UNIVERSITÀ POLITECNICA DELLE MARCHE

FACULTY OF ENGINEERING

Master’s degree in Biomedical Engineering

Design and Programming of a Rehabilitative Collaborative Robotics Station Enslaved by Artificial Vision for Neuromuscular Rehabilitation

Advisor:

Dr. Giacomo Palmieri

Student:

Alessandro Bottiglione

Academic Year 2020-2021

I. Contents

II. Tables' list

III. Figures' List

IV. Abstract

Collaborative robots, also known as cobots, are particularly safe and reliable and are therefore involved in several sectors, including the medical one, especially in rehabilitation. Unlike traditional robots, the new generation of cobots is designed to be flexible, lightweight and easy to be programmed. The purpose of this thesis is to design a rehabilitation robotics station composed by a cobot produced by Universal Robot (UR5e) and an external vision system produced by Cognex (Cognex In-Sight 7600). In the first part of this study, an optimisation algorithm has been developed to determine the optimal position of the patient with respect to the robot base to design the layout of the rehabilitation station. This algorithm has been entirely developed in Matlab environment and it is divided into three phases: definition of a reachable cloud of points of the upper limb, an optimisation method based on the definition of a global quality index defined in the fronto-lateral region of the patient using manipulability ellipsoids of the human upper limb and the robotic arm, finally a local optimisation based on the results of the global optimisation. The second part of the work focuses on the integration of the vision system with the cobot to locate a moving target on the working plane and define the trajectory of the robot to reach the target. The programming of the cobot has been entirely developed in Polyscope software. The development requires an initial hand-eye calibration phase and a pattern recognition by the vision system developed entirely in Explorer In-Sight software. To perform the rehabilitation task, the upper limb is constrained to a handpiece placed on the end-effector of the robot. In conclusion, this thesis allows the design of the collaborative robotics station that can be used in a clinical environment, outlining the optimal position between the patient and an initial rehabilitation exercise set-up. Future research could include the definition of an individual rehabilitation task for each type of patient to improve the quality of the rehabilitation process.

V. Abstract

I robot collaborativi, conosciuti anche come cobot, risultano essere particolarmente sicuri e affidabili sono quindi coinvolti in diversi settori, tra cui quello medico, in particolare nella riabilitazione. A differenza dei robot tradizionali, la nuova generazione di cobot è progettata per essere flessibile, leggera e facile da programmare. L'obiettivo di questa tesi è di progettare una stazione di robotica riabilitativa composta da un robot collaborativo prodotto da Universal Robot (UR5e) e un sistema di visione esterno prodotto da Cognex (Cognex In-Sight 7600). Nella prima parte del lavoro, è stato definito un algoritmo di ottimizzazione in grado di definire la posizione ottimale del paziente rispetto alla base del robot per realizzare il layout della stazione di riabilitazione. Questo algoritmo è stato interamente sviluppo in ambiente Matlab ed è diviso in tre fasi: definizione di una nuvola di punti raggiungibile dell'arto superiore, ottimizzazione basata sulla definizione di un indice globale di qualità definito nella regione fronto-laterale del paziente mediante ellissoidi di manipolabilità dell'arto superiore e della struttura robotica permettendo una mappatura globale, infine un'ottimizzazione locale basata sui risultati dell'ottimizzazione globale. La seconda parte del lavoro si concentra sull'integrazione del sistema di visione con il cobot per localizzare un target mobile sul piano di lavoro e definire la traiettoria del robot. La programmazione del cobot è stata interamente sviluppata in Polyscope. Per definire la traiettoria è necessaria una prima fase di calibrazione hand-eye e il riconoscimento del pattern da parte del sistema di visione sviluppo interamente nel software Explorer In-Sight. Per eseguire questo task riabilitativo l'arto superiore è vincolato ad un manipolo posto sull'end-effector del robot. In conclusione, questo lavoro di tesi permettere di progettare la stazione di robotica collaborativa utilizzabile in ambiente clinico, delineando la posizione ottimale tra il paziente e un primo set-up di esercizio riabilitativo. La ricerca futura potrebbe prevedere la definizione di una task riabilitativa propria per ogni tipo di paziente per migliorare la qualità del processo di riabilitazione.

1. Introduction

Conventional rehabilitation training for regaining motor functions requires therapists' assistance, which is a labour-intensive and repetitive task for therapists. Therefore, there is a strong need for rehabilitation robots to assist, improve and assess the rehabilitation training process for stroke patients. Collaborative robots are adopted not only in industrial applications, but also in the medical field. In particular, the reliability and security of collaborative robots have applications in neuromuscular rehabilitation. Since the increasing scientific and technology potential and the rapid ageing of the society, many research groups have proposed robots to make the rehabilitation process easier. The purpose of this study is to build up a first setup of a rehabilitative station for patient affected by neuromuscular disease, executing a task where the robot is guided toward a target. This station is composed by the collaborative robot UR5e of Universal Robot company and the Cognex In-sight 7600, a smart camera of Cognex company. Firstly, the vision system acquires and elaborates the position of the target, then it transmits the information to the controller of the robotic arm. Subsequently, the robotic arm guides the upper limb toward the target. Thus, this rehabilitative station has been designed and programmed to complete a task where the patient is guided toward a target. To complete this study, two different phases have been finalized. The first step, developed entirely on Matlab environment, is related to the design of an optimization algorithm capable to find the position between the base of the robotic arm and the right shoulder of the patient using a quality index based on the manipulability ellipsoids. The second step is the programming and definition of the rehabilitative task completed using Polyscope. The calibration and the robot guidance are completed using Explorer In-Sight software. At the end, the aim of this rehabilitative process is to stimulate the patient neuroplasticity that is the ability of neurons to modify the strength of existing synapses [1].

1.1 Collaborative Robot for Rehabilitation Process

Since collaborative robots show high reliability and safety level, they have been applied for a new purpose: neuromuscular rehabilitation. In fact, these robotic structures can be widely applied in motor rehabilitation process of the upper limb in patients affected by neurological disease, spinal cord injury or ictus [2]. The major distinction between robot-guided and classic rehabilitation process is the possibility to reach, using collaborative robots, reliable measures of physical proprieties with level of speed, accuracy, power and endurance over time. Furthermore, among all the advantages in the usage of robot-guided rehabilitation application, there are the best reproducibility, as well as a proper limb monitoring, supervised by the physiotherapist [3]. Reliability in the execution of repetitive task is very high. In fact, the application of collaborative robots in rehabilitation permits to compensate for the patient's insufficient strength and in the meantime the robot is capable to continuously transmit feedbacks for the subjective perception of improvement. These features make robotics a potential support in the rehabilitation process for both patients and trainers [4]. Currently, there are not guidelines regarding the manner to choose the robotic intervention, as well as how to customize it on the patient's residual functional abilities because the picture of stroke patient is complex [5]. Nowadays, on the market, there are some rehabilitation devices as MOTORE and PABLO. The first medical device is used to restore the arm and shoulder functionality, based on a planar movement of the limb on a work plane getting the visual feedback. The main limitation of this instrument is related to the admissible movement that can be only planar, as shown in Figure 1. On the other hand, as shown in Figure 2, PABLO is based on wireless sensors used only for specific tasks as rehabilitation of the force hand during gripping, without force feedback. Other medical devices, present in the market, have been designed for the rehabilitation of the fingers without involving the entire limb.

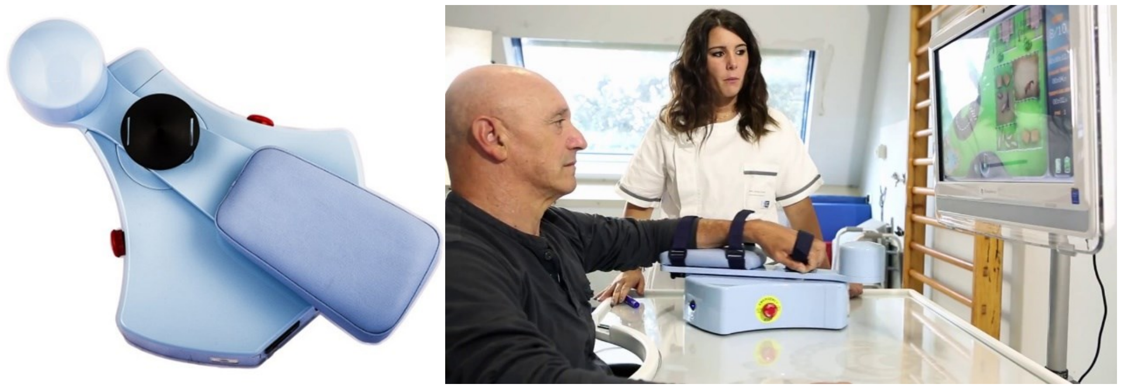


Figure 1. On the left MOTORE used for upper limb rehabilitation. On the right, an application of the medical device during the rehabilitation process.

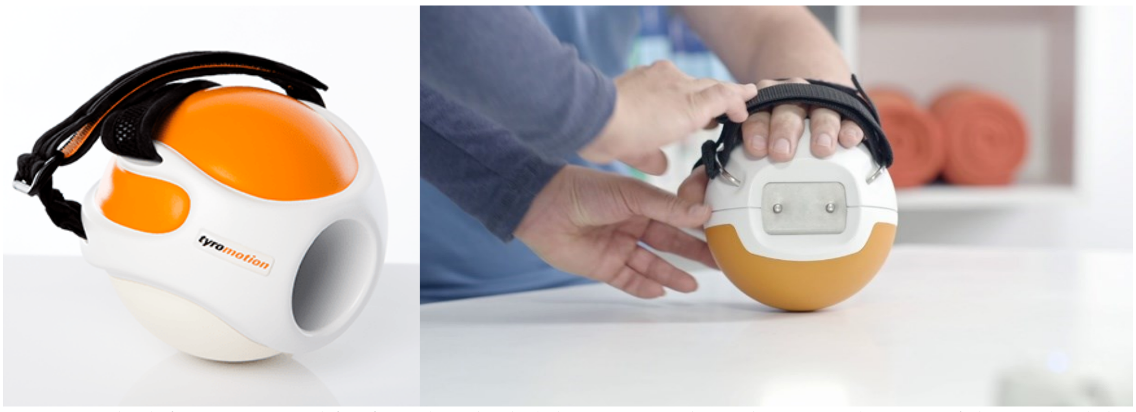


Figure 2. On the left PABLO used for force hand rehabilitation. On the right, an application of the medical device during the rehabilitation process.

In the last years, the rehabilitation system is moving towards the collaborative robots with handle systems on their end-effector. This solution promotes the investigation of robot-human interface, that can be characterized by simple handle, or a glove assisted by a camera system, as shown in Figure 3. This solution permits to solve complex tasks in the three-dimensional workspace [6]. An important advantage of end-effector based robot is related to the possibility to adjust easily the system providing a custom-made approach for the patient [7].

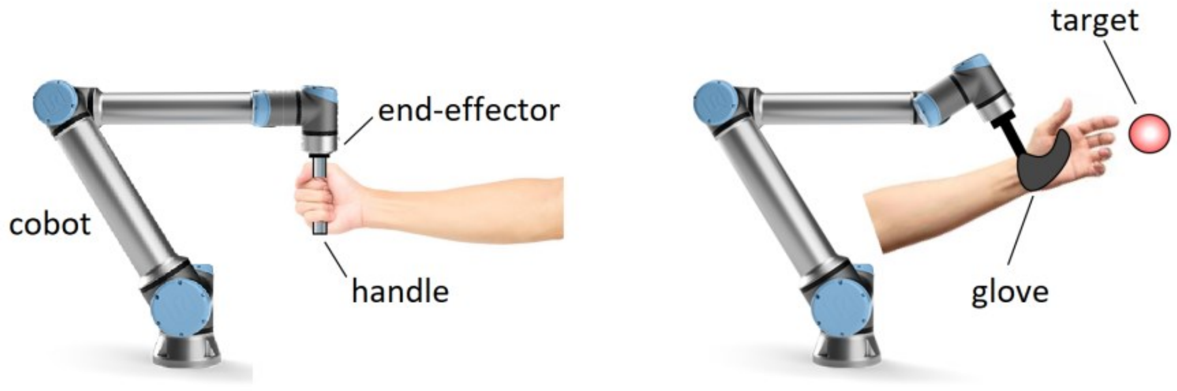


Figure 3. Possible human-robot interface.

The robot can be programmed to work in different modalities, according to the severity of injury or the rehabilitation process. The first is the passive modality, where the patient is relaxed and the robot is capable to move the limb, following an already planned trajectory. This kind of solution is useful to measure and explore the joint motion range of the patient. The second modality, the active-assisted one, on the other hand is based on the initiation of the movement that must be triggered by the patient with his own force and the force has to be kept during the task, in the meantime, the robot guides and supports the limb movement in order to allow the achievement of the exercise. The third is the active modality, the robot does not facilitate the task but on the contrary is capable to provide a viscous resistance in the desired trajectory and, during the task, all the other directions are blocked. This modality permits to the patient to reach the target with maximal effort [8]. In Table 1 are presented the rehabilitation exercises according to the functional performance of the patient.

Table 1. Manual muscle test and exercise types [9].

Class	Description	Exercise type
Zero	No palpation of any spasm	Passive
Very Weak	No movement of any joint, but small amount of spasm	Passive
Weak	Muscle spasm: when gravity is removed, the completion of the movement occurs	Active Assistive
Moderate	Completion of joint mobility against gravity	Active (No Resistance)
Good	Completion of full joint movement against a middle-level resistance and gravity	Active (Resistive)
Normal	Completion of full joint mobility against the maximum resistance and gravity	Active (Resistive)

Different solutions could be found to show the application of cobots during the rehabilitation process. As for regards the early mobilization of patients, ROBERT, an end-effector based robot, is already employed during the rehabilitation process. This solution provides the active mobilization, where the patient's function and strength are collected to determine the activity level during the exercise, and the passive mobilization for immobilized patients, as shown in Figure 4. ROBERT is easy to be set and used to provide an intense treatment, without increasing the team costs. On the other hand, Universal RoboTrainer, as shown in Figure 5, from Universal Robots company is a robot that finds its major potential as a training partner for patients with injury caused by stroke. Based on literature findings, it has been demonstrated that numerous repetitions of training exercise are an efficient way to help the human brain to learn to control paralysed muscles to restore gradually functional abilities, this capacity of the brain is called neuroplasticity. According to exercise, the robot is capable to help the patient needs because of the presence of appropriate sensors and allow the correct task execution. Another end-effector industrial based robot customized for rehabilitation purpose is Mitsubishi PA-10, as shown in Figure 6.

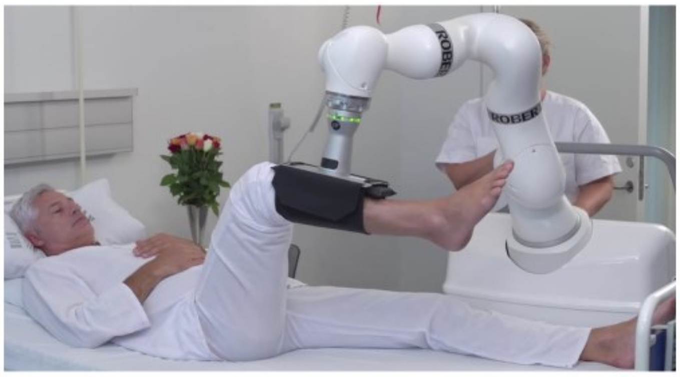


Figure 4. ROBERT Life Science.

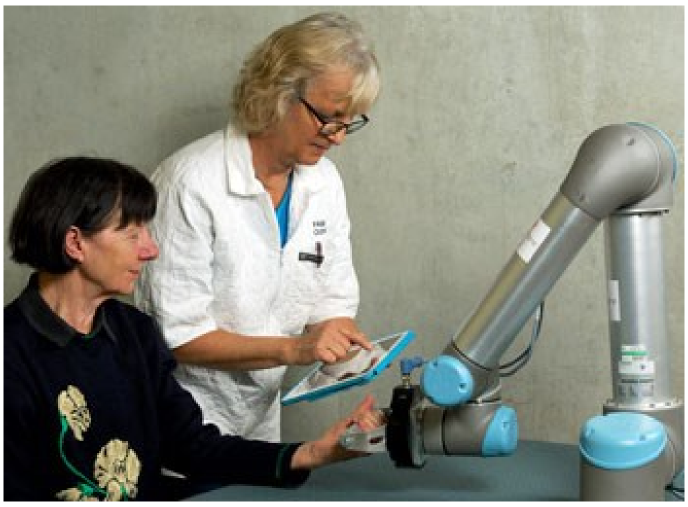


Figure 5. Universal RoboTrainer.

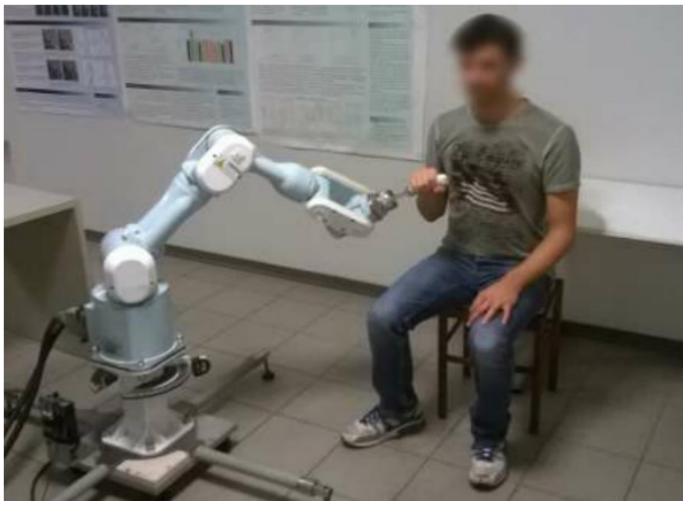


Figure 6. The MitsubishiPA10-7 robot platform.

The main purpose of this robot is to guide the rehabilitation for patients affected by neurological disease. To accomplish this intent, optical systems are integrated in Mitsubishi PA-10 to record and acquire the movement of the arm focusing on the patient deficit and determine custom-made rehabilitation protocols. The progress of technology allows to adapt the robot in response to patient's behaviour dynamically. It has been deducted from many studies and trials that collaborative robots applied for rehabilitation process will improve the quality of the physical exercise. Moreover, because of the possibility to accomplish task in the three-dimensional space, characterized by all the points reachable by both the robot and the patient, all the joints of the upper limb are involved during the treatment. In particular, the recovery time using cobots, instead of traditional therapy, is reduced and especially in movements as the forearm flexion or shoulder addiction. The economic impact, another aspect to be investigated, results to be moderate because robots available to perform upper limb rehabilitation cost about 30k euros that is a relatively low cost for cobots. Automating rehabilitation training for the affected upper limb by robotic system has been proposed to increase the number of repetitions of exercise without overburdening therapists and to maximize the patient's attention and effort as well. To increase the motivation context during the rehabilitation, the robotic therapy could be often coupled with computer gaming environment. This solution facilitates the subject learning and make the improvement faster. As a result, this is new therapy approach decreases the recovery of the patient, thus, lowers the costs for healthcare systems. In the end, the use of robots in the medical field guarantees a high intensity of treatment and the monitoring, or changing, of rehabilitation processes at distance, promoting patient independence and permits to properly stimulate the neuroplasticity of the patient.

1.1.1 Universal Robot UR5e

Unlike traditional robot that are big and expensive, this new generation of collaborative robots is designed to be flexible, easy to program and safe for every cooperation activity. The Universal Robot (UR), funded in 2005, has developed three types of machines: UR3, UR5 and UR10, as shown in Figure 7. These collaborative robots can be employed in different industrial activities and, depending on the application, it is possible to choose the best structure that best fit the environment and the load to be moved.

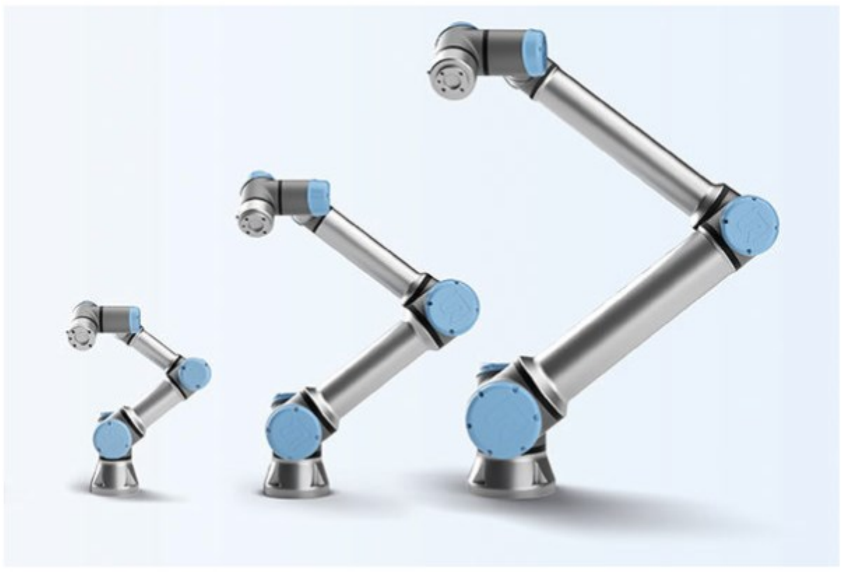


Figure 7. Universal Robot: UR3, UR5, UR10.

A new series of robots, improving the previous ones, has been recently developed by UR company. The e-series cobots (eUR3, eUR5, eUR10) are equipped with more intuitive programming and additional components to be employed in multiple industry production. They are more versatile with an increased number of safety functions. However, being the e-series an upgrade of the UR versions, technical specifications, the mechanical and electrical structures are the same. According to the payload (kilograms), each robot is named. In fact, the UR3 is the smallest one and it suits perfectly light assembly. While, the largest robot arm (UR10), instead, is the one with most power capable to reach radius of up to 1300 mm.

The most widespread and usable for rehabilitation purpose is the UR5, slightly bigger than UR3 and strikes the perfect balance between size and power. In particular, if the force action on the tool-centre-point (TCP) exceeds 25 Kgm/s the system has internal controller safety mechanism that stops all movements of the manipulator within 500 ms. This solution is ideal for automating low-weight processing tasks (up to 5 Kg) as picking, placing and testing. This structure can reach 850 mm from the base joint as maximum distance in its workspace that is combined with compact measures (about 20 Kg with a footprint of 148 mm). The robot, as shown in Figure 8, is an arm with 6 DoF composed by extruded aluminium and six joints that characterize the base, shoulder, elbow and three wrist joints. All the joints are involved in the movement of the end-effector [10].

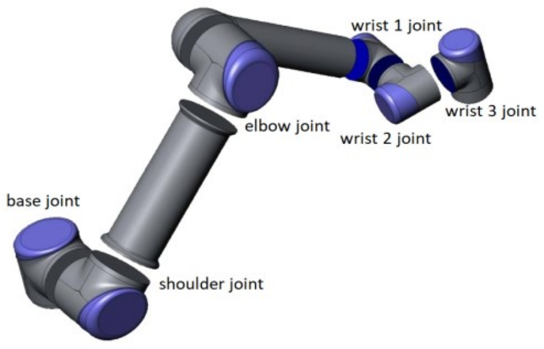


Figure 8. UR5 joints.

The UR5 weight is 18.4 Kg with each joint range of ± 360°. The speed limits of all joints correspond to ± 180°/s, while the tool speed is typical 1 m/s. The repeatability is ± 0.1mm/± 0.0039 and the robot can work in a temperature range of 0-50°C [10]. The electrical equipment of UR5 is composed by a controller, a small connector at the tool end that provides power and control signals for gripper and sensors, an Ethernet connection is provided at the bottom of the control box. To energize the robot, the control box needs to be connected to the mains with an input voltage range of 100-240 VAC and input frequency range of 47-63 Hz [10]. Some of the main technical specifications of the UR robots are reported in the Table 2.

Table 2. Technical Specifications [10].

		UR3	UR5	UR10
Weight		11 Kg	18.4 Kg	28.9 Kg
Payload		3 Kg	5 Kg	10 Kg
Reach		500 mm	850 mm	1300 mm
Joint Ranges		± 360°	± 360°	± 360°
Speed	Base Joint	± 180°/s	± 180°/s	± 180°/s
	Shoulder Joint	± 180°/s	± 180°/s	± 180°/s
	Elbow Joint	± 180°/s	± 180°/s	± 180°/s
	1° Wrist Joint	± 180°/s	± 180°/s	± 180°/s
	2° Wrist Joint	± 180°/s	± 180°/s	± 180°/s
	3° Wrist Joint	± 180°/s	± 180°/s	± 180°/s
	Tool	1 m/s	1 m/s	1 m/s

The robot can be programmed with UR script language or directly through the interface using the teach pendant, as shown in Figure 9. One of the most important advantage using the remote script commands is related to the possibility to develop and edited while the robot is running. Different programming languages as C++, Java, Python, permit to control remotely the robot. On the other hand, using the teach pendant the program has to be stopped while editing, and the actual work of the robot has to be stopped. The main advantage to use teach pendant is related to a easily execution using Polyscope interface. Furthermore, the UR5 can be moved directly by the operator manually, adopting the self-learning modality, in which the robot records each point of the track to define a motion. In manual hand-guiding applications with linear movements, is suggested to put the joint speed limit of the base and shoulder at 40°/s to prevent fast movements.

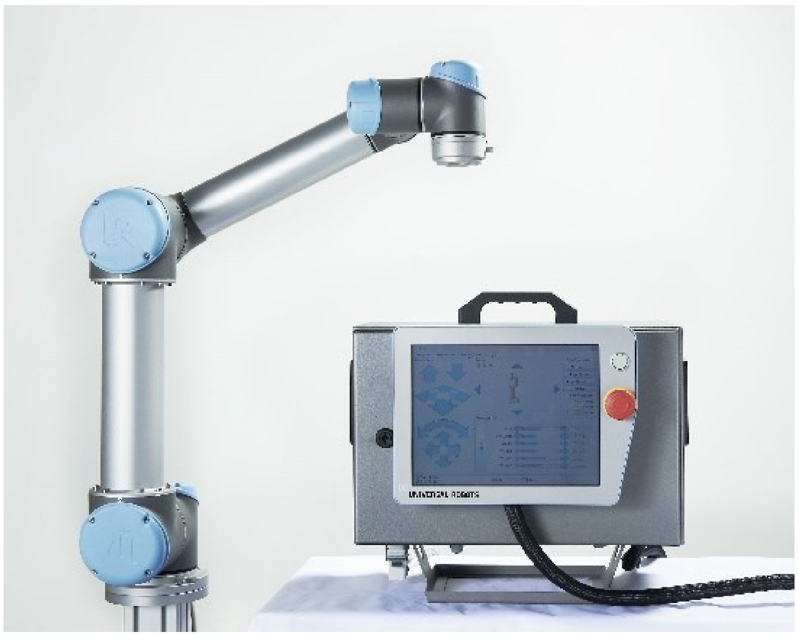


Figure 9. UR10 and teach pendant with Polyscope Robot User Interface.

According to EN ISO 10218, the robot can operate in close proximity to humans and has mechanisms that allow the protection system if external forces over 150 N is applied on the robot [11]. The basic limiting safety-related functions are reported in Table 3, designed for collaborative robot applications. Also advanced limiting safety functions have been reached. Indeed, when the robot is no more in normal conditions (e.g. in human collision), the reduced configuration is adopted, in which more restrictive safety limits are present. In case of violation of safety limits, the robot stops, and the safety system has to be restarted by the recovery mode. In this modality, the robot has to be moved within the ranges using manually movement. In extreme cases, there is the stop button that block the structure immediately. Moreover, the robot can be easily assembled and installed on the floor, on the wall, on the ceiling and even on a moving platform. Hence, these cobots can be easily re-deployed to multiple applications without changing the production layout. Easy programming, fast set-up, flexible deployment and safe collaborative applications are the main advantages of the UR production.

Table 3. Basic limiting safety-related functions in UR5 [10].

Limiting Safety Function	Description
Joint Position	Min. and Max. angular joint position
Joint Speed	Max. angular joint speed
TCP Position	Planes in Cartesian space limiting robot TCP position
TCP Speed	Max. speed of the robot TCP
TCP Force	Max. pushing force of the robot TCP
Momentum	Max. momentum of the robot arm
Power	Max. applied robot arm power

The robot can be adapted to various industry processes with different accessories: from grippers for end-effector, to vision system and software, as shown in Figure 10. In this study, the robotic structure has been coupled with a vision system that is compatible with the robot itself. This choice permits to have the guidance of the human upper limb on the target. In fact, the vision system is capable to acquire the frame and process it, finding the target. Then, the smart camera communicates directly with the controller of the robotic structure. In particular, the position and orientation of the target on a specific plane is shared with the collaborative robot. Before completing the guidance process, it is necessary to calibrate the camera with respect to the end-effector of the robotic structure. Finally, since the human upper limb and the robot end-effector move together because of tools, compatible with the robot end-effector, developed to perform the rehabilitative task.

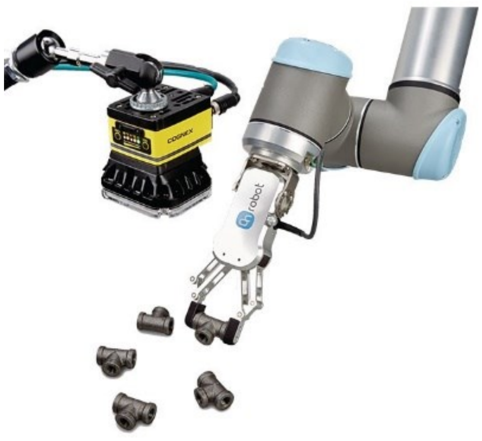


Figure 10. Example of gripper and vision system suitable for UR5.

1.2 **Smart Camera**

In industrial applications, smart cameras have been employed for twenty years but improvement in processor technologies have made these devices more accessible within the past seven years. There is not a widespread agreement upon the definition of the smart camera. However, it is generally accepted that the smart camera does not include only the image sensor, but also some processing device. The most important advantage of the smart camera is related to the output that is not an image but a decision or information. The result of the processing phase can be transmitted directly to another device connected to the vision sensor network [12]. A typical block scheme representing the main functions of a smart camera is shown in Figure 11.

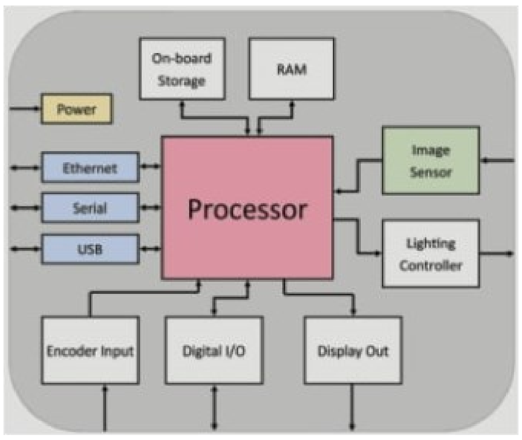


Figure 11. A smart camera typically includes an image sensor, processor, and some type of I/O. Other features can be added, such as direct drive lighting control, industrial I/O, and a display port.

Some essential proprieties must be investigated in smart cameras as the size, image sensor, ruggedness and integration. One benefit to using a smart camera is that multiple components of a vision system are integrated into a single package, resulting in a small size and the potential to save a lot of space. With sizes smaller than 55 x 50 mm and weights of less than 60 grams, these types of smart cameras have been significantly growing in market availability [12].

Being a camera, also smart camera must acquire images. Both CMOS and CCD sensors can be found in smart cameras with resolutions of 5 MP, available in both colour and monochrome. Smart cameras are not just limited to area scan anymore. Line-scan smart cameras are also available with frequencies over 10 kHz. Most smart cameras today come with relatively simple-to-use software. Advanced knowledge of programming is not required to use the technology, but it is important to keep flexibility and scalability during the usage of this software. Sensor and processor technologies are advancing rapidly, so the best-case scenario takes place when the smart camera model and software are well integrated, but not exclusive to each other [12]. As a result, changing smart camera models to a new version or need to move to a different hardware platform, such as a PC or operating system, a complete rewrite of the application or IP should not be required. The level of ruggedness required is dependent upon the environment in which the smart camera is to be deployed. It is important to note that many applications take place in harsh environments. A typical design of smart camera is shown in Figure 12.

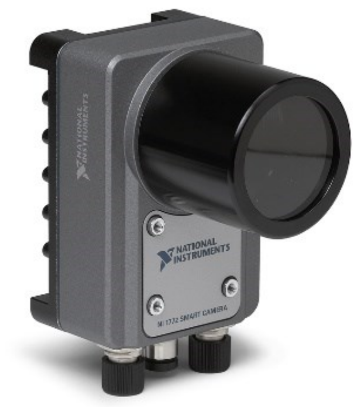


Figure 12. Smart cameras like the NI 1772 camera from National Instruments have high IP ratings for protection against environmental exposures like dust and water. Higher IP ratings are beneficial for applications in harsh environments, including outdoor monitoring and industrial vision inspection.

Since the primary output of a smart camera is a decision, result, or some other information beyond an image, most smart cameras have built-in I/O to communicate or control other devices in the system, as shown in Figure 13. With industrial automation, the smart camera may need to control actuators to sort products; communicate inspection results to a robot controller, PLC, or communicate inspection parameters and results to a local or remote user interface [12].

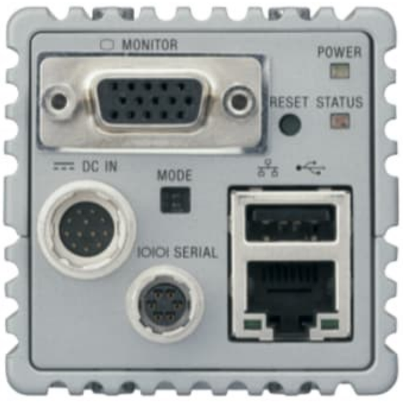


Figure 13. Lighting, digital I/O, serial, Ethernet, and USB buses, as well as display ports for user interfaces.

Models are also available with built-in lighting that can modify illumination directly from the smart camera. To effectively communicate to other devices, more and more industrial communication protocols, including Ethernet IP, and serial, are also being supported natively in smart cameras. These days, smart cameras can come in all shapes, sizes, and performance levels, but there is still one attribute that still defines them as smart cameras: the ability to perform image processing and make decisions directly on the camera. It is the decision making that makes a camera smart, and with the potential cost savings, ease of integration, and increasing performance, smart cameras are a cutting-edge option for many vision applications.

1.2.1 *Cognex Insight 7600*

Cognex is the leading supplier of machine vision and industrial barcode reading solutions. With over 3 million systems installed in facilities around the world and over forty years of experience, Cognex is focused on industrial machine vision and image-based barcode reading technology. Deployed by the world's top manufacturers, suppliers and machine builders, its products ensure that manufactured items meet the stringent quality requirements of each industry. Cognex solutions improve manufacturing quality and performance by eliminating defects, verifying assembly and tracking information at every stage of the production process. Smarter automation using Cognex vision and barcode reading systems means fewer production errors, which equates to lower manufacturing costs and higher customer satisfaction [13]. Cognex comprehensive line of vision sensors and 2D and 3D vision systems, as shown in Figure 14, all use machine vision technology to perform inspections but are engineered for different tasks.

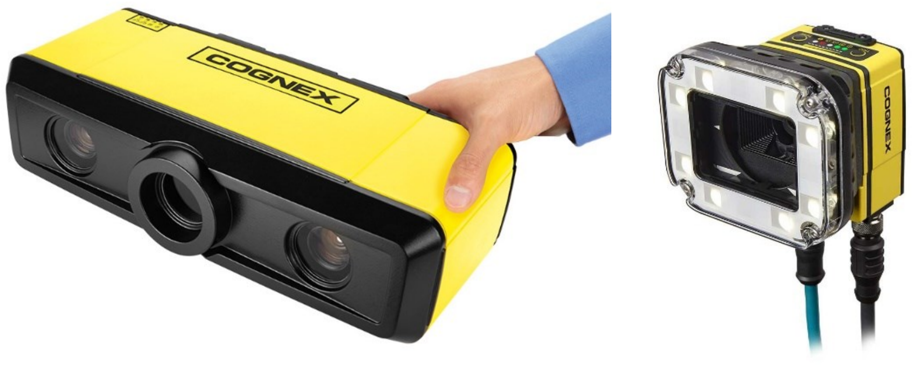


Figure 14. On the left a 3D Cognex vision system (A5000). On the right a 2D Cognex vision sensor (In-Sight 7600).

All In-Sight 2D products, are configured with the powerful and intuitive In-Sight Explorer software. The easy-to-use interface permits to analyse step-by-step through the setup process and provides the power and flexibility of the vision spreadsheet for more difficult applications. In-Sight Explorer also offers the widest range of built-in communication protocols that interface directly to any PLC, robot, or HMI on the factory network [13].

Cognex In-Sight 2D vision systems are unmatched in their ability to inspect, identify, and align parts. These self-contained, industrial-grade vision systems combine a library of advanced vision tools with high-speed image acquisition and processing, as shown in Table 4. All the sensors present the most important vision tools as pattern matching, edge, measurement, flaw detection and image filters [14].

Table 4. 2D vision sensors and vision systems specifications.

	2000 Series	**5705 Series**	**7000 Series**	**8000 Series**	**9000 Series**	**D9000 Series**
Imager Type	Monochrome/ Colour Area Scan	Monochrome/ Colour Area Scan	Monochrome/ Colour Area Scan	Monochrome/ Colour Area Scan	Monochrome/ Colour Area Scan, Monochrome line scan	Monochrome/ Colour Area Scan
Resolution	Up to 1.2 MP	5 MP	Up to 5 MP	Up to 5 MP	12 MP, 32 MP for line scan	Up to 5 MP
Acquisition Speed	75 fps	16 fps	Up to 217 fps	Up to 217 fps	Up to 14 fps	Up to 51 fps
Lenses	S-Mount, Autofocus	C-Mount	C-Mount, S-Mount, Autofocus	C-Mount	C-Mount	C-Mount, S-Mount, Autofocus
Lighting	Integrated	N/A	Integrated, External light via light control connector	N/A	External light via light control connector	Integrated, External light via light control connector
Length	92 mm	124.1 mm	90.1 mm	75.5 mm	121.0 mm	121.0 mm
Width	60 mm	61.4 mm	60.5 mm	35 mm	60.5 mm	60.5 mm
Depth	52 mm	52 mm	Up to 2 MP: 35.7 mm, 5 MP: 49.4 mm	32 mm	53.4 mm	53.4 mm
Protection	IP65	IP67	IP67	IP40	IP67	IP67

To design the rehabilitative station, among all the vision systems, it has been chosen a tool capable to locate an object location in a 2D application In-Sight 7000 Series combines modular integrated lighting and optics for optimal image formation with powerful vision tools, as shown in Figure 15.

Figure 15. In-Sight 7000 Series different lighting and optics.

It has been chosen the 7600 Series vision system to build the rehabilitative station. This vision system is a compact, network-ready, stand-alone machine vision system used for automated inspection, measurement, identification and robot guidance applications. It can be configured remotely over a network using an intuitive user interface. The standard components of this vision system are the sensors that provides image acquisition, vision processing, job storage, ethernet connectivity and discrete I/O [14]. A Cognex system is characterized by a modular design to have maximum flexibility, according to the purpose. The connectors, on the bottom of the vision system, are the PWR that connects breakout cable which provides connections to an external power supply, the ENET connector that links the ethernet cable which is necessary to permit the communication between the camera and other devices and the LIGHT connector that connects the vision system to an external lighting device, as shown in Figure 16.

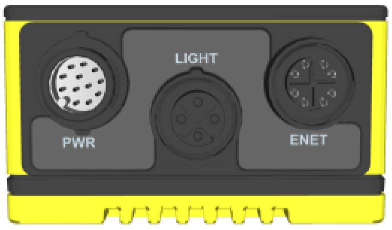


Figure 16. Connectors in Cognex In-Sight 7600.

The indicators, on the top of the vision system as shown in Figure 17, are the TRIG Button used to manually triggers an image acquisition, the Power Led that illuminates to indicate that the vision system is powered on, the SD Card Status LED that illuminates if the SD card is within the tool, the Pass/Fail Led that is user configurable, the Network LED that illuminates to indicate network activity, the Error LED that is user configurable and the TUNE Button that is used as an event trigger within In-Sight Explorer.

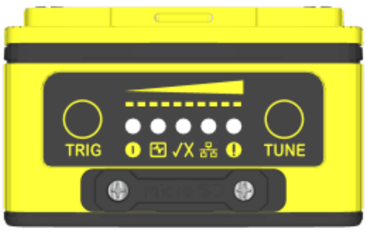


Figure 17. Indicators in Cognex In-Sight 7600. The indicators are presented from left to right.

To build a vision system it has been necessary to choose the lenses, the light source, colour filters and a polarizer. In particular, the rehabilitative station has been implemented with a vision system characterized only by an autofocus lens that permits to focalize automatically a selected point or area. Then, the image type of the camera is thee monochrome one and there is not present lighting mechanism.

2. Kinematic Models

Kinematics studies the motion of bodies without consideration of the forces or moments that cause the motion [15]. Mathematical models of robotic and human arm are indispensable tools to study both forward and inverse kinematic.

2.1 Kinematic Model of Human Arm

The analysis of the upper limb functioning is necessary to define the mathematical model of the human arm. In the studies of Desmurget and Prablanc [16], Lemay and Crago [17] and Raikova et al [18], a serial mechanism of three joints (shoulder, elbow and wrist) and three segments (upper arm, forearm and hand) have been used to characterize the human arm. The hand is greatly simplified because of its complexity and high number of DoF, and consequently, should be studied separately [19]. The proposed kinematic model of the arm consists in the representation of the human arm as three rigid segments connected by frictionless joints with a total of 7 DoFs [20]. To model the human joints is necessary to conduct an important simplification because human joints do not hold a reference location to the segment they connect, as well as in the upper limb, the forearm and the hand are approximated as rigid bodies. The human shoulder is considered as a simple ball-and-socket joint, where the ball-shaped surface of one rounded bone fits into the cup-like depression of another bone. This joint allows three rotational DoFs, attributed to flexion-extension (q1 joint) uction-adduction (q2 joint), and external-internal rotation of the humerus relative to the scapula (q3 joint). The elbow joint, instead, is associate to a universal joint with two DoFs, which are flexion- extension (q4 joint) and pronation-supination (q5 joint). The wrist joint is also related to a universal joint able to perform the flexion-extension (q6 joint) and abduction-adduction (q7 joint), as shown in Figure 18.

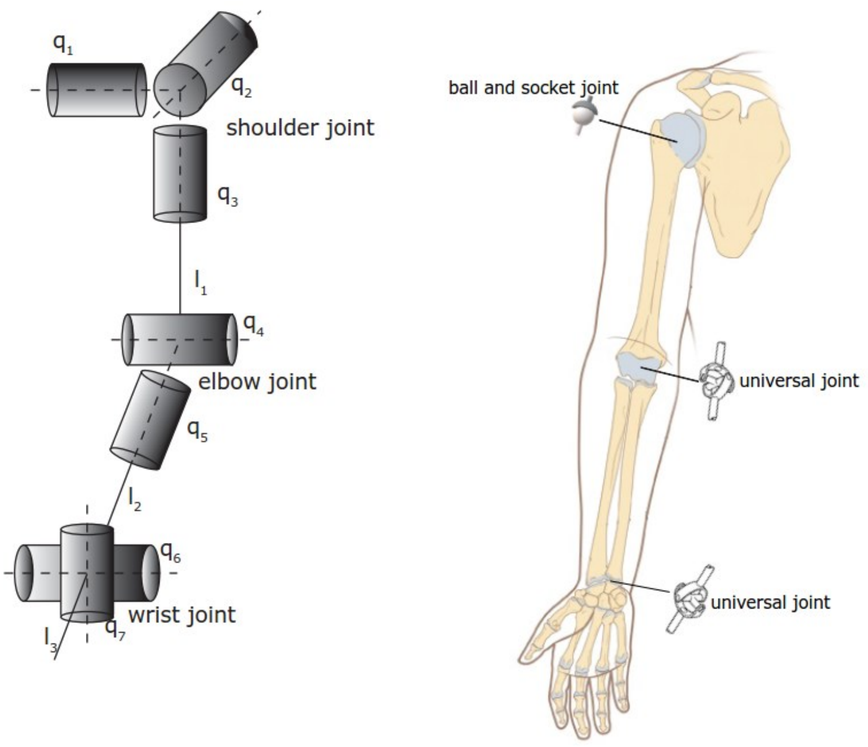


Figure 18. On the left, the kinematic structure of human arm. On the right, joints with 3 DoF (ball and socket joint) and 2DoF (universal joint).

The axes in elbow and wrist joints are assumed to be orthogonal and intersecting. Even though this model is largely accepted, it is not totally exact. In particular, the forearm segment is not rigid because of the rotations of the radius relative to the ulna and the centre of the shoulder joint as well as the axes of rotations in the elbow and wrist can move relative to the adjacent segments during the movements. However, it is a reasonable compromise between accuracy and simplicity of the human arm description. To define the kinematic model, it is necessary to define the length of the upper limb segments and the joint limits. In fact, it has been decided to determine the segments lengths according to anthropometric values [20] based on the knowledge of the heigh of the patient. Thus, it is possible to develop a kinematic model of the upper limb for all the patients. The anthropometric values used to build the kinematic model are reported in Table 5.

. To define the range of each joint, it has been utilized the joint limits present in the OpenSim software [21]. In Table 6 are presented the joint limits.

Table 5. Anthropometric values for human upper limb.

Upper Limb Segment	**Lengths (m)**
Arm	$0.161 \cdot height$
Forearm	$0.146 \cdot height$
Hand	$0.106 \cdot height$
Hand Centre of Gravity	$0.506 \cdot hand$

Table 6. Admissible range for human arm joints.

Joint		**Minimum Value**	**Maximum Value**
Shoulder	Flexion – Extension	- 90°	+90°
	Abduction – Adduction	- 120°	+90°
	External – Internal Rotation	- 90°	+90°
Elbow	Flexion - Extension	0°	+150°
	Pronation - Supination	0°	+180°
Wrist	Flexion – Extension	- 70°	+70°
	Abduction - Adduction	- 25°	+35°

2.2 Kinematic Model of UR5e Robotic Arm

The robot used to design the collaborative station is the UR5e of Universal Robot company and its mathematical model is presented. The UR5 robot has six DoF with six revolute joints: the base joint (q1), the shoulder joint (q2), the elbow joint (q3) and three wrist joints (q4, q5, q6), as show in Figure 19. These last three do not act as a coincidental wrist. All the six joints contribute to the transformational and rotational movements of its end-effector [22]. This characteristic makes the kinematics analysis more complex in comparison with other manipulators having coincidental wrist. In the Figure 20 are shown the lengths of the robotic links. Since this robotic arm has 6 DoFs, the system is not redundant. Therefore, the arm manipulability is not maximum, and the singularities can occur more [23].

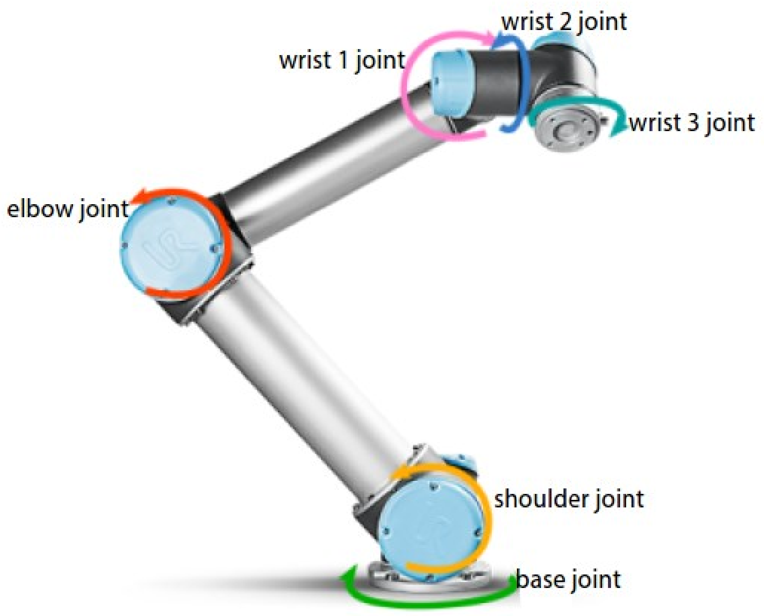


Figure 19. Revolute joints of UR5e.

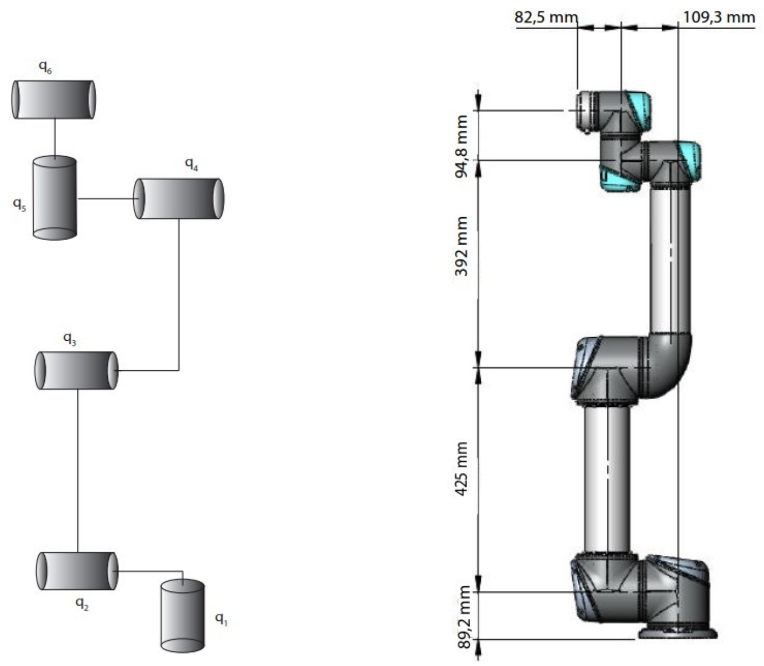


Figure 20. On the left the kinematics of UR5e. On the right the UR5e dimensions.

2.3 **Robotic Toolbox**

The Robotic Toolbox, implemented in Matlab environment, provides several functions that result to be useful for the study and simulation of classical robotic arms. In fact, it is possible to develop kinematics, dynamics and trajectory planning. The toolbox contains functions and classes to represent orientation and pose in 2D and 3D as matrices, quaternions, twists, triple angles, and matrix exponentials. The Toolbox also provides functions for manipulating and converting between datatypes such as vectors, homogeneous transformations and unit-quaternions which are necessary to represent three-dimensional position and orientation. The main advantage of the Toolbox is that the routines are generally written in a straightforward manner which allows for easy understanding, perhaps at the expense of computational efficiency. It is also possible to rewrite the function to be more efficient. Robotic arms can be created by the user for any serial-link manipulator because the Toolbox uses a general method to represent the kinematics and dynamics of the robot. In fact, both the upper limb of the patient and the robotic arm have been defined using the Robotic Toolbox functions. To delineate the associated robotic structures, it has been used the Toolbox function 'SerialLinlk'. In detail, the inputs of this function are a series of joint definitions and homogeneous transformation matrixes to define the base with respect to the global reference frame and the end-effector frame. To obtain reliable robotic structures, the kinematic models have been developed to reproduce the OpenSim model, for the human upper limb, that allows to visualize the real pose of the hand according to joints configuration, while for the robotic arm it has been followed the kinematic model already present in literature [22]. The main limitation of the OpenSim model is related to the first joint limits. In fact, this model does not allow to represent an arm fully extended upward because the range of the shoulder flexion-extension is from - 90° to + 90°.

To represent both robotic arm and human upper limb as a robotic structure, it has been necessary, according to the Denavit-Hartenberg definition, to specify the four parameters (θ, α, d, a) for each joint and then, combine all the joints to form the related robotic structure using the function 'SerialLink'. All the joints have been set as revolute, based on the already described kinematic models. Thus, the degree of freedom associated to each joint is θ and it has been necessary to define the other three parameters, fixed over time. To accomplish this part of the study, it has been used the Robotic Toolbox function 'Revolute' to define the revolute joint where the three set parameters are the inputs of this function. In this study, during the definition of the robotic structure related to the patient upper limb, the values of a and d for each joint are dependent on the height of the patient to obtain a tool that can be employed also for different patient. To calculate the length of the body segment, it has been used the anthropometric values shown in Table 5.

. In this study has been considered a patient whose height is 1.70 m. The segment lengths are shown in Table 7. On the other hand, the definition of the robotic structure related to the UR5e is fixed.

Table 7. Patient upper limb segment lengths.

Upper Limb Segment	Lengths (m)
Arm	0.2737
Forearm	0.2482
Hand	0.1802
Hand Centre of Gravity	0.0912

The Denavit-Hartenberg parameters used to build the robotic structures associated with the human upper limb and the UR5e are shown in Table 8 and in Table 9, respectively. An important difference between the robotic structures is related to the presence of joint limits in the upper limb, according to the OpenSim model, while in the robotic arm there are not joint constraints.

Table 8. DH parameters for the human upper limb.

	d (m)	a (m)	α	off-set
q1 Joint	0	0	90°	90°
q2 Joint	0	0	90°	90°
q3 Joint	-0.2737	0	90°	90°
q4 Joint	0	0	-90°	0°
q5 Joint	-0.2482	0	90°	0°
q6 Joint	0	0	90°	90°
q7 Joint	0	-0.0912	-90°	0°

Table 9. DH parameters for the UR5e.

	d (m)	a (m)	α	off-set
q1 Joint	0.089160	0	90°	90°
q2 Joint	0.14	-0.425	0°	0°
q3 Joint	-0.12	-0.39225	0°	0°
q4 Joint	0.10915	0	90°	0°
q5 Joint	0.09456	0	90°	180°
q6 Joint	0.055	0	0°	0°

The off-set value represents the joint variable shift. The base matrix for both the robotic arm and the upper limb, respectively, have been defined as follow:

$$\mathbf{T}_{\mathbf{base\_robotic\ arm}} = \begin{bmatrix} 0 & -1 & 0 & a \\ 1 & 0 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_{\mathbf{base\_upper\ limb}} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Both characterize the position and the orientation of the robot with respect to global reference frame in Matlab environment. The main difference is related to the origin of the base of the robotic structure that is fixed for the human upper limb, while it is mobile for the robotic arm. For the human upper limb, this matrix is necessary to build a kinematic model that follows the OpenSim one. In fact, all the DoFs present in the OpenSim software model are correctly represented.

It is also presented the homogeneous transformation matrix that allow to rotate the frame orientation of the end-effector of the robotic structure associated to the upper limb:

$$\mathbf{T}_{\mathbf{ee\_upper\ limb}} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Thus, this homogeneous transformation matrix permits to solve, properly, in the next steps the inverse kinematic of the human upper limb. In Figure 21 are shown the initial configuration, where all the joints are set to a value equal to zero of the human upper limb using both the kinematic model developed in Robotic Toolbox and the model in OpenSim software.

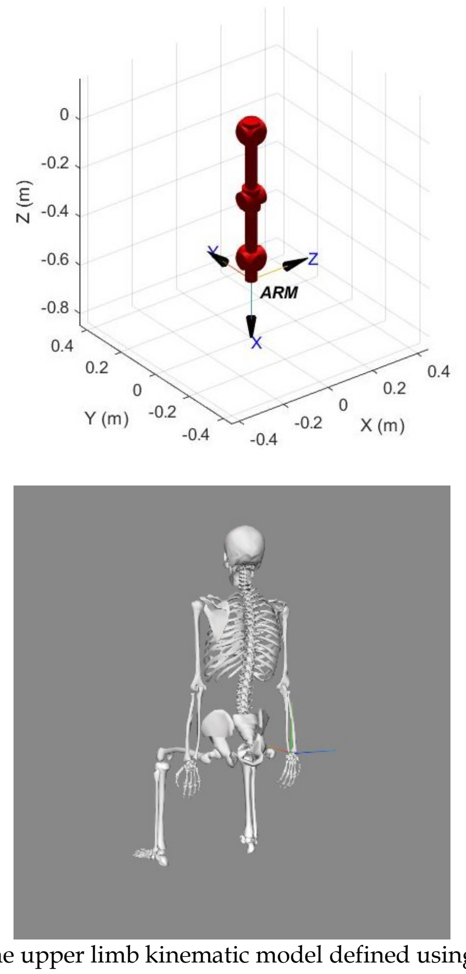


Figure 21. Upper panel shows the upper limb kinematic model defined using Robotic Toolbox. On the lower panel the kinematic model in OpenSim software.

The Z-axis of the end-effector of the robotic structure associated to the human upper limb is in the same direction of the thumb (considering an open hand). The difference in the X-axis and Y-axis direction of the human arm end-effector is not a limitation. As for regard the robotic structure related to the UR5e, the kinematic model obtained using Robotic Toolbox, given a particular base position and all the joints values set to zero, has been compared with Polyscope software, as shown in Figure 22.

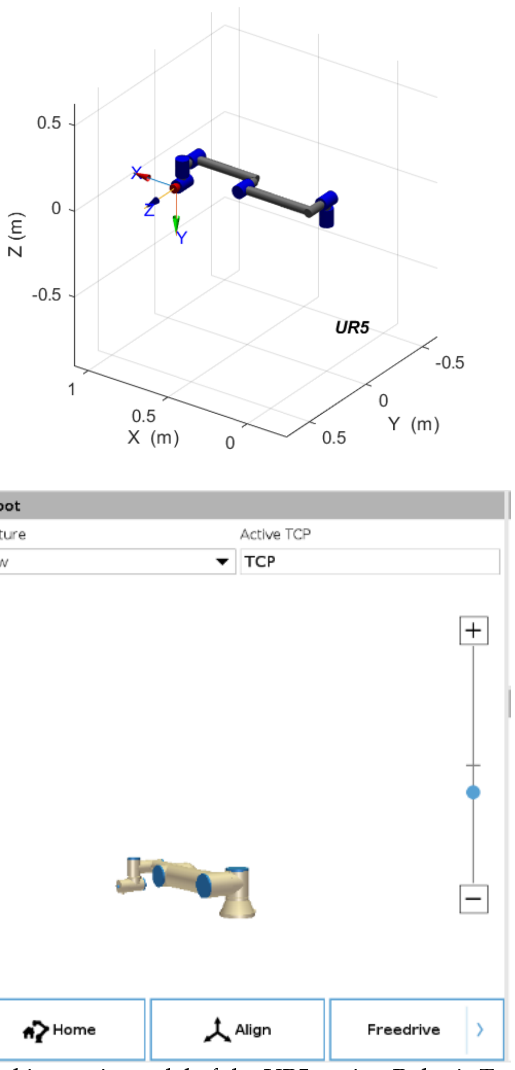


Figure 22. On the upper panel the kinematic model of the UR5e using Robotic Toolbox. On the lower panel, the kinematic model integrated in Polyscope software. In this case, the origin of the robotic arm has been set in the origin.

Once both the kinematic models have been defined, using Robotic Toolbox, it is possible to compute the forward kinematic, the inverse kinematic and the Jacobian matrix. These are the main tools used to design the optimization algorithm. To solve the forward kinematic, the Robotic Toolbox uses the Denavit-Hartenberg method, while to solve the inverse kinematics, the Robotic Toolbox implements a numerical solution where, given the initial guess $\mathbf{q_0}$, the configuration vector $\mathbf{q}$ is then optimized a function such that the error $|\mathbf{f}(\mathbf{q}) - \mathbf{x_D}|$ approach 0. $\mathbf{x_D}$ represents the desired pose of the end-effector (input of the inverse kinematic algorithm) and $\mathbf{f}(\mathbf{q})$ represents the forward kinematic calculated at the actual joints configuration. Finally, Robotic Toolbox permits to also calculate the Jacobian matrix, using the geometrical approach, with respect to the global reference frame.

3. Optimization Algorithm

The first part of this work is based on the development of a Matlab function capable to find the optimal position between the UR5e robot base and the right shoulder of the patient. The optimization problem is focused on three translational parameters, defined in the homogeneous transformation matrix $\mathbf{T}_{\mathbf{base\_robotic\ arm}}$. It is important to underline that the origin of the global coordinate frame coincide with the origin of the patient shoulder coordinate system. The input of the function is the height of the patient, while the function output is the position of the robotic arm origin according to a quality index based on the manipulability ellipsoids. This part of the thesis has been completely developed on Matlab environment. In literature, different authors have focalised their studies in manipulability ellipsoids. In particular, the analysis of the manipulability ellipsoids has been performed for both collaborative robots used for rehabilitative purpose, as well as for designing proper exoskeleton. Sun et al. [24] have studied a rehabilitative system constituted by the human upper limb modelled by five DoFs, considering the human wrist without relative motion during the rehabilitation task and the KUKA collaborative robot. This study is focused on the velocity manipulability ellipsoid intersection volume (VMEIV) index, defined as the intersection between the human arm and the robotic arm velocity ellipsoids intersection. This index has been used to evaluate the dexterity index that correspond to the human-robot interaction movement, and it provides a reference for the training trajectory selection for rehabilitation robot. Yamashita et al. [25] propose a robotic assistive control system for rehabilitation of human arm. The goal of this study is to use the main axis of the manipulability ellipsoids to find the easiest direction to move the upper limb. In fact, from a clinical point of view, this robotic system is used to initialize the rehabilitation process similarly to when an experience therapist leads the arms of the patient in the easiest direction to initialize the movement. Thus, the control system has been employed to guarantee low stiffness in the moving direction and high stiffness in the orthogonal directions to initialize the movement.

Panagiotis et al. [26] aims to study the relationship between the manipulability ellipsoids and the activations of the musculoskeletal system using EMG acquisitions in a rehabilitative system characterized by the PA-10 robot. It has been proved that when the major axes of both ellipsoids (human arm and robot) are aligned the muscular effort is lower. Thus, the axes of the manipulability ellipsoids play a significant role on the muscular activation of the muscle while the arm is interacting with the environment. Petri et al. [27] propose a control approach of the exoskeleton control approach based on the arm muscular manipulability. The purpose of this procedure is to augment the human arm force manipulability to execute task equally in the workspace of the arm. The force capability of the human arm is heavily dependent on its current configuration, giving rise to minor axis where the ability to produce force il low while the major axis represents the direction in which high force-level can be produced. The purpose of this study is to modify the actual shape of the manipulability ellipsoids to a sphere where force production is equal in all the directions, augmenting the performance of the human user. Chen et al. [28] aims to provide an innovative method to evaluate the design of ergonomic rehabilitation robot. The authors have verified the manipulability of the robots while the patient is performing activities of daily living (ADLs) to design an ergonomic robotic structure. In particular, the manipulability ellipsoids of the robot and the human upper limb have been compared to obtain a safety and comfortable robot.

3.1 Cloud Of Points

A set of points, used to represent different starting position for different rehabilitative tasks, has been defined. Spherical coordinate system has been used to define this set of points. The following method allows to represent a point in the three-dimensional space using three variables: the radial distance (length between the origin of the system and the point), the polar angle measured from a fixed zenith direction and the azimuth angle of its orthogonal projection on a reference plane that passes through the origin and is orthogonal to the zenith, measured according to a fixed reference direction on that plane. Thus, each point position is defined as $\mathbf{p}(\mathrm{r}, \theta, \varphi)$, as shown in Figure 23. To define the cloud of points, two different radii have been calculated starting from the total length of the upper limb (considering the hand centre of gravity as end-effector of the human upper limb) as presented in Table 5.

$$\text{upper limb} = (0.161 + 0.146 + 0.106 \ \cdot 0.506) * \text{height}$$

$$\text{radius}_1 = 5/6 * \text{upper limb}$$

$$\text{radius}_2 = 1/2 * \text{upper limb}$$

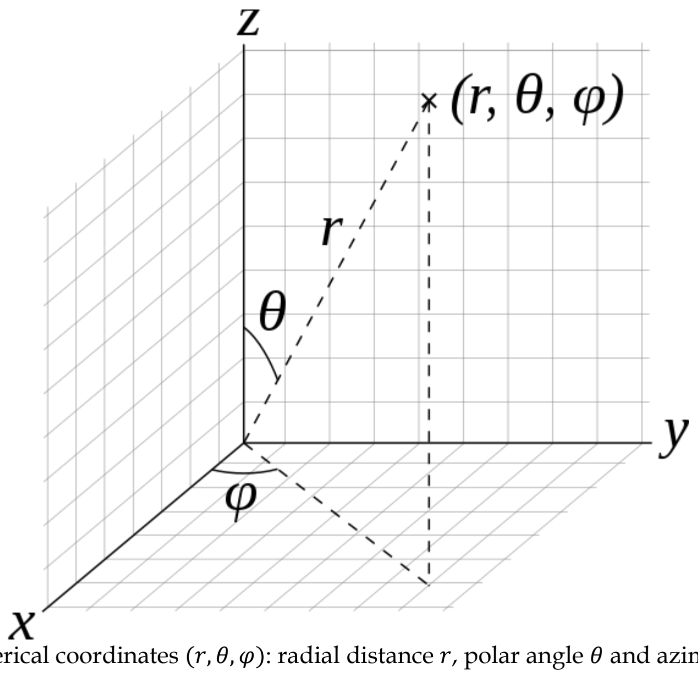


Figure 23. Spherical coordinates (r, θ, φ): radial distance r, polar angle θ and azimuthal angle φ.

In particular, the first radius has been obtained considering the 83% of the total upper limb length and the second radius has been computed with reference to half of the total length of the upper limb. Consequently, also the cloud of points results to be normalized according to the height of the patient. As for regard the polar and the azimuth angles, for each radius, nine different couples of angles have been considered as shown in Table 10.

Table 10. Angles definition for a desired radius.

	θ	φ
A	90°	90°
B	70°	90°
C	110°	90°
D	90°	60°
E	70°	60°
F	110°	60°
G	90°	120°
H	70°	120°
I	110°	120°

Once all the variables have been defined, to calculate the position of the points in the three-dimensional cartesian-coordinate system, the following equation have to be solved:

$$\begin{cases} x = r * \sin(\theta) * \cos(\varphi) \\ y = r * \sin(\theta) * \sin(\varphi) \\ z = r * \cos(\theta) \end{cases}$$

Any spherical coordinate triplet specifies a single point of three-dimensional space. Thus, a common choice is:

$$r \geq 0$$
$$0 \leq \theta \leq \pi$$
$$0 \leq \varphi \leq 2\pi$$

According to the height of the patient and the angles previously presented to build the cloud of points, in Table 11 are presented all the points expressed in the global reference frame.

Table 11. Cartesian coordinate in global frame of the cloud of points.

	X (m)	Y(m)	Z(m)		X(m)	Y(m)	Z(m)
A	0	0.5851	0	**A′**	0	0.3510	0
B	0	0.5497	0.2001	**B′**	0	0.3298	0.1200
C	0	0.5497	-0.2001	**C′**	0	0.3298	-0.1200
D	0.2925	0.5066	0	**D′**	0.1755	0.3040	0
E	0.2748	0.4761	0.2001	**E′**	0.1649	0.2856	0.1200
F	0.2748	0.4761	-0.2001	**F′**	0.1649	0.2856	-0.1200
G	-0.2925	0.5066	0	**G′**	-0.1755	0.3040	0
H	-0.2748	0.4761	0.2001	**H′**	-0.1649	0.2856	0.1200
I	-0.2748	0.4761	-0.2001	**I′**	-0.1649	0.2856	-0.1200

The purpose of this cloud of points is to build a symmetric set of points with respect to an upper limb fully extended in the forward direction with thumb pointing up. From an operative point of view, this choice permits to have a symmetric evaluation of the workspace. At the end, the cloud of points is characterized by eighteen points, nine for each spherical shell, as shown in Figure 24, where for each polar angle (110° 90° 70°) it has been identified a plane characterized by six points. Together with the position, for each point must be defined also the orientation to obtain the desired pose. This is necessary to provide the proper input to the inverse kinematic algorithm of the human upper limb that will be investigated later.

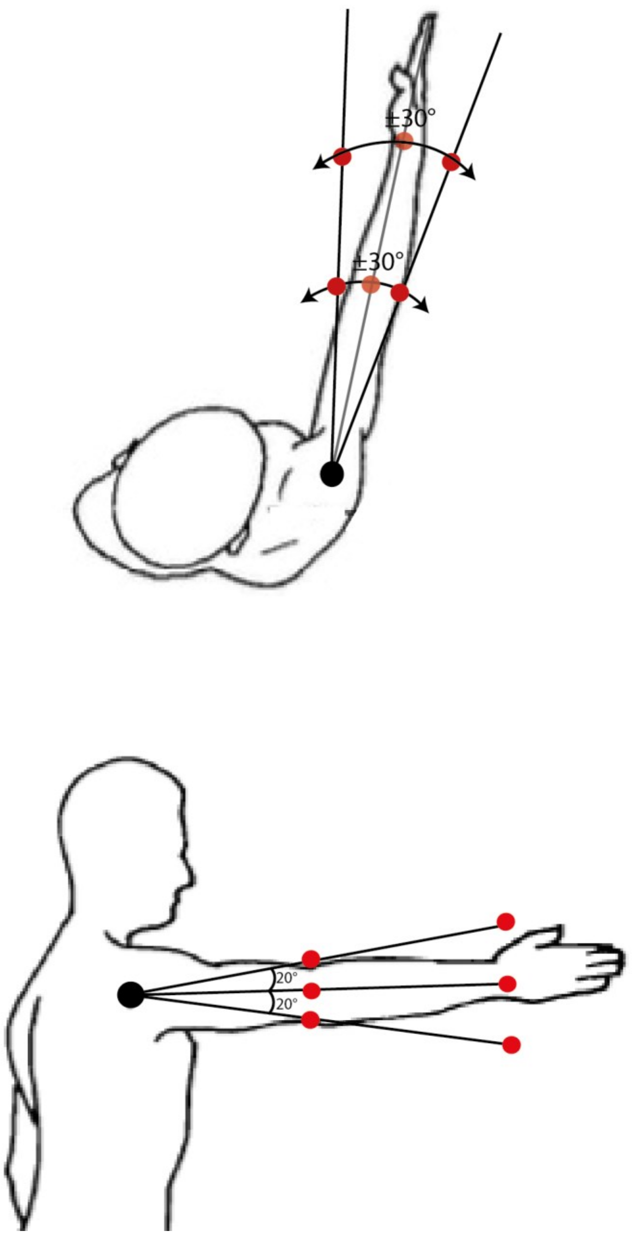


Figure 24. Representation of the cloud of points with respect to the patient upper limb.

Three different orientations have been used to characterize the orientation of all the points. Starting from the point A, characterized by azimuthal angle equal to 90°, the Z-direction of the local frame has been defined parallel to the global Z-direction, the X-direction of the local frame has been defined parallel to the global Y-axis, while the Y-direction of the local frame has been defined anti-parallel to the global X-axis. Then, all the points that are characterized by the azimuthal angle equal to 90°, are characterized by this orientation. As for regards the points that are identified by azimuthal angle equal to 60° it has been applied a -30° elementary rotation around Z-axis with respect to the local frame previously described, while the points related to azimuthal angle equal to 120°, it has been applied a 30°elementary rotation around Z-axis with respect to the local frame previously described. Position and orientation of the points belonging to cloud are shown in Figure 25. From a computational point of view, this definition of the cloud of points allows to guarantee the possibility to always solve the inverse kinematic, according to the defined kinematic model of the upper limb, because the set of points is always within the dexterous workspace of the upper limb being all the points normalized with respect to the height of the patient. Furthermore, the usage of radii shorter than the total upper limb length permits to guarantee boundary singularities avoidance. As for regards the orientation definitions, it has been defined a local frame that has always its own X-axis in line with the forearm to obtain a configuration similar to everyday activity life.

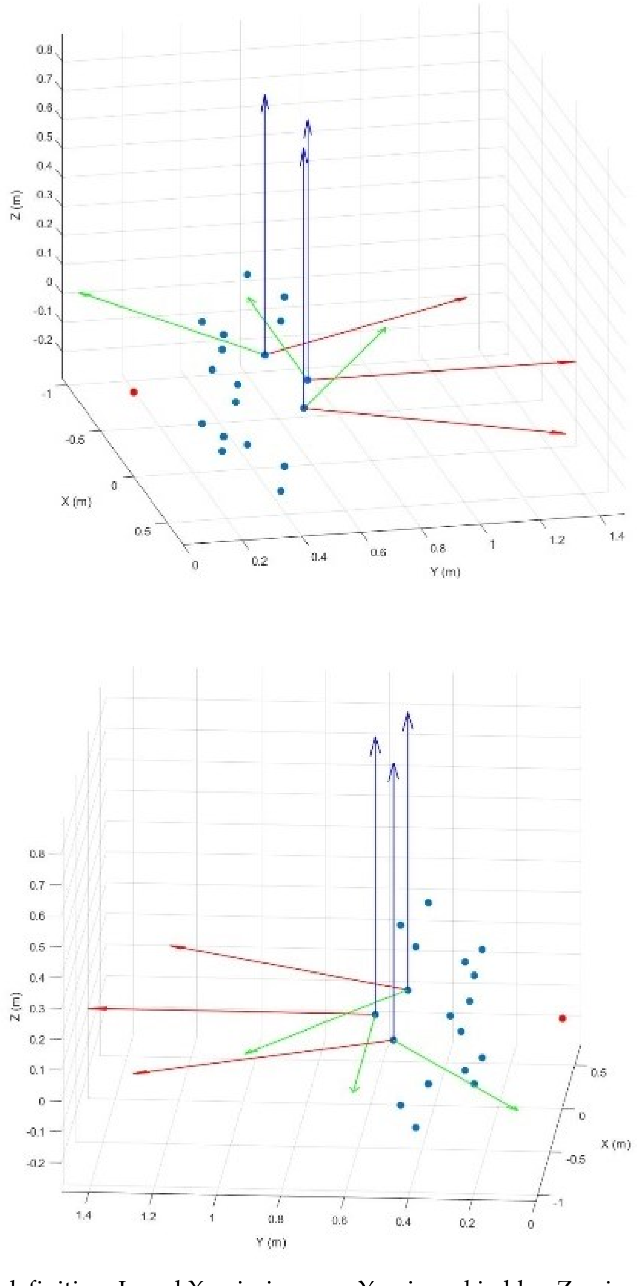


Figure 25. Cloud of point definition. In red X-axis, in green Y-axis and in blue Z-axis representing the orientation of the points. The red point represents the origin of the frame relative to the right shoulder of the patient.

3.2 Human Arm Inverse Kinematic

The robotic structure defined to describe the human arm is composed by 7 DoFs. Based on the knowledge of robotics, it is a redundant structure, consequently, the human arm has high level of mobility and can reach smooth trajectory during its movement. However, because of the redundancy, to reach a pose in the three-dimensional space infinite solutions are present. Thus, to solve the inverse kinematic problem, using the Robotic Toolbox implemented in Matlab environment, it has been proposed a numerical approach. Generally, this method is implemented using the initial guess $\mathbf{q_0}$ (input of the inverse kinematic function), the actual configuration vector $\mathbf{q}$ and being $\mathbf{f}(\mathbf{q})$ the forward kinematic calculated at the actual joint configuration, it is performed the minimization problem based on the calculation of the error $|\mathbf{f}(\mathbf{q}) - \mathbf{x_D}|$ that has to approach 0, where $\mathbf{x_D}$ represents the desired pose of the end-effector and it is the second input of the inverse kinematic algorithm. Using numerical solution, it is important to underline that not always is possible to reach the desired pose because the algorithm may stop earlier, and the output of the function is also strictly dependent on the initial guess $\mathbf{q_0}$. Thus, the output solution is expected to be the one nearest to the initial guess of the algorithm. The initial joint configuration $\mathbf{q_0}$, defined to solve the numerical inverse kinematic algorithm for all the points belonging to the cloud of point already defined for a specific patient, has been chosen the upper limb resting on the worktable with the thumb up, as shown in Figure 26.

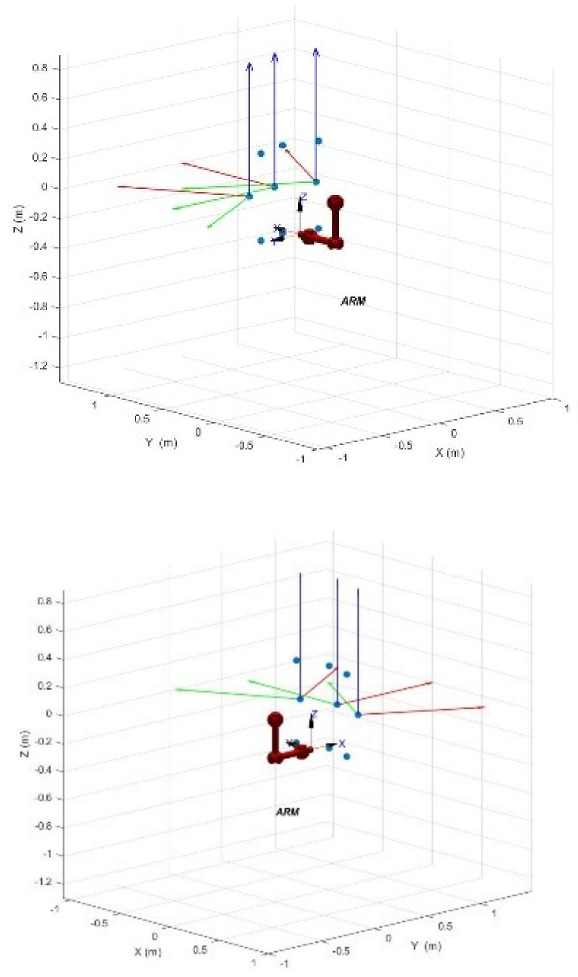


Figure 26. Initial upper limb configuration with respect to the defined cloud of point.

Once the robot structure has been defined, the Robotic Toolbox permits to use different algorithm to solve the inverse kinematic. Among all the possibilities, it has been chosen the function ‘ikcon’. The inputs of the function are the robot end-effector pose defined as a homogeneous transform matrix (4x4), the initial guess and the options vector that allow to modify the settings and threshold used to stop the algorithm. On the other hand, the output of the function is the joint coordinates as vector (1xN) where N is the number of robot joints. To obtain reliable results from the inverse kinematic algorithm the following options have been used: ‘TolFun’ 1E-20, ‘TolX’, 1E-10, ‘MaxFunEvalals’, 3000, ‘MaxIter’, 500, ‘StepTolerance’, 1e-15, ‘FunctionTolerance’, 1e-30.

‘TolFun’ and ‘TolX’ permit to define the termination tolerance for the function value and the for the x-axis of the function respectively. ‘StepTolerance’ permits to halts the algorithm, after a successful poll, if the distance from the previous best point to current best point is less than the set value. ‘FunctionTolerance’ permits to halts the algorithm, after a successful poll, if the distance between the function value at the previous best point and the function value at the current best point is less than the set value.

However, the main advantage of this function is related to the possibility to solve the optimization problem according to joint limits of the robotic structure related to the upper limb. This is necessary because the human joints limits are already defined in Table 6, from OpenSim software model. This optimization algorithm, already developed in the Robotic Toolbox, is based on the ‘fmincon’ Matlab function. At the end of this step, it has been calculated the joints configuration for all the points belonging to the cloud of points previously defined. Thus, a matrix (7x18), containing the joints configuration for each pose in the cloud of points, has been obtained as result. Furthermore, the number of rows is related to the DoFs of the upper limb, while the number of columns is related to the number of points defined. In Figure 27 is shown the starting cloud of points defined using the spherical-coordinate system and the actual set of points that is generated by the calculation of the forward kinematic of the human arm, using as joints configuration the output of the inverse kinematic algorithm, presented in Table 12.

Only the position of the cloud of points is reported. Because of joint limits definition, as presented in paragraph 2.1, not all the poses are correctly solved. In fact, it is important to notice that when the flexo-estension at the level of the shoulder (q1 joint) reaches its maximum value, the algorithm is not able to guarantee the exact solution.

Table 12. Solution of the inverse kinematic algorithm.

	q1 Joint	q2 Joint	q3 Joint	q4 Joint	q5 Joint	q6 Joint	q7 Joint
A	72°	0°	0°	38°	90°	0°	20°
B	90°	0°	0°	41°	90°	0°	35°
C	51°	0°	0°	33°	90°	0°	-6°
D	65°	-51°	59°	121°	39°	-59°	35°
E	90°	-56°	65°	120°	20°	-58°	35°
F	11°	-20°	15°	119°	78°	-23°	35°
G	69°	-30°	-6°	38°	86°	-1°	20°
H	90°	-30°	0°	41°	90°	0°	35°
I	46°	-25°	-13°	33°	82°	-2°	-6°
A′	21°	-73°	11°	121°	39°	-59°	35°
B′	-9°	-105°	-3°	119°	0°	-64°	27°
C′	-1°	-23°	-17°	119°	78°	-23°	35°
D′	71°	24°	25°	38°	74°	-6°	19°
E′	90°	30°	0°	41°	90°	0°	35°
F′	47°	22°	24°	33°	86°	-1°	-6°
G′	73°	-23°	72°	121°	39°	-59°	35°
H′	90°	-26°	65°	120°	20°	-58°	35°
I′	7°	4°	30°	118°	90°	0°	35°

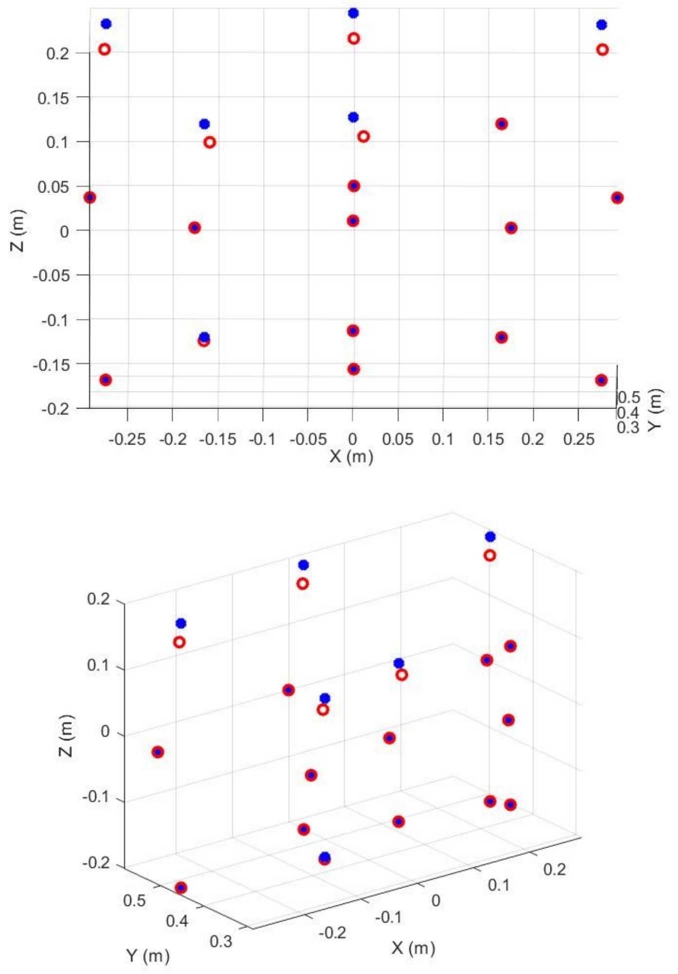


Figure 27. In red the original clou of points, in blue the points position obtained from the forward kinematic of the upper limb using the obtained joints configuration.

Then, all the joints configurations have been checked in OpenSim software, to guarantee that the numerical solution provides a possible arm configuration, as well as the upper limb reaches the desired pose in a manner that is consistent with everyday life. Some solutions are reported in Figure 28 Figure 29 Figure 30 Figure 31 Figure 32 Figure 33.

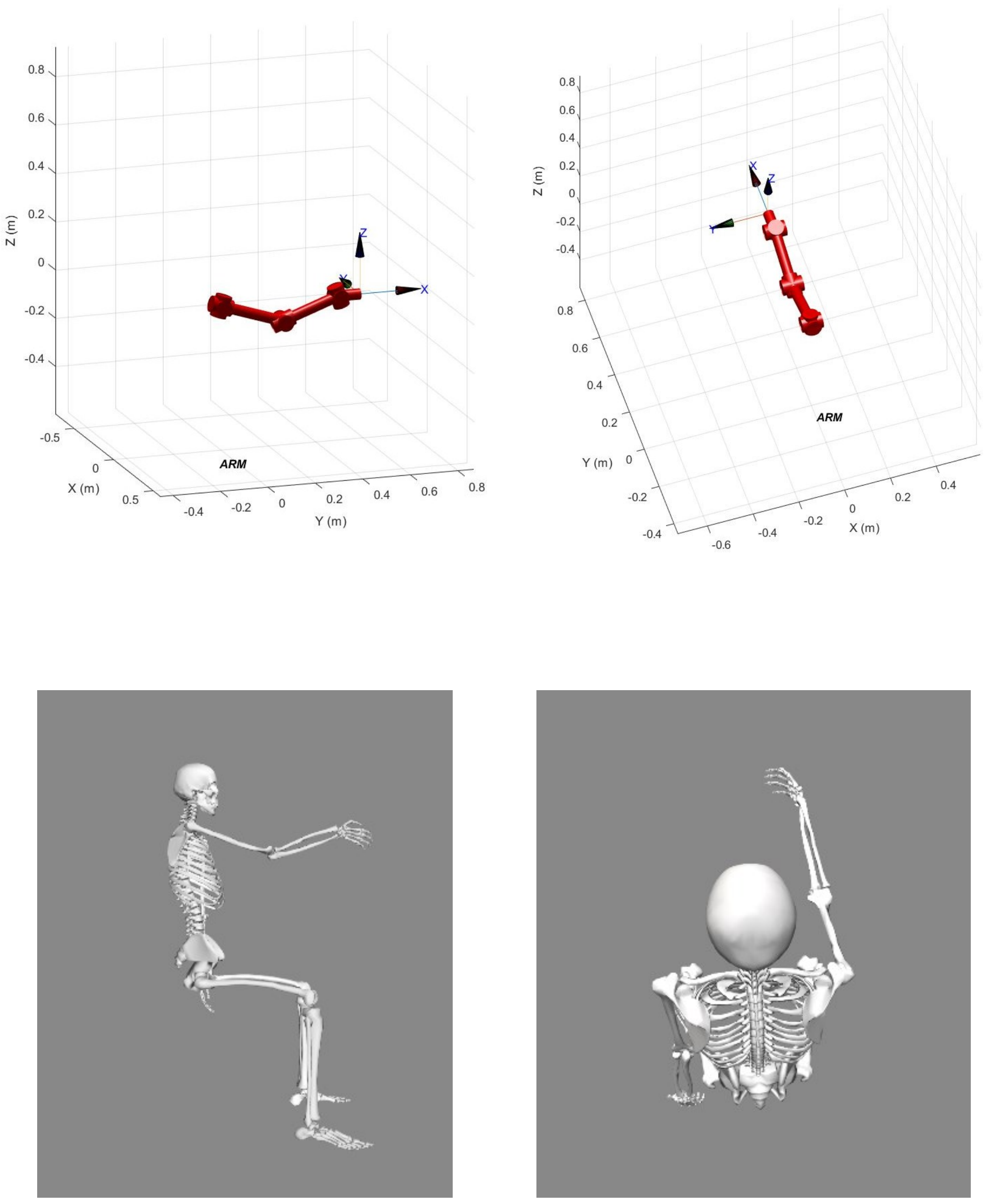
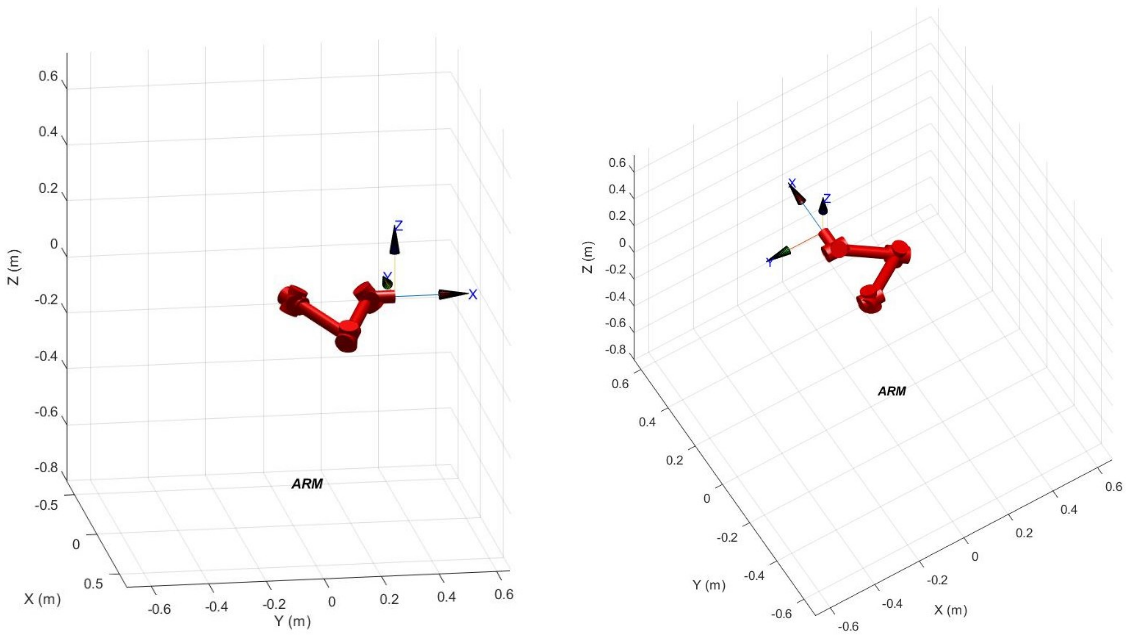


Figure 28. Solution for the point A.



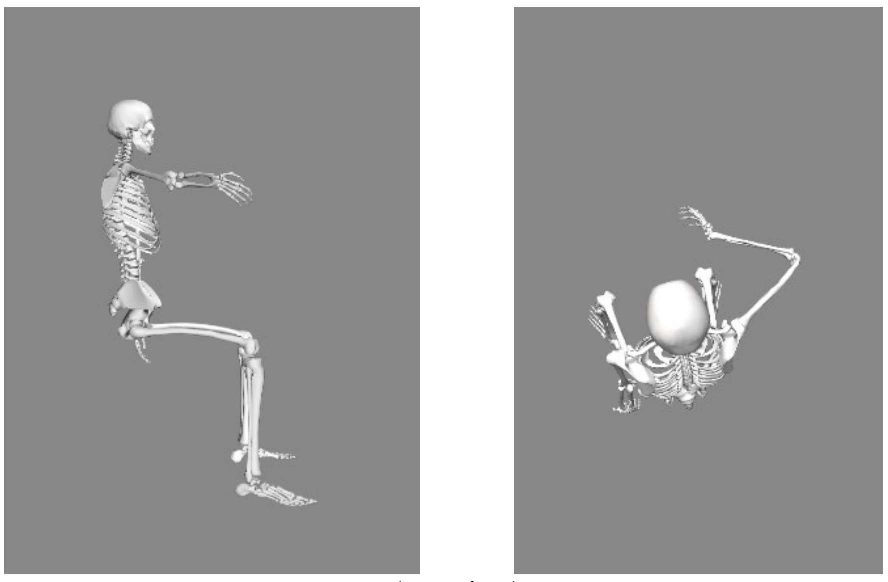


Figure 29. Solution for the point D.

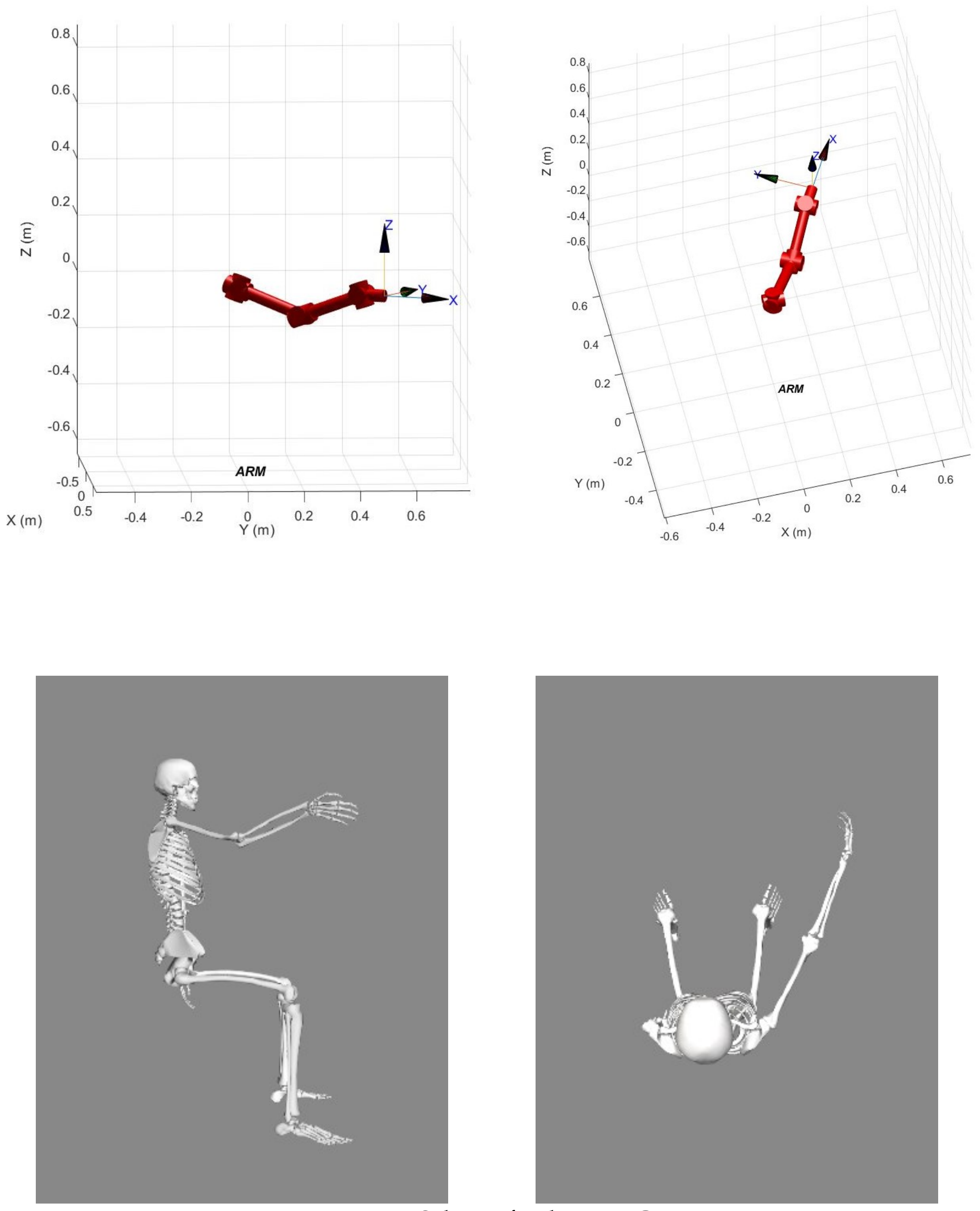


Figure 30. Solution for the point G.

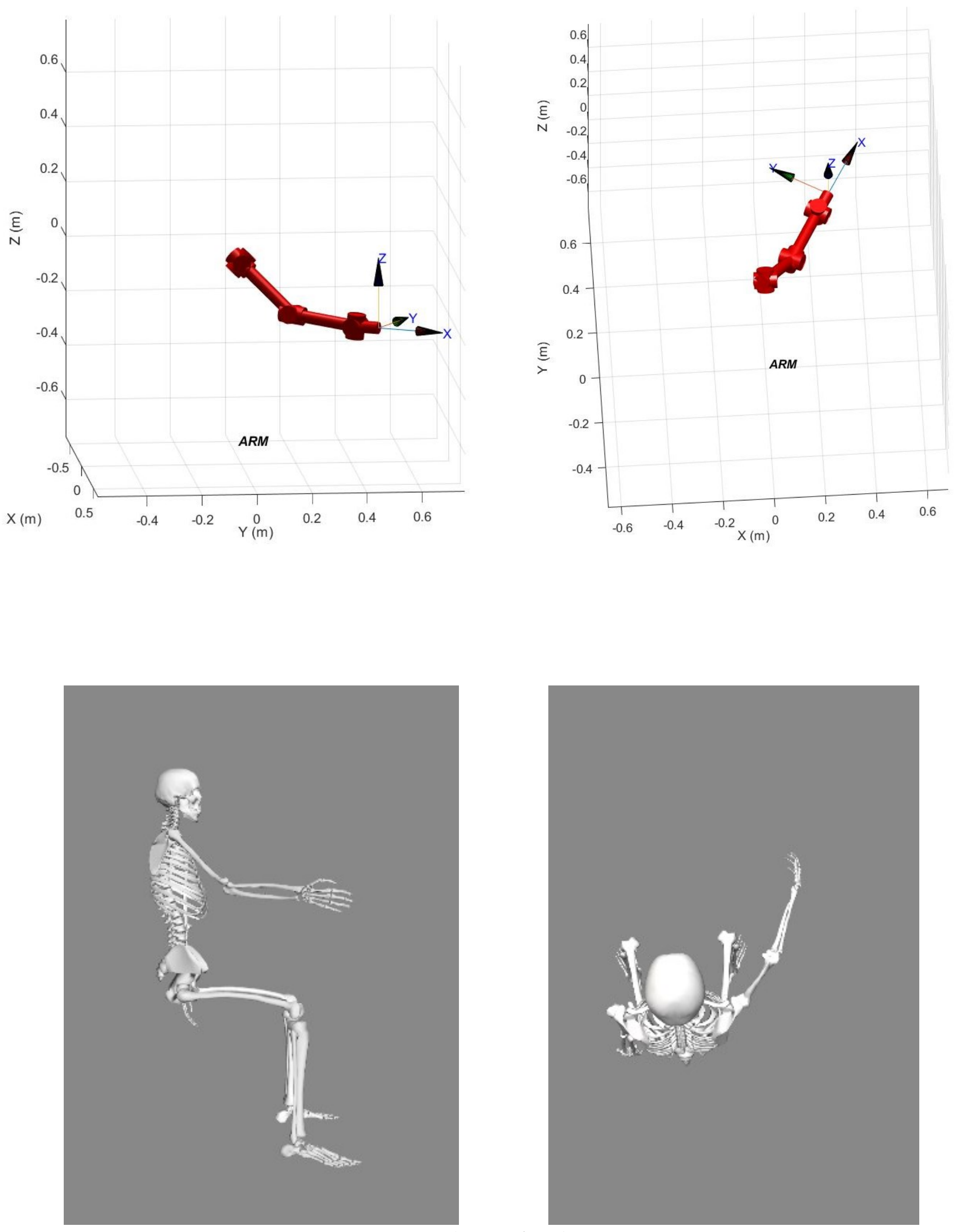


Figure 31. Solution for the point I.

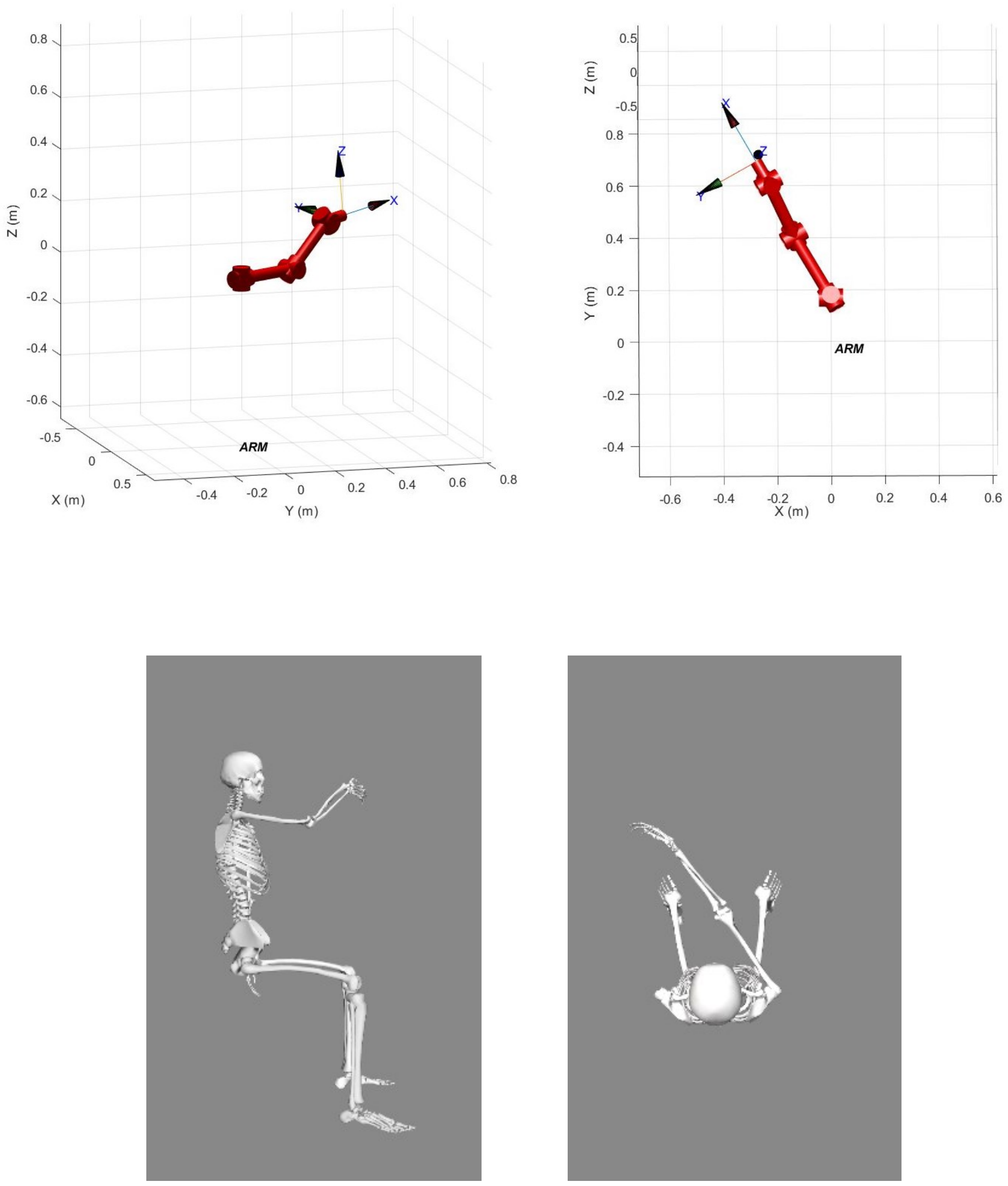


Figure 32. Solution for the point E′.

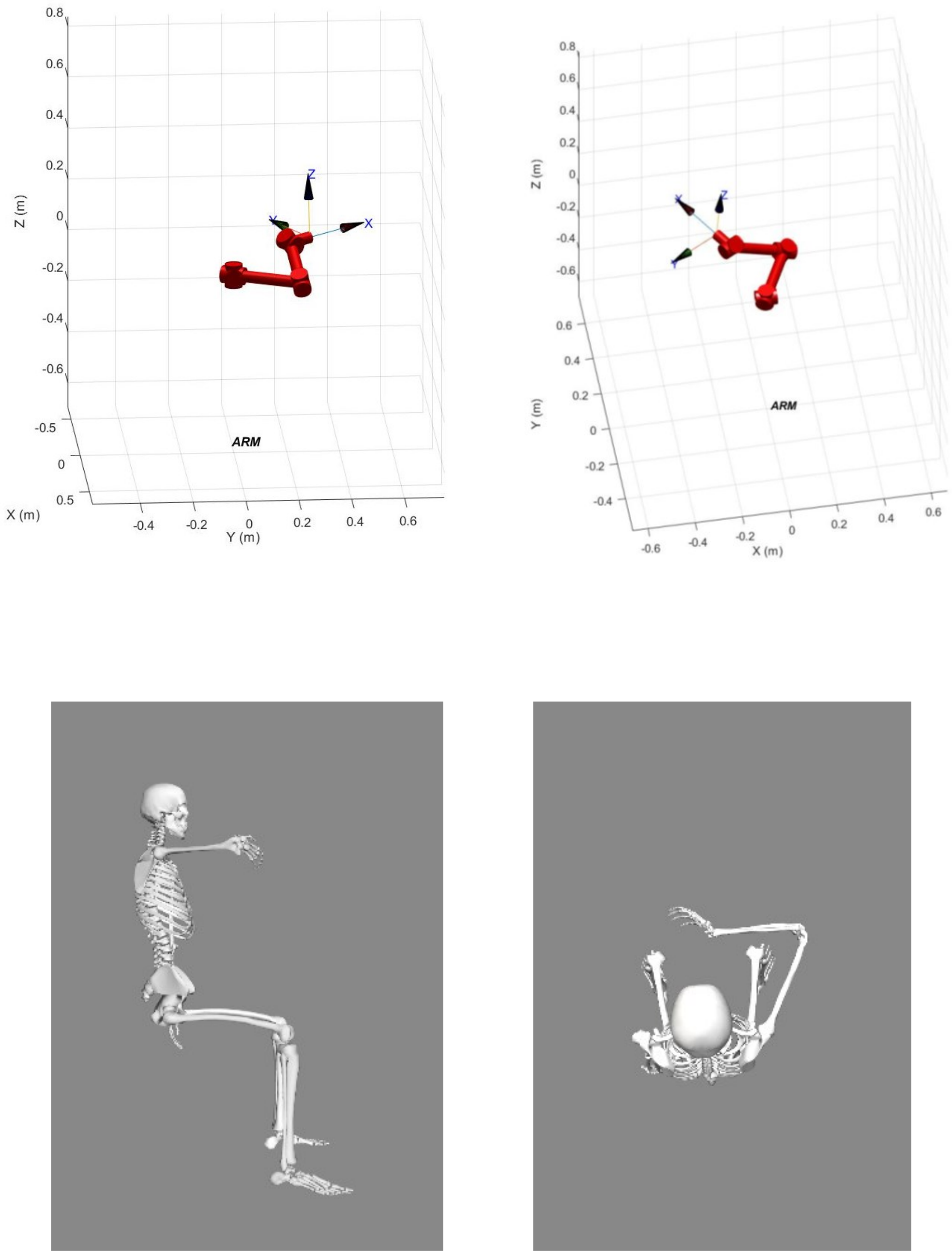


Figure 33. Solution for the point H'.

3.3 **Relative Position Patient and Robotic Arm**

The optimization problem has been defined to find the position between the base of the robotic arm and the right shoulder of the patient, according to a quality index. In this study, rotational parameters have been not included. Both robotic structures have been defined with respect to the absolute reference frame in Matlab environment. The main difference between the two structures is related to the homogeneous transformation matrix that is used to define the base of the robotic structures, as shown in paragraph 3.3. The origin of the robotic structure associated to the human upper limb is fixed in the origin of the absolute global frame, while the origin of the robotic structure associated to the UR5e moves with respect to the global coordinate reference system. Thus, the fourth column $\mathbf{v}$ of the $\mathbf{T_{base\_robotic\ arm}}$ matrix has to be defined in the global coordinate frame. If it is necessary to define the origin of the robotic arm base with respect to the base of the robotic structure associated to the human upper limb $\mathbf{x}$, the following calculation is essential:

$$\mathbf{x} = \mathrm{inv}\left(\mathbf{T_{base\_uppper\ limb}}\right)\mathbf{v}$$

This thesis section proposes an optimization algorithm to find the proper position of the robotic arm base, aims to solve $\mathbf{v}$.

3.4 **Inverse Kinematic of Robotic Arm**

The robot used to design the rehabilitation station is the UR5e, a six DoFs manipulator. Based on the knowledge of robotics, it is not a redundant structure because the number of the DoFs is equal to the task space. Therefore, it is always possible to solve the inverse kinematics, unless the desired pose is outside the workspace of the robotic arm. In fact, it is interesting to notice that having a movable robot base, not always will be possible to reach the desired pose but depends on the position of the robot base. On the other hand, if the pose can be reached by the manipulator, multiple solutions will be admissible. As well as for the human upper limb, also for the UR5e it has been chosen a numerical algorithm to solve the inverse kinematics. Thus, given the initial guess $\mathbf{q_0}$ and the configuration vector $\mathbf{q}$ is then calculated the error $|\mathbf{f}(\mathbf{q}) - \mathbf{x_D}|$ that has to be minimized. In the calculation of the error, $\mathbf{f}(\mathbf{q})$ is the forward kinematic calculated in the actual joint configurations, while $\mathbf{x_D}$ is the desired pose that has to be reached. In addition, to solve the inverse kinematic of the robotic arm it is necessary to well define the initial guess $\mathbf{q_0}$ because the solution will depend on this choice. To perform the algorithm, it has been chosen the starting joints configuration where the robotic is fully extended upward and the wrist is approaching, as shown in Figure 34.

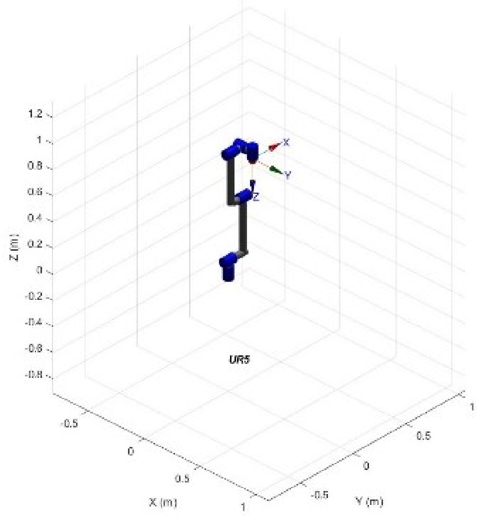


Figure 34. Initial configuration of the robotic arm given its base position in the origin of the global frame.

The algorithm used to solve the inverse kinematic is different than the human upper limb one. In UR5e there are not joint limits, thus, it is not necessary to use a function that optimize the solution according to these constrains. In fact, it has been used the function 'ikunc'. The inputs of the function are the robot end-effector pose defined as a homogeneous transform matrix (4x4), the initial guess and the options vector that allow to modify the settings and threshold used to stop the algorithm. On the other hand, the output of the function is the joint coordinates as vector (1xN) where N is the number of robot joints. The options used to solve the algorithm have been not changed with respect to the standard ones. To obtain the homogeneous transformation matrix used as input of the inverse kinematic algorithm of the robotic arm, it is necessary to solve firstly the forward kinematic of the human upper limb. In fact, given the joints configuration resulting from the paragraph 3.2, using Robotic Toolbox the pose of the end-effector of the human upper limb is obtained using the function 'fkine'. This function, given a set of joints configuration, calculates the pose (position and orientation) of the end-effector as a homogeneous transformation matrix (4x4):

$$\mathbf{T}_{\mathbf{fk\_human\_upperlimb}} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To build the homogeneous transformation matrix uses as input of the inverse kinematic algorithm of the robotic arm, it is necessary to preserve the position of the end-effector of the human upper limb that corresponds to the fourth column of this matrix $\mathbf{T}_{\mathbf{fk\_human\_upperlimb}}$. Furthermore, it is important to remember that, according to the definition of the kinematic model of the human arm, the Z-axis of the end-effector of the robotic structure related to the human upper limb is always in the direction of the thumb. Thus, in order to allow the correct grip, the Z-axis of the robotic arm has been defined antiparallel with respect to the Z-axis of the human arm end-effector.

Then, to guarantee a right-handed frame, the X-axis and Y-axis of the robotic arm end-effector have been defined antiparallelly to the Y-axis and X-axis of the human arm end-effector respectively. At the end the homogeneous transformation matrix used as input of the inverse kinematic algorithm of the robotic arm is defined as follow:

$$\mathbf{T_{ik\_roboticarm}} = \begin{bmatrix} -r_{12} & -r_{11} & -r_{13} & x \\ -r_{22} & -r_{21} & -r_{23} & y \\ -r_{32} & -r_{31} & -r_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In Figure 35 are shown the frames related to the human arm end-effector, obtained from its forward kinematic, and the robotic arm expected end-effector frame, and finally the solution of the inverse kinematic algorithm (given a specific robotic arm base position), shown in Figure 36. Using this approach, if the inverse kinematic of the robotic arm is well solved, it overcomes limitation, presented in paragraph 3.2, that can occur during the calculation of the inverse kinematic of the human arm as shown in Figure 27 and the two robotic structures form a close kinematic chain, thus the human arm is constrained to the movement of the robot. On the other hand, if the inverse kinematic of the robotic arm is not solved, a gap is presents between the two robotic structures. In Figure 37 are reported the two possible solutions.

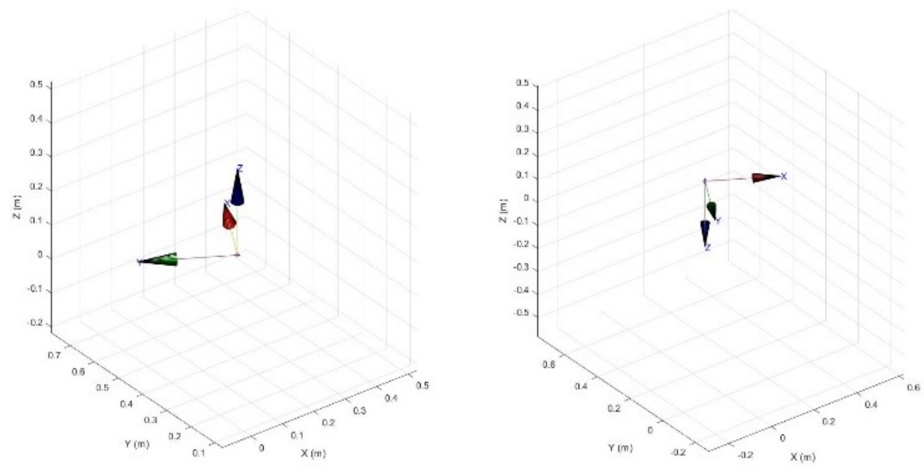


Figure 35. On the left, the coordinate system associated with the human upper limb end-effector. On the right the resulting coordinate frame used to solve the inverse kinematic in the robotic arm.

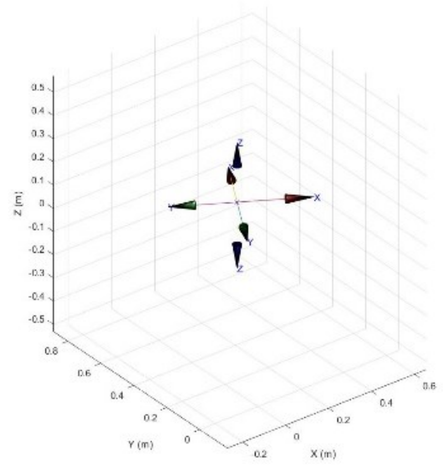
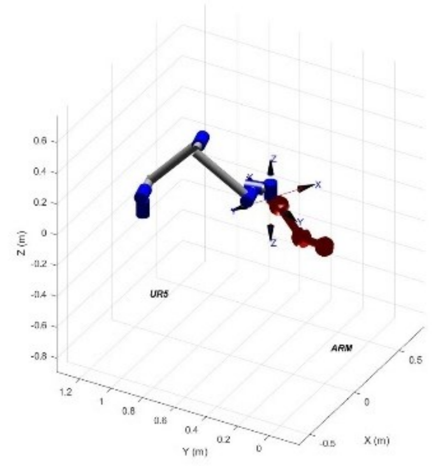


Figure 36. Overlapping between two local reference frames.



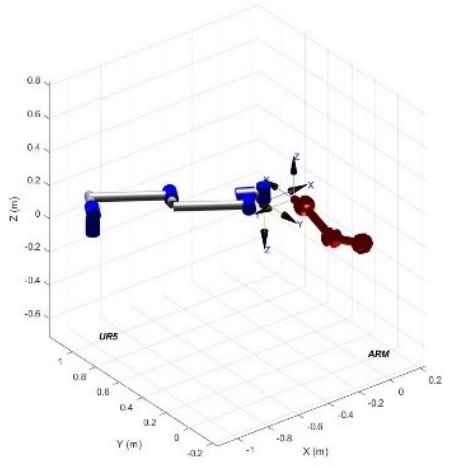


Figure 37. On the upper panel, closed chain representation. On the lower panel, open chain representation.

3.5 Manipulability Ellipsoids

To calculate the manipulability ellipsoids, it is necessary to have already computed the joint configurations and the Jacobian matrix for both the robotic and the human arms. As for regards the joint configurations has been used the output of the inverse kinematics algorithms. On the other hand, to calculate the Jacobian matrix for both the human arm and robotic arm, it has been employed the Robotic Toolbox. In fact, the function 'jacob0' permits to calculate the geometric Jacobian matrix. The output of this function is the Jacobian matrix (6xN), where N represents the number of joints belonging to the robotic structure, while the number of rows is fixed because represents both the linear and angular contributions to the velocity vector [29]. This function has been used because accounts for the base homogeneous transformation matrixes set in paragraph 3.3, for both the robotic and human arm. Then, the joint configurations and the Jacobian matrix have been used to calculate the manipulability ellipsoids, using a M-file function where all the steps necessary to calculate the manipulability ellipsoids have been defined, according to literature [29]. Thus, the inputs of this function are the joints configuration and the Jacobian matrix, while the outputs are the surfaces used to visualize the ellipsoids (force and velocity), their three semiaxes in descending order and the rotational matrixes that relate the local ellipsoid coordinate system with respect to the cartesian global frame. All these outputs have been stored for both the velocity and the force ellipsoids. The triplet related to the three ellipsoid semiaxes is normalized according to the maximum length of the semiaxes, so the first number of the triplet is always equal to one. Based on the robotics principle, the force and velocity ellipsoids that refer to the same robotic structure, are normalized with respect to different values. Furthermore, these ellipsoids are orthogonal each other, as shown in Figure 38 and in Figure 39 where ellipsoids are represented scaled by a factor equal to 1/5 and the longest semiaxis is presented in always in blue and the shortest one, always in red. The calculation of the semiaxes for both force and velocity ellipsoids, in both robotic structures, is an essential step toward the completion of the optimization algorithm because they will be necessary to compute the quality index.

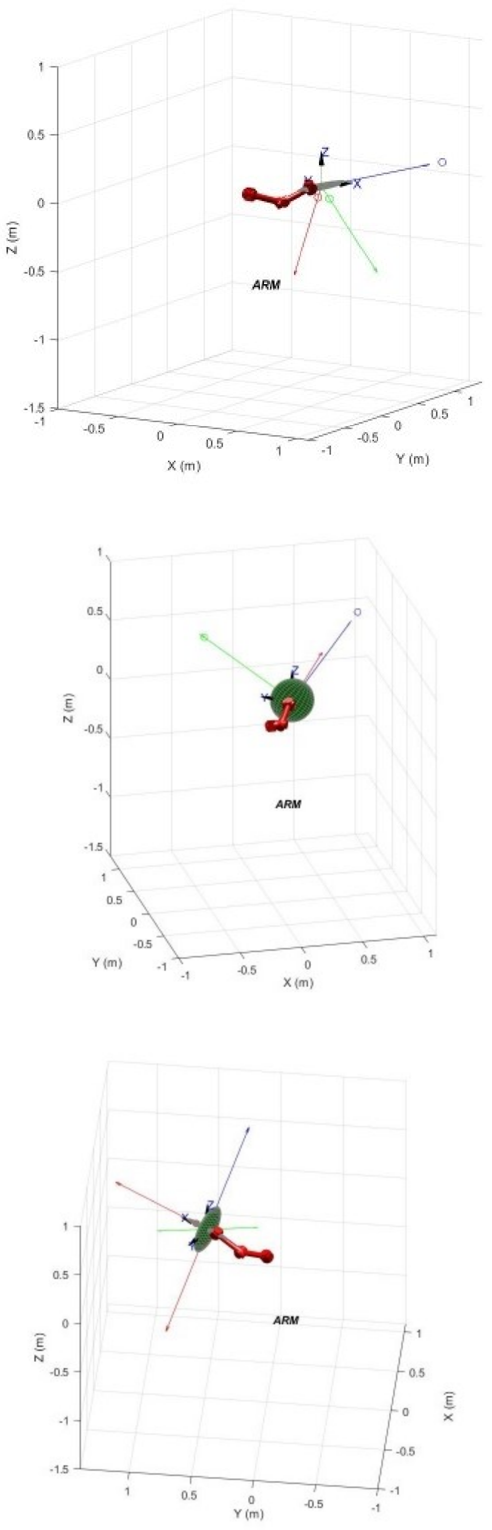
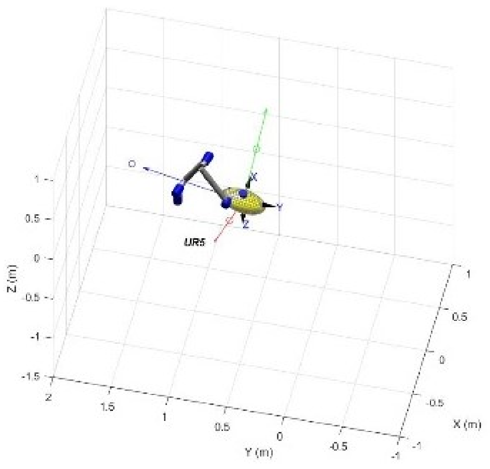
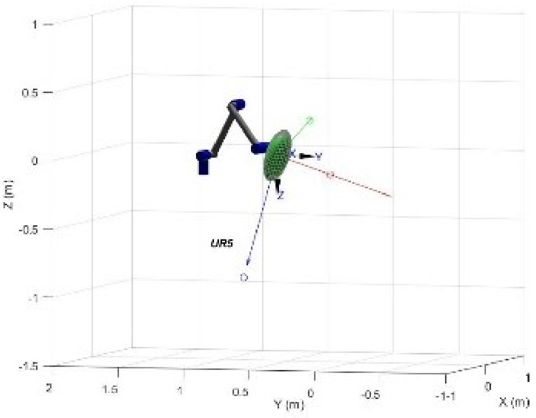


Figure 38. First panel, velocity ellipsoid. Second panel, force ellipsoid. Third panel, both ellipsoids in a configuration of the human upper limb.





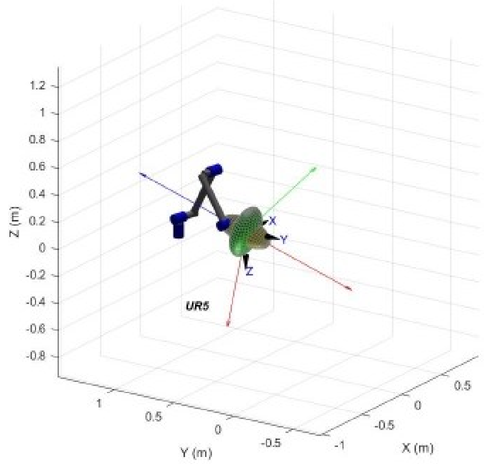


Figure 39. First panel, velocity ellipsoid. Second panel, force ellipsoid. Third panel, both ellipsoids in a configuration of the robotic arm, given a specific robot base position.

As expected from literature [29], since the force and velocity ellipsoids of each robotic structure are orthogonal each other, the longest semiaxis of the velocity (or force) ellipsoid corresponds to the shortest semiaxis of the force (or velocity) and vice versa. This propriety is confirmed in Figure 38 and Figure 39. Then, the manipulability ellipsoids obtained through this approach have been compared to the ellipsoids resulting from forward kinematic and Jacobian matrix calculated using already defined M-file functions that solve the forward kinematic using another method with respect to the D-H one and calculate the Jacobian analytic matrix, as shown in Figure 40.

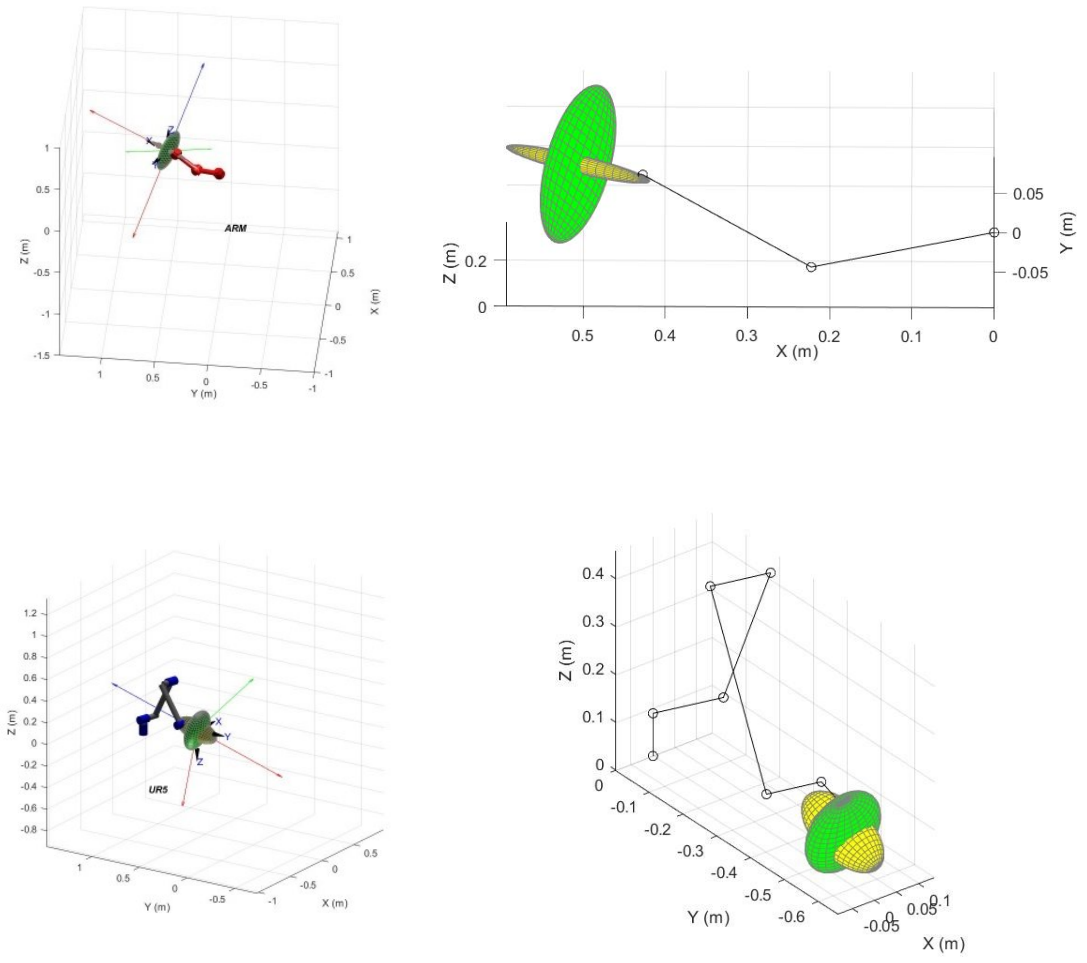


Figure 40. On the left side, manipulability ellipsoids obtained using Robotic Toolbox. On the right side, manipulability ellipsoids, in the same configurations, obtained using M-file.

The global reference system used to build the kinematic model of the human upper limb is different between the M-file and Robotic Toolbox, but both models are capable to correctly represent the DoFs of the OpenSim software model. Therefore, the unique difference between the models is the Jacobian matrix that results to be characterized by the same value but in different directions. In the end, the ellipsoids obtained are equivalent. On the other hand, for the UR5e both kinematic models have been defined in the same global frame so, all the quantities involved in the calculations of the ellipsoids are equal.

3.6 **Index Definition**

To complete the optimization algorithm, it has been necessary to define the quality index that compares ellipsoids (force or velocity) of the two different robotic structures. Therefore, for each robotic and human arm joint configurations, the index has been calculated twice to quantify both the similarity between the force ellipsoids, as well as the velocity ellipsoids. The quality index designed to solve the optimization algorithm is described as follows:

$$\text{index} = \frac{\sum_1^3 \left|\mathbf{a_{i,r}} \cdot \mathbf{a_{i,a}}\right|}{\sum_1^3 \mathrm{a_{i,r}} \mathrm{a_{i,a}}}$$

Where $\mathbf{a_{i,r}}$ corresponds to the ellipsoid semiaxes of the robotic arm, expressed in the human arm ellipsoid coordinate system, and $\mathbf{a_{i,a}}$ corresponds to the ellipsoid semiaxes of the robotic structure associated to the human upper limb. In the numerator it has been calculated, in descending order according to the triplet defined in paragraph 3.5, the three scalar products between the robotic and human arm ellipsoid semiaxes. Being the result of a scalar product a quantity positive or negative, it has been chosen to calculate its absolute value. This choice has been proposed considering that the perfect matching between force or velocity ellipsoids, that occur only if all the semiaxes are coincident, occurs also if the corresponding semiaxes are antiparallel each other.

Therefore, it is sufficient to have semiaxes that have the same direction, while the same orientation is not necessary. According to this index definition, the maximum value of the index is equal to one. This value is reached only if the two compared ellipsoids are coincident. Furthermore, the alignment of the longest semiaxes is weighted more than the alignment of the shortest semiaxes since the definition of the denominator. To properly calculated the index, since ellipsoid semiaxes are defined in different local reference frame, it has been necessary to move the semiaxes of the robotic arm in the local ellipsoid reference system of the human upper limb. This local frame has the origin centred in the human arm end-effector position and the orientation of the axes depends on the ellipsoid obtained by a particular joints configuration and they correspond to the semiaxes of the ellipsoid. Firstly, it has been moved the robotic arm ellipsoid semiaxes from their local frame (that is centred in the same point of the human arm ellipsoid frame because both inverse kinematic are already solved) to global reference system using the rotational matrix, obtained from paragraph 3.5, as follows:

$$\mathbf{a}_{\mathbf{i,r\ global}} = \mathbf{R}_{\mathbf{global}}^{\mathbf{robot\ ellispoid}}\ \mathbf{a}_{\mathbf{i}}$$

Where $\mathbf{a}_{\mathbf{i}}$ represents the semiaxis of the robotic arm in its local ellipsoid frame, $\mathbf{R}_{\mathbf{global}}^{\mathbf{robot\ ellispoid}}$ represents the rotational matrix that allows to move from the local robotic ellipsoid frame to the global reference system and $\mathbf{a}_{\mathbf{i,r\ global}}$ corresponds to the ellipsoid semiaxes of the robotic arm expressed in the global system coordinate system. This procedure has been applied for all the semiaxes. Once this step has been completed, all the semiaxes have to be moved in the local ellipsoid frame of the human upper limb, as follows:

$$\mathbf{a}_{\mathbf{i,r}} = \mathrm{inv}(\mathbf{R}_{\mathbf{arm\ ellipsoid}}^{\mathbf{global}})\ \mathbf{a}_{\mathbf{i,r\ global}}$$

Where $\mathbf{a}_{\mathbf{i,r\,global}}$ represents the semiaxis of the robotic arm in global frame, $\mathbf{R}_{\mathbf{arm\ ellipsoid}}^{\mathbf{global}}$ represents the rotational matrix that allows to move from the local human arm ellipsoid frame to the global reference system and $\mathbf{a}_{\mathbf{i,r}}$ corresponds to the ellipsoid semiaxes of the robotic arm expressed in the human arm ellipsoid coordinate system. Only at the end of this process it has been possible to calculate the index because all the vectors are expressed in the same reference frame. It is important to highlight that if the closed kinematic chain is obtained, the two local frames related to different ellipsoids have the same origin but different axes directions. Therefore, the norm of each semiaxis have been not modified during the transformation from one frame to another. On the other hand, if the open chain has been formed at the end of the inverse kinematic algorithms, the index has been set equal to zero because the two robotic structures do not share the same origin position, so ellipsoids cannot be compared.

In Figure 41 is shown an index value equal to 0.2000. This result is obtained because the semiaxes of the ellipsoids are almost perpendicular each other, then the scalar product is close to zero for the three scalar products.

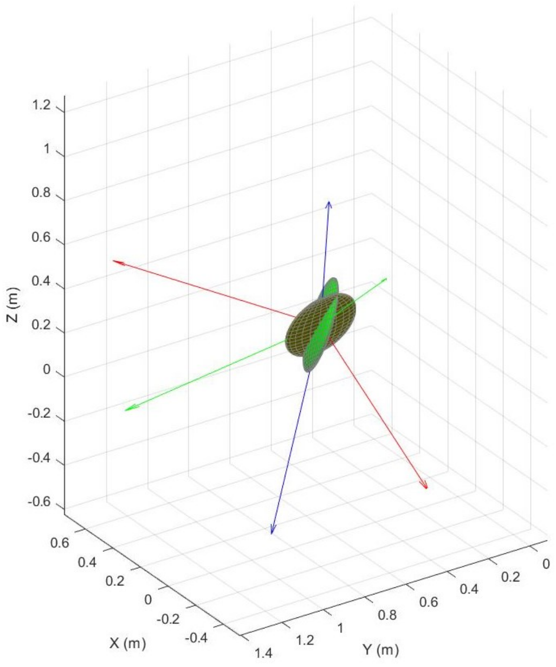


Figure 41. Index value equal 0.2000.

In Figure 42 is shown an index value equal to 0.5766. The angle between the two longest semiaxes that is close to 45° to prove that the longest semiaxes scalar product weights more than the others. In fact, even if the shortest and the intermediate semiaxes of the two ellipsoids are perpendicular each other, the final value of the index depends more on the scalar product of the longest semiaxes. In Figure 43 is shown an index value equal to 0.9763. This result is obtained because all the semiaxes are parallel each other.

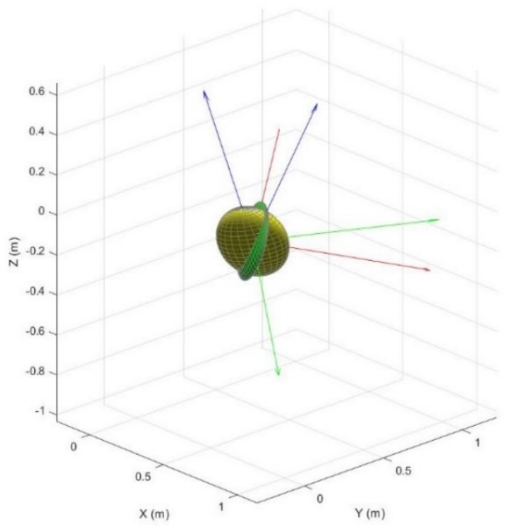


Figure 42. Index value equal to 0.5766.

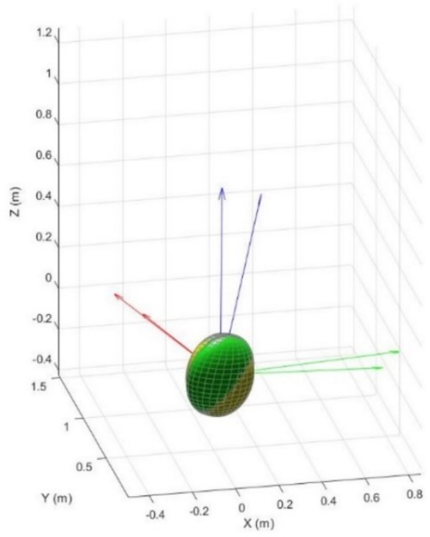


Figure 43. Index value equal to 0.9763.

3.7 **Optimization Problem**

The optimization problem has been defined to assign the three translational parameters to the homogeneous transformation matrix, determining the position of the robotic arm base in the global reference frame. Two main steps have been performed to solve this problem. Firstly, it has been mapped the frontal and the lateral regions with respect to the patient. Then, using the local maxima found during this step, it has been performed a local optimization algorithm. Only the frontal and lateral sides of the patient have been mapped because the collaborative robot works side-by-side with humans, as well as it is the space where the patient performs the rehabilitative task. The mapped space is characterized by two planes placed at different hights with respect to the patient shoulder. The first plane is place at the same level of the patient and the other one is placed at the level of the elbow, considering a fully extended downward human arm. For each plane, it has been defined a rectangular region with respect to the patient right shoulder that goes from -1.5 m to 1.5 m in the global X-direction and from 0 m to 1.5 m in the global Y-direction. The resolution used to map the different planes is 10 cm, as shown in Figure 44. This choice permits to obtain high level of reliability because the base of the robot has a footprint of 148 mm.

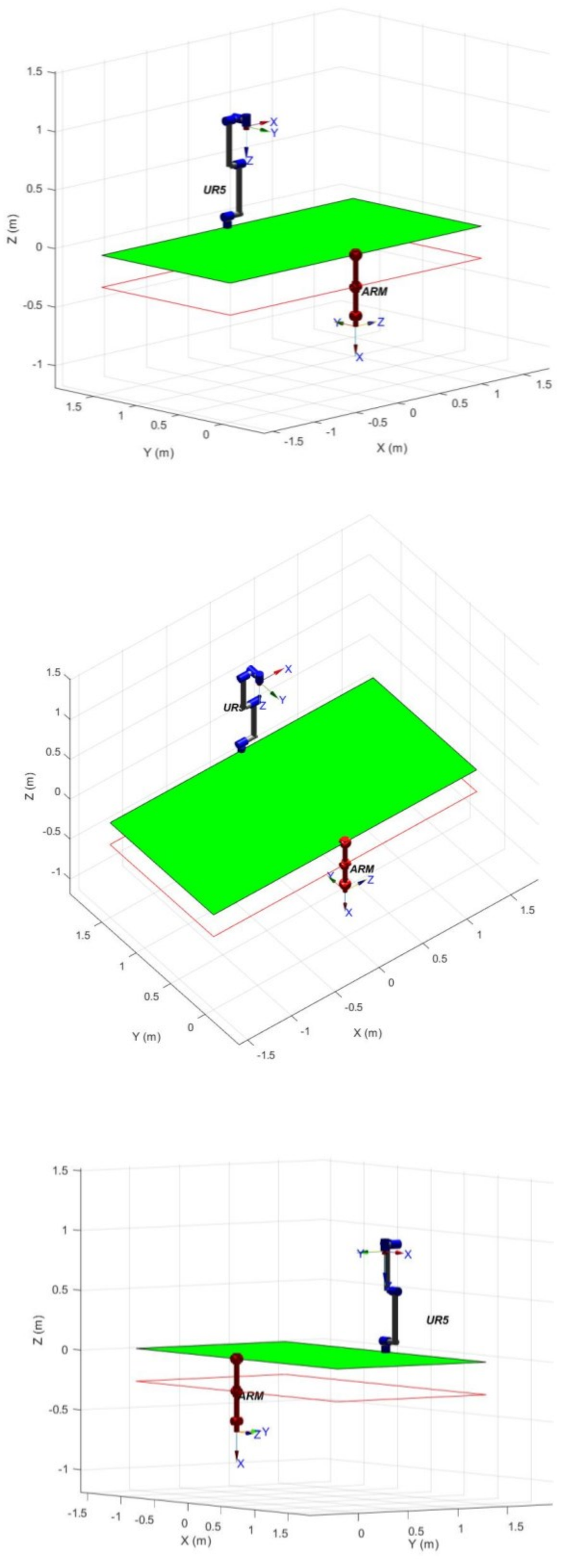


Figure 44. Planes used to solve the global optimization. The first plane (black contour) at the same level of the patient shoulder and the second one (red contour) at the level of the patient elbow.

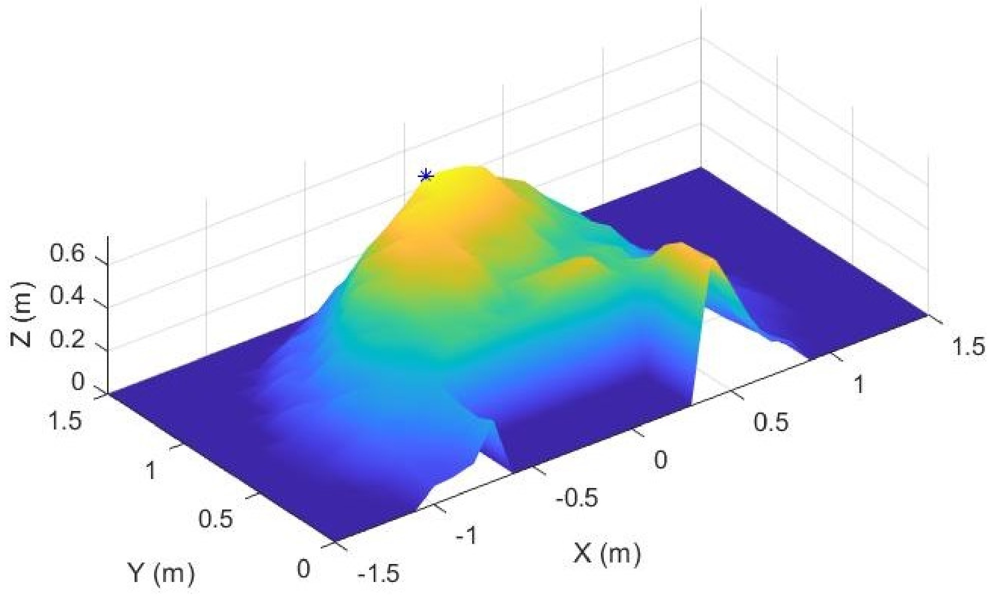
It has been implemented a loop in Matlab environment that changes the position of the robotic arm base. For each position, it has been solved the forward of the human upper limb according to each joints configuration in Table 12. For each pose obtained from this calculation, it has been solved the inverse kinematic algorithm of the robotic arm, as described in paragraph 3.4. Consequently, Jacobian matrixes of human and robotic arm have been calculated. Therefore, only if the close kinematic chain is obtained, as shown in Figure 37, force and velocity ellipsoids of both robotic structures have been calculated to calculate the quality index, as presented in paragraph 3.6.

At the end eighteen index values are obtained for velocity ellipsoids, as well as eighteen index values are obtained for force ellipsoids. Thus, the mean value is computed to obtain a value from zero to one. This is possible because the index values are within the range from zero to one. At the end, for all the points in the mapped three-dimensional global spaced, two values are obtained, one is related to the mean index value of velocity ellipsoids and the other one is related to the mean index value of force ellipsoids. Interpolated results are shown in Figure 45, Figure 46, Figure 47 and Figure 48. In the previous figures, it has been not considered a rectangle near to the patient, index values have been set equal to zero, in order to guarantee the possibility to conclude correctly a certain task and to ensure the absence of obstacles during the rehabilitation process.

In Table 13 are summarized the local maxima according to the mapped space around the patient.

Table 13. Local maxima resulting from the first step of the optimization process.

	X (m)	Y (m)	Z (m)	Index Value
Velocity Ellipsoids	-0.2000	1.100	0	0.7367
Force Ellipsoids	-0.2000	1.100	0	0.8342
Velocity Ellipsoids	0	1.100	-0.2737	0.7038
Force Ellipsoids	0	1.100	-0.2737	0.7857



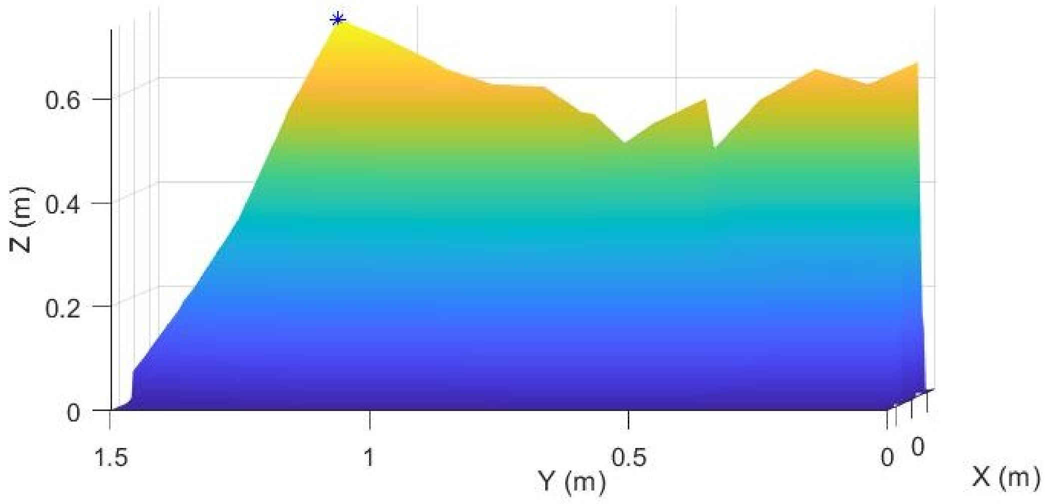
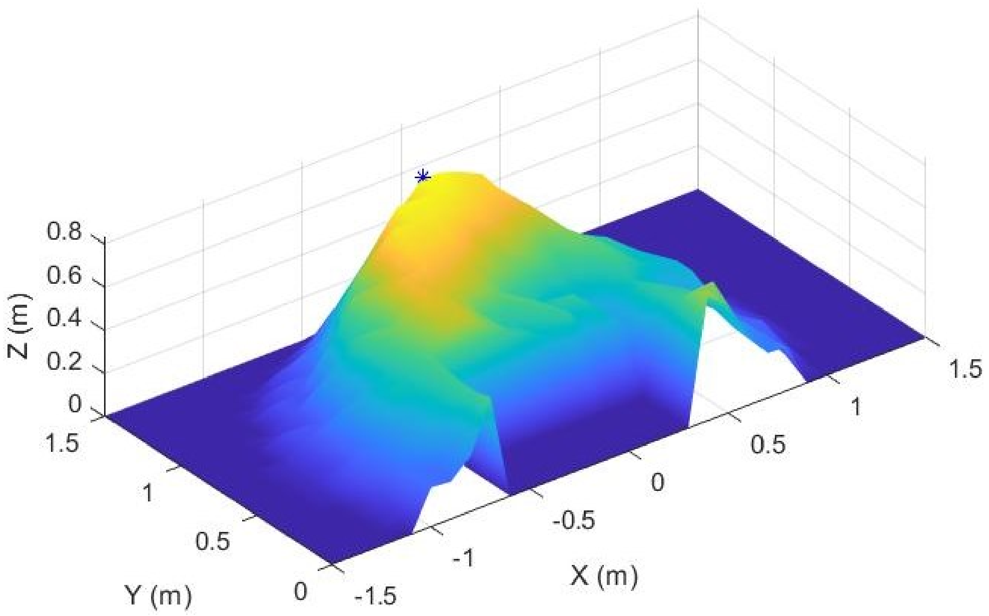


Figure 45. Velocity index evaluated at the height of the patient shoulder. Highlighted the maximum value.



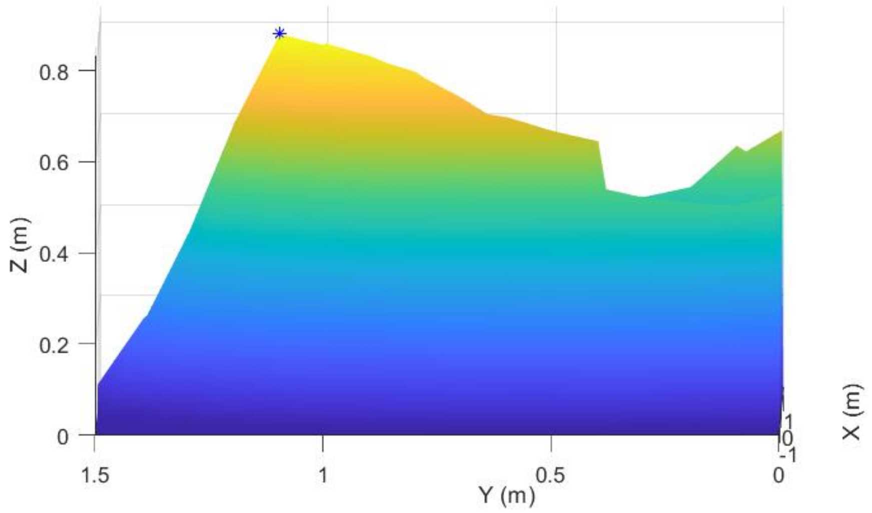
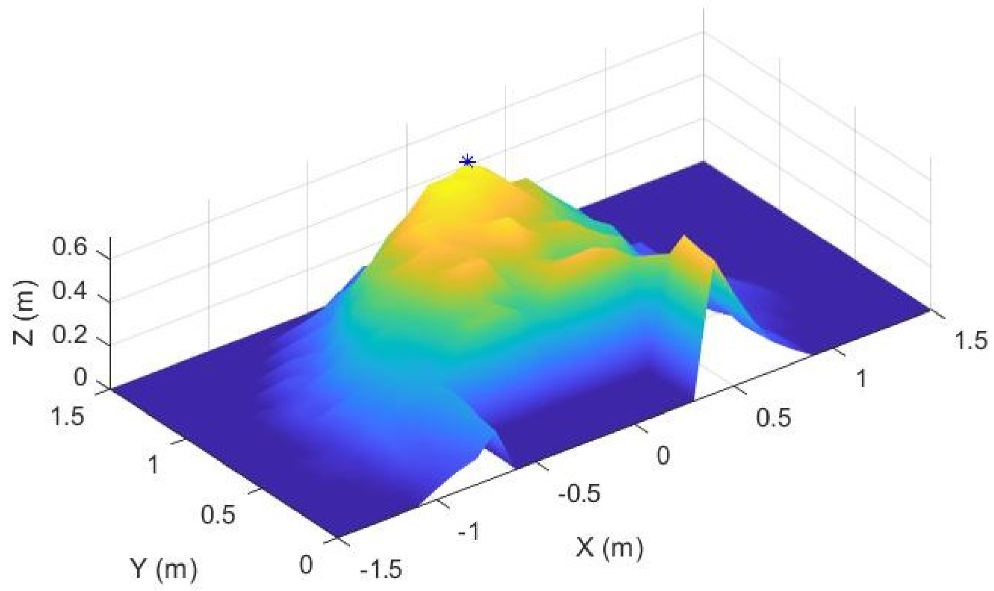


Figure 46. Force index evaluated at the height of the patient shoulder. Highlighted the maximum value.



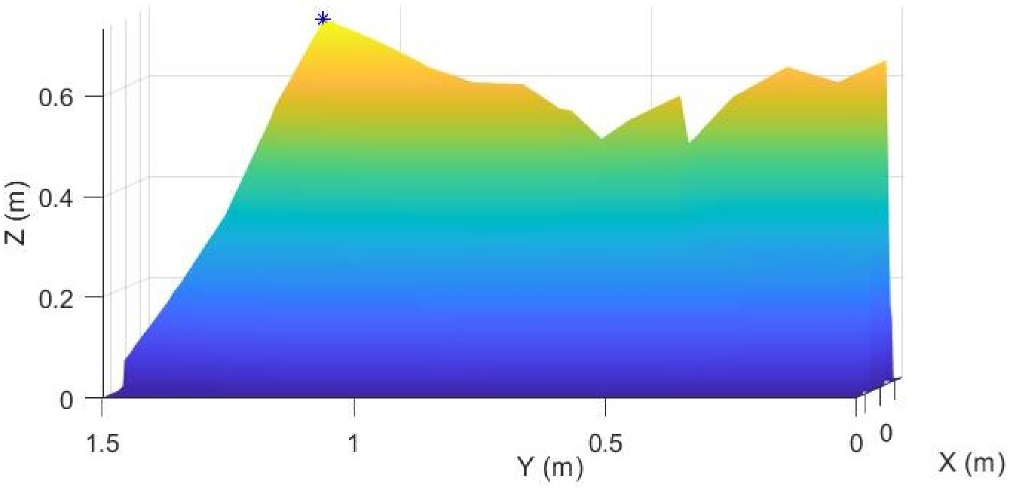
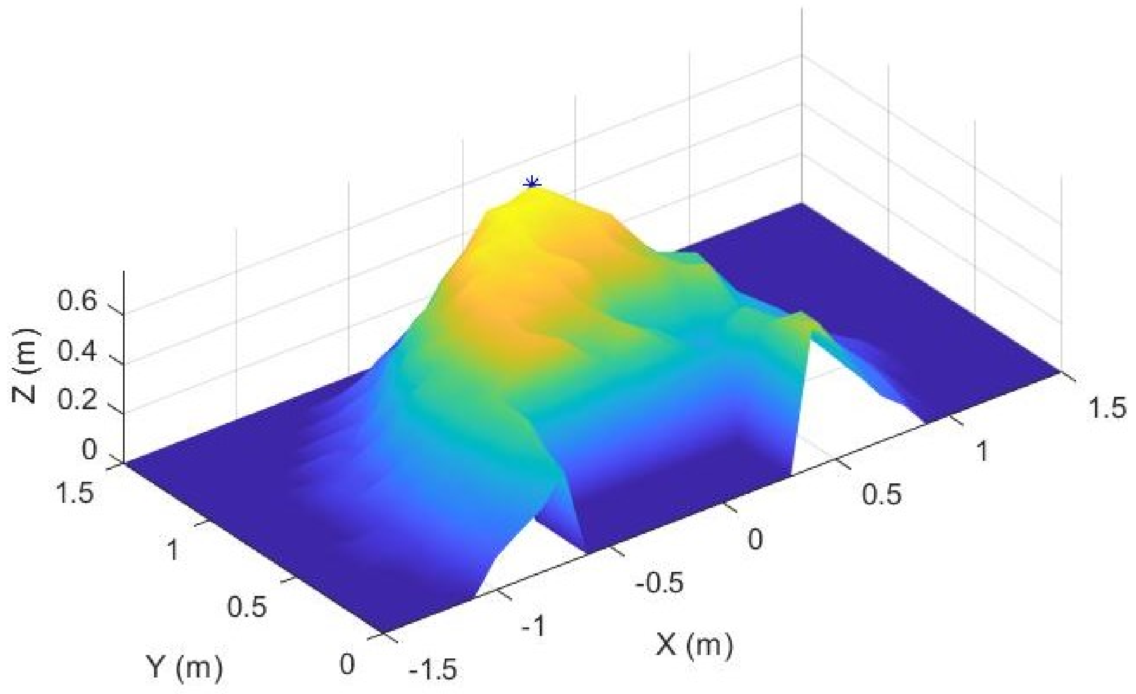


Figure 47. Velocity index evaluated at the height of the patient elbow. Highlighted the maximum value.



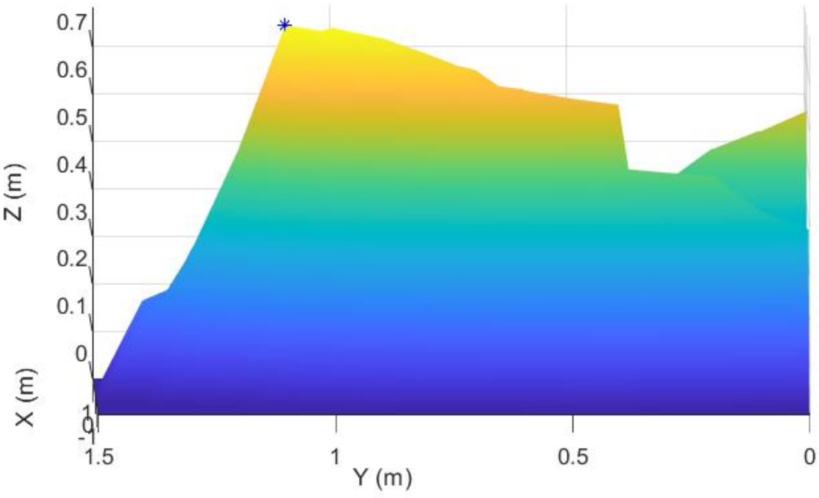


Figure 48. Force index evaluated at the height of the patient elbow. Highlighted the maximum value.

Starting from the local maxima, the second part of the optimization problem aims to explore new workspace regions. To perform this second part of the algorithm it has been used the 'fminsearch' Matlab function. In detail, this Matlab function solves optimization problems finding the minimum of unconstrained multivariable function using derivate-free method. The inputs are the function that has to be minimized, a vector that includes the starting values of this function, that correspond to the rows in Table 13 and the option vector, while the output of this function is a vector of the same dimension of the input vector that incorporate the optimal parameters that permit to minimize the function. This Matlab function permits to solve only minimization problem. Thus, using the function 'fminsearch' it has been minimized the reciprocal of the index. The options used to solve the minimization problem are the standard ones.

In particular, the robot base has been programmed to move in the three-dimensional space without any constrains, differently from the first step where the robot had to move in predefined planes. As a result of the second step, new position of robotic arm base has been obtained in the three-dimensional global reference frame. These values are the three translation parameters that solve the optimization problem, as shown in Table 14.

Table 14. Maxima resulting from the entire optimization process.

	X (m)	Y (m)	Z (m)	Index Value
Velocity Ellipsoids	-0.1184	1.1572	0.001	0.7491
Force Ellipsoids	-0.1937	1.1292	0.001	0.8386
Velocity Ellipsoids	0.001	1.1579	-0.1127	0.7392
Force Ellipsoids	0.001	1.1471	-0.1036	0.8127

These results suggest that to design the rehabilitative station, independently from the exercise (force task or velocity task), it is necessary to place the robot base in front of the patient, as shown in Figure 49, in order to obtain high level of manipulability, according to the defined index.

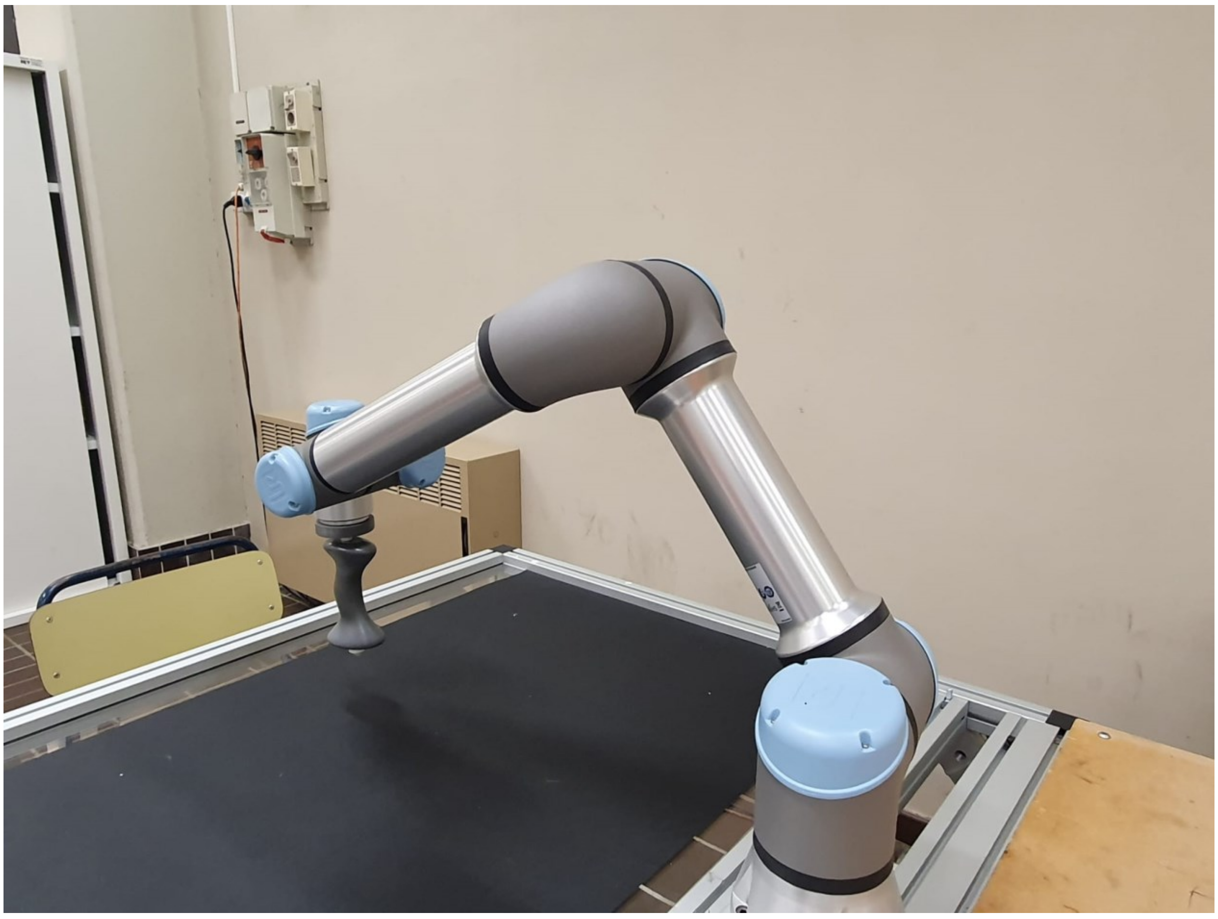


Figure 49. Robot mounted in front of the patient seat.

4. Task Definition

The collaborative robot UR5e and the smart camera Cognex In-Sight 7600 constitute the rehabilitative station. The purpose of the second part of this study is to build a system capable to perform the rehabilitative task. Initially, it has been performed the calibration of the camera according to the hand-eye method. Then, it has trained the vision system to recognize the target that the patient has to reach. Finally, it has been programmed the robot to reach the chosen target and complete the desired task. The calibration and the target recognition have been finalized in Explorer Insight software, while the programming of the robot has been complete in Polyscope interface. At the end, the aim of this rehabilitative process is to stimulate the neuroplasticity of the patient that is the ability of neurons to modify the strength of existing synapses [1].

4.1 Camera Calibration

Before entering in the details of the calibration method, it is necessary to define where the camera is mounted with respect to the working plane to obtain a proper field of view. In fact, to build the rehabilitative station it has been chosen to mount the camera above the table to reach greater field of view because it is possible to increase the operative distance, as shown in Figure 50. In particular, the Cognex website provides a tool to know a-priori the field of view of the camera using information related to the camera sensor and the ideal working distance. At the end, it has been chosen to set the vision system 80 cm above the working plane to obtain a field of view equal to 72 cm and 54 cm on the working plane (width and height respectively).

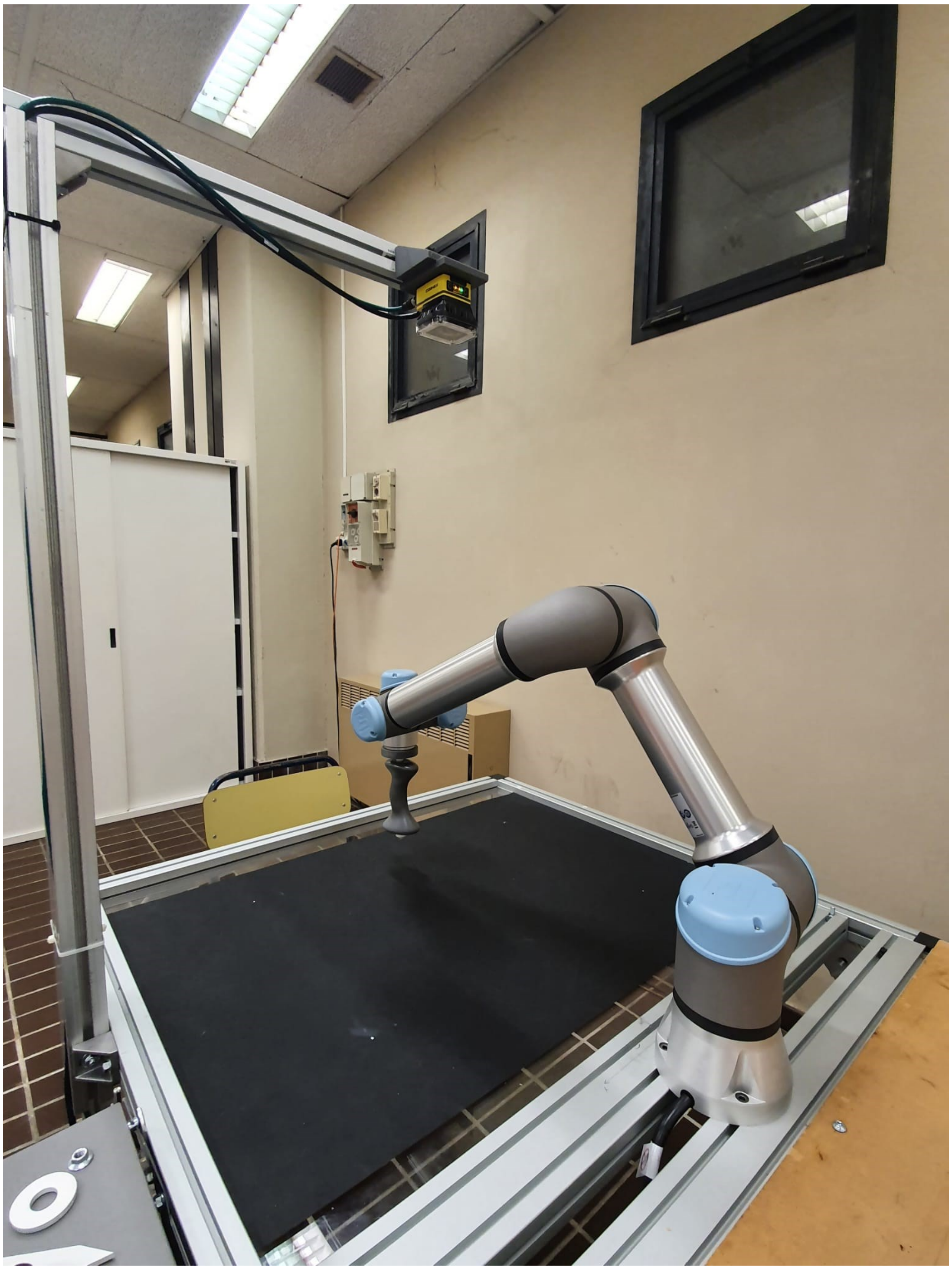


Figure 50. Rehabilitation station final design.

Once the vision system has been mounted properly, the first step toward the definition of the rehabilitative task is the calibration of the camera. The method used to calibrate the vision system and the collaborative robot is the hand-eye robot calibration. Being the camera fixed and the target free to move with respect to the global reference system, it is an eye-to-hand calibration.

This method enables proper perception of the environment in which a vision guided robot operates. Additionally, it enables the mapping of the scene in the robots frame [30]. This algorithm is based on the knowledge of the homogeneous transformation matrix (4x4) from the robot base to the robot-end-effector, that correspond to the forward kinematic, and the homogeneous transformation matrix (4x4) from the target to the camera system in order to calculate the homogeneous transformation matrix that allow to move from the camera reference frame to the end-effector reference frame. Using this method, it is possible to move from the camera system to the robotic base frame [30]. From a mathematical point of view, this eye-to-hand calibration aims to solve the following equation:

$$\mathbf{AX} = \mathbf{XB}$$

Where A and B are the known homogeneous transformation matrixes because A represent the movement of the target with respect to the camera reference system and B represent the movement of the end-effector with respect to the robot base coordinate system, as shown in Figure 51.

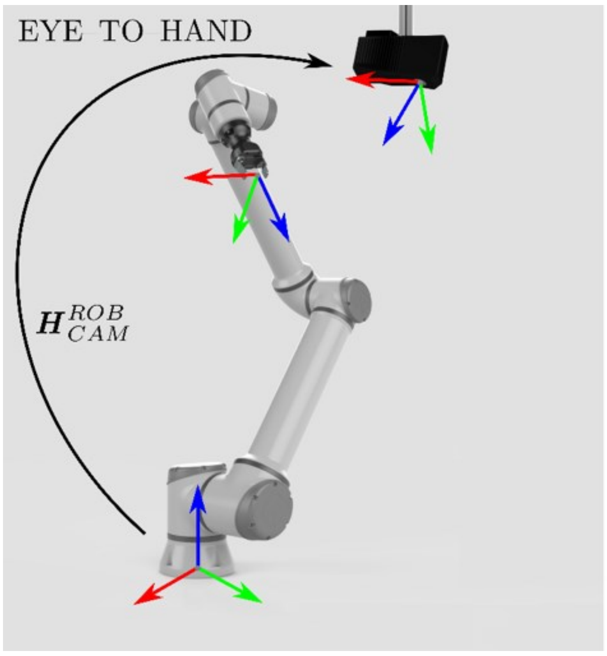


Figure 51. Calibration eye-to-hand representation.

To find the matrix X it is necessary to solve an optimization problem. In fact, least 10 or 15 frames have to be acquired, obtaining different A and B matrixes to properly find the calibration matrix. The matrix X correspond to the movement of the end-effector with respect to the robot base, as well as of the target with respect to the camera reference system. To complete this algorithm, being the vision system used to build the rehabilitative station a smart camera it guarantees the possibility to process images. Using the Explorer-insight software the camera has been trained to recognize a certain target, used to calibrate the working plane.

Firstly, it is necessary to acquire the image where is visible the target used to calibrate the system. Then, using the software options it is possible to modify parameters, as the autofocus that is mounted on the Cognex In-Sight 7600, to improve the quality of the image. Once the image is defined, using a 'Locate Part tool', in detail PathMax Pattern, the camera is trained to recognize the pattern (custom-made printed tip), as shown in Figure 52. This target has been designed in order to guarantee the possibility to perform the calibration process maintaining the camera focal axis and the end-effector Z-axis parallel each other.

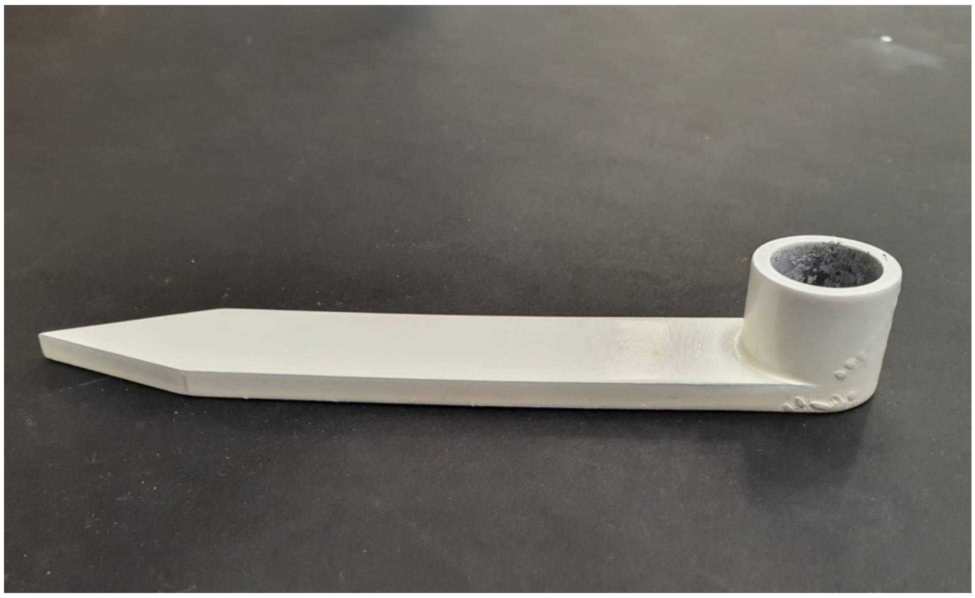


Figure 52. Target used to calibrate the camera.

Then, it is crucial to select in the Explorer In-Sight software the 'Inspect Part tool' and in the calibration field, choose the robot calibration and assign to the pattern the one that has been previously defined. Once the target and the calibration tool have been properly defined, the second part of the calibration is completed in the Polyscope software where three points have to be defined to find the robot working plane, as shown in Figure 53.

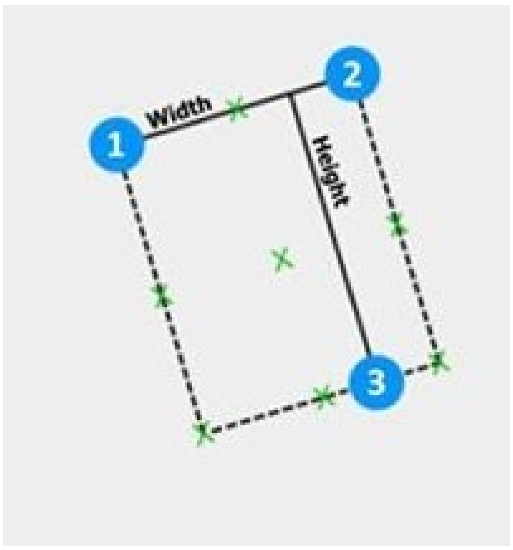


Figure 53. Polyscope representation of the working plane.

In fact, using the robot interface, the end-effector together with the defined target has to be moved manually to define two points for the width of the working plane and one point to define then, the height. At the end of this process, it is necessary to start the 'Robot Calibration' from Explorer In-Sight software and the robot will move in the defined field of view determining 20 points that are recorder from the camera and used to solve the calibration problem. Finally, a calibration file is created from the vision system software, where the working plane has been defined and is possible to move from the smart camera coordinate system to the robotic reference frame.

During the calibration process, it is necessary that the controller of the collaborative robot, the vision system and the workstation are connected each other using an Ethernet switch and maintaining all the device on the same network.

4.2 Robot Guidance

Since the camera calibration has been accomplished, the robot guidance has to be performed. In this rehabilitative task, the end-effector of the robotic arm has been designed with an ergonomic handful, as shown in Figure 54.

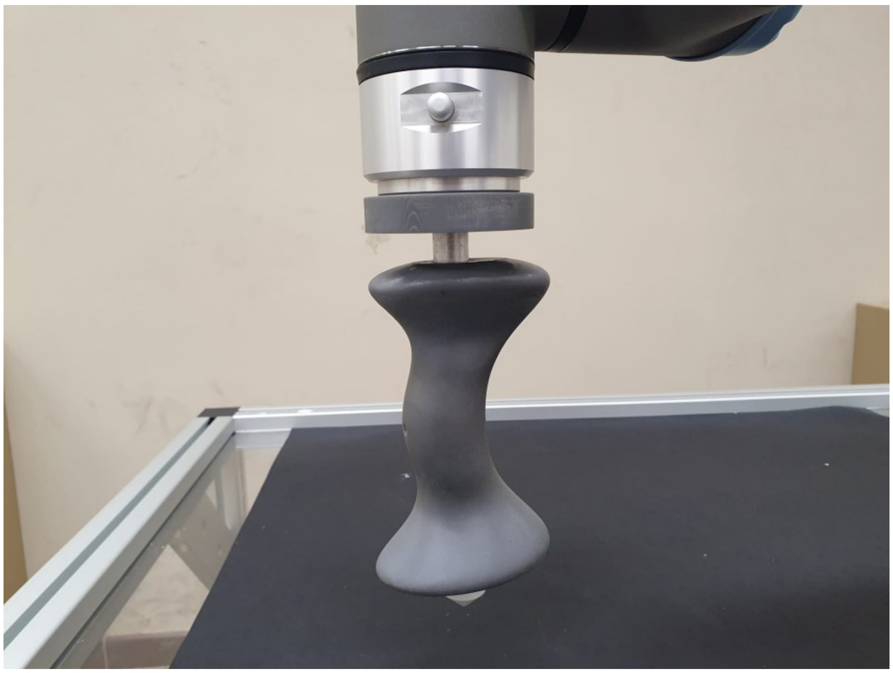


Figure 54. Ergonomic handful to ensure the grip of the patient.

The essential step of the robot guidance is to load the calibration file that is previously created to characterize the working plane for both the vision system and the collaborative robot. Once the calibration has been loaded, it is necessary to define the target of the rehabilitative task, using the function PathMax Pattern that is capable to train to vision system and recognize a defined geometry. In particular, it has been designed a circular crown and a plastic printed tip at the bottom of the robot end-effector that has to reach the centre of the circular crown, as shown in Figure 55. Thus, the point that the robot has to reach has been placed at the centre of the circular crown using the settings related to vertical and horizontal offsets.

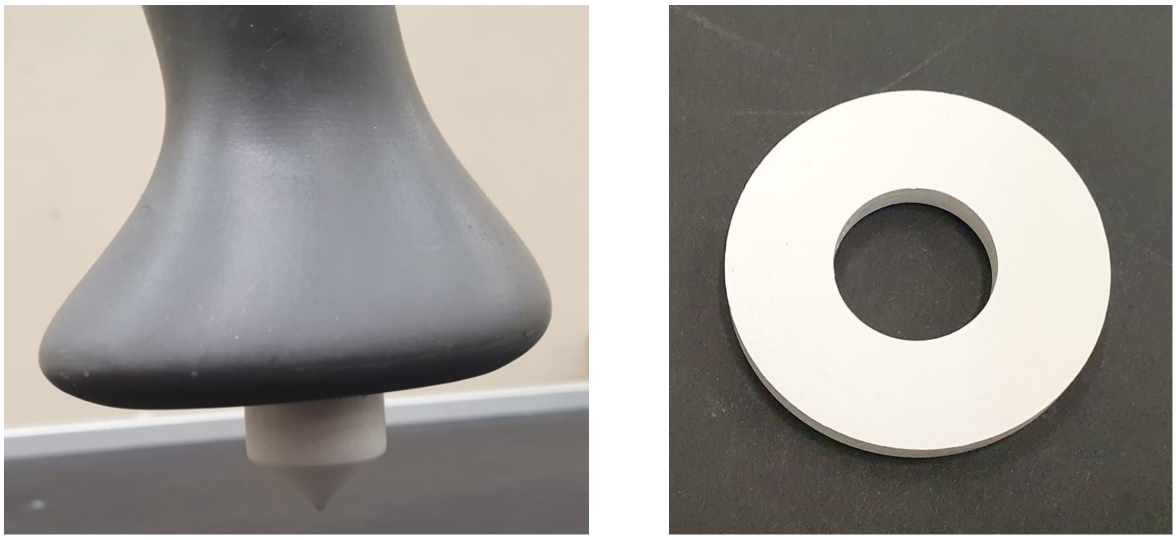


Figure 55. On the left the printed tip. On the right the circular crown.

Within the ‘Inspect Part’ there is the robot guidance under the math and logic tool. The input of this function is to select the pattern that has to be associated as target.

The last step is in the Polyscope interface. In fact, it permits to define the variable pose where inside is contained information of the target with respect to the three-dimensional space. This variable Is related to a trigger of the camera that is capable to acquire the position of the target each time the program is launched.

At the end, it is necessary to build a Polyscope program where there is the output of the camera that correspond to the target position and define a movement of the robotic arm from a waypoint (decided by the user) to the camera output. To complete the robot guidance, it is not necessary to have all the devices connected each other but only the robot controller and the vision system.

4.3 Rehabilitative Task

Once the design of the rehabilitative station has been defined and the correct target for both the calibration and the robot guidance process has been correctly recognize by the vision system, the rehabilitative task has to be defined. The aim of the rehabilitative task is to visualize the target on the working plane using the vision system and starting from a defined position, the robot is capable to guide the human upper limb toward the target and then, moving back in the starting position.

In particular, the first step of the rehabilitative task is to define the initial position of the human upper limb. One of the possible approaches is to choose one of the points belonging to the cloud of points described in paragraph 3.1 in order to quantify the manipulability indexes values. Then, the human-robot interface permits to the patient to reach the target according to the rehabilitative purpose. In fact, it is possible to implement a passive, active or active resistance rehabilitative task. Thus, the first part of the exercise ends with the target reach, in particular, the pointer designed at the end of the robot end-effector has to be centred in the circular target.

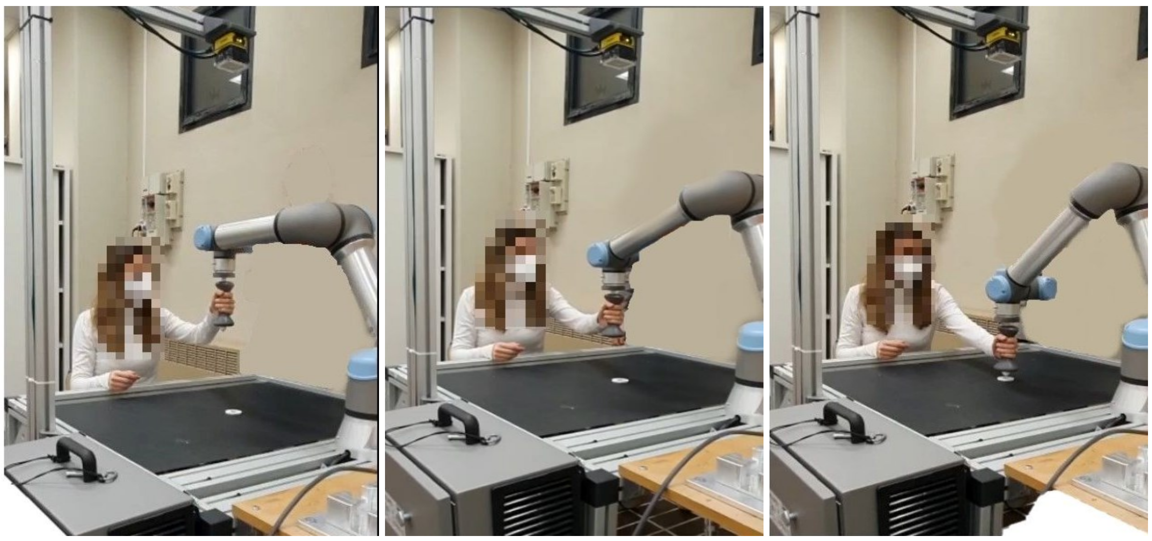


Figure 56. From left to right, the first part of the rehabilitative task. From the starting position to the target reach.

The second part of the rehabilitative task is the movement back of the human upper limb to the starting position. In this way, having a vision system capable to acquire the position of the circular target dynamically, it is possible to change the target position and repeat the task continuously.

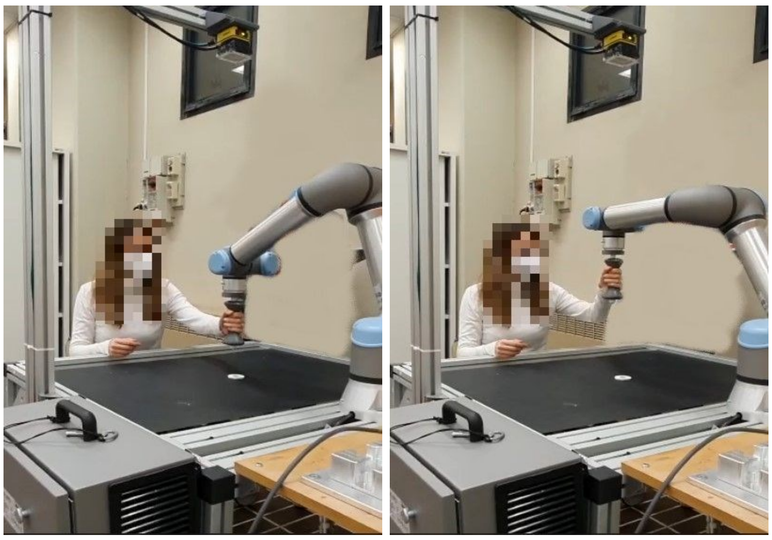


Figure 57. The second part of the rehabilitative task. The return to the initial position.

5. Conclusion

In the last years, coupling the human upper limb with robotic devices is gaining increasing attention. The advantages of more intensiveness, long duration, repeatability and task-orientation, robot-assistant training has become a promising technology in rehabilitation. To design a rehabilitation collaborative station and obtain the proper robot-patient interface usable in clinical environment, it has been necessary firstly to develop a suitable optimization algorithm that set the correct position of the collaborative robot with respect to the patient shoulder and then to plan the proper rehabilitative task.

In this study, the input used to build the optimization algorithm is the joints configuration of the human upper limb that have been obtained solving a proper numerical inverse kinematic method of a height-based cloud of points. This approach permits to obtain a standard solution for all the patients. The core of the optimization algorithm is based on the solution of the inverse kinematic algorithm of the movable collaborative robot to calculate the quality index based on the manipulability ellipsoids (force and velocity). The output of the function is a three-dimensional map of the indexes according to specific frontal and lateral workspace dimensions. Once the global optimization process has been concluded, the local optimization has been performed to obtain unconstrained position of the robotic structure. The results obtained from the optimization core, differently from the definition of the starting cloud of points, are highly dependent on the patient height. It is important to notice that even if the height of the patients is different, generally, the optimal solution for the algorithm is always frontal with respect to the patient, giving rise to a specification for the design of the rehabilitation collaborative station. In fact, according to this result, the collaborative station has been designed to have the robot base in front of the patient. Furthermore, this algorithm results to be a reliable tool in clinical practise to obtain the optimal position of the robot base for all the patients.

The second part of this study has been based on the definition of the task where the patient is guided by the robotic structure toward a movable object. To accomplish this task, in the design of the rehabilitation collaborative station has been mounted a vision system that permits to acquire a movable target. To define the task using an external vision system the hand-eye calibration has been performed as well as, the correct pattern recognition has to be completed to perform the robot guidance. In particular, it was assumed that a non-specific patient was capable of grabbing the robot's handle and able to reach toward the target with maximal effort.

In the realization of human and robot models it has relied on the literature, focusing only on models which have standard measures and known reference frames. The kinematics for the UR5 is accurate, while the human upper limb model can be optimized including the synergy of muscles, tendons and ligaments. One more simplification was adopted considering the body segments as rigid bodies for the purpose of describing the human motion. Even though the two systems are simplified, they allow a good quantitative and qualitative estimation.

The physiological motions have been defined on the basis of possible rehabilitation procedures. Depending on the nature of the functional impairment, the patient is often unable to perform some simple exercises. Therefore, there are several rehabilitation protocols suitable for each pathological condition. Moreover, future studies could involve the analysis of the three different working modalities (active, active-assisted and passive), focusing on the control system of human-robot. The research could make possible the analysis of the impact of rehabilitation on brain plasticity to adapt treatment resources to meet the needs of each patient and optimize the recovery process. Future research should identify what features are essential to the efficacy of robotic manipulator. Integration of the collaborative robots into current rehabilitation practice holds the promise of improving the quality of physical rehabilitation, alleviating its labour-intensive aspects, and increasing the efficiency of therapists.

6. Matlab Code

In this section are reported the Matlab scripts adopted in this project, organized in the following list. The Matlab scripts start using the syntax 'clear all, close all, clc' while, the functions start with its own name.

- Definition of the workspace according to the spherical coordinates
- Inverse kinematic of the human upper limb
- Optimization algorithm solutions (global optimization)
- Processing of the indexes
- Local minimization
- Functions used to build the code

```
clear all, close all, clc

h = 1.70; % [m]
[~, braccio, avambraccio, ~, limb] = arm_dimension(h);
ARM = upper_limb_7Dof(h);

r = 5*limb/6;gamma = deg2rad(20); beta = pi/6; alfa = pi/6; % angles for spherical coordinates
p = workspace(r, alfa, beta, gamma); % circular shell first radius
p2 = workspace(limb/2, alfa, beta, gamma); % circular shell second radius
[R_0, R_alfa, R_beta] = orientation_arm_ik(alfa, beta); % definition of the orientation for
each point

P = [p p2];

save('workspace', 'P')
teta1 = 0; teta2 = 0; teta3 = 0; teta4 = pi/2; teta5 = pi/2; teta6 = 0; teta7 = 0;


q0_arm = [teta1, teta2, teta3, teta4, teta5, teta6, teta7]';


% figure
% plot3(p(1,:), p(2,:), p(3,:), '*')
% grid on
% axis equal
% xlabel('X')
% ylabel('Y')
% zlabel('Z')
% hold on
% ARM.plot(q0_arm', 'floorlevel', - 1.3, 'workspace', [-1 1 -1 1 -1.3 1], 'jointcolor', [1 0
0], 'linkcolor', [1 0 0])
% hold on
% quiver3(p(1,1), p(2,1), p(3,1), R_0(1,1), R_0(2,1), R_0(3,1),'r') %x
% hold on
% quiver3(p(1,1), p(2,1), p(3,1), R_0(1,2), R_0(2,2), R_0(3,2),'g') %y
% hold on
% quiver3(p(1,1), p(2,1), p(3,1), R_0(1,3), R_0(2,3), R_0(3,3),'b') %z
% hold on
% quiver3(p(1,4), p(2,4), p(3,4), R_alfa(1,1), R_alfa(2,1), R_alfa(3,1),'r') %x
% hold on
% quiver3(p(1,4), p(2,4), p(3,4), R_alfa(1,2), R_alfa(2,2), R_alfa(3,2),'g') %y
% hold on
% quiver3(p(1,4), p(2,4), p(3,4), R_alfa(1,3), R_alfa(2,3), R_alfa(3,3),'b') %z
% hold on
% quiver3(p(1,7), p(2,7), p(3,7), R_beta(1,1), R_beta(2,1), R_beta(3,1),'r') %x
% hold on
% quiver3(p(1,7), p(2,7), p(3,7), R_beta(1,2), R_beta(2,2), R_beta(3,2),'g') %y
% hold on
% quiver3(p(1,7), p(2,7), p(3,7), R_beta(1,3), R_beta(2,3), R_beta(3,3),'b') %z
% hold on
% plot3(0, 0, 0, 'r*')
% hold off
```

```
clear all, close all, clc

load('workspace.mat')

teta1 = 0; teta2 = 0; teta3 = 0; teta4 = pi/2; teta5 = pi/2; teta6 = 0; teta7 = 0;
q0_arm = [teta1, teta2, teta3, teta4, teta5, teta6, teta7]';

h = 1.70; % [m]
[~, braccio, avambraccio, ~, limb] = arm_dimension(h);
ARM = upper_limb_7Dof(h);

% orientation definition
beta = pi/6; alfa = pi/6;
[R_0, R_alfa, R_beta] = orientation_arm_ik(alfa, beta);

q_i = [];
p = [];

for i = 1 : max(size(P))

    if i >= 1 && i <=3 || i >= 10 && i <=12
    T_inv = [R_0 P(:,i); 0 0 0 1];
    q = ARM.ikcon(T_inv,q0_arm);
    T = ARM.fkine(q);
    end

    if i >= 4 && i <=6 || i >= 13 && i <=15
    T_inv = [R_alfa P(:,i); 0 0 0 1];
    q = ARM.ikcon(T_inv,q0_arm);
    T = ARM.fkine(q);
    end

    if i >= 7 && i <=9 || i >= 16 && i <=18
    T_inv = [R_beta P(:,i); 0 0 0 1];
    q = ARM.ikcon(T_inv,q0_arm);
    T = ARM.fkine(q);
    end

    q_i = [q_i q'];
    t = T.t;
    p = [p t];

end

q_i = [q_i(:,1:3) q_i(:,10:12) q_i(:,4:6) q_i(:,13:15) q_i(:,7:9) q_i(:,16:18)];

save('q_inv_arm', 'q_i')

figure
plot3(P(1,:),P(2,:),P(3,:), 'b*')
grid on
axis equal
xlabel('X')
ylabel('Y')
zlabel('Z')
hold on
plot3(p(1,:),p(2,:),p(3,:), 'ro')
hold on
ARM.plot(q0_arm')
hold off
```

```
clear all, close all, clc

load('q_inv_arm.mat');
q_inv_arm = q_i;

teta1 = 0; teta2 = 0; teta3 = 0; teta4 = pi/2; teta5 = pi/2; teta6 = 0; teta7 = 0;
q0_arm = [teta1, teta2, teta3, teta4, teta5, teta6, teta7]';

h = 1.70; % [m]
[~, braccio, avambraccio, ~, limb] = arm_dimension(h);
ARM = upper_limb_7Dof(h);
k = 0;


final_v = []; final_f = [];


q0_ur = [0 -pi/2 0 0 -pi/2 0];


for a = -1.5 : 0.1 : 1.5 % coordinata X globale
    for b = 0 : 0.1 : 1.5 % coordinata Y globale

            index_v = [];
            index_f = [];

            T_base_ur = [0 -1 0 a; 1 0 0 b; 0 0 1 -braccio; 0 0 0 1];
            UR5=UR5_definition(T_base_ur);

        for i = 1 : max(size(q_inv_arm))

            %Points that are reached by the arm
            T_dir = ARM.fkine(q_inv_arm(:,i));

            %Inverse Kinematic of the UR5
            R_axes_ur = [-T_dir.o -T_dir.n -T_dir.a];
            T_inv_ur = [R_axes_ur T_dir.t; 0 0 0 1];
            q_inv_ur = UR5.ikunc(T_inv_ur, q0_ur);

            %Calcolo degli ellissoidi
            J_arm = ARM.jacob0(q_inv_arm(:,i));
            T_fk_arm = ARM.fkine(q_inv_arm(:,i));
            [~, ~, ~, ~, ~, ~, sa_v_arm, sa_f_arm, R_v_arm, R_f_arm, cn_v_arm, ~] =
ellissoidi_uomo(J_arm,T_fk_arm.t);

            T_ur = UR5.fkine(q_inv_ur);
            J_ur = UR5.jacob0(q_inv_ur);
            [~, ~, ~, ~, ~, ~, sa_v_ur, sa_f_ur, R_v_ur, R_f_ur, cn_v_ur, ~] =
ellissoidi_UR(J_ur,T_ur.t);

            diff_pose = norm(T_fk_arm.t - T_ur.t);

            if norm(diff_pose) < 0.01

            TT_v_ur = [R_v_ur, T_ur.t; 0 0 0 1]; % da ellissoide di velocità dell'ur
all'assoluto
            TT_f_ur = [R_f_ur, T_ur.t; 0 0 0 1]; % da ellissoide di forza dell'ur all'assoluto

            TT_v_arm = [R_v_arm, T_fk_arm.t; 0 0 0 1]; % da ellissoide di velocità della
spalla all'origine
            TT_f_arm = [R_f_arm, T_fk_arm.t; 0 0 0 1]; % da ellissoide di forza della spalla
all'origine

        %    semiassi del braccio nel loro sistema di rif. (ellissoide velocità mano)
            semiasse1_varm = [0 0 sa_v_arm(3)]';
            semiasse2_varm = [0 sa_v_arm(2) 0]';
            semiasse3_varm = [sa_v_arm(1) 0 0]';
            semiassi_v_arm = [semiasse1_varm, semiasse2_varm, semiasse3_varm];

        %    semiassi del braccio nel loro sistema di rif. (ellissoide forza mano)
            semiasse1_farm = [0 0 sa_f_arm(3)]';
            semiasse2_farm = [0 sa_f_arm(2) 0]';
            semiasse3_farm = [sa_f_arm(1) 0 0]';
            semiassi_f_arm = [semiasse1_farm, semiasse2_farm, semiasse3_farm];
        %    semiassi ellissoide ur nel sistema di riferimento ellissoide mano
            semiassi_v_ur = semiassi_ur(TT_v_ur, TT_v_arm, sa_v_ur);
            semiassi_f_ur = semiassi_ur(TT_f_ur, TT_f_arm, sa_f_ur);

        %    calcolo index per ciascun ellissoide
            index_norm_v = index(semiassi_v_arm, semiassi_v_ur);
            index_norm_f = index(semiassi_f_arm, semiassi_f_ur);

            index_v = [index_v index_norm_v];
            index_f = [index_f index_norm_f];

            else

            index_v = [index_v 0];
            index_f = [index_f 0];

            end

        end

            final_v = [final_v mean(index_v)];
            final_f = [final_f mean(index_f)];

            k = k + 1

    end
end

zbraccio.v = final_v; zbraccio.f = final_f; save('zbraccio', 'zbraccio')


k = 0;


final_v = []; final_f = [];


q0_ur = [0 -pi/2 0 0 -pi/2 0];


for a = -1.5 : 0.1 : 1.5 % coordinata X globale
    for b = 0 : 0.1 : 1.5 % coordinata Y globale

            index_v = [];
            index_f = [];

            T_base_ur = [0 -1 0 a; 1 0 0 b; 0 0 1 0; 0 0 0 1];
            UR5=UR5_definition(T_base_ur);

        for i = 1 : max(size(q_inv_arm))

            %Points that are reached by the arm
            T_dir = ARM.fkine(q_inv_arm(:,i));

            %Inverse Kinematic of the UR5
            R_axes_ur = [-T_dir.o -T_dir.n -T_dir.a];
            T_inv_ur = [R_axes_ur T_dir.t; 0 0 0 1];
            q_inv_ur = UR5.ikunc(T_inv_ur, q0_ur);

            %Calcolo degli ellissoidi
            J_arm = ARM.jacob0(q_inv_arm(:,i));
            T_fk_arm = ARM.fkine(q_inv_arm(:,i));
            [~, ~, ~, ~, ~, ~, sa_v_arm, sa_f_arm, R_v_arm, R_f_arm, cn_v_arm, ~] =
ellissoidi_uomo(J_arm,T_fk_arm.t);

            T_ur = UR5.fkine(q_inv_ur);
            J_ur = UR5.jacob0(q_inv_ur);
            [~, ~, ~, ~, ~, ~, sa_v_ur, sa_f_ur, R_v_ur, R_f_ur, cn_v_ur, ~] =
ellissoidi_UR(J_ur,T_ur.t);

            diff_pose = norm(T_fk_arm.t - T_ur.t);

            if norm(diff_pose) < 0.01
            TT_v_ur = [R_v_ur, T_ur.t; 0 0 0 1]; % da ellissoide di velocità dell'ur
all'assoluto
            TT_f_ur = [R_f_ur, T_ur.t; 0 0 0 1]; % da ellissoide di forza dell'ur all'assoluto

            TT_v_arm = [R_v_arm, T_fk_arm.t; 0 0 0 1]; % da ellissoide di velocità della
spalla all'origine
            TT_f_arm = [R_f_arm, T_fk_arm.t; 0 0 0 1]; % da ellissoide di forza della spalla
all'origine

        %    semiassi del braccio nel loro sistema di rif. (ellissoide velocità mano)
            semiasse1_varm = [0 0 sa_v_arm(3)]';
            semiasse2_varm = [0 sa_v_arm(2) 0]';
            semiasse3_varm = [sa_v_arm(1) 0 0]';
            semiassi_v_arm = [semiasse1_varm, semiasse2_varm, semiasse3_varm];

        %    semiassi del braccio nel loro sistema di rif. (ellissoide forza mano)
            semiasse1_farm = [0 0 sa_f_arm(3)]';
            semiasse2_farm = [0 sa_f_arm(2) 0]';
            semiasse3_farm = [sa_f_arm(1) 0 0]';
            semiassi_f_arm = [semiasse1_farm, semiasse2_farm, semiasse3_farm];

        %    semiassi ellissoide ur nel sistema di riferimento ellissoide mano
            semiassi_v_ur = semiassi_ur(TT_v_ur, TT_v_arm, sa_v_ur);
            semiassi_f_ur = semiassi_ur(TT_f_ur, TT_f_arm, sa_f_ur);

        %    calcolo index per ciascun ellissoide
            index_norm_v = index(semiassi_v_arm, semiassi_v_ur);
            index_norm_f = index(semiassi_f_arm, semiassi_f_ur);

            index_v = [index_v index_norm_v];
            index_f = [index_f index_norm_f];

            else

            index_v = [index_v 0];
            index_f = [index_f 0];

            end

        end

            final_v = [final_v mean(index_v)];
            final_f = [final_f mean(index_f)];

            k = k + 1

    end
end

z0.v = final_v; z0.f = final_f; save('z0', 'z0')
```

```
clear all, close all, clc

load('q_inv_arm.mat');
q_inv_arm = q_inv;

% teta1 = pi/3; teta2 = 0; teta3 = 0; teta4 = pi/6; teta5 = pi/2; teta6 = 0; teta7 = 0;
teta1 = 0; teta2 = 0; teta3 = 0; teta4 = pi/2; teta5 = pi/2; teta6 = 0; teta7 = 0;
q0_arm = [teta1, teta2, teta3, teta4, teta5, teta6, teta7]'; % initial condition for the
inverse kinematic of the upper limb

h = 1.70; % height of the patient [m]
[~, braccio, avambraccio, ~, limb] = arm_dimension(h);
ARM = upper_limb_7Dof(h); % arm definition using robotic toolbox

X = [-0.50    0.50   -braccio]; % check a postion of the robotic arm in global coordinates
index_v = [];


index_f = [];


% Relative position UR5 in global frame
T_base_ur = [0 -1 0 X(1); 1 0 0 X(2); 0 0 1 X(3); 0 0 0 1];

% Robot definition using robotic toolbox
UR5=UR5_definition(T_base_ur);
q0_ur = [0 -pi/2 0 0 -pi/2 0]; %initial configuration of the robot

% figure
% ARM.plot(q0_arm', 'floorlevel', - 1.3, 'workspace', [-1 1 -1 1 -1.3 1], 'jointcolor', [1 0
0], 'linkcolor', [1 0 0])
% hold on
% UR5.plot(q0_ur, 'floorlevel', - 1.3, 'workspace', [-1 1 -1 1 -1.3 1], 'scale', 0.5,
'jointcolor', [0 0 1], 'linkcolor', [0.5 0.5 0.5])
% hold off

for i = 1 : max(size(q_inv_arm))

    % Points that are reached by the arm
    T_dir = ARM.fkine(q_inv_arm(:,i));

    %Inverse Kinematic of the UR5
    R_axes_ur = [-T_dir.o -T_dir.n -T_dir.a]; % end-effector frame of the robotic arm
    T_inv_ur = [R_axes_ur T_dir.t; 0 0 0 1]; % input inverse kinematic robotic arm
    q_inv_ur = UR5.ikunc(T_inv_ur, q0_ur); % robotic arm inverse kinematic

    %Calcolo degli ellissoidi
    J_arm = ARM.jacob0(q_inv_arm(:,i));
    T_fk_arm = ARM.fkine(q_inv_arm(:,i));
    [X_v0_arm, Y_v0_arm, Z_v0_arm, X_f0_arm, Y_f0_arm, Z_f0_arm, sa_v_arm, sa_f_arm, R_v_arm,
R_f_arm, cn_v_arm, cn_f_arm] = ellissoidi_uomo(J_arm,T_fk_arm.t);

    T_ur = UR5.fkine(q_inv_ur);
    J_ur = UR5.jacob0(q_inv_ur);
    [X_v0_ur, Y_v0_ur, Z_v0_ur, X_f0_ur, Y_f0_ur, Z_f0_ur, sa_v_ur, sa_f_ur, R_v_ur, R_f_ur,
cn_v_ur, cn_f_ur] = ellissoidi_UR(J_ur,T_ur.t);

    diff_pose = norm(T_fk_arm.t - T_ur.t);

    if norm(diff_pose) < 0.01

%     figure
%     UR5.plot(q_inv_ur, 'floorlevel', - 1.3, 'workspace', [-1 1 -1 1 -1.3 1], 'scale', 0.5,
'jointcolor', [0 0 1], 'linkcolor', [0.5 0.5 0.5])
%     hold on
%     surf(X_f0_ur, Y_f0_ur, Z_f0_ur,'EdgeColor','[0.5 0.5 0.5]','FaceColor','[1 1
1]','FaceLighting','gouraud')
%     hold on
%     surf(X_v0_ur, Y_v0_ur, Z_v0_ur,'EdgeColor','[0.5 0.5 0.5]','FaceColor','[1 1
0]','FaceLighting','gouraud')
%     hold off
%
%     figure
%     ARM.plot(q_inv_arm(:,i)', 'floorlevel', - 1.3, 'workspace', [-1 1 -1 1 -1.3 1],
'jointcolor', [1 0 0], 'linkcolor', [1 0 0])
%     hold on
%     surf(X_f0_arm, Y_f0_arm, Z_f0_arm,'EdgeColor','[0.5 0.5 0.5]','FaceColor','[1 1
1]','FaceLighting','gouraud')
%     hold on
%     surf(X_v0_arm, Y_v0_arm, Z_v0_arm,'EdgeColor','[0.5 0.5 0.5]','FaceColor','[1 1
0]','FaceLighting','gouraud')
%     hold off

    TT_v_ur = [R_v_ur, T_ur.t; 0 0 0 1]; % da ellissoide di velocità dell'ur all'assoluto
    TT_f_ur = [R_f_ur, T_ur.t; 0 0 0 1]; % da ellissoide di forza dell'ur all'assoluto

    TT_v_arm = [R_v_arm, T_fk_arm.t; 0 0 0 1]; % da ellissoide di velocità della spalla
all'origine
    TT_f_arm = [R_f_arm, T_fk_arm.t; 0 0 0 1]; % da ellissoide di forza della spalla
all'origine

%    semiassi del braccio nel loro sistema di rif.
    semiasse1_varm = [0 0 sa_v_arm(3)]';
    semiasse2_varm = [0 sa_v_arm(2) 0]';
    semiasse3_varm = [sa_v_arm(1) 0 0]';
    semiassi_v_arm = [semiasse1_varm, semiasse2_varm, semiasse3_varm];

%    semiassi del braccio nel loro sistema di rif.
    semiasse1_farm = [0 0 sa_f_arm(3)]';
    semiasse2_farm = [0 sa_f_arm(2) 0]';
    semiasse3_farm = [sa_f_arm(1) 0 0]';
    semiassi_f_arm = [semiasse1_farm, semiasse2_farm, semiasse3_farm];

%    semiassi ellissoide ur nel sistema di riferimento ellissoide mano
    semiassi_v_ur = semiassi_ur(TT_v_ur, TT_v_arm, sa_v_ur);
    semiassi_f_ur = semiassi_ur(TT_f_ur, TT_f_arm, sa_f_ur);

%    calcolo index per ciascun ellissoide
    index_norm_v = index(semiassi_v_arm, semiassi_v_ur);
    index_norm_f = index(semiassi_f_arm, semiassi_f_ur);

    index_v = [index_v index_norm_v];
    index_f = [index_f index_norm_f];

    av_arm = TT_v_arm * [0 0 sa_v_arm(3,1) 1]'; %questi sono solo per il plot
    bv_arm = TT_v_arm * [0 sa_v_arm(2,1) 0 1]';
    cv_arm = TT_v_arm * [sa_v_arm(1,1) 0 0 1]';

    av_ur = TT_v_ur * [0 0 sa_v_ur(3,1) 1]'; %questi sono solo per il plot
    bv_ur = TT_v_ur * [0 sa_v_ur(2,1) 0 1]';
    cv_ur = TT_v_ur * [sa_v_ur(1,1) 0 0 1]';

    af_arm = TT_f_arm * [0 0 sa_f_arm(3,1) 1]'; %questi sono solo per il plot
    bf_arm = TT_f_arm * [0 sa_f_arm(2,1) 0 1]';
    cf_arm = TT_f_arm * [sa_f_arm(1,1) 0 0 1]';

    af_ur = TT_f_ur * [0 0 sa_f_ur(3,1) 1]'; %questi sono solo per il plot
    bf_ur = TT_f_ur * [0 sa_f_ur(2,1) 0 1]';
    cf_ur = TT_f_ur * [sa_f_ur(1,1) 0 0 1]';

%     Check Ellipsoids
%     figure
%     xlabel('X [m]')
%     ylabel('Y [m]')
%     zlabel('Z [m]')
%     xlim([-1 1])
%     ylim([-1 1])
%     zlim([-1.3 1])
%     surf(X_v0_arm, Y_v0_arm, Z_v0_arm,'EdgeColor','[0.5 0.5 0.5]','FaceColor','[0 1
0]','FaceLighting','gouraud')
%     hold on
%     plot3(av_arm(1,1), av_arm(2,1), av_arm(3,1), 'bo')
%     hold on
%     plot3(bv_arm(1,1), bv_arm(2,1), bv_arm(3,1), 'go')
%     hold on
%     plot3(cv_arm(1,1), cv_arm(2,1), cv_arm(3,1), 'ro')
%     hold on
%     plot3(T_fk_arm.t(1,1), T_fk_arm.t(2,1), T_fk_arm.t(3,1), '*')
%     hold on
%     quiver3(T_fk_arm.t(1,1), T_fk_arm.t(2,1), T_fk_arm.t(3,1), R_v_arm(1,1), R_v_arm(2,1), 
R_v_arm(3,1),'r')
%     hold on
%     quiver3(T_fk_arm.t(1,1), T_fk_arm.t(2,1), T_fk_arm.t(3,1), R_v_arm(1,2), R_v_arm(2,2), 
R_v_arm(3,2),'g')
%     hold on
%     quiver3(T_fk_arm.t(1,1), T_fk_arm.t(2,1), T_fk_arm.t(3,1), R_v_arm(1,3), R_v_arm(2,3), 
R_v_arm(3,3),'b')
%     hold on
%     surf(X_v0_ur, Y_v0_ur, Z_v0_ur,'EdgeColor','[0.5 0.5 0.5]','FaceColor','[1 1 
0]','FaceLighting','gouraud')
%     hold on
%     plot3(av_ur(1,1), av_ur(2,1), av_ur(3,1), 'bo')
%     hold on
%     plot3(bv_ur(1,1), bv_ur(2,1), bv_ur(3,1), 'go')
%     hold on
%     plot3(cv_ur(1,1), cv_ur(2,1), cv_ur(3,1), 'ro')
%     hold on
%     quiver3(T_ur.t(1,1), T_ur.t(2,1), T_ur.t(3,1), R_v_ur(1,1), R_v_ur(2,1), 
R_v_ur(3,1),'r')
%     hold on
%     quiver3(T_ur.t(1,1), T_ur.t(2,1), T_ur.t(3,1), R_v_ur(1,2), R_v_ur(2,2), 
R_v_ur(3,2),'g')
%     hold on
%     quiver3(T_ur.t(1,1), T_ur.t(2,1), T_ur.t(3,1), R_v_ur(1,3), R_v_ur(2,3), 
R_v_ur(3,3),'b')
%     hold on
%     UR5.plot(q_inv_ur, 'floorlevel', - 1.3, 'workspace', [-1 1 -1 1 -1.3 1], 'scale', 0.5, 
'jointcolor', [0 0 1], 'linkcolor', [0.5 0.5 0.5])
%     hold on
%     ARM.plot(q_inv_arm(:,i)', 'floorlevel', - 1.3, 'workspace', [-1 1 -1 1 -1.3 1], 
'jointcolor', [1 0 0], 'linkcolor', [1 0 0])
%     hold off

%     figure
%     xlabel('X [m]')
%     ylabel('Y [m]')
%     zlabel('Z [m]')
%     xlim([-1 1])
%     ylim([-1 1])
%     zlim([-1.3 1])
%     surf(X_f0_arm, Y_f0_arm, Z_f0_arm,'EdgeColor','[0.5 0.5 0.5]','FaceColor','[0 1 
0]','FaceLighting','gouraud')
%     hold on
%     plot3(af_arm(1,1), af_arm(2,1), af_arm(3,1), 'bo')
%     hold on
%     plot3(bf_arm(1,1), bf_arm(2,1), bf_arm(3,1), 'go')
%     hold on
%     plot3(cf_arm(1,1), cf_arm(2,1), cf_arm(3,1), 'ro')
%     hold on
%     plot3(T_fk_arm.t(1,1), T_fk_arm.t(2,1), T_fk_arm.t(3,1), '*')
%     hold on
%     quiver3(T_fk_arm.t(1,1), T_fk_arm.t(2,1), T_fk_arm.t(3,1), R_f_arm(1,1), R_f_arm(2,1), 
R_f_arm(3,1),'r')
%     hold on
%     quiver3(T_fk_arm.t(1,1), T_fk_arm.t(2,1), T_fk_arm.t(3,1), R_f_arm(1,2), R_f_arm(2,2), 
R_f_arm(3,2),'g')
%     hold on
%     quiver3(T_fk_arm.t(1,1), T_fk_arm.t(2,1), T_fk_arm.t(3,1), R_f_arm(1,3), R_f_arm(2,3), 
R_f_arm(3,3),'b')
%     hold on
%     surf(X_f0_ur, Y_f0_ur, Z_f0_ur,'EdgeColor','[0.5 0.5 0.5]','FaceColor','[1 1 
0]','FaceLighting','gouraud')
%     hold on
%     plot3(af_ur(1,1), af_ur(2,1), af_ur(3,1), 'bo')
%     hold on
%     plot3(bf_ur(1,1), bf_ur(2,1), bf_ur(3,1), 'go')
%     hold on
%     plot3(cf_ur(1,1), cf_ur(2,1), cf_ur(3,1), 'ro')
%     hold on
%     quiver3(T_ur.t(1,1), T_ur.t(2,1), T_ur.t(3,1), R_f_ur(1,1), R_f_ur(2,1), 
R_f_ur(3,1),'r')
```

```
%     hold on
%     quiver3(T_ur.t(1,1), T_ur.t(2,1), T_ur.t(3,1), R_f_ur(1,2), R_f_ur(2,2),
R_f_ur(3,2),'g')
%     hold on
%     quiver3(T_ur.t(1,1), T_ur.t(2,1), T_ur.t(3,1), R_f_ur(1,3), R_f_ur(2,3),
R_f_ur(3,3),'b')
%     hold on
%     UR5.plot(q_inv_ur, 'floorlevel', - 1.3, 'workspace', [-1 1 -1 1 -1.3 1], 'scale', 0.5,
'jointcolor', [0 0 1], 'linkcolor', [0.5 0.5 0.5])
%     hold on
%     ARM.plot(q_inv_arm(:,i)', 'floorlevel', - 1.3, 'workspace', [-1 1 -1 1 -1.3 1],
'jointcolor', [1 0 0], 'linkcolor', [1 0 0])
%     hold off

%     Check inverse kinematic
%     figure
%     xlabel('X [m]')
%     ylabel('Y [m]')
%     zlabel('Z [m]')
%     UR5.plot(q_inv_ur, 'floorlevel', - 1.3, 'workspace', [-1 1 -1 1 -1.3 1], 'scale', 0.5,
'jointcolor', [0 0 1], 'linkcolor', [0.5 0.5 0.5])
%     hold on
%     ARM.plot(q_inv_arm(:,i)', 'floorlevel', - 1.3, 'workspace', [-1 1 -1 1 -1.3 1],
'jointcolor', [1 0 0], 'linkcolor', [1 0 0])
%     hold off

    else

    index_v = [index_v 0];
    index_f = [index_f 0];

%     figure
%     UR5.plot(q_inv_ur, 'floorlevel', - 1.3, 'workspace', [-1 1 -1 1 -1.3 1], 'scale', 0.5,
'jointcolor', [0 0 1], 'linkcolor', [0.5 0.5 0.5])
%     hold on
%     ARM.plot(q_inv_arm(:,i)', 'floorlevel', - 1.3, 'workspace', [-1 1 -1 1 -1.3 1],
'jointcolor', [1 0 0], 'linkcolor', [1 0 0])
%     hold off

    end

    v = mean(index_v);
    f = mean(index_f);

end
```

```
clear all, close all, clc

load('z0.mat')
load('zbraccio.mat')
pos = [];

for a = -1.5 : 0.1 : 1.5 % coordinata X
    for b = 0 : 0.1 : 1.5 % coordinata Y
        x = [a b]';
        pos = [pos x];
    end
end

X = [z0.v;pos]; Y = [z0.f;pos];
K = [zbraccio.v;pos]; Z = [zbraccio.f;pos];

for i = 1 : max(size(X))
    if X(2,i) < 0.4
        if X(2,i) > -0.7
            if X(3,i) < 0.4
%                 X(1,i) = NaN;
                X(1,i) = 0;
            end
        end
    end
end

for i = 1 : max(size(Y))
    if Y(2,i) < 0.4
        if Y(2,i) > -0.7
            if Y(3,i) < 0.4
%                 Y(1,i) = NaN;
                Y(1,i) = 0;
            end
        end
    end
end

for i = 1 : max(size(K))
    if K(2,i) < 0.4
        if K(2,i) > -0.7
            if K(3,i) < 0.4
%                 K(1,i) = NaN;
                K(1,i) = 0;
            end
        end
    end
end

for i = 1 : max(size(Z))
    if Z(2,i) < 0.4
        if Z(2,i) > -0.7
            if Z(3,i) < 0.4
%                 Z(1,i) = NaN;
                Z(1,i) = 0;
            end
        end
    end
end

z0.v = X(1,:); z0.f = Y(1,:);
zbraccio.v = K(1,:); zbraccio.f = Z(1,:);

[pv_z0, ma_z0, pf_z0, mb_z0] = index_max(X(1,:), Y(1,:), pos);
plot_index(z0, pos, pv_z0, ma_z0, pf_z0, mb_z0)
[pv_zb, ma_zb, pf_zb, mb_zb] = index_max(K(1,:), Z(1,:), pos);
plot_index(zbraccio, pos, pv_zb, ma_zb, pf_zb, mb_zb)
```

```
clear all, close all, clc
h = 1.70;
[~, braccio, ~, ~, limb] = arm_dimension(h);

x0 = [0 1.1 -0.2737];

options = optimset('OutputFcn', @outfun, 'PlotFcns',@optimplotfval);
x = fminsearch(@MainTask,x0,options);

function f = MainTask(X)
load('q_inv_arm.mat');
q_inv_arm = q_inv;

h = 1.70; % [m]
ARM = upper_limb_7Dof(h);

index_v = []; index_f = [];


% Posizione relativa tra UR5 e uomo
T_base_ur = [0 -1 0 X(1); 1 0 0 X(2); 0 0 1 X(3); 0 0 0 1];

% Defizione del robot con il toolbox
UR5=UR5_definition(T_base_ur);
q0_ur = [0 -pi/2 0 0 -pi/2 0]; %steso alto

for i = 1 : max(size(q_inv_arm))

    T_dir = ARM.fkine(q_inv_arm(:,i));

    %Inverse Kinematic of the UR5
    R_axes_ur = [-T_dir.o -T_dir.n -T_dir.a];
    T_inv_ur = [R_axes_ur T_dir.t; 0 0 0 1]; %input per inverse kinematic
    q_inv_ur = UR5.ikunc(T_inv_ur, q0_ur); %cinematica inversa

    %Calcolo degli ellissoidi
    J_arm = ARM.jacob0(q_inv_arm(:,i));
    T_fk_arm = ARM.fkine(q_inv_arm(:,i));
    [~, ~, ~, ~, ~, ~, sa_v_arm, sa_f_arm, R_v_arm, R_f_arm, ~, ~] =
ellissoidi_uomo(J_arm,T_fk_arm.t);

    T_ur = UR5.fkine(q_inv_ur);
    J_ur = UR5.jacob0(q_inv_ur);
    [~, ~, ~, ~, ~, ~, sa_v_ur, sa_f_ur, R_v_ur, R_f_ur, ~, ~] = ellissoidi_UR(J_ur,T_ur.t);

    diff_pose = norm(T_fk_arm.t - T_ur.t);

    if norm(diff_pose) < 0.01

    TT_v_ur = [R_v_ur, T_ur.t; 0 0 0 1]; % da ellissoide di velocità dell'ur all'assoluto
    TT_f_ur = [R_f_ur, T_ur.t; 0 0 0 1]; % da ellissoide di forza dell'ur all'assoluto

    TT_v_arm = [R_v_arm, T_fk_arm.t; 0 0 0 1]; % da ellissoide di velocità della spalla
all'origine
    TT_f_arm = [R_f_arm, T_fk_arm.t; 0 0 0 1]; % da ellissoide di forza della spalla
all'origine

%    semiassi del braccio nel loro sistema di rif. (ellissoide velocità mano)
    semiasse1_varm = [0 0 sa_v_arm(3)]';
    semiasse2_varm = [0 sa_v_arm(2) 0]';
    semiasse3_varm = [sa_v_arm(1) 0 0]';
    semiassi_v_arm = [semiasse1_varm, semiasse2_varm, semiasse3_varm];

%    semiassi del braccio nel loro sistema di rif. (ellissoide forza mano)
    semiasse1_farm = [0 0 sa_f_arm(3)]';
    semiasse2_farm = [0 sa_f_arm(2) 0]';
    semiasse3_farm = [sa_f_arm(1) 0 0]';
    semiassi_f_arm = [semiasse1_farm, semiasse2_farm, semiasse3_farm];

%    semiassi ellissoide ur nel sistema di riferimento ellissoide mano
    semiassi_v_ur = semiassi_ur(TT_v_ur, TT_v_arm, sa_v_ur);
    semiassi_f_ur = semiassi_ur(TT_f_ur, TT_f_arm, sa_f_ur);
```

```
%    calcolo index per ciascun ellissoide
    index_norm_v = index2(semiassi_v_arm, semiassi_v_ur);
    index_norm_f = index2(semiassi_f_arm, semiassi_f_ur);

    index_v = [index_v index_norm_v];
    index_f = [index_f index_norm_f];

    else

    index_v = [index_v 0];
    index_f = [index_f 0];

    end

    f = 1/mean(index_v);

end
end

function stop = outfun( x,~,~ )
 stop=false;
 x
 end
```

```matlab
function [busto, braccio, avambraccio, baricentromano, limb] = arm_dimension (altezza)

braccio = altezza * 0.161;
avambraccio = altezza * 0.146;
mano = altezza * 0.106;
baricentromano = mano * 0.506;
busto = altezza*0.100;

limb = braccio + avambraccio + mano;

end
```

```matlab
function index_norm = index(semiasse_arm, semiasse_ur)

index_1 = abs(dot(semiasse_arm(1:3,1), semiasse_ur(1:3,1)));
index_2 = abs(dot(semiasse_arm(1:3,2), semiasse_ur(1:3,2)));
index_3 = abs(dot(semiasse_arm(1:3,3), semiasse_ur(1:3,3)));

a = norm(semiasse_arm(1:3,1))*norm(semiasse_ur(1:3,1));
b = norm(semiasse_arm(1:3,2))*norm(semiasse_ur(1:3,2));
c = norm(semiasse_arm(1:3,3))*norm(semiasse_ur(1:3,3));

index_norm = (index_1 + index_2 + index_3) / (a + b + c);

end
```

```matlab
function semiassi_ur = semiassi_ur(TT_ur, TT_arm, sa_ur)

% dal sistema ellissoide dell'ur al sistema fisso
semiasse_1_ur_ass = TT_ur * [0 0 sa_ur(3,1) 1]'; % ordine decrescente
semiasse_2_ur_ass = TT_ur * [0 sa_ur(2,1) 0 1]';
semiasse_3_ur_ass = TT_ur * [sa_ur(1,1) 0 0 1]';

% dal sistema fisso al sistema ellissoide del braccio
semiasse1_ass_arm = TT_arm \ semiasse_1_ur_ass;
semiasse2_ass_arm = TT_arm \ semiasse_2_ur_ass;
semiasse3_ass_arm = TT_arm \ semiasse_3_ur_ass;

semiassi_ur = [semiasse1_ass_arm(1:3,1), semiasse2_ass_arm(1:3,1), semiasse3_ass_arm(1:3,1)];

end
```

```matlab
function [R_0, R_alfa, R_beta] = orientation_arm_ik(alfa, beta)

%il pollice segue sempre l'asse Z
n_0 = [0 1 0]';
o_0 = [-1 0 0]';
a_0 = [0 0 1]';

%orientamento per ik dell'upper limb
R_0 = [n_0 o_0 a_0];
R_alfa = R_0 * [cos(-alfa) -sin(-alfa) 0; sin(-alfa) cos(-alfa) 0; 0 0 1];
R_beta = R_0 * [cos(beta) -sin(beta) 0; sin(beta) cos(beta) 0; 0 0 1];

end
```

```
function [X_v0, Y_v0, Z_v0, X_f0, Y_f0, Z_f0, sa_v, sa_f, R_v, R_f, cn_v, cn_f] =
ellissoidi_uomo(J,posizione)
% Ellissoide Velocità
J_p = J(1:3,:);
[V_v,lambda_v]= eig(inv(J_p*J_p'));

%autovalori
lambda1_v = lambda_v(1,1);
lambda2_v = lambda_v(2,2);
lambda3_v = lambda_v(3,3);

%autovettori associati agli autovalori modulo unitario
v1_v = V_v(:,1);
v2_v = V_v(:,2);
v3_v = V_v(:,3);

%matrice di rotazione da ellissoide a sistema fisso
R_v=[v1_v v2_v v3_v];

%semiassi dell'ellissioide in ordine crescente
sa1_v = sqrt(lambda1_v);
sa2_v = sqrt(lambda2_v);
sa3_v = sqrt(lambda3_v);

%misura della manipolabilità
w = sqrt(det(J_p*J_p'));

%ottengo semiassi scalati di ampizza massima = 1
fattore_scala_v = max ([sa1_v,sa2_v,sa3_v]);
sa1_v = sa1_v/fattore_scala_v;
sa2_v = sa2_v/fattore_scala_v;
sa3_v = sa3_v/fattore_scala_v;

cn_v = min([sa1_v,sa2_v,sa3_v])/max([sa1_v,sa2_v,sa3_v]);

% Plot
n = 25;
[X_v, Y_v, Z_v] = ellipsoid(0,0,0,sa1_v/5,sa2_v/5,sa3_v/5,n);
X_v0 = zeros(26,26);
Y_v0 = zeros(26,26);
Z_v0 = zeros(26,26);
for i=1:size(X_v,1)
    for j=1:size(X_v,2)
        punti = [X_v(i,j) Y_v(i,j) Z_v(i,j)];
        punti_ruotati = R_v*punti';
        X_v0(i,j) = punti_ruotati(1,1)+posizione(1);
        Y_v0(i,j) = punti_ruotati(2,1)+posizione(2);
        Z_v0(i,j) = punti_ruotati(3,1)+posizione(3);
    end
end

% Ellissoide Forza
M = J_p*J_p';
[V_f,lambda_f]= eig(M);

%autovalori
lambda1_f = lambda_f(1,1);
lambda2_f = lambda_f(2,2);
lambda3_f = lambda_f(3,3);

%autovettori associati agli autovalori modulo unitario
v1_f = V_f(:,1);
v2_f = V_f(:,2);
v3_f = V_f(:,3);

%matrice di rotazione da ellissoide a sistema fisso
R_f=[v1_f v2_f v3_f];

%semiassi dell'ellissioide in ordine crescente
sa1_f = sqrt(lambda1_f);
sa2_f = sqrt(lambda2_f);
sa3_f = sqrt(lambda3_f);
```

```
fattore_scala_f = max ([sa1_f,sa2_f,sa3_f]);
sa1_f = sa1_f/fattore_scala_f;
sa2_f = sa2_f/fattore_scala_f;
sa3_f = sa3_f/fattore_scala_f;

cn_f = min([sa1_f,sa2_f,sa3_f])/max([sa1_f,sa2_f,sa3_f]);

n = 25;


[X_f, Y_f, Z_f] = ellipsoid(0,0,0,sa1_f/5,sa2_f/5,sa3_f/5,n);


X_f0 = zeros(26,26); Y_f0 = zeros(26,26); Z_f0 = zeros(26,26);


for i=1:size(X_f,1)
    for j=1:size(X_f,2)
        punti = [X_f(i,j) Y_f(i,j) Z_f(i,j)];
        punti_ruotati = R_f*punti';
        X_f0(i,j) = punti_ruotati(1,1)+posizione(1);
        Y_f0(i,j) = punti_ruotati(2,1)+posizione(2);
        Z_f0(i,j) = punti_ruotati(3,1)+posizione(3);
    end
end

% Output
sa_v = [sa1_v; sa2_v; sa3_v];
sa_f = [sa1_f; sa2_f; sa3_f];
end
```

```
function [X_v0, Y_v0, Z_v0, X_f0, Y_f0, Z_f0, sa_v, sa_f, R_v, R_f, cn_v, cn_f] =
ellissoidi_UR(J,posizione)
% Ellissoide Velocità
J_p = J(1:3,:);
[V_v,lambda_v]= eig(inv(J_p*J_p'));

%autovalori
lambda1_v = lambda_v(1,1);
lambda2_v = lambda_v(2,2);
lambda3_v = lambda_v(3,3);

%autovettori associati agli autovalori modulo unitario
v1_v = V_v(:,1);
v2_v = V_v(:,2);
v3_v = V_v(:,3);

%matrice di rotazione da ellissoide a sistema fisso
R_v=[v1_v v2_v v3_v];

%semiassi dell'ellissioide in ordine crescente
sa1_v = sqrt(lambda1_v);
sa2_v = sqrt(lambda2_v);
sa3_v = sqrt(lambda3_v);

%misura della manipolabilità
w = sqrt(det(J_p*J_p'));

%ottengo semiassi scalati di ampizza massima = 1
fattore_scala_v = max ([sa1_v,sa2_v,sa3_v]);
sa1_v = sa1_v/fattore_scala_v;
sa2_v = sa2_v/fattore_scala_v;
sa3_v = sa3_v/fattore_scala_v;

cn_v = min([sa1_v,sa2_v,sa3_v])/max([sa1_v,sa2_v,sa3_v]);

% Plot
n = 25;
[X_v, Y_v, Z_v] = ellipsoid(0,0,0,sa1_v/5,sa2_v/5,sa3_v/5,n);
```

```
X_v0 = zeros(26,26);
Y_v0 = zeros(26,26);
Z_v0 = zeros(26,26);
for i=1:size(X_v,1)
    for j=1:size(X_v,2)
        punti = [X_v(i,j) Y_v(i,j) Z_v(i,j)];
        punti_ruotati = R_v*punti';
        X_v0(i,j) = punti_ruotati(1,1)+posizione(1);
        Y_v0(i,j) = punti_ruotati(2,1)+posizione(2);
        Z_v0(i,j) = punti_ruotati(3,1)+posizione(3);

    end
end

% Ellissoide Forza
M = J_p*J_p';
[V_f,lambda_f]= eig(M);

%autovalori
lambda1_f = lambda_f(1,1);
lambda2_f = lambda_f(2,2);
lambda3_f = lambda_f(3,3);

%autovettori associati agli autovalori modulo unitario
v1_f = V_f(:,1);
v2_f = V_f(:,2);
v3_f = V_f(:,3);

%matrice di rotazione da ellissoide a sistema fisso
R_f=[v1_f v2_f v3_f];

%semiassi dell'ellissioide in ordine crescente
sa1_f = sqrt(lambda1_f);
sa2_f = sqrt(lambda2_f);
sa3_f = sqrt(lambda3_f);

fattore_scala_f = max ([sa1_f,sa2_f,sa3_f]);
sa1_f = sa1_f/fattore_scala_f;
sa2_f = sa2_f/fattore_scala_f;
sa3_f = sa3_f/fattore_scala_f;

cn_f = min([sa1_f,sa2_f,sa3_f])/max([sa1_f,sa2_f,sa3_f]);
n = 25;


[X_f, Y_f, Z_f] = ellipsoid(0,0,0,sa1_f/5,sa2_f/5,sa3_f/5,n);


X_f0 = zeros(26,26); Y_f0 = zeros(26,26); Z_f0 = zeros(26,26);


for i=1:size(X_f,1)
    for j=1:size(X_f,2)
        punti = [X_f(i,j) Y_f(i,j) Z_f(i,j)];
        punti_ruotati = R_f*punti';
        X_f0(i,j) = punti_ruotati(1,1)+posizione(1);
        Y_f0(i,j) = punti_ruotati(2,1)+posizione(2);
        Z_f0(i,j) = punti_ruotati(3,1)+posizione(3);

    end
end

% Output
sa_v = [sa1_v; sa2_v; sa3_v];
sa_f = [sa1_f; sa2_f; sa3_f];
end
```

```
function plot_index(struc, pos, pv, ma, pf, mb)

x = pos(1,:); y = pos(2,:);
xv = linspace(min(x), max(x), length(x));
yv = linspace(min(y), max(y), length(y));

z_v = struc.v;
z_f = struc.f;
[X,Y] = meshgrid(xv, yv);
Z_v = griddata(x,y,z_v,X,Y);
Z_f = griddata(x,y,z_f,X,Y);

% teta1 = 0; teta2 = 0; teta3 = 0; teta4 = pi/2; teta5 = pi/2; teta6 = 0; teta7 = 0;
% q0_arm = [teta1, teta2, teta3, teta4, teta5, teta6, teta7]';
% h = 1.70;
% ARM = upper_limb_7Dof(h);

% figure
% subplot(1,2,1);
% stem3(x,y,z_v)
% hold on
% surf(X, Y, Z_v);
% grid on
% set(gca, 'ZLim',[0 1])
% shading interp
% title('Velocity Index')
% xlabel('X')
% ylabel('Y')
% zlabel('Z')
% hold off
% subplot(1,2,2);
% stem3(x,y,z_f)
% hold on
% surf(X, Y, Z_f);
% grid on
% set(gca, 'ZLim',[0 1])
% shading interp
% title('Force Index')
% xlabel('X')
% ylabel('Y')
% zlabel('Z')
% hold off

figure
surf(X, Y, Z_v);
grid on
set(gca, 'ZLim',[0 1])
shading interp
grid on
xlabel('X (m)')
ylabel('Y (m)')
zlabel('Z (m)')
hold on
plot3(pv(1), pv(2), ma, 'b*')
hold off

figure
surf(X, Y, Z_f);
grid on
set(gca, 'ZLim',[0 1])
shading interp
grid on
xlabel('X (m)')
ylabel('Y (m)')
zlabel('Z (m)')
hold on
plot3(pf(1), pf(2), mb, 'b*')
hold off

end
```

```
function ARM=upper_limb_7Dof(altezza)

a1 = 0; a2 = 0; a3 = 0; a4 = 0; a5 = 0; a6 =  0; a7 = -altezza*0.106*0.506; %a7 = 0;
d1 = 0; d2 = 0; d3 = -altezza*0.161; d4 = 0; d5 = -altezza*0.146; d6 = 0; d7 = 0; %*0.506
alfa1 = pi/2; alfa2 = pi/2; alfa3 = pi/2; alfa4 = -pi/2; alfa5 = pi/2; alfa6 = pi/2;  alfa7 =
-pi/2;

L1 = Revolute('d', d1, 'a', a1, 'alpha', alfa1, 'offset', pi/2);
L2 = Revolute('d', d2, 'a', a2, 'alpha', alfa2, 'offset', pi/2);
L3 = Revolute('d', d3, 'a', a3, 'alpha', alfa3, 'offset', pi/2);
L4 = Revolute('d', d4, 'a', a4, 'alpha', alfa4);
L5 = Revolute('d', d5, 'a', a5, 'alpha', alfa5);
L6 = Revolute('d', d6, 'a', a6, 'alpha', alfa6, 'offset', pi/2);
L7 = Revolute('d', d7, 'a', a7, 'alpha', alfa7);

T_base = [0 0 1 0; 1 0 0 0; 0 1 0 0; 0 0 0 1];
T_tool = [-1 0 0 0; 0 -1 0 0; 0 0 1 0; 0 0 0 1];
ARM = SerialLink([L1 L2 L3 L4 L5 L6 L7], 'base', T_base, 'tool', T_tool, 'name', 'ARM');

lb = [deg2rad(-90) deg2rad(-120) deg2rad(-90) deg2rad(0) deg2rad(0) deg2rad(-70) deg2rad(-
25)];
ub = [deg2rad(90) deg2rad(90) deg2rad(90) deg2rad(150) deg2rad(90) deg2rad(70) deg2rad(35)];

ARM.qlim(:,1) = lb;
ARM.qlim(:,2) = ub;

end
```

```
function UR5=UR5_definition(T_base_ur)

% [m] Parametri UR5e
% a_2=0.425;
% a_3=0.39225;
% d_1=0.08916;
% d_4=0.10915;
% d_5=0.09456;
% d_6=0.055;
% s_1=0.14;
% s_3=0.12;

d1 = 0.089160; d2 = 0.14; d3 = -0.12; d4 = 0.10915; d5 = 0.09456; d6 = 0.055;
a1 = 0; a2 = -0.425; a3 = -0.39225; a4 = 0; a5 = 0; a6 = 0;
alfa1 = pi/2; alfa2 = 0; alfa3 = 0; alfa4 = pi/2; alfa5 = pi/2; alfa6 = 0;

L1 = Revolute('d', d1, 'a', a1, 'alpha', alfa1, 'offset', pi/2);
L2 = Revolute('d', d2, 'a', a2, 'alpha', alfa2);
L3 = Revolute('d', d3, 'a', a3, 'alpha', alfa3);
L4 = Revolute('d', d4, 'a', a4, 'alpha', alfa4);
L5 = Revolute('d', d5, 'a', a5, 'alpha', alfa5, 'offset', pi);
L6 = Revolute('d', d6, 'a', a6, 'alpha', alfa6);

UR5 = SerialLink([L1 L2 L3 L4 L5 L6], 'base', T_base_ur, 'name', 'UR5');

end
```

7. Bibliography

[1] R. von Bernhardi, L. Eugenín-Von Bernhardi, and J. Eugenín, “What is neural plasticity?,” in *Advances in Experimental Medicine and Biology*, vol. 1015, Springer New York LLC, 2017, pp. 1–15. doi: 10.1007/978-3-319-62817-2_1.

[2] P. Maciejasz, J. Eschweiler, K. Gerlach-Hahn, A. Jansen-Troy, and S. Leonhardt, “JNER JOURNAL OF NEUROENGINEERING AND REHABILITATION REVIEW Open Access A survey on robotic devices for upper limb rehabilitation,” 2014. [Online]. Available: http://www.jneuroengrehab.com/content/11/1/3

[3] E. D. Oña, R. Cano-de la Cuerda, P. Sánchez-Herrera, C. Balaguer, and A. Jardón, “A review of Robotics in Neurorehabilitation: Towards an automated process for upper limb,” *Journal of Healthcare Engineering*, vol. 2018. Hindawi Limited, 2018. doi: 10.1155/2018/9758939.

[4] L. Pignolo, “Robotics in neuro-rehabilitation,” in *Journal of Rehabilitation Medicine*, Oct. 2009, vol. 41, no. 12, pp. 955–960. doi: 10.2340/16501977-0434.

[5] M. Caimmi *et al.*, “Predicting Functional Recovery in Chronic Stroke Rehabilitation Using Event-Related Desynchronization-Synchronization during Robot-Assisted Movement,” *BioMed Research International*, vol. 2016, 2016, doi: 10.1155/2016/7051340.

[6] C. Duret, A. G. Grosmaire, and H. I. Krebs, “Robot-assisted therapy in upper extremity hemiparesis: Overview of an evidence-based approach,” *Frontiers in Neurology*, vol. 10, no. APR. Frontiers Media S.A., 2019. doi: 10.3389/fneur.2019.00412.

[7] T. Nef, M. Guidali, and R. Riener, “ARMin III - arm therapy exoskeleton with an ergonomic shoulder actuation,” *Applied Bionics and Biomechanics*, vol. 6, no. 2, pp. 127–142, 2009, doi: 10.1080/11762320902840179.

[8] P. S. Lum, C. G. Burgar, P. C. Shor, M. Majmundar, and M. van der Loos, “Robot-assisted movement training compared with conventional therapy techniques for

the rehabilitation of upper-limb motor function after stroke," *Archives of Physical Medicine and Rehabilitation*, vol. 83, no. 7, pp. 952–959, 2002, doi: 10.1053/apmr.2001.33101.

[9] E. Akdoğan, "Upper limb rehabilitation robot for physical therapy: Design, control, and testing," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 24, no. 3, pp. 911–934, 2016, doi: 10.3906/elk-1310-50.

[10] "User Manual UR5/CB3 Original instructions (en)."

[11] E. Kyrkjebø, M. J. Laastad, and Ø. Stavdahl, "Feasibility of the UR5 Industrial Robot for Robotic Rehabilitation of the Upper Limbs After Stroke."

[12] Y. Shi and S. Lichman, "Smart Cameras: A Review."

[13] "MACHINE VISION VISION SYSTEMS ▪ VISION SENSORS ▪ DEEP LEARNING ▪ VISION SOFTWARE." [Online]. Available: www.cognex.com/machine-vision

[14] "In-Sight ® 7000 Gen2 Series Vision System Reference Guide," 2019.

[15] M. J. Grimble *et al.*, "Advanced Textbooks in Control and Signal Processing Series Editors Other titles published in this series: Introduction to Optimal Estimation."

[16] M. Desmurget and C. Prablanc, "Postural Control of Three-Dimensional Prehension Movements," 1997.

[17] M. A. Lemay and P. E. Crago, "Pergamon A DYNAMIC MODEL FOR SIMULATING MOVEMENTS OF THE ELBOW, FOREARM, AND WRIST," 1996.

[18] R. Raikova, "A GENERAL APPROACH FOR MODELLING AND MATHEMATICAL INVESTIGATION OF THE HUMAN UPPER LIMB," 1992.

[19] A. Gams and J. Lenari, "Humanoid arm kinematic modeling and trajectory generation."

[20] G. P. G. Legnani, *Fondamenti di meccanica e biomeccanica del movimento.*, CittàStudi. 2016.

[21] S. L. Delp *et al.*, "OpenSim: Open-source software to create and analyze dynamic simulations of movement," *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 11, pp. 1940–1950, Nov. 2007, doi: 10.1109/TBME.2007.901024.

[22] Parham M. Kebria, Saba Al-wais, Hamid Abdi, and Seid Nahavandi, *Kinematic and Dynamic Modelling of UR5 Manipulator*.

[23] S. Moe, G. Antonelli, A. R. Teel, K. Y. Pettersen, and J. Schrimpf, "Set-based tasks within the singularity-robust multiple task-priority inverse kinematics framework: General formulation, stability analysis, and experimental results," *Frontiers Robotics AI*, vol. 3, no. APR, Apr. 2016, doi: 10.3389/frobt.2016.00016.

[24] Q. Sun, S. Guo, and L. Zhang, "Kinematic dexterity analysis of human-robot interaction of an upper limb rehabilitation robot," *Technology and Health Care*, vol. 29, no. 5, pp. 1029–1045, Jan. 2021, doi: 10.3233/thc-202633.

[25] M. Yamashita, "Robotic Rehabilitation System for Human Upper Limbs Using Guide Control and Manipulability Ellipsoid Prediction," *Procedia Technology*, vol. 15, pp. 559–565, 2014, doi: 10.1016/j.protcy.2014.09.016.

[26] P. K. Artemiadis, P. T. Katsiaris, M. v Liarokapis, and K. J. Kyriakopoulos, "On the Effect of Human Arm Manipulability in 3D Force Tasks: Towards Force-controlled Exoskeletons."

[27] T. Petrič, L. Peternel, J. Morimoto, and J. Babič, "Assistive arm-exoskeleton control based on human muscular manipulability," *Frontiers in Neurorobotics*, vol. 13, 2019, doi: 10.3389/fnbot.2019.00030.

[28] J. J. Steil, S. Gopinathan, and P. Mohammadi, "Improved Human-Robot Interaction: A manipulability based approach," 2018. [Online]. Available: https://www.researchgate.net/publication/325871439

[29] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, "Robotics: modelling, planning and control."

[30] I. Enebuse, M. Foo, B. S. K. K. Ibrahim, H. Ahmed, F. Supmak, and O. S. Eyobu, "A comparative review of hand-eye calibration techniques for vision guided

robots," *IEEE Access*, vol. 9, pp. 113143–113155, 2021, doi: 10.1109/ACCESS.2021.3104514.