719	1.224	56.315	0.371
10	3	100	2.896	9.342	29.487	4.668	0.052	14.540	0.145	56.878	0.041
10	3	200	2.943	9.241	28.933	5.673	0.025	14.002	0.040	55.038	0.202
10	3	300	2.949	9.233	29.148	5.674	0.020	13.930	0.041	55.240	0.187
10	3	400	2.950	9.235	29.152	5.667	0.016	13.926	0.071	55.238	0.188
10	4	10	2.665	8.929	28.590	4.639	0.003	13.539	0.096	55.747	0.332
10	4	50	2.635	8.810	28.734	4.979	0.061	13.104	0.004	55.955	0.418
10	4	100	2.643	8.675	29.365	5.393	0.017	12.699	0.008	53.755	0.010
10	4	200	2.635	8.688	29.874	5.186	0.022	12.650	0.004	54.589	0.229
10	4	300	2.571	8.700	29.958	4.985	0.102	12.695	0.007	54.591	0.250
10	4	400	2.559	8.705	29.994	4.958	0.080	12.691	0.010	54.611	0.252
10	5	10	2.607	8.444	29.084	4.994	0.003	13.022	0.016	55.815	0.379
10	5	50	2.587	8.438	29.041	4.908	0.086	12.721	0.044	55.792	0.432
10	5	100	2.624	8.553	30.812	5.953	0.060	12.314	0.020	54.854	0.210
10	5	200	2.521	8.567	30.670	6.110	0.057	12.374	0.019	55.609	0.233
10	5	300	2.516	8.538	30.740	6.096	0.044	12.357	0.063	55.547	0.217
10	5	400	2.510	8.448	30.793	6.050	0.048	12.095	0.044	55.483	0.248
10	6	10	2.460	8.205	29.712	4.755	0.084	12.305	0.073	55.576	0.050
10	6	50	2.445	8.290	30.166	4.871	0.113	12.497	0.025	56.113	0.009
10	6	100	2.417	8.242	30.382	5.438	0.097	12.432	0.100	55.886	0.337
10	6	200	2.478	8.250	30.265	5.313	0.022	12.369	0.025	55.024	0.454
10	6	300	2.501	8.253	30.189	5.343	0.078	12.308	0.036	54.126	0.151
10	6	400	2.523	8.248	30.159	5.380	0.039	12.323	0.105	53.841	0.097
10	7	10	2.541	8.200	29.924	5.278	0.012	11.810	0.067	55.623	0.436
10	7	50	2.517	8.174	30.045	5.320	0.093	12.116	0.051	55.833	0.288
10	7	100	2.470	8.185	30.178	5.252	0.025	12.750	0.013	56.484	0.128
10	7	200	2.451	8.183	30.528	5.564	0.018	11.914	0.033	57.472	0.198
10	7	300	2.441	8.171	30.494	5.556	0.055	11.942	0.210	56.957	0.296
10	7	400	2.440	8.230	30.609	5.556	0.041	12.039	0.147	56.880	0.314
10	8	10	2.457	8.134	30.513	5.112	0.114	12.285	0.091	55.001	0.298
10	8	50	2.525	8.039	30.617	5.477	0.064	12.134	0.064	55.879	0.049
10	8	100	2.609	7.879	30.389	6.541	0.061	11.943	0.132	55.417	0.250
10	8	200	2.675	7.952	30.355	6.755	0.149	11.712	0.002	54.938	0.083
10	8	300	2.707	7.992	30.412	6.901	0.051	11.737	0.019	54.594	0.226
10	8	400	2.720	8.009	30.447	7.113	0.083	11.810	0.034	54.424	0.492

Table B.1: *ANFIS quality evaluation parameters - about $\theta_1, \theta_2 and \theta_3$ - step 10 deg*

Step [deg]	Infis	n epoch	MaxErrorX [cm]	MinErrorX [cm]	MaxErrorY [cm]	MinErrorY [cm]	XRMSE	YRMSE
10	2	10	1.984	0.852	1.225	0.009	1.449	0.615
10	2	50	1.759	0.796	1.171	0.018	1.368	0.639
10	2	100	1.694	0.030	1.552	0.011	1.198	0.936
10	2	200	1.483	0.024	1.157	0.003	1.106	0.786
10	2	300	1.462	0.008	1.133	0.007	1.097	0.772
10	2	400	1.441	0.004	1.130	0.010	1.086	0.759
10	3	10	1.469	0.008	1.449	0.005	0.916	0.640
10	3	50	1.384	0.002	1.334	0.009	0.880	0.619
10	3	100	1.165	0.042	0.926	0.018	0.801	0.477
10	3	200	1.109	0.146	0.760	0.013	0.769	0.523
10	3	300	1.104	0.146	0.765	0.005	0.768	0.515
10	3	400	1.105	0.153	0.759	0.004	0.768	0.512
10	4	10	1.187	0.018	0.830	0.016	0.771	0.493
10	4	50	1.113	0.001	0.739	0.003	0.746	0.475
10	4	100	1.030	0.010	0.709	0.002	0.706	0.468
10	4	200	0.998	0.007	0.653	0.007	0.690	0.471
10	4	300	0.987	0.010	0.652	0.015	0.693	0.453
10	4	400	0.991	0.007	0.654	0.009	0.692	0.451
10	5	10	1.050	0.012	0.817	0.008	0.704	0.450
10	5	50	1.061	0.004	0.794	0.003	0.698	0.454
10	5	100	1.096	0.010	0.881	0.009	0.672	0.512
10	5	200	1.116	0.008	0.975	0.001	0.672	0.469
10	5	300	1.115	0.014	0.972	0.012	0.671	0.467
10	5	400	1.107	0.005	0.879	0.004	0.666	0.461
10	6	10	1.068	0.003	0.719	0.000	0.664	0.441
10	6	50	1.043	0.015	0.707	0.012	0.660	0.451
10	6	100	0.972	0.000	0.714	0.021	0.651	0.448
10	6	200	0.997	0.004	0.681	0.025	0.648	0.460
10	6	300	1.025	0.025	0.736	0.004	0.648	0.474
10	6	400	1.031	0.001	0.785	0.014	0.648	0.486
10	7	10	1.061	0.065	0.731	0.010	0.658	0.460
10	7	50	1.059	0.074	0.724	0.019	0.654	0.463
10	7	100	1.069	0.002	0.667	0.002	0.643	0.464
10	7	200	1.059	0.004	0.738	0.005	0.644	0.460
10	7	300	1.059	0.006	0.776	0.001	0.634	0.476
10	7	400	1.067	0.013	0.791	0.002	0.643	0.473
10	8	10	1.038	0.008	0.761	0.011	0.636	0.463
10	8	50	1.098	0.013	0.770	0.001	0.620	0.471
10	8	100	1.039	0.055	0.751	0.032	0.590	0.510
10	8	200	0.970	0.012	0.745	0.031	0.592	0.525
10	8	300	0.949	0.002	0.791	0.016	0.595	0.525
10	8	400	0.978	0.004	0.832	0.011	0.598	0.521

Table B.2: *ANFIS quality evaluation parameters - about $X and Y values$ - step 10 deg*

Step [deg]	Infis	n epoch	Th1RMSE	Th2RMSE	Th3RMSE	MaxTh1Err [cm]	MinTh1Err [cm]	MaxTh2Err [deg]	MinTh2Err [deg]	MaxTh3Err [deg]	MinTh3Err [deg]
5	2	10	4.993	12.802	27.461	9.397	0.072	17.329	0.027	55.084	0.289
5	2	50	4.688	12.264	27.600	8.738	0.099	15.810	0.115	55.435	0.282
5	2	100	3.301	11.566	28.339	5.194	0.061	16.018	0.025	55.652	0.234
5	2	200	3.093	11.047	28.465	4.750	0.028	15.387	0.196	55.555	0.300
5	2	300	3.096	10.921	28.471	4.771	0.015	15.214	0.075	55.580	0.299
5	2	400	3.081	10.877	28.476	4.747	0.007	15.149	0.027	55.586	0.299
5	3	10	3.636	9.896	30.228	5.621	0.066	17.037	0.837	56.739	0.133
5	3	50	3.374	9.479	30.056	5.164	0.024	16.028	0.436	56.672	0.429
5	3	100	2.570	8.511	29.485	5.354	0.095	12.616	0.037	56.487	0.370
5	3	200	2.724	7.931	29.276	6.380	0.017	11.605	0.009	56.404	0.311
5	3	300	2.707	7.913	29.253	6.351	0.004	11.559	0.117	56.386	0.275
5	3	400	2.712	7.897	29.256	6.359	0.023	11.511	0.012	56.384	0.273
5	4	10	2.549	8.246	28.979	5.500	0.115	12.557	0.018	56.315	0.016
5	4	50	2.517	8.088	29.079	5.825	0.104	12.121	0.085	56.475	0.102
5	4	100	2.538	7.880	29.837	6.165	0.109	11.645	0.016	54.879	0.117
5	4	200	2.496	7.908	30.766	5.967	0.000	11.712	0.041	56.109	0.079
5	4	300	2.458	7.912	30.854	5.884	0.069	11.700	0.132	56.185	0.023
5	4	400	2.447	7.913	30.859	5.867	0.079	11.707	0.125	56.199	0.009
5	5	10	2.510	7.631	29.852	5.514	0.009	11.896	0.095	56.228	0.278
5	5	50	2.486	7.619	29.903	5.512	0.055	11.538	0.083	55.898	0.304
5	5	100	2.514	7.740	30.950	6.143	0.026	11.225	0.034	55.429	0.330
5	5	200	2.438	7.724	31.113	6.194	0.068	11.149	0.058	55.531	0.154
5	5	300	2.436	7.712	31.151	6.188	0.070	11.083	0.033	55.889	0.240
5	5	400	2.436	7.695	31.081	6.189	0.068	11.014	0.024	56.280	0.322
5	6	10	2.447	7.492	29.818	5.540	0.065	11.048	0.007	56.047	0.248
5	6	50	2.449	7.580	30.109	5.598	0.040	10.976	0.156	56.234	0.231
5	6	100	2.471	7.596	30.875	5.933	0.077	11.070	0.103	55.838	0.327
5	6	200	2.495	7.533	30.828	6.068	0.030	10.950	0.045	57.182	0.006
5	6	300	2.491	7.495	30.844	6.104	0.010	10.850	0.019	57.155	0.054
5	6	400	2.474	7.472	30.856	6.142	0.013	10.752	0.016	57.225	0.058
5	7	10	2.554	7.524	30.584	6.139	0.011	10.454	0.046	57.751	0.352
5	7	50	2.539	7.528	30.719	6.126	0.038	10.658	0.005	57.610	0.379
5	7	100	2.505	7.572	30.766	6.211	0.007	10.917	0.022	57.996	0.313
5	7	200	2.551	7.316	30.901	6.268	0.038	10.319	0.097	58.647	0.358
5	7	300	2.542	7.229	30.916	6.071	0.012	10.241	0.099	58.657	0.364
5	7	400	2.527	7.235	30.933	5.954	0.029	10.168	0.120	58.538	0.362
5	8	10	2.449	7.470	30.864	6.025	0.030	10.724	0.156	56.190	0.082
5	8	50	2.485	7.423	30.845	5.884	0.006	10.600	0.109	56.622	0.047
5	8	100	2.541	7.546	30.950	5.953	0.044	10.810	0.066	57.627	0.110
5	8	200	2.628	7.501	31.049	6.007	0.008	10.660	0.088	58.229	0.205
5	8	300	2.629	7.498	31.113	5.889	0.041	10.609	0.120	58.659	0.298
5	8	400	2.635	7.504	31.142	5.853	0.051	10.600	0.103	58.836	0.347

Table B.3: *ANFIS quality evaluation parameters - about $\theta_1, \theta_2 and \theta_3$ - step 5 deg*

Step [deg]	Infis	n epoch	MaxErrorX [cm]	MinErrorX [cm]	MaxErrorY [cm]	MinErrorY [cm]	XRMSE	YRMSE
5	2	10	1.872	0.713	1.179	0.008	1.359	0.605
5	2	50	1.633	0.640	1.135	0.003	1.269	0.638
5	2	100	1.605	0.027	1.655	0.021	1.131	1.022
5	2	200	1.464	0.018	1.335	0.024	1.062	0.877
5	2	300	1.428	0.030	1.290	0.032	1.045	0.854
5	2	400	1.417	0.009	1.270	0.033	1.038	0.837
5	3	10	1.407	0.002	1.337	0.005	0.872	0.575
5	3	50	1.319	0.007	1.232	0.001	0.835	0.554
5	3	100	0.993	0.076	0.728	0.008	0.709	0.396
5	3	200	0.947	0.002	0.731	0.000	0.669	0.428
5	3	300	0.950	0.006	0.719	0.001	0.669	0.420
5	3	400	0.947	0.024	0.720	0.008	0.667	0.418
5	4	10	1.054	0.007	0.794	0.014	0.702	0.453
5	4	50	0.981	0.007	0.713	0.003	0.672	0.437
5	4	100	0.934	0.003	0.596	0.003	0.620	0.428
5	4	200	0.890	0.011	0.598	0.018	0.601	0.423
5	4	300	0.879	0.013	0.599	0.012	0.606	0.399
5	4	400	0.884	0.018	0.599	0.008	0.608	0.394
5	5	10	0.941	0.012	0.773	0.002	0.614	0.408
5	5	50	0.944	0.011	0.757	0.005	0.611	0.411
5	5	100	0.892	0.008	0.707	0.006	0.591	0.432
5	5	200	0.914	0.002	0.650	0.001	0.589	0.404
5	5	300	0.930	0.010	0.645	0.009	0.593	0.397
5	5	400	0.931	0.016	0.643	0.017	0.592	0.397
5	6	10	0.995	0.003	0.708	0.002	0.597	0.419
5	6	50	0.959	0.007	0.690	0.003	0.599	0.430
5	6	100	0.888	0.000	0.638	0.004	0.570	0.441
5	6	200	0.871	0.003	0.628	0.009	0.568	0.429
5	6	300	0.864	0.002	0.619	0.011	0.564	0.425
5	6	400	0.877	0.001	0.611	0.022	0.564	0.418
5	7	10	0.915	0.041	0.697	0.008	0.583	0.425
5	7	50	0.899	0.074	0.677	0.011	0.577	0.426
5	7	100	0.901	0.000	0.661	0.004	0.582	0.418
5	7	200	1.002	0.018	0.673	0.002	0.557	0.434
5	7	300	0.943	0.001	0.671	0.010	0.549	0.435
5	7	400	0.917	0.010	0.663	0.005	0.550	0.429
5	8	10	0.936	0.014	0.706	0.006	0.568	0.431
5	8	50	0.927	0.000	0.694	0.000	0.562	0.442
5	8	100	0.895	0.070	0.676	0.001	0.571	0.431
5	8	200	0.908	0.006	0.720	0.009	0.564	0.433
5	8	300	0.947	0.005	0.715	0.015	0.563	0.435
5	8	400	0.958	0.002	0.701	0.026	0.563	0.439

Table B.4: *ANFIS quality evaluation parameters - about $X and Y values$ - step 5 deg*

Step [deg]	Infis	n epoch	Th1RMSE	Th2RMSE	Th3RMSE	MaxTh1Err [cm]	MinTh1Err [cm]	MaxTh2Err [deg]	MinTh2Err [deg]	MaxTh3Err [deg]	MinTh3Err [deg]
2	2	10	4.717	12.224	27.522	8.954	0.049	16.724	0.023	55.218	0.020
2	2	50	4.414	11.696	27.643	8.283	0.065	15.249	0.163	55.527	0.126
2	2	100	3.199	10.146	28.367	4.971	0.034	14.158	0.031	55.577	0.091
2	2	200	3.029	9.881	28.470	4.629	0.019	13.773	0.127	55.461	0.146
2	2	300	3.023	9.671	28.473	4.624	0.028	13.452	0.067	55.462	0.150
2	2	400	2.971	9.485	28.477	4.518	0.061	13.120	0.007	55.474	0.150
2	3	10	3.478	9.432	30.347	5.377	0.060	16.371	0.879	57.048	0.043
2	3	50	3.230	9.007	30.177	4.941	0.031	15.363	0.366	56.969	0.351
2	3	100	2.515	7.976	29.583	5.750	0.082	11.969	0.039	56.577	0.387
2	3	200	2.661	7.501	29.481	6.737	0.048	10.725	0.012	56.675	0.284
2	3	300	2.654	7.505	29.461	6.733	0.057	10.689	0.035	56.687	0.278
2	3	400	2.651	7.498	29.462	6.728	0.060	10.732	0.023	56.659	0.293
2	4	10	2.489	7.788	29.273	5.953	0.100	11.915	0.085	56.584	0.201
2	4	50	2.472	7.628	29.348	6.274	0.075	11.458	0.112	56.453	0.056
2	4	100	2.522	7.436	29.938	6.588	0.102	11.001	0.060	56.885	0.330
2	4	200	2.489	7.474	31.015	6.401	0.071	11.107	0.013	56.186	0.256
2	4	300	2.468	7.475	31.031	6.355	0.045	11.112	0.000	56.179	0.288
2	4	400	2.461	7.475	31.038	6.360	0.040	11.115	0.013	56.161	0.309
2	5	10	2.532	7.207	30.253	5.896	0.102	11.288	0.101	56.461	0.459
2	5	50	2.510	7.191	30.462	5.950	0.035	10.920	0.069	56.446	0.279
2	5	100	2.530	7.309	31.386	6.420	0.007	10.555	0.028	56.161	0.323
2	5	200	2.534	7.288	30.934	6.551	0.006	10.552	0.081	55.582	0.281
2	5	300	2.521	7.281	30.808	6.516	0.011	10.534	0.026	55.090	0.417
2	5	400	2.515	7.273	30.777	6.526	0.014	10.495	0.044	55.359	0.386
2	6	10	2.498	7.082	30.060	6.029	0.035	10.229	0.227	56.398	0.004
2	6	50	2.507	7.158	30.543	6.154	0.043	10.137	0.101	56.115	0.089
2	6	100	2.531	7.185	31.146	6.513	0.067	10.447	0.155	56.436	0.013
2	6	200	2.563	7.202	31.033	6.624	0.030	10.471	0.033	57.490	0.034
2	6	300	2.563	7.194	31.025	6.627	0.039	10.483	0.031	57.426	0.110
2	6	400	2.563	7.184	31.015	6.628	0.040	10.465	0.025	57.391	0.170
2	7	10	2.594	7.155	30.902	6.599	0.076	10.003	0.069	58.419	0.358
2	7	50	2.586	7.148	31.170	6.622	0.006	10.154	0.020	58.242	0.412
2	7	100	2.558	7.167	30.800	6.583	0.008	10.310	0.104	58.731	0.343
2	7	200	2.571	6.954	30.944	6.424	0.018	9.771	0.033	59.044	0.019
2	7	300	2.574	6.942	30.953	6.424	0.017	9.710	0.020	59.005	0.085
2	7	400	2.578	6.936	30.960	6.444	0.016	9.675	0.090	59.066	0.118
2	8	10	2.502	7.138	31.119	6.425	0.032	10.078	0.025	57.237	0.103
2	8	50	2.535	7.104	31.096	6.433	0.018	9.949	0.029	57.931	0.004
2	8	100	2.597	7.148	30.986	6.638	0.058	10.190	0.022	58.281	0.080
2	8	200	2.613	7.085	31.164	6.715	0.007	9.793	0.033	58.173	0.092
2	8	300	2.615	7.071	31.199	6.708	0.032	9.863	0.045	58.130	0.180
2	8	400	2.628	7.060	31.211	6.647	0.022	9.879	0.032	58.139	0.293

Table B.5: *ANFIS quality evaluation parameters - about $\theta_1, \theta_2 and \theta_3$ - step 2 deg*

Step [deg]	Infis	n epoch	MaxErroXr [cm]	MinErrorX [cm]	MaxErrorY [cm]	MinErrorY [cm]	XRMSE	YRMSE
2	2	10	1.795	0.632	1.113	0.001	1.294	0.563
2	2	50	1.570	0.560	1.060	0.020	1.209	0.582
2	2	100	1.340	0.027	1.169	0.006	0.988	0.785
2	2	200	1.283	0.024	1.040	0.001	0.970	0.695
2	2	300	1.240	0.002	1.055	0.007	0.956	0.669
2	2	400	1.210	0.014	1.021	0.003	0.940	0.624
2	3	10	1.341	0.001	1.285	0.003	0.823	0.553
2	3	50	1.252	0.007	1.179	0.003	0.784	0.532
2	3	100	0.923	0.068	0.699	0.011	0.656	0.377
2	3	200	0.888	0.003	0.698	0.008	0.619	0.405
2	3	300	0.886	0.000	0.694	0.006	0.621	0.401
2	3	400	0.886	0.008	0.690	0.005	0.620	0.399
2	4	10	0.994	0.018	0.759	0.009	0.649	0.433
2	4	50	0.913	0.010	0.681	0.010	0.616	0.424
2	4	100	0.863	0.000	0.564	0.000	0.576	0.400
2	4	200	0.834	0.007	0.545	0.006	0.553	0.395
2	4	300	0.815	0.004	0.546	0.009	0.552	0.377
2	4	400	0.813	0.003	0.550	0.011	0.554	0.372
2	5	10	0.876	0.013	0.744	0.011	0.566	0.388
2	5	50	0.872	0.005	0.729	0.015	0.563	0.389
2	5	100	0.797	0.002	0.691	0.011	0.536	0.403
2	5	200	0.797	0.002	0.645	0.001	0.528	0.395
2	5	300	0.815	0.000	0.639	0.003	0.537	0.386
2	5	400	0.823	0.008	0.637	0.001	0.539	0.383
2	6	10	0.933	0.002	0.680	0.002	0.554	0.403
2	6	50	0.905	0.001	0.669	0.006	0.554	0.416
2	6	100	0.907	0.010	0.629	0.007	0.543	0.419
2	6	200	0.815	0.000	0.618	0.001	0.535	0.396
2	6	300	0.814	0.004	0.617	0.005	0.535	0.397
2	6	400	0.813	0.009	0.616	0.003	0.535	0.397
2	7	10	0.852	0.047	0.664	0.005	0.540	0.403
2	7	50	0.830	0.057	0.640	0.003	0.534	0.408
2	7	100	0.821	0.003	0.618	0.002	0.541	0.384
2	7	200	0.850	0.005	0.636	0.001	0.528	0.380
2	7	300	0.851	0.007	0.643	0.001	0.531	0.379
2	7	400	0.850	0.019	0.643	0.001	0.531	0.382
2	8	10	0.880	0.008	0.669	0.003	0.531	0.411
2	8	50	0.867	0.016	0.658	0.007	0.526	0.419
2	8	100	0.835	0.014	0.629	0.012	0.536	0.402
2	8	200	0.850	0.004	0.630	0.010	0.544	0.386
2	8	300	0.857	0.002	0.629	0.034	0.543	0.383
2	8	400	0.865	0.012	0.624	0.009	0.542	0.387

Table B.6: *ANFIS quality evaluation parameters - about $X and Y values$ - step 2 deg*

Bibliography

[1] Ankarali A. and Cilli M. "ANFIS Inverse Kinematics and Hybrid Control of a Human Leg Gait Model". In: *Academic Platform Journal of Engineering and Science* 1 (Jan. 2013), pp. 34–49.

[2] Dongare A., Kharde R. R., and Kachare A. D. "Introduction to Artificial Neural Networks". In: (2012).

[3] J. M. Blain. *The complete guide to Blender graphics: computer modeling & animation*. English. CRC Press. 2016.

[4] Giulia Caserta. "Evaluation and improvement of the embodiment of an upper limb prosthesis: the Hannes system". In: (2021).

[5] Jennifer Cheesborough et al. "Targeted Muscle Reinnervation and Advanced Prosthetic Arms". In: *Seminars in Plastic Surgery* 29.01 (2015), pp. 062–072. ISSN: 1535-2188. DOI: 10.1055/s-0035-1544166. URL: https://dx.doi.org/10.1055/s-0035-1544166.

[6] J. C. K. Chou. "Quaternion kinematic and dynamic differential equations". In: *IEEE Transactions on Robotics and Automation* 8.1 (1992), pp. 53–64. ISSN: 1042-296X. DOI: 10.1109/70.127239. URL: https://dx.doi.org/10.1109/70.127239.

[7] Francesca Cordella et al. "Literature review on needs of upper limb prosthesis users". In: *Frontiers in neuroscience* 10 (2016), p. 209.

[8] Armaghani D.J. and Asteris P.G. "A comparative study of ANN and ANFIS models for the prediction of cement-based mortar materials compressive strength". In: *Neural Comput and Applic.* 33 (2021), pp. 4501–4532.

[9] Chen D.W. and Zhang J.P. "Time series prediction based on ensemble ANFIS." In: *2005 International Conference on Machine Learning and Cybernetics* (2005), pp. 3552–3556.

[10] Dario Di Domenico. "Hannes Prosthesis Control Based on Regression Machine Learning Algorithms". Politecnico di Torino, 2020.

[11] D. Erdmann. *Fast IK*. URL: http://www.giuseppesottile.it/mathfis/quaternioni.php.

[12] H. Flor. "Phantom-limb pain: characteristics, causes, and treatment". In: *The Lancet Neurology* Volume 3 (), Pages 182–189. ISSN: 1474-4422.

[13] Anders Fougner et al. “Control of Upper Limb Prostheses: Terminology and Proportional Myoelectric Control—A Review”. In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 20.5 (2012), pp. 663–677. ISSN: 1534-4320. DOI: 10.1109/tnsre.2012.2196711. URL: https://dx.doi.org/10.1109/tnsre.2012.2196711.

[14] Robert Gailey et al. “Unilateral lower-limb loss: Prosthetic device use and functional outcomes in servicemembers from Vietnam war and OIF/OEF conflicts”. In: *The Journal of Rehabilitation Research and Development* 47.4 (2010), p. 317. ISSN: 0748-7711. DOI: 10.1682/jrrd.2009.04.0039. URL: https://dx.doi.org/10.1682/jrrd.2009.04.0039.

[15] Purushothaman Geethanjali. “Myoelectric control of prosthetic hands: state-of-the-art review”. In: *Medical Devices: Evidence and Research* Volume 9 (2016), pp. 247–255. ISSN: 1179-1470. DOI: 10.2147/mder.s91102. URL: https://dx.doi.org/10.2147/mder.s91102.

[16] Will Goldstone. *Unity game development essentials*. Packt Publishing Ltd, 2009.

[17] Harold H. Sears and Julie Shaperman. “Proportional myoelectric hand control: an evaluation”. In: *American Journal of Physical Medicine & Reabilitation* (1991).

[18] Zach T Harvey et al. “Prosthetic advances”. In: *Journal of surgical orthopaedic advances* 21.1 (2012), p. 58.

[19] Dembys J., Gao Y., and Desouza G.N. “A Study on Solving the Inverse Kinematics of Serial Robots using Artificial Neural Network and Fuzzy Neural Network”. In: *International Journal of Computers, Communications and Control* (June 2019), pp. 1–6.

[20] A. Klein, C. Chu-Jenq, and S. Ahmed. “A new formulation of the extended Jacobian method and its use in mapping algorithmic singularities for kinematically redundant manipulators”. In: *IEEE Transactions on robotics and automation* (1995).

[21] Serdar Küçük. *Serial and parallel robot manipulators: Kinematics, dynamics, control and optimization*. BoD–Books on Demand, 2012.

[22] T. A. Kuiken et al. “The use of targeted muscle reinnervation for improved myoelectric prosthesis control in a bilateral shoulder disarticulation amputee”. In: *Prosthetics & Orthotics International* 28.3 (2004), pp. 245–253. ISSN: 0309-3646. DOI: 10.3109/03093640409167756. URL: https://dx.doi.org/10.3109/03093640409167756.

[23] Wei L.X., Wang H.R., and Li Y. “A new solution for inverse kinematics of manipulator based on neural network”. In: *International Journal of Computers, Communications and Control* 2 (2003), pp. 1201–1203.

[24] M. Laffranchi et al. “The Hannes hand prosthesis replicates the key biological properties of the human hand”. In: *Science Robotics* (2020).

[25] Hong-You Lee and Charles F Reinholtz. “Inverse kinematics of serial-chain manipulators”. In: (1996).

[26] I-Min Lee and David M Buchner. "The importance of walking to public health". In: *Medicine & Science in Sports & Exercise* 40.7 (2008), S512–S518.

[27] A. Liégeois. "Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms". In: *IEEE Transactions on systems, man, and cybernetics* SMC - 7 (1977).

[28] D. S. McKenzie. "THE RUSSIAN MYO-ELECTRIC ARM". In: *J Bone Joint Surg Br* 47 (1965), pp. 418–20. ISSN: 0301-620X (Print) 0301-620x.

[29] M Merad et al. "Intuitive prosthetic control using upper limb inter-joint coordinations and IMU-based shoulder angles measurement: a pilot study". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2016).

[30] D. R. Merrill et al. "Development of an implantable myoelectric sensor for advanced prosthesis control". In: *Artif Organs* 35.3 (2011), pp. 249–52. ISSN: 1525-1594 (Electronic) 0160-564X (Linking). DOI: 10.1111/j.1525-1594.2011.01219.x. URL: https://www.ncbi.nlm.nih.gov/pubmed/21371058.

[31] Ingram Andrew Murray. "Determining upper limb kinematics and dynamics during everyday tasks". PhD thesis. Newcastle University, 1999.

[32] Y. Nakamura, H. Hanafusa, and T. Yoshikawa. "Task-Priority Based Redundancy Control of RObot Manipulators". In: *The International Journal of Robotics Research* 6 (1987).

[33] Max Ortiz-Catalan et al. "On the viability of implantable electrodes for the natural control of artificial limbs: Review and discussion". In: *BioMedical Engineering OnLine* 11.1 (2012), p. 33. ISSN: 1475-925X. DOI: 10.1186/1475-925x-11-33. URL: https://dx.doi.org/10.1186/1475-925x-11-33.

[34] Jang J.-S. R. "ANFIS: adaptive-network-based fuzzy inference system". In: *IEEE Transactions on Systems, Man, and Cybernetics* 23(3) (1993), pp. 665–685.

[35] Lorenzo Rapetti et al. "Model-Based Real-Time Motion Tracking Using Dynamical Inverse Kinematics". In: *Algorithms* 13.10 (2020), p. 266. ISSN: 1999-4893. DOI: 10.3390/a13100266. URL: https://dx.doi.org/10.3390/a13100266.

[36] A. Saradjian, A. R. Thompson, and D. Datta. "The experience of men using an upper limb prosthesis following amputation: positive coping and minimizing feeling different". In: *Disabil Rehabil* 30.11 (2008), pp. 871–83. ISSN: 0963-8288 (Print) 0963-8288 (Linking). DOI: 10.1080/09638280701427386. URL: https://www.ncbi.nlm.nih.gov/pubmed/17852212.

[37] E. Soldati. *Stato dell'arte delle protesi di arto superiore*. Generic. 2015. DOI: 10.13140/RG.2.1.1017.2565.

[38] G. Sottile. *H come Hamilton: Quaternioni, Ipercomplessi e rotazioni*. URL: https://assetstore.unity.com/packages/tools/animation/fast-ik-139972.

[39] André T Sugawara et al. "Abandonment of assistive products: assessing abandonment levels and factors that impact on it". In: *Disability and Rehabilitation: Assistive Technology* 13.7 (2018), pp. 716–723.

[40] Kamel T.S., Moustafa Hassan M.A., and El-Morshedy A. *Using a Combined Artificial Intelligent Approach In Distance Relay For Transmission Line Protection In EPS*. Sept. 2009, pp. 1–6.

[41] Alan J. Thurston. "PARÉ AND PROSTHETICS: THE EARLY HISTORY OF ARTIFICIAL LIMBS". In: *ANZ Journal of Surgery* 77.12 (2007), pp. 1114–1119. ISSN: 1445-1433. DOI: 10.1111/j.1445-2197.2007.04330.x. URL: https://dx.doi.org/10.1111/j.1445-2197.2007.04330.x.

[42] S. Traversaro and A. Saccon. *Multibody dynamics notation*. English. Dept. of Mechanical Engineering. Report locator DC 2016.064. Technische Universiteit Eindhoven, July 2016.

[43] Ivan Vujaklija, Dario Farina, and Oskar Aszmann. "New developments in prosthetic arm systems". In: *Orthopedic Research and Reviews* Volume 8 (2016), pp. 31–39. ISSN: 1179-1462. DOI: 10.2147/orr.s71468. URL: https://dx.doi.org/10.2147/orr.s71468.

[44] Jun Wan et al. "A study on avoiding joint limits for inverse kinematics of redundant manipulators using improved clamping weighted least-norm method". In: *Journal of Mechanical Science and Technology* 32.3 (2018), pp. 1367–1378. ISSN: 1738-494X. DOI: 10.1007/s12206-018-0240-7. URL: https://dx.doi.org/10.1007/s12206-018-0240-7.

[45] Andrea Maria Zanchettin et al. "Kinematic analysis and synthesis of the human arm motion during a manipulation task". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 2692–2697. DOI: 10.1109/ICRA.2011.5979654.

[46] M Zecca et al. "Control of multifunctional prosthetic hands by processing the electromyographic signal". In: *Critical Reviews™ in Biomedical Engineering* 45.1-6 (2017).