ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE SISTEMAS

UNIDAD DE TITULACIÓN

DESIGN AND IMPLEMENTATION OF AN AGENT USING REINFORCEMENT LEARNING TO THE OPERATION OF A MYOELECTRIC PROSTHESIS USING EMG SIGNALS

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL GRADO DE MAGISTER EN COMPUTACIÓN

JONATHAN ALEJANDRO ZEA GUACHAMÍN

jonathan.zea@epn.edu.ec

Director: MARCO E. BENALCÁZAR, PhD.

marco.benalcazar@epn.edu.ec

Codirector: ÁNGEL LEONARDO VALDIVIESO, PhD.

angel.valdivieso@epn.edu.ec

Quito,junio de 2022

DIRECTOR'S APPROVAL

As director of the research thesis "Design and implementation of an agent using reinforcement learning to the operation of a myoelectric prosthesis using EMG signals" carried on by Jonathan Alejandro Zea Guachamín, student of the master's degree in Computer Science, I certify that I have supervised the completion of this work and made the corresponding corrections, therefore I give my approval of the final written document, in order to continue with the corresponding procedures required prior the oral defense.

Firmado electrónicamente por:
MARCO ENRIQUE
BENALCAZAR
PALACIOS

Marco E. Benalcázar, PhD.

DIRECTOR

CODIRECTOR'S APPROVAL

As codirector of the research thesis "Design and implementation of an agent using reinforcement learning to the operation of a myoelectric prosthesis using EMG signals" carried on by Jonathan Alejandro Zea Guachamín, student of the master's degree in Computer Science, I certify that I have supervised the completion of this work and made the corresponding corrections, therefore I give my approval of the final written document, in order to continue with the corresponding procedures required prior the oral defense.

ANGEL LEONARDO VALDIVIESO CARAGUAY

Firmado digitalmente por ANGEL LEONARDO VALDIVIESO CARAGUAY
DN: cn=ANGEL LEONARDO VALDIVIESO CARAGUAY c=EC o=SECURITY DATA S.A. 2 ou=ENTIDAD DE CERTIFICACION DE INFORMACION
Motivo:Acepto los términos que define el emplazamiento de mi firma en este documento
Ubicación:
Fecha:2022-06-24 11:02-05:00

Ángel Leonardo Valdivieso, PhD.

CODIRECTOR

AUTORSHIP DECLARATION

I, Jonathan Alejandro Zea Guachamín, hereby declare under oath that the following work is of my own; that has not been previously submitted for any degree nor professional qualification; and that I have reviewed the bibliographic references cited in this research.

The *Escuela Politécnica Nacional* may use the corresponding rights of this work, as established by the Intellectual Property Law, by its regulations and by the current institutional normative.

Firmado electrónicamente por:
JONATHAN ALEJANDRO ZEA GUACHAMIN

Jonathan Alejandro Zea Guachamín

ACKNOWLEDGMENT

Agradezco a los miembros del Laboratorio de Inteligencia y Visión Artificial por ser quienes me han abierto las puertas y me han mostrado siempre su apoyo.

También a mis padres, de quienes sigo aprendiendo.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

RESUMEN

En el mundo existen aproximadamente 65 millones de amputados, de los cuales el 38,7% corresponden a amputaciones de miembros superiores, y la mayoría de estos se encuentran en países en vías de desarrollo. Esto lleva a la necesidad de una prótesis de mano asequible y de fácil fabricación que alivie la situación de estas personas. Este proyecto propone el uso del aprendizaje por refuerzo para la operación de una prótesis mioeléctrica. Se construyó en 3D un prototipo de prótesis con materiales flexibles a partir de un modelo de código abierto, de bajo costo con 4 grados de libertad, 1 para cada dedo excepto el meñique. El entrenamiento del agente que opera la prótesis se dividió en dos etapas, simulación y fine-tuning. Para ello, se implementó un modelo simplificado de la dinámica de la prótesis para simulación. El entrenamiento en simulación permitió rápidas pruebas de concepto y selección de hiperparámetros. Fue necesario un ajuste fino para adaptar el agente al hardware real. Se consideraron tres enfoques para la función de recompensa, los mejores resultados se obtuvieron con la combinación de 2 de ellos: la distancia al objetivo y una recompensa discreta dependiendo de la acción seleccionada por el agente. El desempeño de la prótesis se midió utilizando la tasa de éxito en las tareas de agarrar y soltar obteniendo un 86%. A partir de esto, concluimos que se dio un paso exitoso hacia la meta de una prótesis de mano mioeléctrica completamente funcional.

Palabras clave: prótesis mioeléctrica, EMG, operación de prótesis usando aprendizaje por refuerzo, Deep Q-learning

ABSTRACT

In the world exists approximately 65 million amputees, 38.7% of which correspond to upper limb amputations, and most of them are found in developing countries. This leads to the necessity of an affordable, easy-to-made hand prosthesis to alleviate these people situation. This project proposes the use of reinforcement learning to train an agent for the operation of a myoelectric prosthesis. A prosthesis prototype with flexible materials was built from an open-source model chosen from the literature. The prothesis was modified and built as a low-cost 3D printed prototype with 4 degrees of freedom, 1 for each finger but the little. A commercial electromyography sensor is used, and just during evaluation a glove with flexion sensors. The training of the agent that operates the prosthesis was divided in two stages, simulation and fine-tuning. For this matter, a simplified model of the prosthesis dynamics was implemented for simulation. Simulation training allowed fast proof of concept testing and hyperparameters selection. Fine-tuning was necessary to adapt the agent to the real hardware. Three approaches for the reward function were considered, the best results were obtained with the combination of 2 of them: the distance to the target, and a discrete reward depending on the selected action by the agent. The performance of the prosthesis was measured using the success rate of the grasp and release tasks obtaining an 86%. From this, we conclude that this research was a successful first step towards achieving a fully functional myoelectric hand prosthesis.

Keywords: myoelectric prosthesis, EMG, reinforcement learning for prosthesis operation, deep Q-learning

CHAPTER 1:
INTRODUCTION

1.1 Background

It is estimated that in the world are 65 million amputees due to trauma and different diseases. Most of traumatic amputations are found in developing countries, and 38.7% of it corresponds to upper limb amputations (McDonald et al., 2020). In Ecuador, Consejo Nacional para la Igualdad de Discapacidades (CONADIS by its acronym in Spanish) in January 2022 estimated that 215 thousand people suffer physical disability, considering people with hand amputation. Amputees can improve their quality of life by using active prosthesis, as these allow different degrees of mobility, instead of passive prosthesis that are intended to be more cosmetic. Among active prothesis, the ones that have the greatest potential to reproduce the 27 degrees of freedom of a human hand are the myoelectric prostheses. This kind of prosthesis measures the electromyography signals EMG that transmit the movement intention sent from the brain to the skeletal muscles, causing muscle contraction, the base of force and movement. Unfortunately, commercial myoelectric prostheses are unaffordable for most people in developing countries due to its high cost (between 20 and 30 thousand US dollars). Furthermore, its functionality is still very limited, reason for which they have high abandonment rates.

In this project, we propose the development of an agent using reinforcement learning for the operation of a myoelectric prosthesis. The agent's input will be EMG signals from a user and kinematic information related to the posture of the prosthesis. The final goal would be to implement and train the machine algorithms on amputees, but in this project, the EMG signals will be acquired from the right forearm of a user with no disability. A prototype of a myoelectric prosthesis for the right hand will be built with 4 degrees of freedom and will be able to imitate a whole-hand grasping movement. This prototype will be based on an open-source design and will be built using 3D printing.

1.2 General objective

Design, train and implement an agent using reinforcement learning to the operation of a myoelectric transradial prosthesis using EMG signals.

1.3 Specific objective

1. Review the literature about prosthesis control using EMG, and 3D printed Open-Source active prosthesis designs to compare results with the present project.
2. Build a transradial myoelectric prosthesis with 4 degrees of freedom and develop its communication interface with a central PC.
3. Design an agent using reinforcement learning that uses the EMG signal from an 8-channel commercial sensor and the relative encoders position to control the actuators of the hand prosthesis in real time.
4. Train the agent so that the hand prosthesis can imitate a whole-hand grasp movement from the right hand of a user with no disability.
5. Evaluate the agent's real-time operation and performance using the success rate on the execution of whole-hand grasp and release tasks.

1.4 Hypothesis

An agent trained with reinforcement learning that uses an 8-channel EMG commercial device can operate a 4 degrees of freedom myoelectric prosthesis to imitate a whole-hand grasp movement.

CHAPTER 2:
THEORETICAL FRAMEWORK AND LITERATURE REVIEW

2.1 Prosthesis overview

There are two kinds of prostheses: active and passive. Passive or aesthetic prostheses do not have any type of movement and only replicate the morphology of the hand. Meanwhile, active or functional prostheses are much more versatile because they have actuators that allow finger movement. In the case of active prostheses, the essential movement they must perform is to open and close the hand (Fajardo et al., 2020). More versatile models allow independent finger control (Vasan & Pilarski, 2017), but its functionality is still limited. Among the active prostheses, there are three types of activation: manual, body-powered and myoelectric. Manual prostheses use a button or a similar mechanism for activation. Body-powered prostheses use the movement of another part of the body to activate the prosthesis. For its part, myoelectric prostheses try to predict the user's movement intention, by measuring the EMG signal that travels from the brain to the muscles through the nervous system (Castellini, 2019). Via this mechanism it is commanded the activation of electric motors that give movement to the prosthesis. Body-powered prostheses are the most widely used, as they are generally considered more robust. This is because myoelectric prostheses significantly reduce their accuracy due to poor placement, electrode displacement, bad conductivity contact with the skin, and require continuous adaptations over time (Farina et al., 2014). However, the great opportunity with myoelectric prostheses is that EMG signals travel through the arm of the individual with the movement information, even though the limb has been amputated. Potentially, it can also be used in lower limb prosthesis because they have the same principle. This technology —when using surface electrodes— is also non-invasive, and thus reduces the risk of complications as it does not need chirurgical intervention. Unfortunately, commercial myoelectric prostheses currently have a high cost (between 20 and 30 thousand dollars in the US), far from the budget of most families in Ecuador. This generates the need to develop low-cost active prostheses.

A recent option that has emerged as an affordable alternative for the construction of prostheses is 3D printing. These initiatives have the advantage of being cheaper to manufacture, although they still have performance and durability limitations. In this sense, there are several Open-Source models for 3D printing of prostheses. Some of these initiatives allow the dimensions and shapes of the prosthesis to be customized to better fit the specific

measurements of a user. Most of these prostheses, however, are body-powered, which significantly limits the mobility they provide.

2.2 Prosthesis research challenges

Scientific research regarding active prostheses can be synthesized in the following 6 sub-problems (Castellini, 2019).

Table 19.2 Seven Subproblems in Upper Limb Prosthetics, and Related Challenges/Ideas/Perspectives to be Addressed in the Upcoming Years, for the Benefit of Active Upper Limb Prostheses Users

	Current	Short Term	Mid-Term	Long Term
Socket technology	Different for each patient and in each country; no standard mechanization	3D scanning; assessment of loci of residual activity; CAD design of socket	Semimechanized procedure becomes standard in the clinics	Completely automatic socket design, tailored to each patient
Prosthetic hardware	Electric motors; few or no sensors; up to 10 DoFs	Impedance control; sensors + closed-loop (position/velocity) control	3D printing to industrial strength; low-cost upper limb prosthetics	Tendon-driven activation; semisoft materials; human-likeness (?)
Sensors	Two sEMG sensors with sequential control; eight sEMG sensors with pattern recognition (CoApt)	Validation of novel sensors (tactile, ultrasound, etc.); sensors embedded in bio-compatible silicon	Targeted array of multimodal sensors and built-in miniaturized electronics	Sensor array built contextually with the socket
Proportionality	Present in traditional two-sensors control but only for one DoF at a time	Proportional control over some motors	Proportional coordinated control over all motors	Usage of (novel) muscle synergies to yield physiologically plausible coordinated control
Simultaneous control	None	Simultaneous control over a subset of the motors of the device	Simultaneous control over all motors of the device	
Coadaptation	Scarce evidence; assessment of signal change in the subject	Precise, qualitative, and quantitative assessment of parallel change of subject and control system	Correlation of functional improvement and coadaptation	Structured, standardized quantitative assessment of coadaptation, performed semiautomatically
Functional assessment	ACMC [46] is the only protocol targeting myocontrol—no focus on the control system	Protocols including the characteristics of the myocontrol system	Assessment with both user and control system in the loop	

Figure 2.1 Subproblems in prosthesis research from (Castellini, 2019)

- **Socket technology:** refers to the mechanism by which the patient inserts and adjusts the prosthesis to her body.
- **Hardware:** refers to the number of motors, degrees of freedom and versatility of movement that the prostheses have.

- **Sensor technology:** refers to the capacity, resolution and reliability of the sensor to capture physiological signals related to movement intention and tactile perception.
- **Independent and force control:** refers to the ability to independently control the movement of each finger including the level of force; as part of the control is gesture recognition.
- **Co-adaptation:** refers to both the prosthesis calibration process and the user's learning process.
- **Performance evaluation:** refers to some regulatory framework or procedure that allows comparing or certifying the characteristics of a prosthesis.

The socket is the mechanism by which the user can connect the prosthesis to their body. The challenge in this area is that dimensions and weight —between other characteristics— must be adapted to each person during the prosthesis connection with the elbow or shoulder. Also comfort, sanitation, and the connection with the sensors must be taken into account (Hallworth et al., 2020).

Each of the aforementioned aspects will be analyzed below, explaining the advances and challenges for new research reported by the literature. Special attention will be paid to control algorithms and gesture recognition, because they are the areas in which this project plans to make a scientific contribution.

2.3 Hardware and Degrees of Freedom

With respect to the hardware, Prostheses currently consist of few degrees of freedom DoF when compared to the total versatility of the human hand. The human hand is estimated to have 27 degrees of freedom: 3 for translation, 3 for rotation, and 4 for each finger except the thumb that has 5 (ElKoura & Singh, 2003). Fingers have 3 joints with flexion-extension type of movement, and 1 joint with abduction-adduction. Abduction-adduction is the pair of movements of a finger joint in the same plane of the palm, abduction is moving away, and adduction is coming closer. The other type of movement is flexion-extension: flexion is the movement of a finger trying to touch the palm, and extension is its oppositive, trying to move away. The thumb has 5 degrees of freedom: 3 flexion-extension joints as the rest of the fingers, and 2 abduction/adduction, 1 additional than the other fingers in the trapeziometacarpal joint (between the trapezium and the metacarpal) (Rahman & Al-Jumaily, 2013).

In this sense, 3D printing has allowed the appearance of new prosthesis designs with varying degrees of freedom. In (Fajardo et al., 2020) a low-cost prosthesis with 6 degrees of

freedom is presented, one per finger that allows the hand to open and close, and another to rotate the wrist. In (Vasan & Pilarski, 2017) was used the Bento Arm with 5 degrees of freedom that allows him to pronate-supinate and open-close his hand. In addition, in this work a glove was used to measure the desired position that the prosthesis should follow (see Figure 2.2). Increasing degrees of freedom implies installing more motors in the structure, and therefore increasing the electrical consumption and the total weight of the prosthesis. The selection and placement of degrees of freedom and its driving motors is important as its size correlates with the prosthesis force and thus can only be reduced to certain extend.

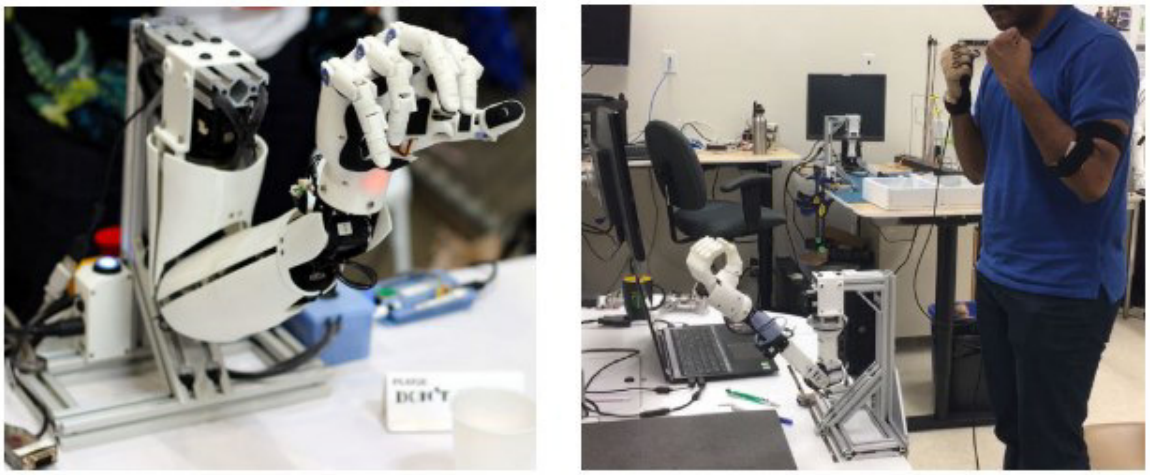


Figure 2.2 Usage of commercial prosthesis in research from (Vasan & Pilarski, 2017)

A modern approach proposes using flexible materials for prosthesis 3D printing. (Mohammadi et al., 2020) developed a soft prosthesis with TPU material, which design is open source. Its hand model was subjected to mechanical stress tests to verify that it can withstand a 1Kg load and 45 000 cycles before the appearance of some cracks (Mohammadi et al., 2020). This model, however, consists of only 5 degrees of freedom, one for each finger. The advantage of using flexible materials is that it reduces the number of moving parts, which may increase its durability.

2.4 Sensors used in prosthesis

Sensors are used to measure biological signals and are the base for gesture recognition systems that predict the user's move intention. Electromyography is a technique that records the electrical activity of skeletal muscles (Jaramillo-Yánez et al., 2020). There are two types of sensors: intramuscular and superficial. Intramuscular electromyography involves inserting needles directly into a person's muscle. In (George et al., 2020) a 96 Utah Slanted

electrode array was inserted into two transradial amputees. These monitoring systems have high accuracy but involve subjecting the patient to medical surgery to install the sensors. In contrast, surface electromyography is a non-invasive technique that uses surface electrodes that are placed in contact with the skin. However, its drawback is lower measurement accuracy due to location, temperature, skin conductivity, and interference between EMG signals from various muscles (Weiss et al., 2015). In (Côté-Allard et al., 2019) is a comparison of EMG commercial devices (view Figure 2.3). The most affordable commercially available surface EMG sensors (less than $2,000) are the Myo Armband and the gForce Pro (view Figure 2.4). These consist of 8 surface channels to measure differential EMG signal and a 9-axis IMU at 50 Hz with Bluetooth communication to the computer. They differ in the EMG sampling rate (gForce Pro at 1Khz and Myo Armband at 200Hz). With respect of the rest of devices, those have higher sampling rates, resolution, number of channels, and higher quality data. A comparison of a medical-grade with a low-cost device was carried out by (Pizzolato et al., 2017) concluding that dexterous control can be performed with a small decreased performance. Based on all this, we consider that wireless low-cost EMG devices may be suitable for myoelectric prosthesis control.

	Delsys Systems Trigno Avanti	Biometrics DataLITE sEMG	Noraxon Ultium EMG	Oymotion gForce-Pro	Thalmic Lab Myo Armband	Hercules	3DC Armband
sEMG channels	up to 16	up to 16	up to 32 (at 2000 sps) or 16 (at 4000 sps)	8	8	8	10
sEMG ADC *	16 bits	13 bits	16 bits	8 bits	8 bits	12 bits	10 bits (ENOB *) (data sent on 16 bits)
sEMG Sampling rate	1960 sps	2000 sps	4000 sps	1000 sps	200 sps	1000 sps	1000 sps
Bandwidth or Built-in Filters	20–450 Hz or 10–850 Hz	10–490 Hz	5/10/20–500/1000/1500 Hz	20–500 Hz	~5–100 Hz	20–500 Hz	20–500 Hz
Contact Dimensions	5 mm^2	78 mm^2	N.A.	~66 mm^2	100 mm^2	78 mm^2	50 mm^2
Contact Material	Silver	Stainless Steel	N.A.	Stainless steel silver coated	Stainless Steel	Gold plated Copper	Electroless nickel immersion gold (ENIG)
Full Scale (Peak to Peak)	+/−11 sps or +/−22 sps	+/−6 sps	+/−24 sps	N.A.	~+/−1 sps (measured)	+/−6 sps	+/−3 sps
Input referred-noise (On system bandwith)	N.A.	<5μV	<1 μV	N.A.	N.A.	N.A.	2.2 μV
IMU * sensors	9-axis Acc, Gyro, Mag	No	9-axis Acc, Gyro, Mag (if EMG set at 2000 sps or below)	9-axis Acc, Gyro, Mag	9-axis Acc, Gyro, Mag	No	9-axis Acc, Gyro, Mag
IMU Sampling rate	24–470 Hz (Acc), 24–360 Hz (Gyro), 50 Hz (Mag)	-	200 Hz	50 Hz	50 Hz	-	50 Hz
Transmitter	BLE 4.2	WiFi	2.4 GHz	BLE 4.1	BLE 4.0	Wi-Fi	Enhanced Shockburst **
Autonomy	4 to 8 h	8 h	8 h	N.A.	16 h	N.A.	6 h
Weight	14 g (per channel)	17 g (per channel)	14 g (per channel)	80 g	93 g	N.A.	62 g
Price	~$20,000 USD (for 16 channels)	~$17,000 USD (for 16 channels)	~$20,000 USD (for 16 channels and free battery replacement)	$1250 USD	$200 USD	N.A.	~$150 USD ***

* ADC: Analog-to-digital converter; ENOB: effective number of bits; IMU: inertial measurement unit; BLE: Bluetooth low energy. ** 2.4 GHz low-power custom protocol (similar to BLE*) from Nordic Semiconductor, Norway. *** The cost of the System-on-Chip was replaced by the cost of a comparable product: the ADS1298 from Texas Instruments, USA.

Figure 2.3 Comparison of different surface EMG devices from (Côté-Allard et al., 2019)

One of the reasons that makes it difficult to compare the performance of available prostheses is the lack of a clear evaluation methodology. Not being clear about the capabilities and limitations of a prosthesis has caused a high level of abandonment among amputees. Procedures in this direction have recently appeared in the literature. (Manero et al., 2019) proposes asking questions related to quality of life and, additionally, assess motor skills when using the prosthesis. Other works have used the TLX load test (NASA Task Load Index) (Fajardo et al., 2020), which evaluates a user's mental, temporal and physical effort when interacting with a prosthesis. Although there exist proposed solutions, in most of the cases, these are isolated efforts.

a) Myo Armband **b) GForce-Pro**

Figure 2.4 EMG commercial devices

Research regarding prosthetics must also take into account psychological factors. Such as that 35% of patients abandon the use of a prosthesis in the long term (Manero et al., 2019). In the adult population, the main reasons for hand prosthesis abandonment are related to the lack of sensory feedback, poor appearance, aesthetics, limited performance characteristics, and low comfort and durability. The challenges related to prosthetics do not end there, because another area of great challenges is related to children. They require continual changes or updates of the prosthesis structure as the individual grows, which complicates the mechanical and electronic design. In addition, the weight of the prosthesis must be adjusted to the body proportions. This inconvenience is more accentuated in children under 5 years of age, since they are still developing their psychomotor skills (Manero et al., 2019). In general, coadaptation is still a challenge as amputee and prosthesis change and degrade over time. For this reason, WHO guidelines state that a person with amputation needs a new prosthesis every 3 years.

2.5 Machine learning applied in prosthesis

The control approaches mentioned so far used some version of an expert system or a supervised learning technique. However, these have the drawback of requiring an expert on the field to model the environment, or —in the case of supervised learning— that each sample must be labeled beforehand. In the case of the prosthesis, labeling the optimal speed and torque for a given scenario can be difficult or arduous to obtain. Approaches such as reinforcement learning can provide additional benefits to these systems, because it can learn directly from interactions with the environment to find an optimal policy. Reinforcement learning algorithms are based on Markov decision processes, where it try to maximize the expected value of the discounted reward given an initial state and action (Sutton & Barto, 2018). An MDP is the system formed between an agent and an environment, the agent can perform actions that modify the environment, and the environment return rewards and observations based on these actions. Reinforcement learning algorithms could help solve the problem of adapting active prostheses to their users, due to their online and continuous learning capacity. In this work we will use reinforcement learning to train the prosthesis to execute the proposed movements based on the EMG signals measured in the operator's forearm.

A challenge in the design and training of an agent using reinforcement learning is to adequately define the actions, observations and rewards representation of the prosthesis. Few works have been found in the scientific literature that use reinforcement learning for the recognition of hand gestures. For example, (Englehart & Hudgins, 2003; Kukker & Sharma, 2018) present reinforcement learning algorithms based on artificial neural networks and deep learning techniques for recognition of hand gestures. These works use signals acquired with commercial Myo armband sensors and the PowerLab 26T. However, the number of samples used, and the number of users during training is too low for it being considered to real world applications. (Seok et al., 2018) proposes a deep reinforcement learning algorithm to recognize human arm movement patterns using a surface EMG sensor. The results show that the model is more stable and is more accurate in predicting arm movement patterns. In addition, it allows the adaptation of the prostheses to the needs of the patient. Although few works have been found in the literature related to the use of reinforcement learning for gesture recognition and/or prosthesis control, the use of these techniques have the potential of helping to solve some of its remaining challenges, especially the challenges related with proportional, and simultaneous prosthesis control.

CHAPTER 3:
METHODOLOGY AND MATERIALS

In the previous chapter was reviewed the different approaches used in prosthetics construction and development; as well as the main control techniques that have been applied based on Machine Learning. This review enabled us to understand the problematic and focus our research efforts on a specific topic. As the objectives dictate, an operator agent had to be trained. For that matter, it was required a prosthesis prototype to develop and train the reinforcement learning models.

This chapter describes, on one hand, the development of the hardware and software components of the prosthesis, and the other, the implementation of a platform for training an operator agent using the reinforcement learning framework. The hardware component consists of the design and construction of the prosthesis prototype with its electronics sensors and actuators. The software component consists of the communication protocol, and interface to support the execution and interaction between the different entities (such as the EMG sensor, prosthesis actuators and agent). The development and selection of the agent, as well as the usage of the reinforcement learning framework, are more related to the research objectives and thus, are taken special attention.

3.1 EMG sensor

The most important sensor in a myoelectric prosthesis is the EMG sensor. This sensor allows to capture the electromyography signal that carries the intention of movement from the brain to the muscles. The Thalmic's Myo Armband (Figure 2.4b) was preferred among the commercial EMG surface sensors for its robust API and affordability. Although it has a lower sampling frequency than the Oymotion's GForce -Pro, from our experience (Zea et al., 2021), the current GForce-Pro SDK version is still in development with some issues, whereas the Myo Armband has a more robust and stable communication.

3.2 Prosthesis hardware construction

The main considerations taken into account for the design and construction of the prosthesis prototype are the following:

- Usage of available local materials

- Affordable material costs
- Usage of 3D print Open-Source prosthesis models

3.2.1 Selected prosthesis base model

Following these considerations an Open-Source model was selected. The selected model called X-Limb (Mohammadi et al., 2020), and developed by The University of Melbourne, has multi-articulating capabilities using a flexible and soft material (Thermoplastic Polyurethane TPU). The finger's movement is achieved by pulling a tendon-like thread that makes the finger bend by force. This special characteristic makes this hand prosthesis design outstands from other prosthesis designs. In this design, each finger may have independent movement capabilities, resulting in 5 Degrees of Freedom (DoF). With respect to durability, the authors report that a finger can withstand 45000 cycles before the appearance of cracks in the membrane joints (Mohammadi et al., 2020). As can be seen in Figure 3.1, this design uses a Quick Wrist Disconnect QWD to connect with the body. The right-hand model was selected as base design for fabrication using 3D printing. This base model required some design modifications that will be briefly mentioned below.

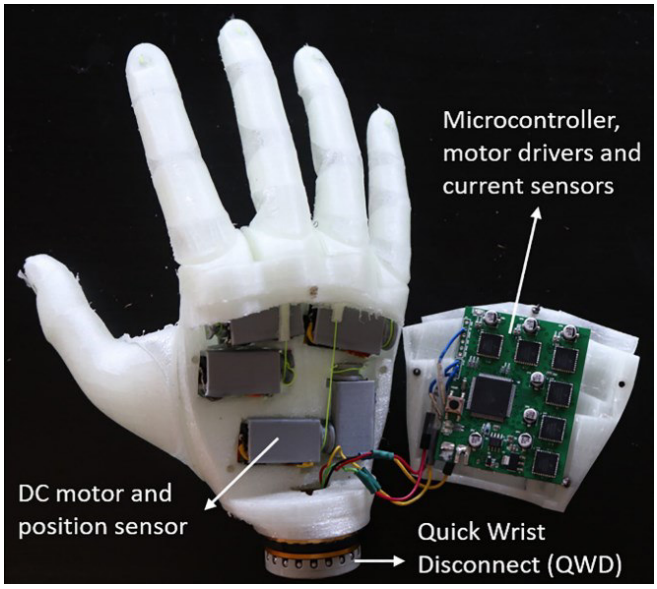


Figure 3.1 X-Limb Open Source anthropomorphic hand design from (Mohammadi et al., 2020)

3.2.2 Degrees of freedom and actuators selection

The prosthesis prototype will have 4 Degrees of Freedom DoF, each one in a different finger of the hand prosthesis except for the little finger. This was decided for several reasons, mainly because we consider that an 8-channel EMG device will be able to capture a maximum of 4 independent movements due to the antagonistic nature of muscle pairs. Although

the prosthesis with 4 fingers may seem odd, we consider that this simplification does not interfere with the research objectives of this project that are focused on training an operator agent with reinforcement learning.

The actuators in the prosthesis are in charge of pulling the tendon threads to close the fingers. Following the base model, the actuators in our prosthesis design are DC motors so that a larger torque in a smaller space can be obtained compared with servo motors. To comply with our objective to build the prosthesis design with local materials, the suggested DC motors must be changed. In Table 3.1 is summarized the specifications of the selected DC motors. The main difference with respect to the suggested DC motors is the gear ratio (1000:1 to 100:1). This motor change implies a faster speed and a lower torque. In this research, the prosthesis prototype is not intended to carry weight yet, so we consider acceptable this difference. The prosthesis will require 4 of these motors, each one for a Degree of Freedom.

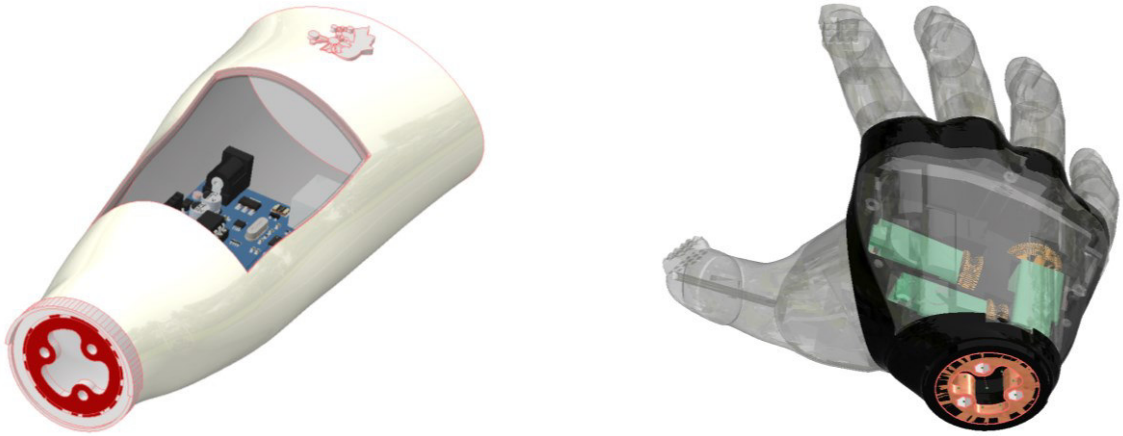
Table 3.1 Motor specifications

Parameter	**Value**
Type	DC motor with extended shaft
Size	10 x 12 x 26 mm (+ 14 mm shaft)
weight	9.5 g
gear ratio	100.37:1
No load speed	310 rpm @ 6v
Stall current	1.6 A @ 6v
speed	250 rpm
torque	0.29 kg * cm
current	0.33 A

The selected motor includes an extended shaft, this is in order to connect a magnetic encoder for measuring the prosthesis fingers position. The base model uses absolute rotary encoders that allows a precise measurement of each motor position but needs to be inserted in the shaft of the motor. In our design, we use incremental magnetic encoders that are widely available. With these encoders each motor speed will be measured, and the angular position will be calculated.

3.2.3 Structure 3D modifications

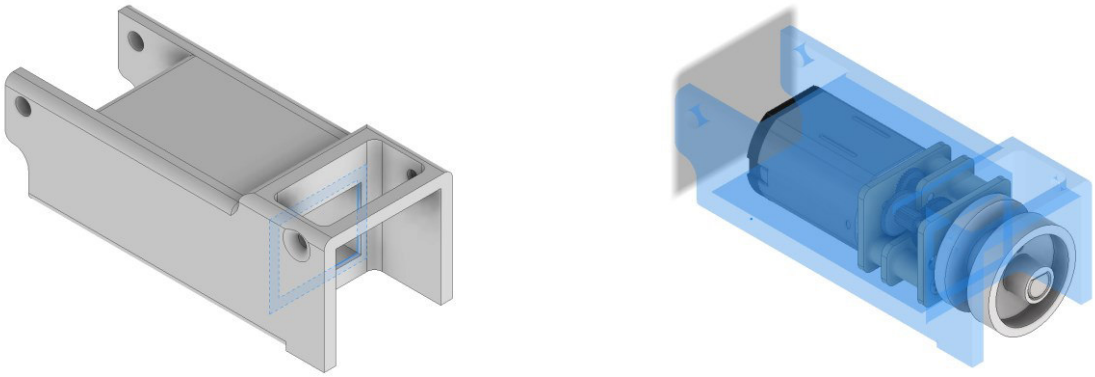
The base prosthesis model requires some adjustments to match the criteria and constraints stablished for this research. The QWD (a medical-grade socket purchased in the base model) was replaced because its high cost turn it to be not of easy acquisition. For that matter, a socket mechanism consisting of a wrist and forearm was developed to allow the connection with the body of an amputee. Also, the motor specifications are different with respect to the original design for availability reasons; to accommodate these motors a spool and a case were developed.



a) forearm **b) wrist**

Figure 3.2 Prosthesis forearm and socket design modifications

In Figure 3.2 is presented the designed forearm and wrist connector. As can be observed, in this prototype, the wrist does not have mobility, and the degrees of freedom come from the finger's movement. The shallow space inside the forearm is intended for hardware allocation. On the wider side of the forearm is reserved an empty space so that the amputee's stump could be inserted. Figure 3.3 shows the motor case and spool designed. The case encloses the motor to avoid glides and vibrations, it also leaves an open space for the encoder wires.



a) case **b) motor and spool in case**

Figure 3.3 Spool and case designs

In Appendix I is included the blueprints of the mechanical design of all these components such as the designed spool, palm connector, motor cases and the forearm socket.

3.2.4 3D Fabrication structure

The prosthesis prototype consists of several parts that have to be 3D printed (fingers, palm, forearm, cases, etc.). Among these, the fabrication of the fingers requires special attention because its flexibility enable the mobility of the prosthesis. The flexibility for the fingers is achieved printing in TPU material. The usage of flexible materials is a new approach under research as an alternative to moving mechanical part. In Table 3.2 is summarized the printing parameters used for the fabrication of the flexible fingers. These parameters are important as the flexibility and durability of the hand depends directly on it and might change significantly for a different set of parameters.

Table 3.2 Flexible fingers 3D printing parameters

Parameter	**Value**
Filament type	TPU
Filament diameter	1.75 ± 0.02 mm
Print speed	26 mm/s
Infill	Gyroid with 2 perimeters
# Top layers	3
# Bottom layers	2
Bed temperature	65 ºC
Extrusor temperature	205 ºC

3.2.5 Power driver and circuitry

The motor driver used is the Adafruit Motor Shield v1.0. This motor driver is capable of driving up to 4 DC motors, and it is based on the integrated circuit L293D that provides a nominal current of 600 mA. Using this driver, the motor speed can be controlled by Pulse Width Modulation PWM. A PCB Connection Board was developed to connect the 4 motors of the prosthesis with the driver shield and with the embedded controller.

More details about the circuitry of the prosthesis are presented in Appendix II. Table 5.1 of this appendix shows the connections between the connections board and the Arduino controller, There is included the wiring used for the motor-encoder connector, the motor connections board with its PCB design; and a hardware-like diagrams.

3.2.6 Embedded controller and Agent Server

An embedded controller is required in the prosthesis to handle the communication between the different sensors, to receive and decode the action chosen by the agent, and to send the corresponding signal to drive the actuators. The embedded controller requires at least 4 PWM and 8 digital inputs for processing the magnetic encoder of each motor. An Arduino Uno R3 is used for this purpose. This contains 8-bit PWM signals, this implies that speed can be controlled in a discrete range between 0-255, forward and backwards direction. The selected motor driver is a board specifically for this microcontroller. This means that we reduce the effort on the electronic design by using compatible hardware.

An important aspect of this research is that the agent (the entity that makes the decisions about the prosthesis movement) is not located in the embedded controller. The system architecture was designed this way to avoid during the development and training of the agent, the constraints of memory and processing speeds innate of embedded systems. The agent is implemented in Matlab R2021b, running in a personal desktop computer called **Agent Server**. The communication interface between the embedded controller and the Agent Server is described in the following Sections.

3.3 Software architecture

As mentioned in the previous Section, the proposed system deploys the agent in a personal desktop computer, aka Agent Server. The Agent Server must transmit the desired action to the prosthesis, as well as to receive the signal from the EMG sensor. Hence, a communication interface needed to be designed between devices.

In Figure 3.4 is summarized the communication intention between devices. The EMG sensor is sending the EMG signal via Bluetooth to the Agent Server, so that the agent can infer the intention of movement of the user. The Agent Server, on the other side, communicates via USB serial with the prosthesis through the embedded controller. In broad terms, the prosthesis reports the angular position of each Degree of Freedom DoF; and receives an action to execute from the Agent Server.

Figure 3.4 Communication diagram between devices

3.3.1 Communication protocol

A communication protocol between the Agent Server and the prosthesis was designed in order to have an efficient data transmission and to reduce its latency. A type of command with its parameter is sent in a *packet*, and it is decoded in the corresponding receiving device. An example of the main packets sent in the communication from the Agent Server to the prosthesis is presented in Table 3.3 and from the prosthesis to the Agent Server in Table 3.4.

Table 3.3 Communication protocol: commands of the Agent Server

Message	Packet Sample	Description
sendAllSpeed(PWM1, PWM2, PWM3, PWM4)	`"aFFB45C00d34"`	Sends the desired speed to the motors. A motor code is encoded with upper case letters (e.g., `B…C`) in the case of forward movement or positive speed, and in lower case letters for backwards movement. The motor speed is sent next to the motor code as a 2-digit hexadecimal number (e.g., `aFF`, motor 1 in reverse direction with 255 PWM).
resetEncoder()	`"R:"`	Resets the internal counter of each encoder to 0.
stop()	`"S:"`	Stops all motors.
changePeriod(p=100)	`"P100"`	Sets the period of the prosthesis for transmitting position values [in milliseconds].

Table 3.4 Communication protocol: messages from the prosthesis

Message	Packet Sample	Description
transmit(p1, p2, p3, p4)	`"x23490y1000z0w796"`	Sends the incremental measure of the encoders. Motors are encoded with the lower cases (e.g., `x…y…z…w…`), and the relative angular position is sent next to it (e.g., `y1000`, for motor 2 the relative angular position is 1000).

The transmission rate of the communication protocol is also important and was set to a reliable pace for each device. In Table 3.5 is presented the transmission rates between the devices in our system. The Flexion Glove has not been described so far but will be introduced in the following Sections.

Table 3.5 Transmission rates

Sender	Receiver	Period | Frequency
EMG sensor	Agent Server	5 ms | 200 Hz
Agent Server	Prosthesis	200 ms | 5 Hz
Prosthesis	Agent Server	100 ms | 10 Hz
Flexion Glove	Agent Server	100 ms | 10 Hz

3.4 Difference between flexion-extension the prosthesis hand

Finishing the description of the prosthesis is important to describe the types of movements it has to perform. From the objectives of this research, the prosthesis requires 2 types of movements:

- Whole-hand grasp of the prosthesis, that from now on will be called flexion, and
- Release, that will be called extension.

Although these movements are complimentary —for they are controlled by moving its corresponding motor forward or backwards— they are not identical. When executing the movement of flexion, the agent will have full control of the prosthesis position, but during extension, the agent will have partial and delayed control. This is due to the tendon-like pulling mechanism implemented to move the fingers: a finger bends (or closes) when the motor pulls its thread; and retreats (or opens) when the motor loose the tension in the thread. The force for this retreat does not come from the motor, but from the elastic force of the bended finger; and was observed to be delayed with respect to the motor movement. For this reason, is considered that the extension movement has a partial and delayed control.

3.5 Reinforcement Learning: agent and environment

Once built the prosthesis prototype, the development of the operator agent using Reinforcement Learning begins. As explained in Chapter 2, it is of crucial importance the differentiation of agent and environment. In Figure 3.5 is presented a diagram of the agent-environment interaction. The environment is composed of the prosthesis, the EMG sensor and a reward function module; and it is defined by its internal *state* s_t. The agent is located in the Agent Server and chooses an *action* A_t to interact with the environment, and in return receives a *reward* r_t. Each of its components will be described in detail in the following sections.

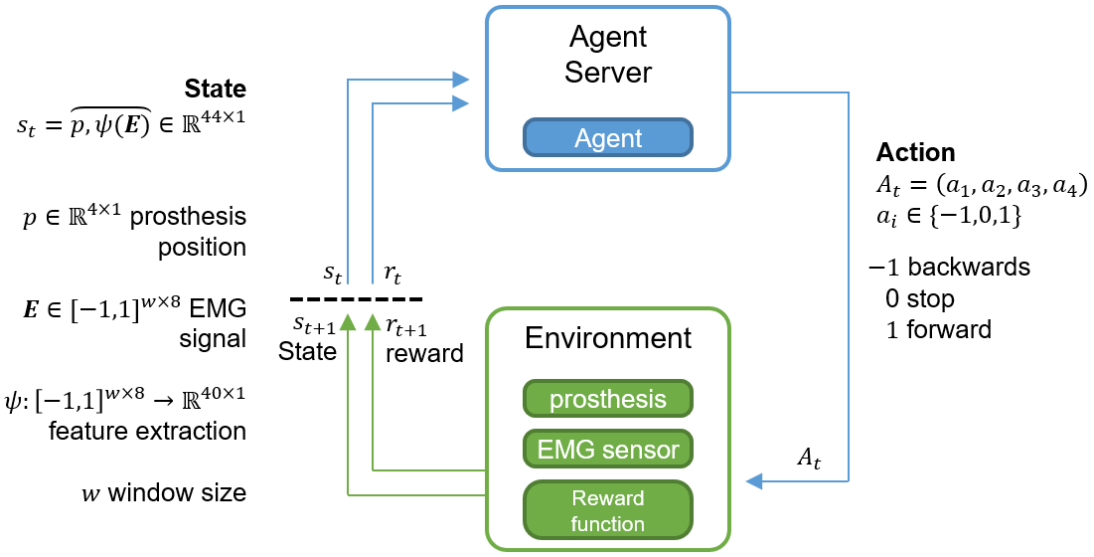


Figure 3.5 States and actions in the agent—environment interaction

3.6 Agent actions

The agent action A_t is vector of 4 components containing the desired movement of each Degree of Freedom in the prosthesis. In each episode step, the agent chooses an action and sends it via serial to the embedded controller in the prosthesis. Each action component corresponds to a motor. In Table 3.6 is included the mapping between motors and prosthesis fingers. As shown in Figure 3.5, each action component has 3 discrete possibilities: **1** that means move the corresponding motor forward, **-1** move the corresponding motor backwards, and **0** stop. For example, an action of the form $(0, -1,1,0)$ would mean to stop motor and 4, move backwards motor 2, and move forward motor 3. The size of the action space is the number of possible combinations of the action components (Equation (1)).

$$|A_t| = |\{-1,0,1\}|^{N_{DoF}} = 3^4 = 81 \quad (1)$$

Table 3.6 Mapping between actions and motors

Finger	Motor index	PWM speed
Little	1	170
Index	2	170
Thumb	3	255
Middle	4	170

The prosthesis listens for an action A_t, and decodes it for its execution. A constant PWM speed was selected heuristically for each motor of the prosthesis (view Table 3.6). The

speed for the thumb was set to the maximum possible (255), because it requires a larger torque for closing the prosthesis hand properly.

3.7 State representation

As shown in Figure 3.5, the state is composed of a total of 44 features: 40 features extracted from the EMG signal, and 4 cinematic features corresponding to the finger positions.

3.7.1 EMG feature extraction

The EMG signal $\boldsymbol{E} \in [-1,1]^{w \times 8}$ in a window of size w is reduced to 40 features using a set of selected feature extraction functions. This set of functions ψ was tested and developed in a previous research in the problem of hand gesture recognition of 5 gestures using EMG signals (Barona López et al., 2020). These functions are: Standard Deviation, Integral Absolute Envelope, Mean Absolute Value, EMG Energy and Root mean square. Because the Agent Server sends the desired action to the prosthesis each 200 ms (view Table 3.5), during this period, the EMG signal is stored in a buffer. Hence, the window size was set to this value 200 ms (40 points@200 Hz).

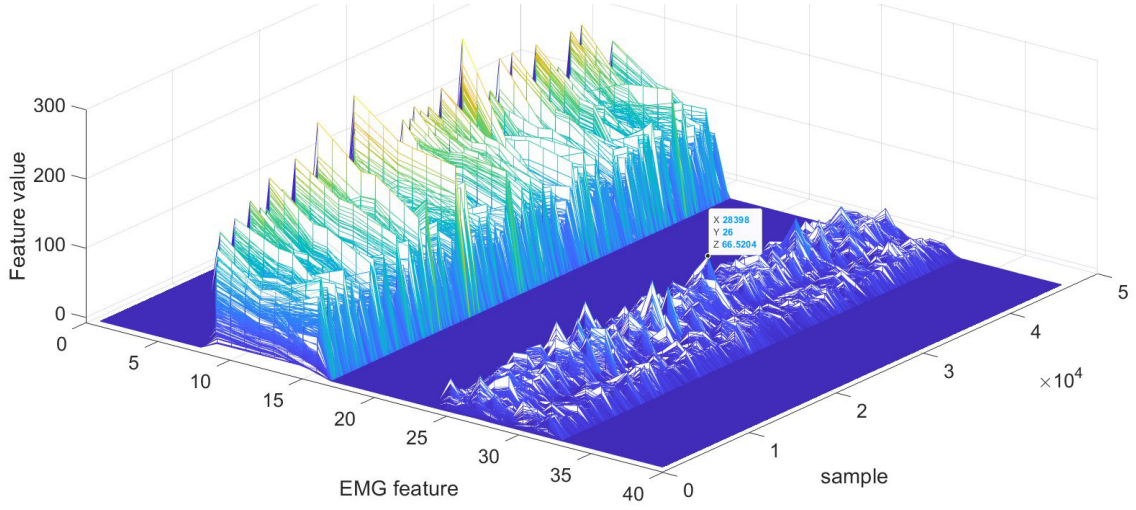


Figure 3.6 EMG feature space (dataset 612-EMG-EPN)

Due to disparity between the order of magnitude between features (view Figure 3.6), the 40 EMG features are normalized using the z-score. For that matter, center and scale values were precalculated (view Figure 3.7) using the training set of the dataset 612-EMG-EPN (Benalcázar et al., 2020).

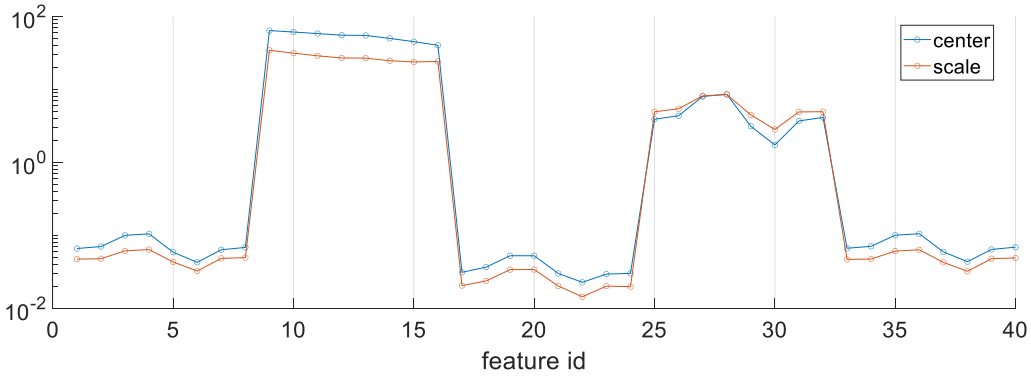


Figure 3.7 z-score normalization values by EMG feature

3.7.2 Cinematic features

The Second component of the agent's state is the cinematic features. These features represent the degree of movement of the 4 fingers in the prosthesis. These cinematic features are inferred from the incremental count of the encoders in the motors. The measurement of the encoders can be used to calculate an angular relative position of the shaft of the motors.

Based on all this, a transformation is required, because the agent requires information about the *degree of closeness* of the fingers, but the available measure of the encoder is angular position. In order to represent the *degree of closeness* of the hand, we use a glove with flexion sensors that will be called Flexion Glove.

3.7.3 Flexion Glove for position feedback

The Flexion Glove is a glove developed in (Estrada Jiménez, 2016) for Sign Language Recognition. It has one flexion sensor for the thumb, and two flexion sensors for the fingers index, middle, ring and little. It samples data at 10 Hz.

The Flexion Glove will be used both by the operator and by the prosthesis (shown in Figure 3.8), in 3 separate instances:

- By the prosthesis, to the estimation of prosthesis hand position.
- By the operator, during a dataset acquisition (prior training) to record hand movements that will be used in offline training.
- And, again by the operator, during online training to measure real-time hand movements, and to calculate agent rewards (view Section 3.10.4).

a) operator **b) prosthesis**

Figure 3.8 Usage of the Flexion Glove

Although, the Flexion Glove in the prosthesis does not fit perfectly (the index and the thumb were a little bit off), it could be adjusted in the transformation after the dataset acquisition.

3.7.4 Dataset acquisition

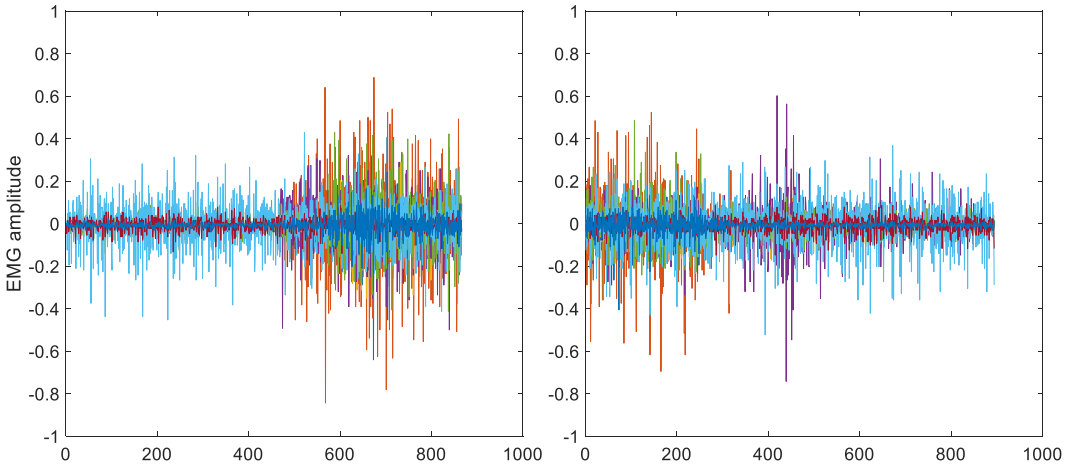


Figure 3.9 EMG signal captured with EMG sensor

A dataset of prerecorded EMG and flexion signals from the hand of the operator was acquired. This dataset is formed of 400 samples in total: 200 samples of the movement **Flexion**, and another 200 samples of **Extension**. For this dataset acquisition, the operator wore the EMG sensor and the Flexion Glove. Each sample recorded lasts between 2 and 5 seconds. In Figure 3.9 (EMG) and in Figure 3.10 (flexion values) is presented a sample of this dataset. The signals corresponding to **Flexion** are on the left side of the axes, while on the right side are the signals corresponding to **Extension**. The flexion signals (Figure 3.10) have its lowest values when the sensor is relaxed (the hand is open) and increase in value when bended (the hand is closed). As a side note, the Flexion Glove returns a Boolean value for when the middle and index fingers are touching, as well as orientation information of the palm (yaw, pitch and roll angles), but both of these signals are ignored in this research.

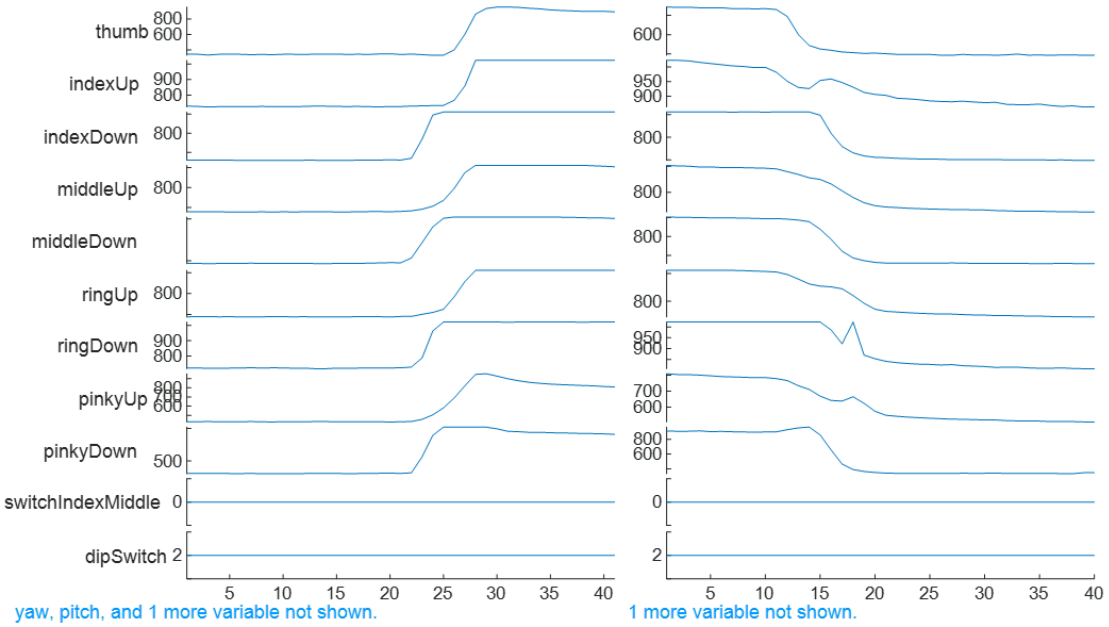


Figure 3.10 Flexion signals captured with Flexion Glove

3.7.5 Transformation from encoder data to prosthesis position

As discussed in Section 3.7.2, the position of the prosthesis fingers p needs to be inferred from the encoder data of its motors. To determine a suitable transformation, measurements of different prosthesis movements while using the Flexion Glove (as shown in Figure 3.8.B) were recorded.

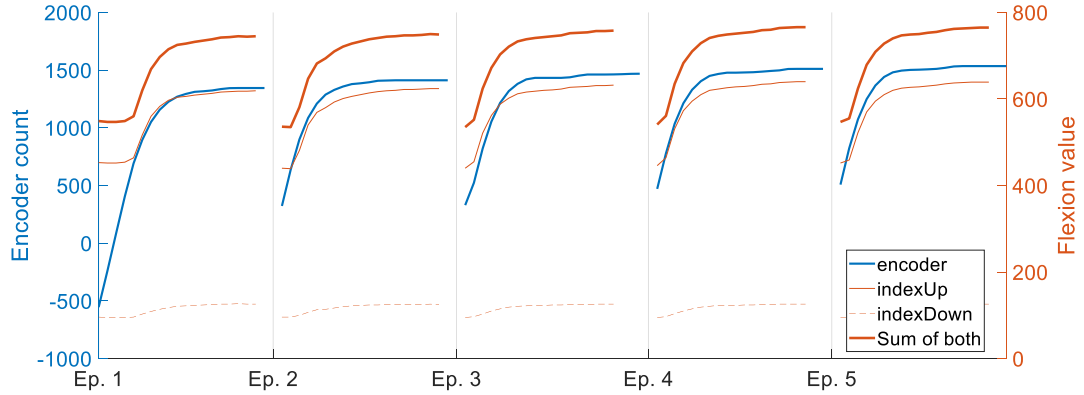


Figure 3.11 Mapping between encoder and flexion data (prosthesis index finger)

This recording consisted of 5 time-separated episodes of the flexion movement. This recording is shown in Figure 3.11 for the index finger, and in Figure 3.12 for the middle finger. In these figures, the encoder signal (*blue*) and the flexion signals are proportional following a similar trend, although in different scales. One of these flexion signals was calculated as the sum of the lower and upper flexion signals of its corresponding finger. It is important to note that the flexion upper signal of a finger is driven mainly by the joint between the palm

and the proximal phalange, while the lower signal is driven mainly by the intermediate and distal phalanges. Also, from the figures can be observed that the upper flexion signal best captures the movement of the prosthesis, while the lower signal is less perceptible.

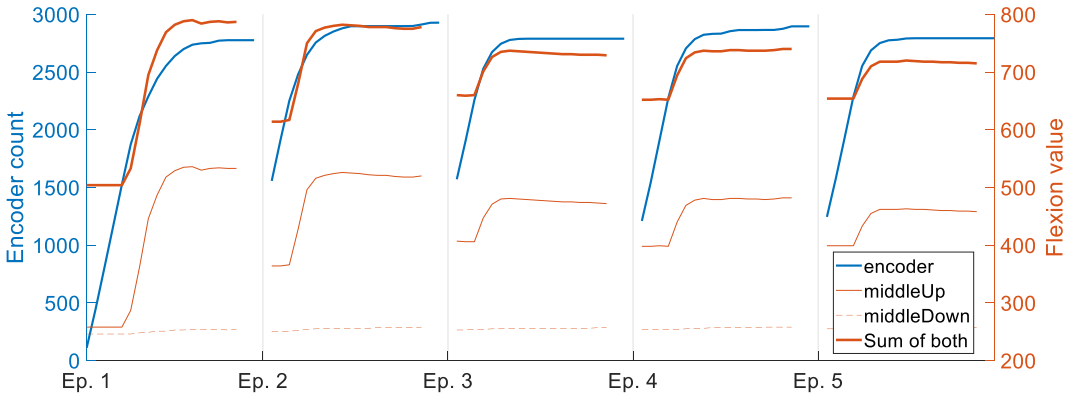


Figure 3.12 Mapping between encoder and flexion data (prosthesis middle finger)

The encoder count will be transformed using a polynomial to obtain an estimate of the position of the prosthesis hand. For each DoF, a polynomial was selected between different d-degree polynomials fit with the flexion and encoder data recorder (view Figure 3.13).

Additionally to the different polynomials fit in Figure 3.13, the **loose zone** is highlighted in red. This zone represents the space that the motor can rotate freely without pulling the finger thread. It is important in our design, as it allows the motor to gain speed to pull the thread strong enough to bend the finger. It was necessary to identify it in every finger to know when the finger does indeed start to bend. Another important zone is the **breaking zone** over on which the finger is completely close and if it keeps pulling it might damage itself. The transformation function considers both of these limits to estimate the prosthesis position.

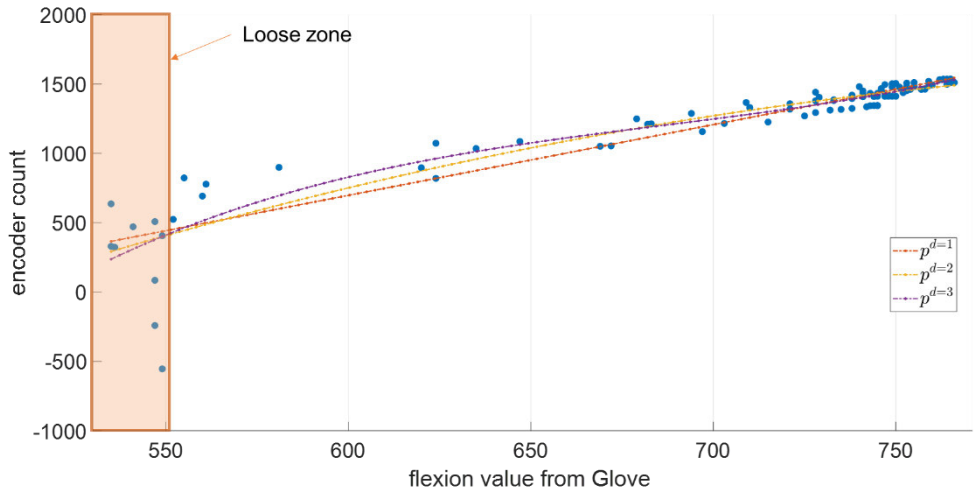


Figure 3.13 Transformation from encoder data to flexion data for the index finger

3.8 Episode definition

In order to train our agent, the episode structure shown in Figure 3.14 will be used. The movements flexion and extension will be separated in continuous episodes. Odd episodes will be loaded with random samples of the flexion movement, and the next episode will be loaded with the corresponding extension movement. This way, the last position of an episode, matches the initial position of the next one. All points inside the loose zone will correspond to the same prosthesis position, this is due to, inside this zone, the finger does not reflect any movement, and hence stay in the same position. The system will block the movement of the agent if it intends to surpass the break limit, avoiding any hardware damage.

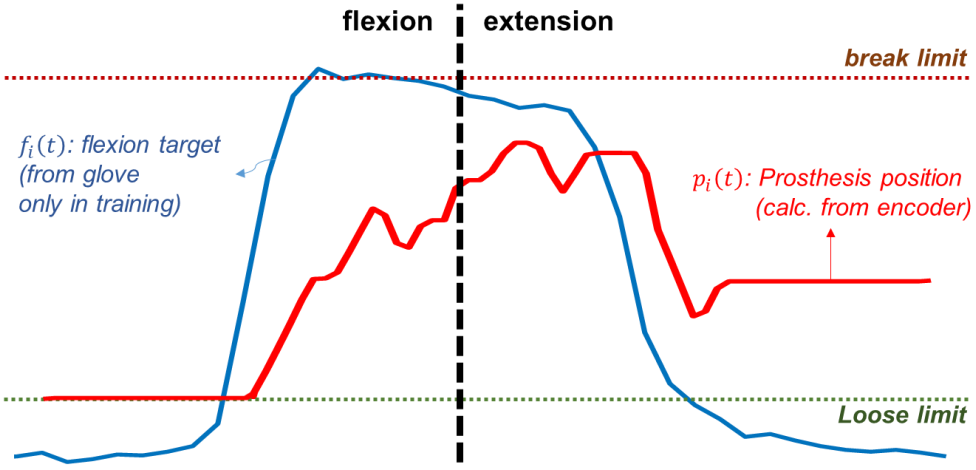


Figure 3.14 Flexion target and prosthesis position in a pair of episodes for a DoF i

3.9 Reward function selection

The reward function is the indirect way through which the desired objective can be thought to the agent. In our case, we want the agent to learn to operate the prosthesis motors so that the prosthesis position $p_i(t)$ of a degree of freedom i follows as close as possible the flexion signal $f_i(t)$ of the finger i (view Figure 3.14). The flexion signal $f_i(t)$ is problematic in the sense that it is obtained from the Flexion Glove, but our objective is not to use flexion sensors (as it measures the flexion of the fingers, unavailable in an amputee), but to use EMG signals. For this reason, the Flexion Glove will only be used during training and during evaluation to calculate the reward function, and not during normal execution.

The reward function that provided us with the best performance (alternative 3) was a combination of 2 approaches: calculating the distance between the target $f_i(t)$ and the prosthesis position $p_i(t)$ (alternative 1), and directly rewarding the action choose by the agent if it

closes the gap to the target $f_i(t)$ (alternative 2). These alternatives will be discussed in further detail in the following subsections.

3.9.1 Alternative 1: rewarding based on distance

In this approach, the reward function R_d (Equation (3)) on a given time instant t is the negative sum of the distances (Equation (2)) between the target $f_i(t)$ and the prosthesis position $p_i(t)$ of each degree of freedom i.

$$d_i = |f_i(t) - p_i(t)|, \qquad i = \{1 \dots 4\} \tag{2}$$

$$R_d = -\sum_{i=1}^{4} d_i \tag{3}$$

The problem with this approach (view Figure 3.15) is that it punishes its reaction time and the transmission delay that does not depend on the agent. It also seemed to cause unstable and abrupt movements of the prosthesis because the rewards were very noisy.

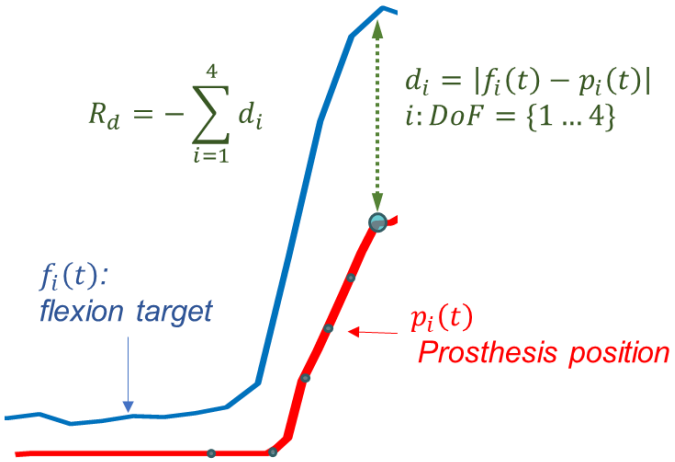


Figure 3.15 Rewarding based on distance

3.9.2 Alternative 2: rewarding based on direction

The second reward approach R_{dir} (Equation (4)) selects among discrete rewards depending on the chosen action of the agent. To calculate this reward the correct action $\bar{a}_i$ is required, additionally it requires a tolerance τ_i and the break limit b_i. Unlike the previous reward function, this function correctly punishes the agent when it wants to go further from the breaking limit.

$$R_{dir} = \sum_{i=1}^{4} \begin{cases} +\mathbf{2}, & if\ a_i = \bar{a}_i \\ -\mathbf{1}, & if\ a_i = 0\ and\ d_i > \tau_i \\ -\mathbf{2}, & if\ a_i = -\bar{a}_i\ and\ p_i < b_i \\ -\mathbf{3}, & if\ a_i = 1\ and\ p_i \geq b_i \end{cases} \tag{4}$$

Where:

a_i is the action taken by the agent for DoF i,

$\bar{a}_i$ is the ideal action that the agent can take,

τ_i is a tolerance range constant,

b_i is the break limit.

This reward function, described in Figure 3.16, works well but does not take in consideration how far the position of the prosthesis $p_i(t)$ is from the target $f_i(t)$. This resulted in an agent satisfied with its performance, because took the correct actions, but finished far from the target.

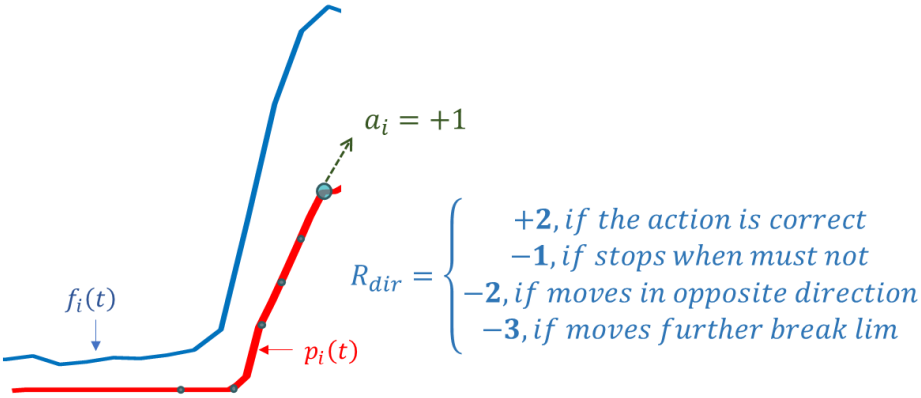


Figure 3.16 Rewarding based on action direction

3.9.3 Alternative 3: Distance and direction rewarding

The best performance was obtained with the combination of both approaches: distance rewarding combined with directional rewarding. This combination uses a constant factor c_i that scales the distance values and defines the relative contribution of each part. This reward function can be understood as a modulation, where the R_{dir} signal is the modulating signal, and R_d is the carrier. This approach has the advantage of rewarding based on the correct action, and how far the prosthesis position is from the target.

$$R = R_{dir} - \sum_{i=1}^{4} c_i * d_i \tag{5}$$

3.10 Training the agent

Training the agent consists in improving its policy based on experience captured from the interaction between the agent and its environment (view Figure 3.5). A reinforcement learning algorithm updates the agent policy to maximize the expected discounted reward. The training framework was developed in Matlab R2021b.

One difficulty in our research was the (relatively) slow movement inherent of the prosthesis hardware. Training completely on hardware would imply limiting the number and time of the different experiments. To handle this limitation, the training of the agent was divided into 2 stages:

- **Stage 1 pretraining in simulation**: in which the majority of hyperparameters are chosen from several parallel tests using a simplified simulation model of the prosthesis.
- **Stage 2 fine tuning in hardware**: in which the *best* agent from Stage 1 was adapted to the hardware.

3.10.1 Prosthesis simulation

A simplified simulation of the prosthesis was developed in order to accelerate the training process during the selection of hyperparameters. This model consists of a parameterized function for each finger that returns the next position (represented as the motor encoder count) after a given time, given an initial speed. This parameterized function has the form of Equation (6). The increasing resistance the finger face when it is bending is reflected in the damping factor ρ, Equation (7). In Figure 3.17 is plot the damping factor vs the encoder position; as can be observed, the resistance increases (tends to cero) with the position.

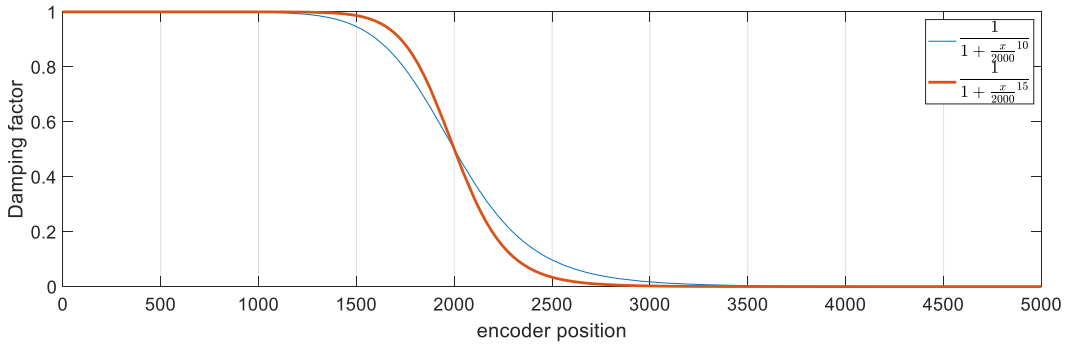


Figure 3.17 Damping factor

$$x_{t+1} = \begin{cases} x_t + k * PWM & if\ sign(x_t) \neq sign(PWM) \\ x_t + k * PWM * \rho(x_t) & if\ sign(x_t) = sign(PWM) \end{cases} \quad (6)$$

$$\rho(x_t) = \frac{1}{1 + \left(\frac{x_t}{2000}\right)^{15}} \tag{7}$$

Where:

x_t is the position simulated

PWM is the PWM speed,

k is a speed constant,

$\rho(x_t)$ is a damping factor, function of the position.

This simulation parameters were calibrated for the motors and the flexible fingers of the prosthesis; and using it, the pretraining stage was carried on.

3.10.2 Selection of training algorithm (Q-Learning with Experience Replay vs Policy Gradient)

Two training algorithms were compared. First preliminary experiments were based on Policy Gradient, but due its negative results this training algorithm was replaced by Deep Q-Learning with Experience Replay.

- **Policy Gradient PG** was used for its advantage of learning directly the policy from the full experience of an episode. However, most of the agent configurations tested resulted in a saturation of the actions. In Figure 3.18 can be observed the first 200 episodes of a training session.

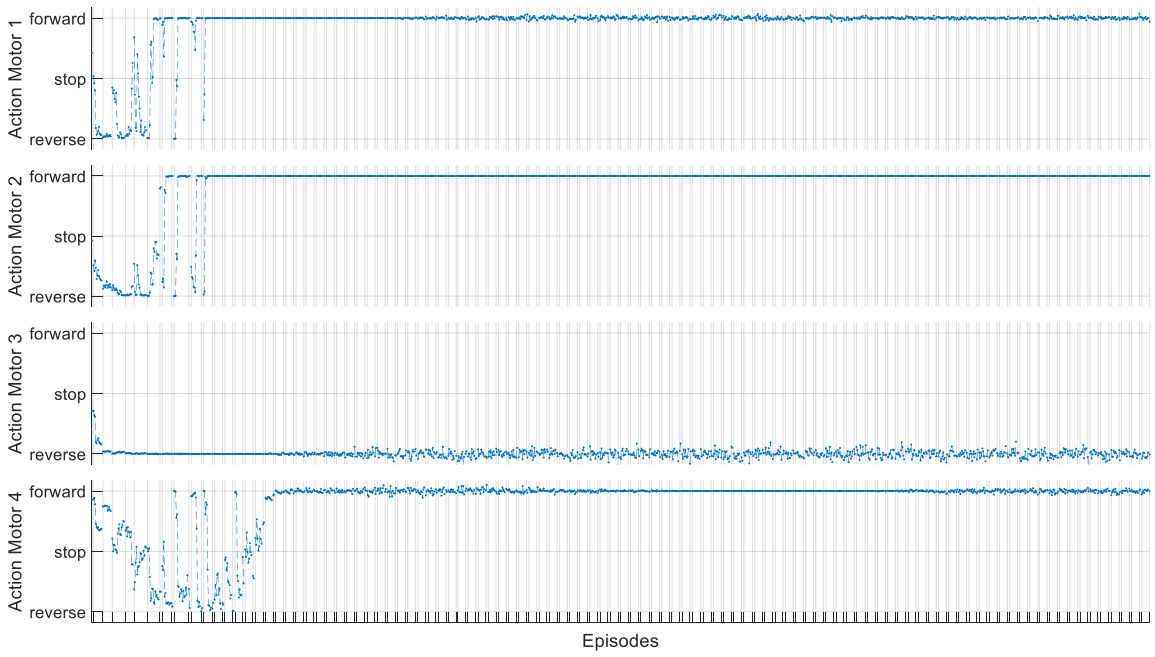


Figure 3.18 Saturation of actions when training with Policy Gradient

For motor 1, 2 and 4, the agent always closes the finger independently of the input. It also can be observed that the first episodes (< 50) the agent tries different actions but afterwards it stabilizes without considering if it is a flexion or extension episode; this behavior was observed up until 2000 episodes. Variations with baseline, no baseline gave similar results; variating the learning rate resulted in accelerating or delaying this process.

- **Deep Q-Learning with Experience Replay** was the algorithm that overcame the saturation problem obtained with PG. For the rest of this report this was the algorithm used with an experience buffer of size 10 000.

Deep Q-Learning requires to be defined a parameterized function to be its critic. In our case, shallow neural network architectures will be tested as critic.

3.10.3 Training Stage 1: simulation pretraining

Using the simulation model of the prosthesis, described above, several simulations in parallel were executed. Different hyperparameters (such as learning rate, discount factor, batch size, etc.) were tested. Further details of the procedure for hyperparameters selection are described in Section 4.1. Pretraining in simulation allowed us to test a larger number of agents and combinations compared with training directly on hardware. A rough estimate is that simulation accelerated training times by a factor of 12. The desktop computer was an AMD FX-8370 at 4 GHz with 16GB of RAM and 8 cores. Around 4 different training instances could run simultaneously on this machine.

3.10.4 Training stage 2: hardware fine tuning

The stage 2 of the training corresponds to the hardware fine tuning. The best model from pretraining will be retrained on hardware and with the operator. The fine tuning is important as the simulation is an approximation of the real behavior of the prosthesis. Learning rate must be reduced to make small adjustments in the weights of the critic network. The exploration factor ε also was carefully reset at the beginning of the fine tuning. Further details of this are described in Section 4.2.

CHAPTER 4:
EXPERIMENTATION AND RESULTS

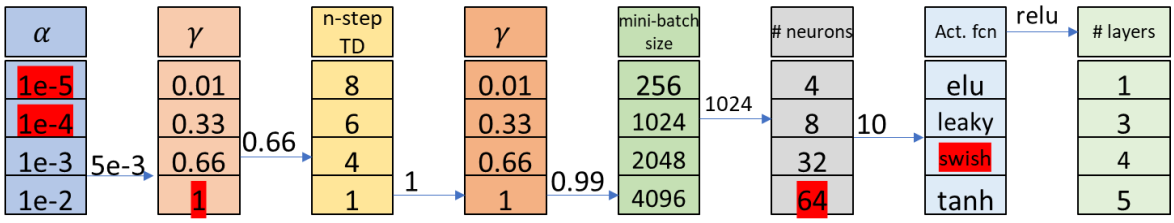
In this Chapter will be described the systematic process for hyperparameter selection, the fine tuning in hardware, and the evaluation of the trained agent.

4.1 Hyperparameters selection

This prosthesis prototype contains several hyperparameters; a few have been set beforehand whether based on the literature or based on preliminary experiments. But for most of it, an iterative procedure was followed to find a suitable set of hyperparameters. On each iteration of this procedure, 4 training simulations were carried on, varying only a single hyperparameter. The number of training simulations (4) where the maximum number of parallel instances that could be run in the personal desktop computer (view Section 3.10.3).

The details of the procedure of hyperparameters selection are described in Table 4.1 (part I) and Table 4.2 (part II). On top of each column is demarked the hyperparameter to test, and below are shown its different values. Highlighted on red are the values for which the performance was *notably worst*, (in this whole procedure there was not any value with notably best performance). Connecting each column is the value chosen from each iteration.

Table 4.1 Procedure of hyperparameters selection (part I)



Where:

$\boldsymbol{\alpha}$ is the learning rate for the training algorithm of the shallow neural network.

$\boldsymbol{\gamma}$ is the discount factor used to calculate the discounted reward.

$\boldsymbol{n_{step}\ TD}$ is the number of future rewards used to estimate the value of the policy during the critic's network update.

$\boldsymbol{mini\text{-}batch\ size}$ is the size of random experience sampled to execute experience replay.

$\#_{\boldsymbol{neurons}}$ is the number of neurons in the hidden layer of the critic's network.

act_{fcn} is the activation function used on the critic's network.

$\#_{layers}$ is the number of hidden layers. For this experiment, the number of weights was maintained close to 1500.

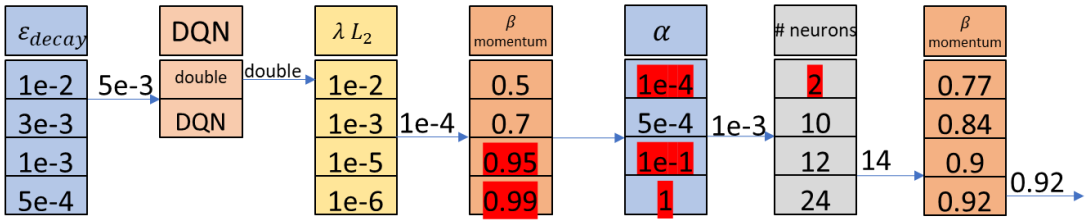
ε is the initial probability threshold to choose a random action instead of the agent's considered *best* action. The epsilon-greedy strategy used reduces the ε by a rate of 0.005 at the end of each training step, up to the minimum value of 0.01.

DQN represents whether the reinforcement algorithm is Deep-Q learning or Double deep Q learning.

λ is the weight decay for L_2 regularization.

β is the momentum for the SGDM optimizer (stochastic gradient descent with momentum).

Table 4.2 Procedure of hyperparameters selection (part II)



It is interesting to note the performance of the hyperparameters, in some cases the same hyperparamter changed its behavior after calibrating another one. For instance, in Table 4.1, $\gamma = 1$ had a *notably* bad performance, but after changing the n-step TD parameter, $\gamma = 1$ improves its performance. In the case where there was not a clear improvement the default value was prefered.

To select the hyperparameter during this iterative process, 2 evaluation metrics where considered: the agent reward and the RMSE between the target and prosthesis position. During training is expected that the episode reward increases over time meaning that it learned; at the same time, the RMSE is expected to decrease implying a closer trajectory of the prosthesis position to the target. In Figure 4.1 is compared the average (between 40 consecutive episodes) agent reward during varying the learning rate on each training, and in Figure 4.2 is shown the RMSE for the same experiment. As can be observed, RMSE and agent reward are complementary, in the sense that when one increases the other tend to decrease, and viceversa. But this relation is not linear and thus both curves were used to tune the hyperparameter.

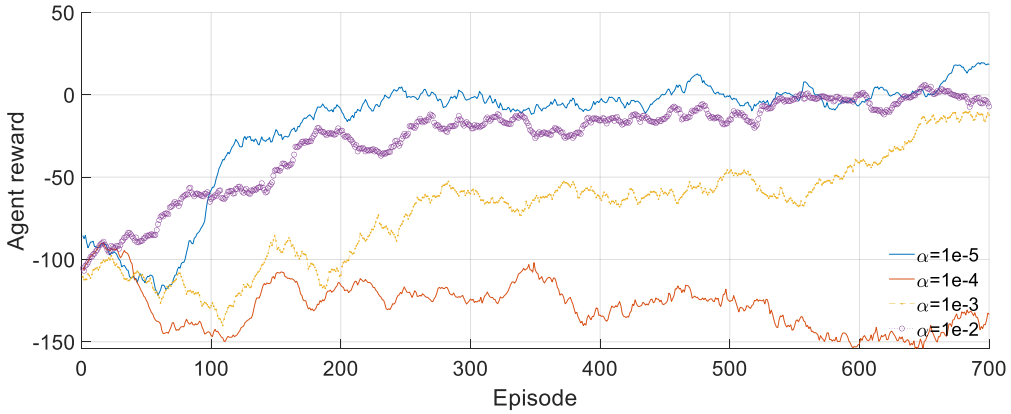


Figure 4.1 Agent training varying learning rate (reward)

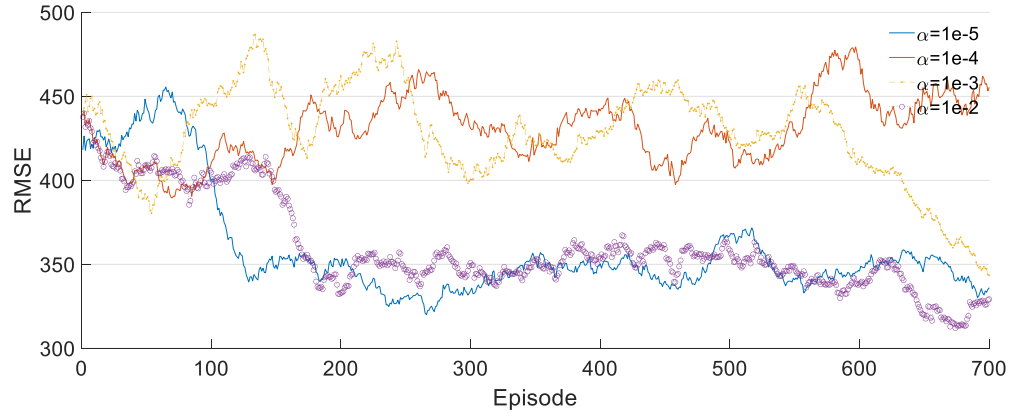


Figure 4.2 Agent training varying learning rate (RMSE)

For most hyperparameters, the different values tested had a similar performance. For instance, in the case of activation function (view Figure 4.3) the functions *relu, elu, leaky rely* at the end of the training oscillated around a similar value. With respect to the activation function *tanh* can be argued that it had a lower performance. Only the *swish* activation function shown a significantly worse performance and thus was discarded. The rest of hyperparameters were tuned for the Stage 1 of training in simulation following an alike procedure.

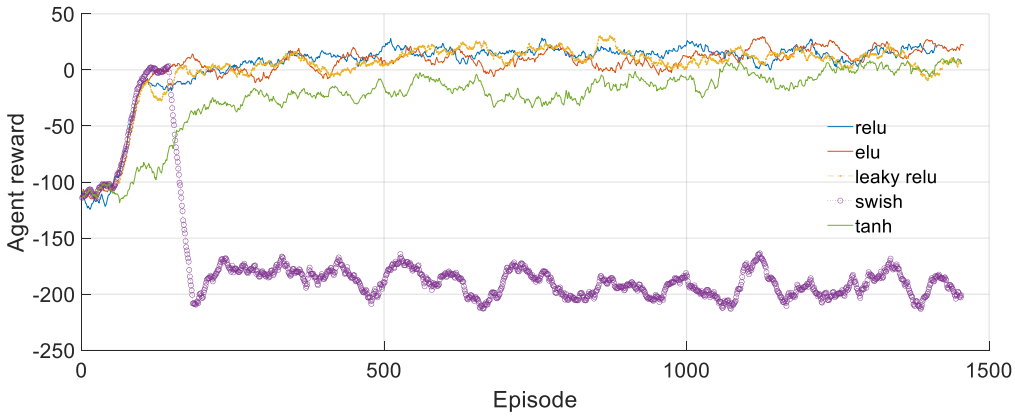


Figure 4.3 Agent training varying activation function (reward)

Using the best combination of hyperparameters found, the agent was trained for 1400 episodes, in the Agent Server for approximately 1 hour 30 minutes. In Figure 4.4 is shown the average reward during training for each episode. As can be seen, most of the learning occurs in the first 200 episodes, after it, the agent improves more slowly its policy. A similar behavior can be observed if the training progress is divided by type of movement (Figure 4.5). It is interesting to note that the agent has a slightly lower reward when closing the hand. This can be because the force required for closing the prosthesis hand makes it harder to follow, lagging behind and hence obtaining a lower reward.

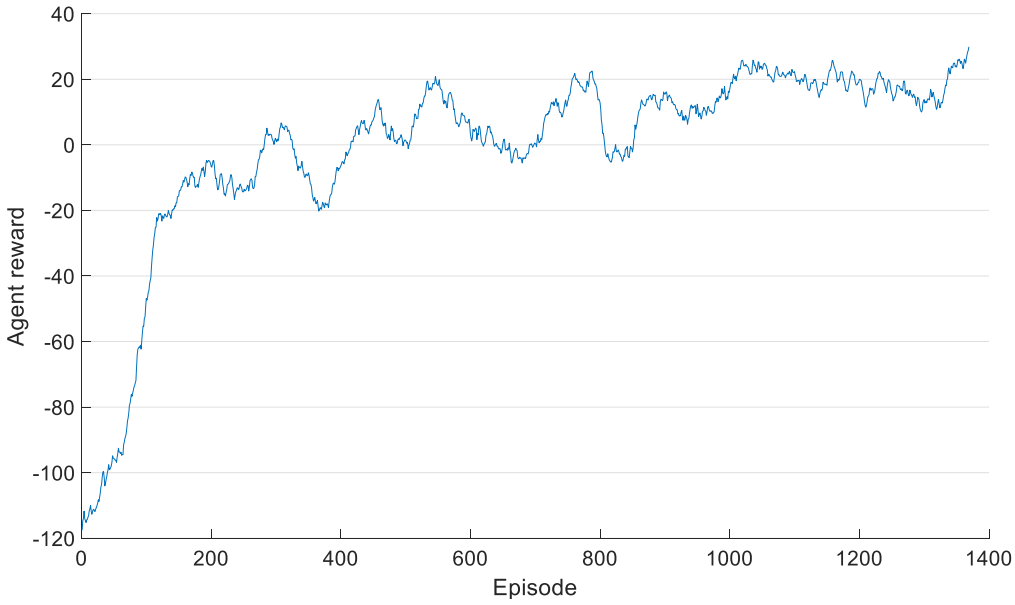


Figure 4.4 Training progress in simulation

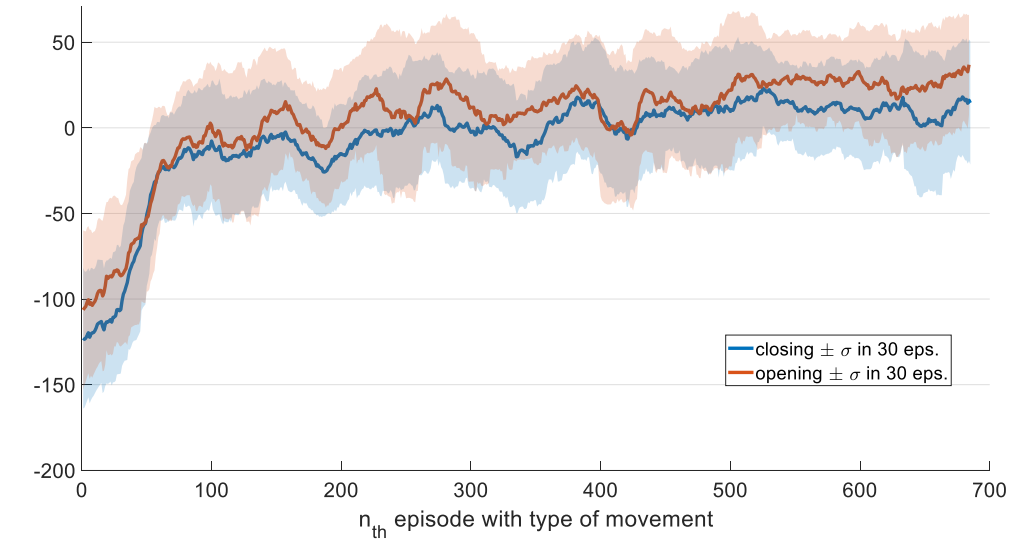
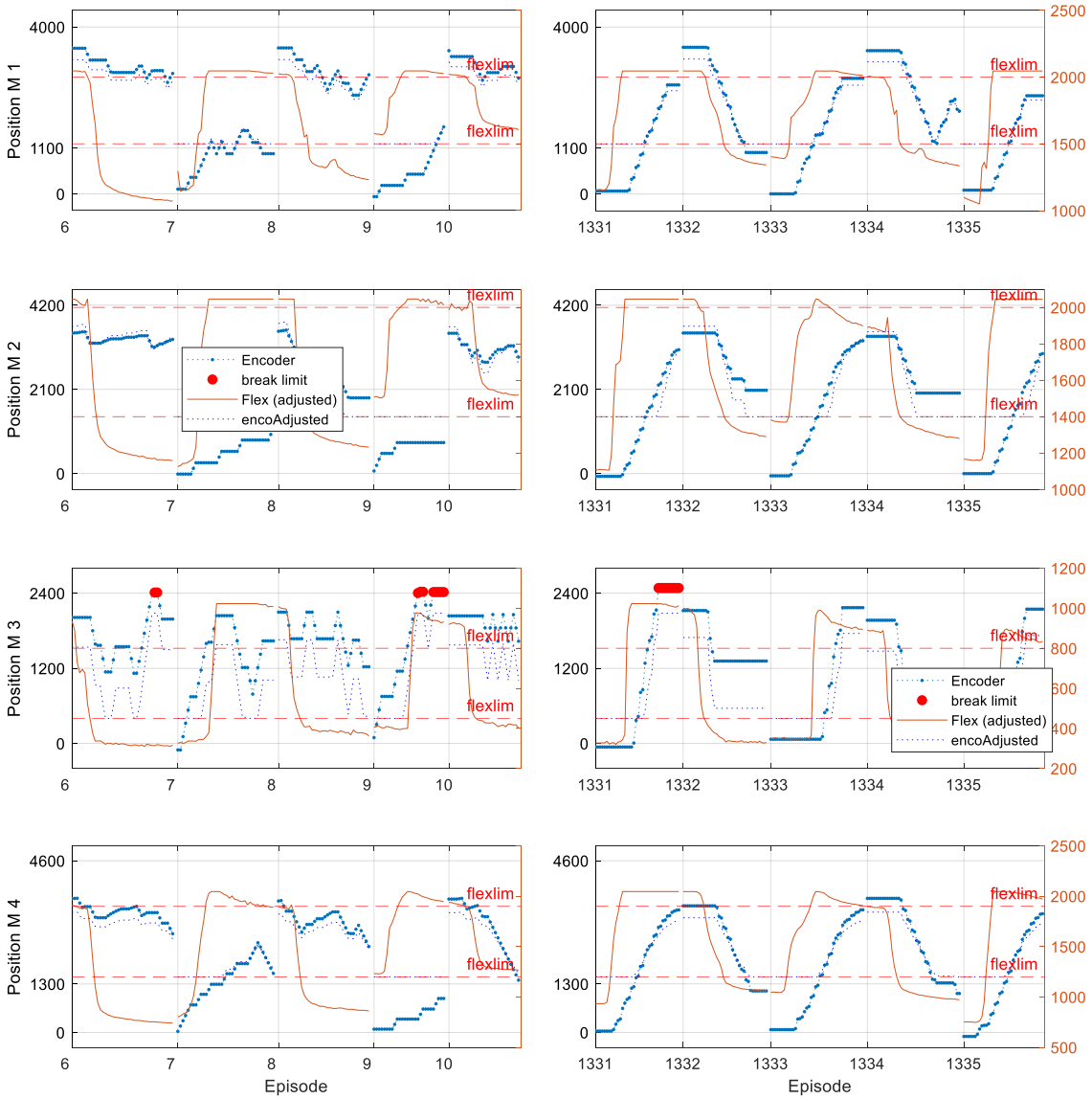


Figure 4.5 Rewards during training by type of movement

The performance of the agent at the beginning and at the end is displayed in Figure 4.6. In this Figure, the red line corresponds with the target flexion signal (right y axis) on each

Degree of Freedom, and the blue line corresponds to the encoder position (left y axis). The agent at the beginning of the training (Figure 4.6 a, episodes 6 to 10) executes random movements, uncoordinated between Degrees of Freedom.



a) beginning of training **b) end of training**

Figure 4.6 Agent performance during training

As was mentioned in Section 3.8, odd episodes were loaded with the flexion movement, while the even episodes have the extension movement. After approximately 1330 training episodes (Figure 4.6 b, episodes 1331 to 1335), the agent follows the target flexion signal as expected. It is important to note that the finger movement is delayed with respect to the target, it happens because the agent starts its motion delayed and because it has a fixed speed. The delayed start may be due to the agent needs some context before deciding itself to move, as well as the system's transmission delay. The second reason is that the agent does not have speed control, it only has an action to move forward with a constant speed,

and therefore there is a limitation on how fast the prosthesis can follow. The only visible error in these episodes is in episode 1334, motor 1, where the motor is opening correctly the hand, but suddenly reverses direction, but it only happened in 1 of the 3 DoF. In the rest of episodes, the agent has a satisfactory behavior. Although, in some cases (episode 1334, DoF 2), it might seem that the motor stopped before reaching the desired position, it is a false impression occasioned when the motor reaches the loose zone (view Section 3.8).

4.2 Fine tuning

The Fine tuning was the second stage of the training process. It consisted of adjusting the agent (that was previously trained in simulation) to the prosthesis prototype. This stage is important because, most probably, the simulation might differ from the real hardware. For this matter, the agent is retrained in the hardware, choosing to this stage a new initial ε and learning rate.

During simulation, the exploration factor was at maximum ($\varepsilon = 1$) for the agent did not have any knowledge and, hence, exploration was recommended. But, during hardware fine tuning, the agent already has some previous knowledge, and therefore the initial exploration can be reduced. To set this ε, an analysis of the reduction of the exploration factor was carried on. In Figure 4.7 is shown different decay rates and highlighted in green the default value of 5e-3. In approximately 50 episodes the agent reaches its minimum value of 0.01 (1% percent of the actions).

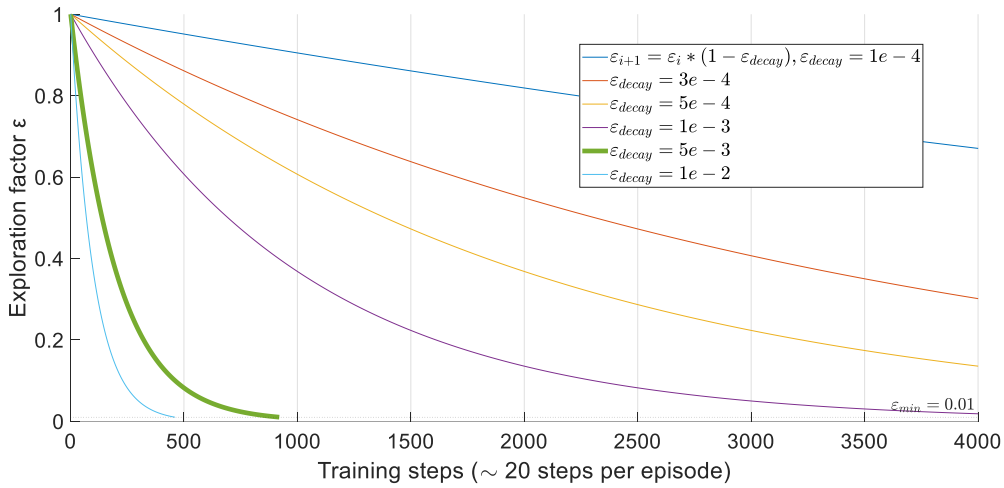


Figure 4.7 Decay rate in epsilon-greedy exploration

With an initial epsilon of 0.3, different training sessions on hardware were executed varying the learning rate as shown in Figure 4.8. As can be observed, the lower learning rate tried (α=1e-5) was the value that achieved good performance and will be used for the agent

evaluation. Greater values of learning rate did not show any learning over time. Each of these experiments took approximately 30 minutes.

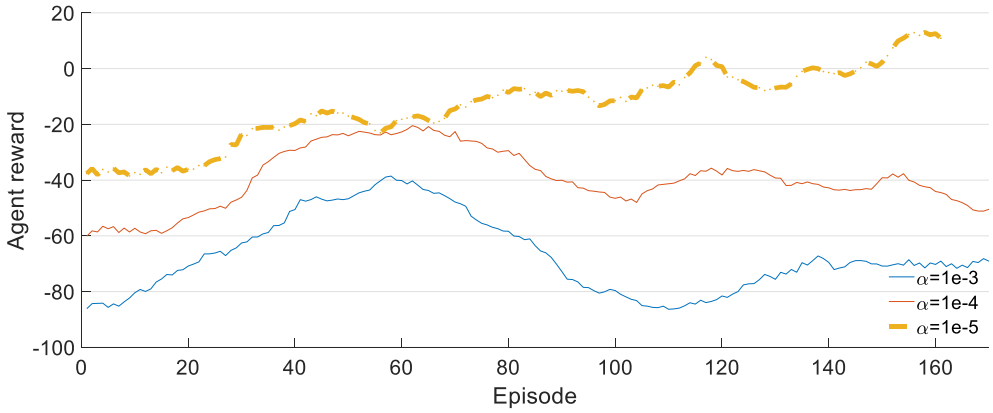


Figure 4.8 Training progress during fine tuning in hardware

The performance curves related with this trained agent (view Figure 4.10) will be analyzed in the following Sections.

4.3 Sample size estimation for testing

The agent performance evaluation is based on the success rate on the execution of the whole-hand grasp and release movements. This evaluation required to record the agent performing these movements, called opening and closing. To obtain a significant result, a power analysis was conducted to determine the sample size. For this matter, a t-test was selected for sample size estimation, as it allows us to probe that there exists a significant difference between the mean of two different groups. To compute the sample size is required selecting an effect size. The effect size d is a measure of the magnitude of the assumed effect. For the t-test selected, the effect size is described with the z-score by Equation (8), where μ_i is the mean of the i_{th} variable and σ is the standard deviation, either of the two groups, or the average between the two. Cohen stablished the following standardized convention: small effect $d = 0.2$, medium effect $d = 0.5$, large effect $d = 0.8$ (Faul et al., 2007). In Figure 4.9 is plotted different effect sizes, as can be observed the greater the effect size the larger the distance between the means of the two variables.

$$d = \frac{\mu_1 - \mu_2}{\sigma} \tag{8}$$

To calculate the sample size, it was defined that the first group will be the RMSE between the target and the position of the prosthesis generated by the trained agent, and the second ground will be the RMSE generated by a random agent. The parameters used to estimate

sample size are: $\alpha = 0.05$ probability of false positives, $power: 1 - \beta = 0.95$ probability of false negatives, and $d = 1$ as effect size. This way, the statistical test will probe that there is at least 1 standard deviation of improvement between the means of the random agent and the trained agent. Using these values was stablished that 25 samples per group will be required.

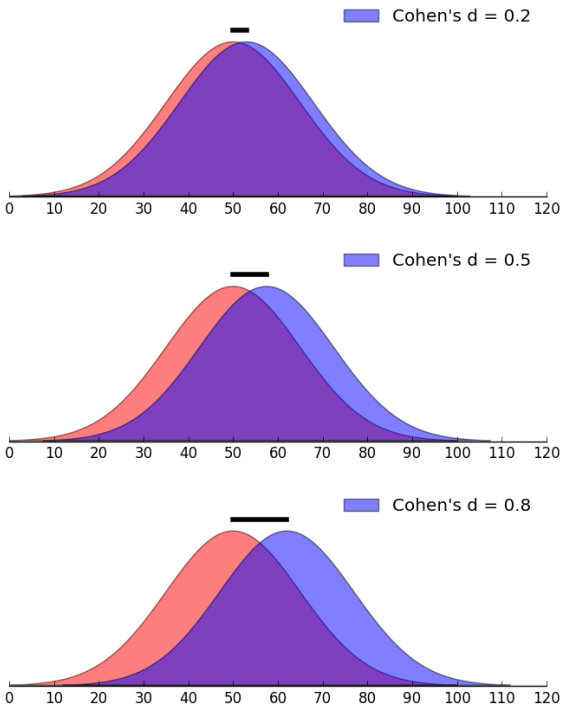


Figure 4.9 Cohen's effect size convention

4.4 Agent performance

Using the trained agent to operate the prosthesis, samples for evaluation were recorded of the opening and closing movements . In Figure 4.10 is showed some of these samples. In similar way than previous Sections, odd episodes correspond to the movement of closing the prosthesis hand, and the even episodes in contrast to the opening of the prosthesis. For each degree of freedom is plotted the desired flexion target (red) and the prosthesis trajectory (blue). It can be observed that in all samples the agent goes to the right direction, albeit in some cases it stops halfway. It is interesting to note that the agent starts to follow the target always delayed and with the same slope. The delay for the most samples can be considered acceptable. As discussed before, this can be due to transmission delays and processing times. The same slope when following the target implies that the degree of freedom has a constant speed and faces the same resistance to movement. The constant speed is a result of the agent's action definition. This has the advantage of a small

architecture but limits the rate at which the agent can follow the target. In rare cases, when opening the hand, for a brief moment, the agent tries to close it further before doing the correct action. This is not an error at all, and it most related with the division between episodes. In general, these results are very similar with the obtained in simulation, meaning that the model of the dynamics of the prosthesis —altogether with the fine-tuning stage— were correctly carried on.

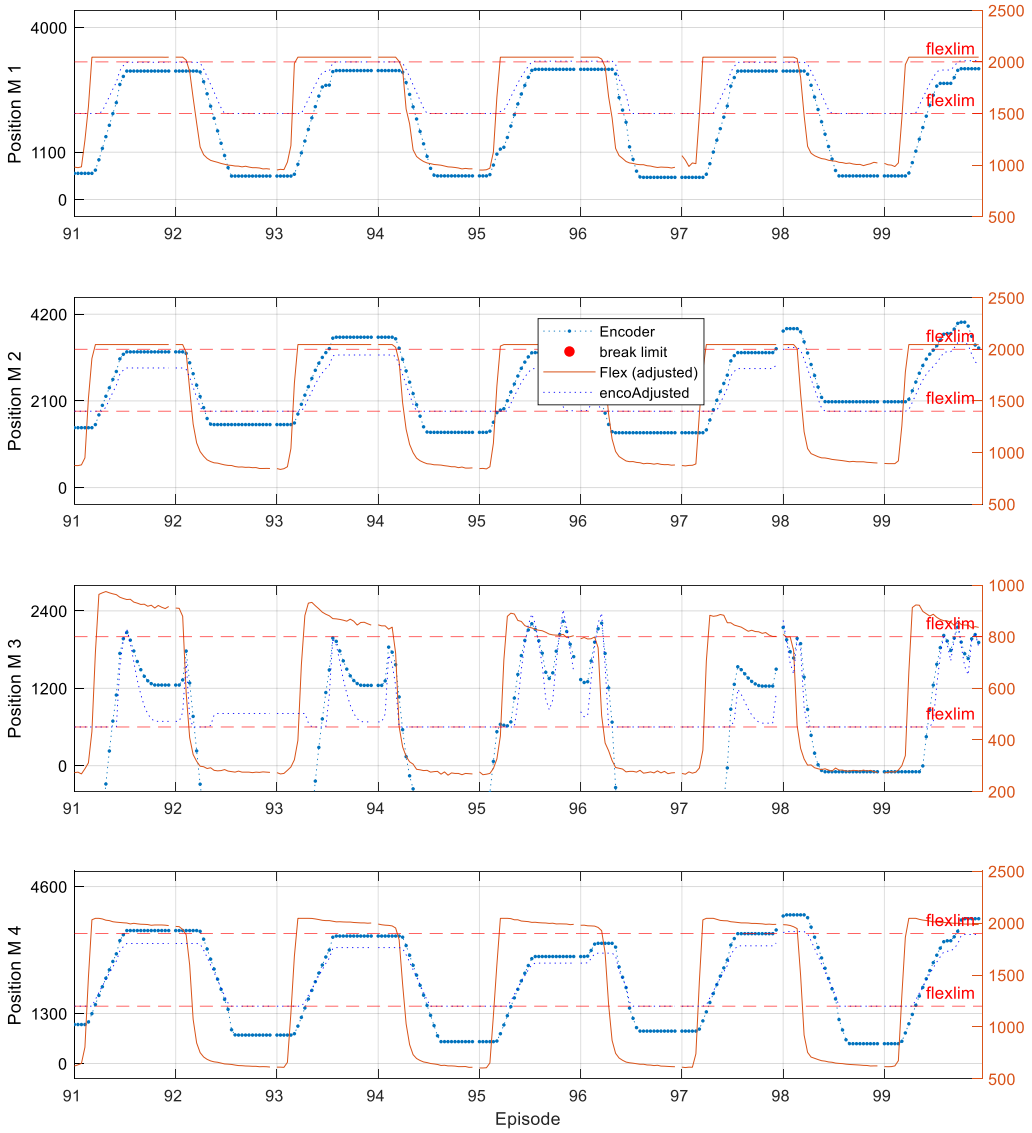


Figure 4.10 Performance of the trained agent by degree of freedom

The biggest concern in the results is related with the 3rd degree of freedom (i.e., the thumb). It shows an oscillation when closing the hand, always when reaching the flex limit. We consider this is due to the motor having a lower torque than required. As can be observed, the agent –for the rest of fingers— learns to stop when reaching this limit. And because they have enough torque, they can withstand the flexion force without retracting. But because

the thumb (by design) requires a larger torque, the motor cannot stay in that position. A motor upgrade could solve this problem but might require a readaptation of the hardware, larger battery capacity, etc. Even in this case, we consider the agent learned the correct behavior; and this inconvenient helps us to identify areas for improvement.

4.5 Results discussion

In Figure 4.11 is presented the results obtained for the tasks of whole-hand grasping and releasing. In blue is represented the performance of the trained agent, and in red the random agent. For each agent was included the histogram with the distribution of samples that fall inside certain RMSE range. A better performance implies a lower RMSE, because the lower the RMSE means a closer prosthesis trajectory to the flexion target. The trained agent obtained 369.97 ± 55.79 RMSE, while the random agent obtained 511.67 ± 43.02 RMSE. It is interesting to note that the trained agent has a larger standard deviation. The obtained effect size —when using the larger standard deviation— is $d = 2.54$ (i.e., the means are apart 2.5 standard deviations). Although any agent obtained a RMSE distribution that resembled a normal distribution, we consider the results are still valid. It is clear that for the majority of samples, the trained agent had a better performance.

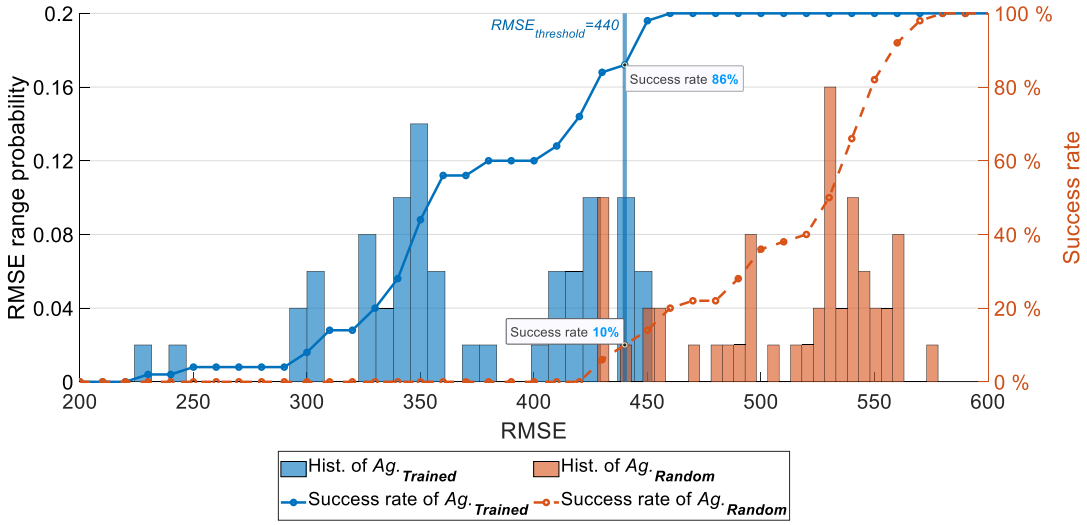


Figure 4.11 Performance of the trained agent vs the random agent

To determine the success rate an RMSE threshold of 440 was established. This way, a closing or opening sample with an RMSE value lower than or equal to it will be consider as correct. With this threshold the success rate for the trained agent was 86% and for the random agent 10%. It is important to note that this value is subjective, and the success rate of the agents will change for a different RMSE threshold. As was noted in the previous section, for some samples the trained agent did not complete the trajectory and stayed at a

halfway position. The RMSE threshold was selected to consider a sample incorrect if it is notably far from its desired final destination. Based on these results we can conclude that the trained agent —and the research project overall— met with its desired goals.

In Summary, the agent that operates the prosthesis was evaluated using the success rate of closing and opening the hand. The t-test for difference between the mean of 2 groups was used to calculate the sample size. The groups are the performance of the trained agent and of a random agent. The corresponding recordings were taken and a threshold for RMSE was selected. With it was calculated the success rate that is presented in Table 4.3.

Table 4.3 Evaluation results of the trained agent

Parameter	Value
Sample size	50
Effect size	2.54
RMSE	369.97 ± 55.79
RMSE threshold	440
Success rate	**86%**

CHAPTER 5:
CONCLUSSION

5.1 Conclusions

The results of this research demonstrate that a myoelectric hand prosthesis was trained successfully to follow the movements of opening and closing the hand (flexion extension). To achieve this goal, a prosthesis model with flexible materials was chosen from the literature, modified to meet our requirements and was built including its driving circuitry. This prothesis resulted in a low-cost 3D printed prototype with 4 degrees of freedom, 1 for each finger but the little. The training of the agent that operates the prosthesis was divided in two stages, simulation and fine-tuning. Simulation training allowed a fast proof of concept test and hyperparameters selection. Fine-tuning was necessary to adapt the agent to the real hardware. The performance of the prosthesis was measured using the success rate of the grasp and release tasks obtaining an 86%. From this, we conclude that this research was a successful first step towards achieving a fully functional myoelectric hand prosthesis.

The implementation of a simulation environment for the prosthesis allowed a faster experimentation and tuning of different hyperparameters. It speeded up the training time by a factor of 12 and reduced the prosthesis wear. Although the prosthesis dynamics had a simplified model —and was required a fine-tuning stage— the results showed that the simulation model met its goal.

The degree of freedom in this research corresponded to flexible fingers 3D printed based on an open-source model available in the literature. The authors of these models stated that the fingers can withstand 45 000 cycles without degradation. But in our experience, at least one joint of the fingers but the thumb suffered some sort of damage that needed to be completely replace during development. Differences in the fabrication process (quality of the TPU material, 3D printing parameters, etc.) might have led to this result. This has important implications for further research needed with respect to the usage of flexible materials in prosthesis building.

The reward function that obtained the best performance was the combination of 2 approaches: distance to the target and rewarding the chosen action. When they are used separately, the former was observed to be a noisy signal surely due to transmission delays and prosthesis reaction times. The later reward function learned to move in the correct

direction but in some episodes stopped halfway not reaching the desired position. The combination of both approaches obtained good performance with a fixed contribution of each part. This contribution was encapsulated in the constant c_i, enforcing a 0 reward when the agent selected the correct action but was located at the furthest possible distance. This parameter was not subject to exhaustive calibration, and hence can be further improved.

5.2 Recommendations

From the experience during the development of this project, we extend the following recommendations for any researcher interested in myoelectric prosthesis operation using reinforcement learning:

5.2.1 Improve prosthesis simulation

This project used a simplified model of the prosthesis dynamics. This allowed a faster testing of proof of concepts and hyperparameters tuning, but required some effort at the time of training in the prosthesis real hardware (fine-tuning). Improving this simulation model might help find a better set of hyperparameters for a better prosthesis performance. In general, we recommend implementing even a simple simulation environment when using reinforcement learning. We consider time spent in the coding of a simulation environment will pay off in a larger agent experience for training and an overall better performance.

5.2.2 Upgrade embedded microcontroller to increase reaction time

The materials selected for the construction of the prosthesis were prioritized to meet a fixed budget. This resulted in several compromises both in performance and capabilities. The one with the greatest impact is the usage of an Arduino Uno as the embedded controller. The usage of this component made the electronic design easier, as there is a commercial motor driver specifically designed for it (the Adafruit Motor Shield V1.0), with the disadvantage of a slow processing time. The reason for the 200 ms of the agent action period is to secure at least one measurement of the prosthesis position in the Agent Server. This made the reaction times of the prosthesis very slow. Upgrading this component implies a cost increase, and a redesign and validation of the electronic circuitry, but would bring significant performance improvements.

5.3 Future work

Although the objectives of this research were met, this does not mean that the research has finished. In the following paragraphs are discussed several aspects that can be improved:

5.3.1 Feature extraction specialized for prosthesis control instead of hand gesture recognition

This work uses a set of features previously validated in the area of hand gesture recognition. Although the fields of prosthesis operation and hand gesture recognition using EMG are related in numerous ways, they are not the same. It can be argued that prosthesis control is a harder problem than hand gesture recognition. We state this, because prosthesis control and hand gesture recognition require the prediction of movement intention in real time, but prosthesis control require additionally speed and force estimation. This opens the possibility that the sets of functions for feature extraction had a different performance in these two fields. Finding the best set of features in a given problem is not a trivial task, as it is a combinatorial problem that increases exponentially with the number of possible options.

Another possibility in this respect is the inclusion of automatic feature extraction methods such as Convolutional Neural Networks or Autoencoders. A comparison of both approaches would reveal helpful insights for achieving a better performance of myoelectric prosthesis.

5.3.2 Deploying the agent in the embedded controller of the prosthesis

In this research, the agent was deployed in a personal desktop computer called the Agent Server. This allowed us a fast development and training, as well as a reduction in the load of the embedded controller. But, the inclusion of a second processing unit implied the sacrifice of portability, that is fundamental for a fully functional prosthesis. Deploying the agent in the embedded microcontroller of the prosthesis would solve this problem, and would also reduce the latency, improving the time response of the actuators and making it completely portable.

Running the agent in the prosthesis controller would limit the size and complexity of the agent compared to the possibilities available when using a personal desktop computer. It also would increase the energetic demand of the entire system, reducing its run time, or imposing the need of a battery upgrade, increasing its weight. A possible approach would be the usage of microcontrollers with AI modules, such as the Kendrite K210, NVIDIA Jetson Nano or alike.

5.3.3 Smooth prosthesis control

One of the biggest caveats of this research is that it does not have speed control. It means that the prosthesis cannot perform fast or slow movements, as its speed is fixed. It was observed that in these cases, the agent opted for moving and stopping in an attempt to emulate different moving speeds. It is clear the disadvantages of this situation. A possible solution is to increase the agent period to send more frequently the actions. If this period is increase significantly, a speed control could be attained. A more feasible solution would be instead directly including speed variations in the actions that the agent can perform. This implies increasing the size and complexity of the critic network. A larger size might not be a great obstacle as the network has a very small size. But increasing the number of actions —or using a continuous action space— would requires further research.

BIBLIOGRAPHY

Barona López, L. I., Valdivieso Caraguay, Á. L., Vimos, V. H., Zea, J. A., Vásconez, J. P., Álvarez, M., & Benalcázar, M. E. (2020). An Energy-Based Method for Orientation Correction of EMG Bracelet Sensors in Hand Gesture Recognition Systems. *Sensors*, *20*(21), 6327.

Benalcázar, M. E., Barona, L., Valdivieso, L., Aguas, X., & Zea, J. (2020). *EMG-EPN-612 Dataset*. Zenodo. https://doi.org/10.5281/zenodo.4421500

Castellini, C. (2019). Upper limb active prosthetic systems-overview. In *Wearable Robotics: Systems and Applications* (pp. 365–376). Elsevier. https://doi.org/10.1016/B978-0-12-814659-0.00019-9

Côté-Allard, U., Gagnon-Turcotte, G., Laviolette, F., & Gosselin, B. (2019). A low-cost, wireless, 3-D-printed custom armband for semg hand gesture recognition. *Sensors (Switzerland)*, *19*(12), 2811. https://doi.org/10.3390/s19122811

ElKoura, G., & Singh, K. (2003). Handrix: Animating the Human Hand. *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 110–119.

Englehart, K., & Hudgins, B. (2003). A robust, real-time control scheme for multifunction myoelectric control. *IEEE Transactions on Biomedical Engineering*, *50*(7), 848–854.

Estrada Jiménez, L. A. (2016). *Diseño e implementación de un prototipo para la traducción de lenguaje de señas mediante la utilización de un guante sensorizado*. https://bibdigital.epn.edu.ec/handle/15000/16712

Fajardo, J., Ferman, V., Cardona, D., Maldonado, G., Lemus, A., & Rohmer, E. (2020). Galileo hand: An anthropomorphic and affordable upper-limb prosthesis. *IEEE Access*, *8*, 81365–81377. https://doi.org/10.1109/ACCESS.2020.2990881

Farina, D., Jiang, N., Rehbaum, H., Holobar, A., Graimann, B., Dietl, H., & Aszmann, O. C. (2014). The extraction of neural information from the surface EMG for the control of upper-limb prostheses: Emerging avenues and challenges. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *22*(4), 797–809.

Faul, F., Erdfelder, E., Lang, A.-G., & Buchner, A. (2007). G*Power 3: a flexible statistical

power analysis program for the social, behavioral, and biomedical sciences. *Behavior Research Methods*, *39*(2), 175–191. https://doi.org/10.3758/bf03193146

George, J. A., Davis, T. S., Brinton, M. R., & Clark, G. A. (2020). Intuitive neuromyoelectric control of a dexterous bionic arm using a modified Kalman filter. *Journal of Neuroscience Methods*, *330*, 108462. https://doi.org/10.1016/j.jneumeth.2019.108462

Hallworth, B. W., Austin, J. A., Williams, H. E., Rehani, M., Shehata, A. W., & Hebert, J. S. (2020). A Modular Adjustable Transhumeral Prosthetic Socket for Evaluating Myoelectric Control. *IEEE Journal of Translational Engineering in Health and Medicine*, *8*. https://doi.org/10.1109/JTEHM.2020.3006416

Jaramillo-Yánez, A., Benalcázar, M. E., & Mena-Maldonado, E. (2020). Real-Time Hand Gesture Recognition Using Surface Electromyography and Machine Learning: A Systematic Literature Review. *Sensors*, *20*(9), 2467. https://doi.org/10.3390/s20092467

Kukker, A., & Sharma, R. (2018). Neural reinforcement learning classifier for elbow, finger and hand movements. *Journal of Intelligent \& Fuzzy Systems*, *35*(5), 5111–5121.

Manero, A., Smith, P., Sparkman, J., Dombrowski, M., Courbin, D., Kester, A., Womack, I., & Chi, A. (2019). Implementation of 3D Printing Technology in the Field of Prosthetics: Past, Present, and Future. *International Journal of Environmental Research and Public Health*, *16*(9), 1641. https://doi.org/10.3390/ijerph16091641

McDonald, C. L., Westcott-McCoy, S., Weaver, M. R., Haagsma, J., & Kartin, D. (2020). Global prevalence of traumatic non-fatal limb amputation. *Prosthetics and Orthotics International*, 0309364620972258. https://doi.org/10.1177/0309364620972258

Mohammadi, A., Lavranos, J., Zhou, H., Mutlu, R., Alici, G., Tan, Y., Choong, P., & Oetomo, D. (2020). A practical 3D-printed soft robotic prosthetic hand with multi-articulating capabilities. *PLOS ONE*, *15*(5), 1–23. https://doi.org/10.1371/journal.pone.0232766

Pizzolato, S., Tagliapietra, L., Cognolato, M., Reggiani, M., Müller, H., & Atzori, M. (2017). Comparison of six electromyography acquisition setups on hand movement classification tasks. *PloS One*, *12*(10), e0186132–e0186132. https://doi.org/10.1371/journal.pone.0186132

Rahman, A., & Al-Jumaily, A. (2013). Design and Development of a Bilateral Therapeutic

Hand Device for Stroke Rehabilitation. *International Journal of Advanced Robotic Systems*, *10*(12), 405. https://doi.org/10.5772/56809

Seok, W., Kim, Y., & Park, C. (2018). Pattern recognition of human arm movement using deep reinforcement learning. *2018 International Conference on Information Networking (ICOIN)*, 917–919.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (Second). The MIT Press. http://incompleteideas.net/book/the-book-2nd.html

Vasan, G., & Pilarski, P. M. (2017). Learning from demonstration: Teaching a myoelectric prosthesis with an intact limb via reinforcement learning. *IEEE International Conference on Rehabilitation Robotics*, 1457–1464. https://doi.org/10.1109/ICORR.2017.8009453

Weiss, L. D., Weiss, J. M., & Silver, J. K. (2015). *Easy EMG: A Guide to Performing Nerve Conduction Studies and Electromyography*. Elsevier.

Zea, J., Benalcázar, M. E., Barona López, L. I., & Valdivieso Caraguay, Á. L. (2021). An Open-Source Data Acquisition and Manual Segmentation System for Hand Gesture Recognition based on EMG. *ETCM 2021 - 5th Ecuador Technical Chapters Meeting*. https://doi.org/10.1109/ETCM53643.2021.9590811

APPENDIX

Appendix I Forearm and socket blueprints

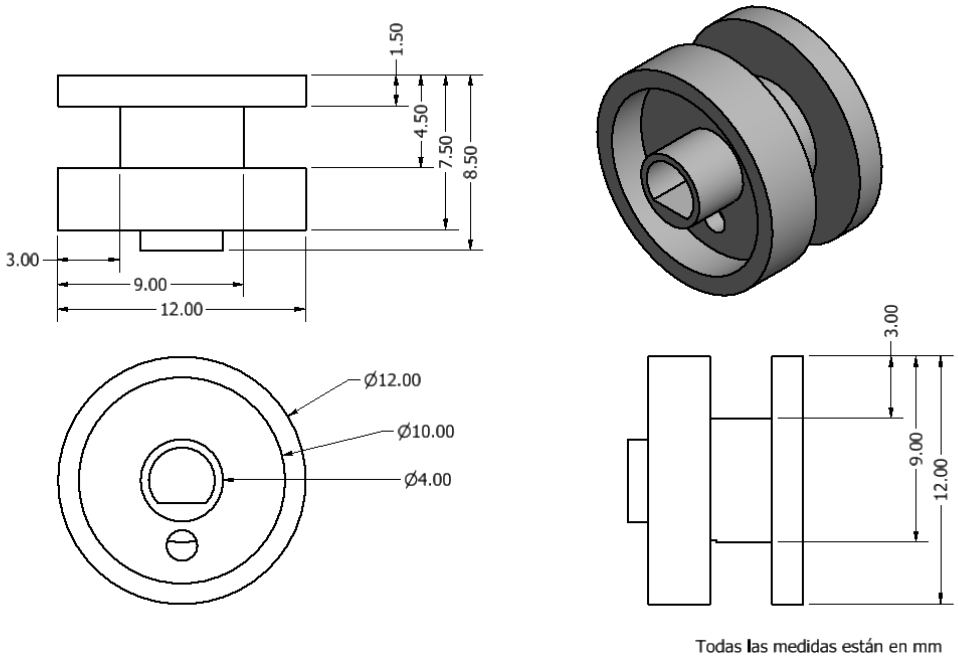


Figure 5.1 Spool dimensions

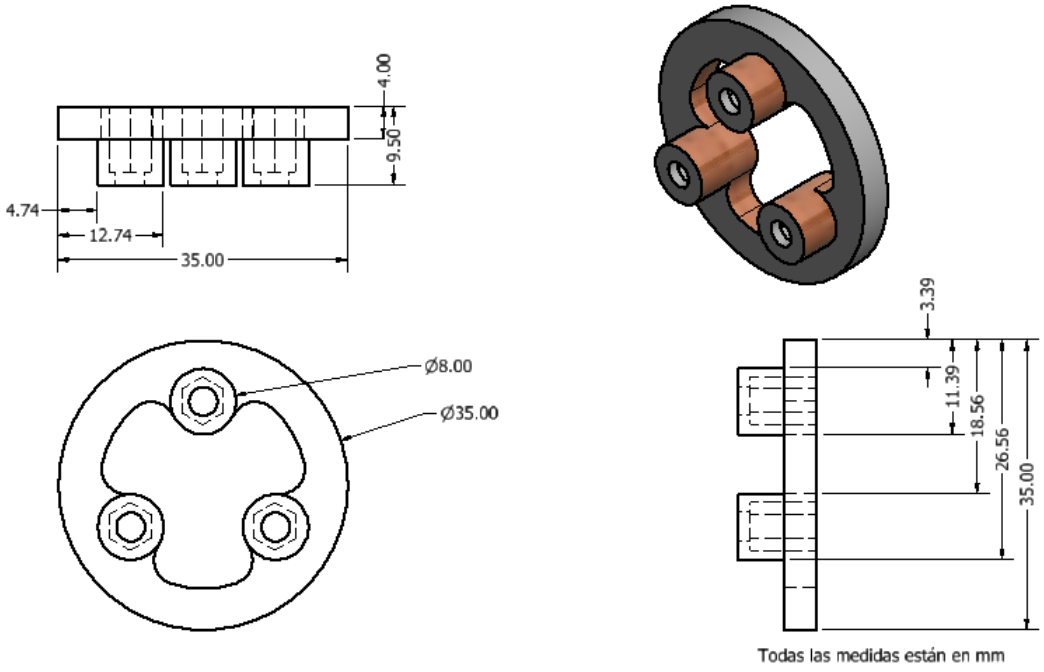


Figure 5.2 Palm connector dimensions

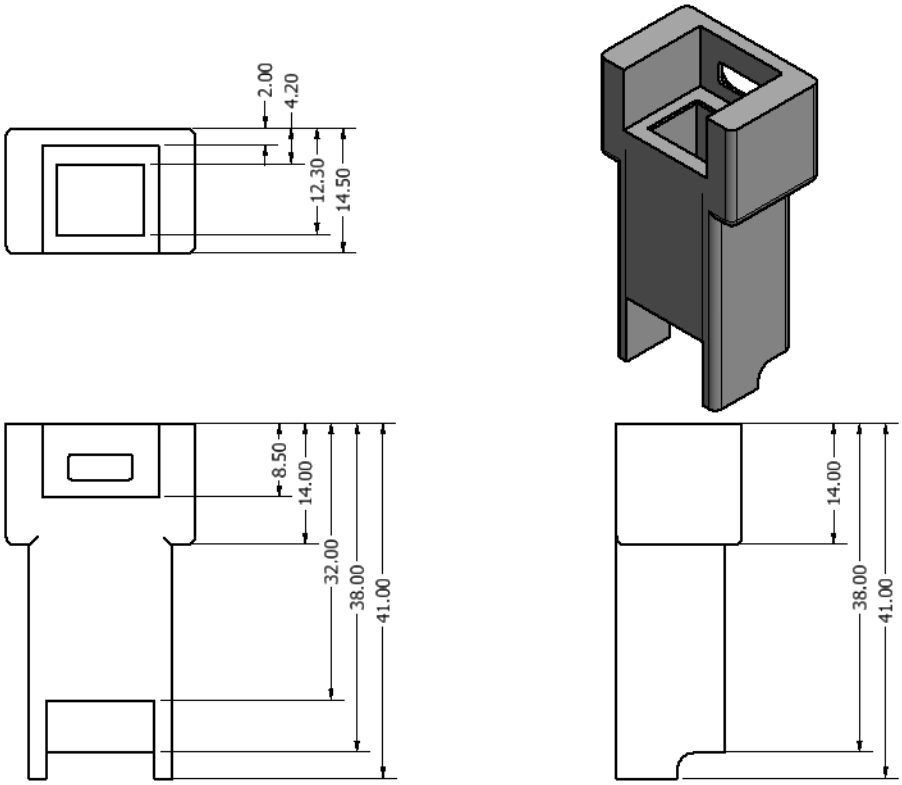


Figure 5.3 Case thumb dimensions

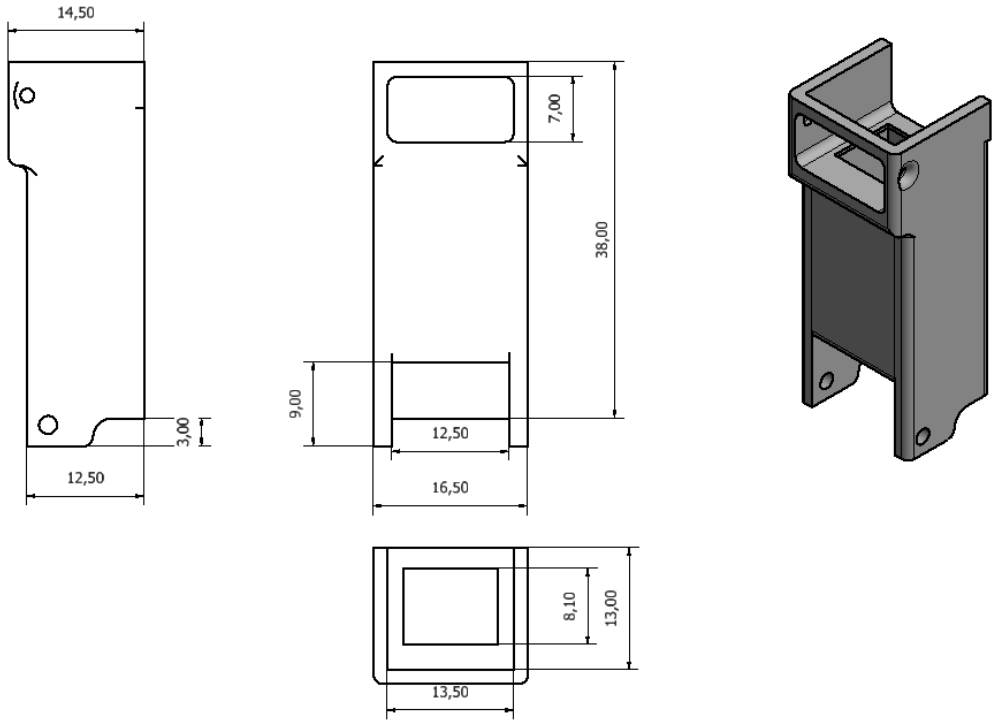


Figure 5.4 Motor case dimensions

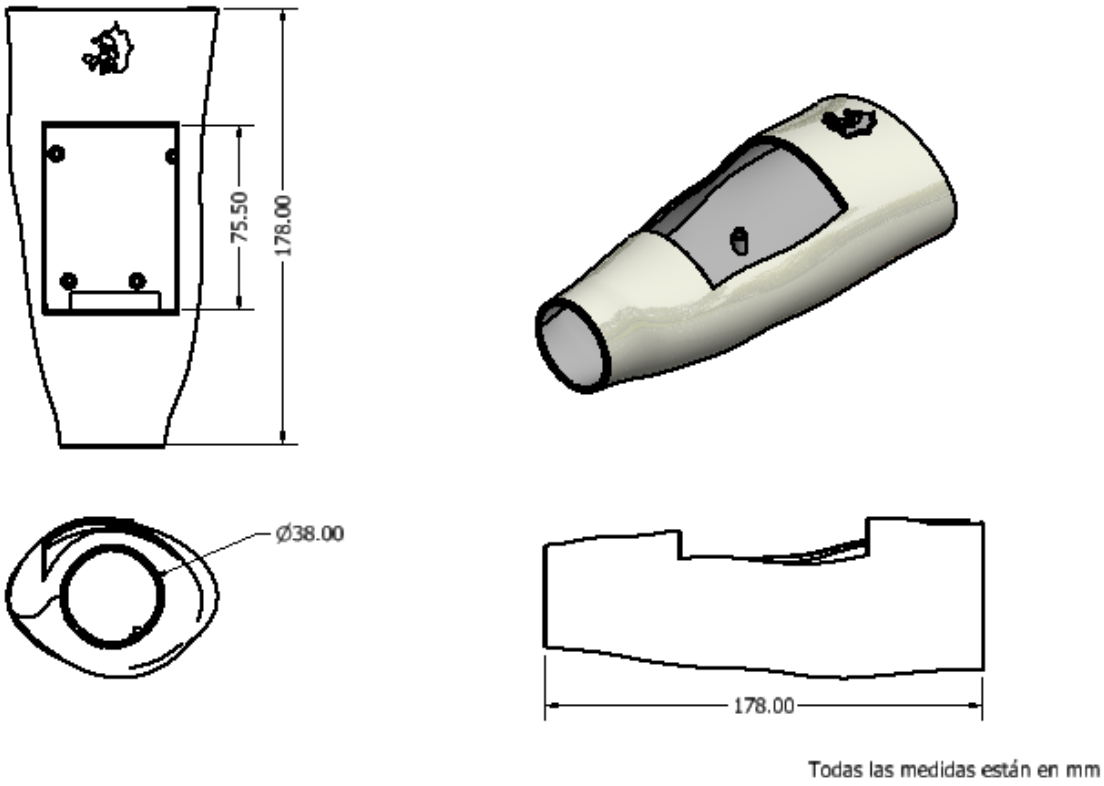


Figure 5.5 Forearm dimensions

Appendix II Electrical connections

Table 5.1 Connections mapping motor to Arduino controller

#	finger	pin	AF-motor shield v1.1
1	-	VCC	VCC
2	-	GND	GND
3	thumb	M3 -	M3 +
4	thumb	M3 +	M3 -
5	thumb	M3 A	A5
6	thumb	M3 B	A4
7	middle	M4 -	M4 -
8	middle	M4 +	M4 +
9	middle	M4 A	gpio 13
10	middle	M4 B	gpio 2
11	index	M2 B	A1
12	index	M2 A	A0
13	index	M2 +	M2 +
14	index	M2 -	M2 -
15	little	M1 B	A3
16	little	M1 A	A2
17	little	M1 +	M1 +
18	little	M1 -	M1 -

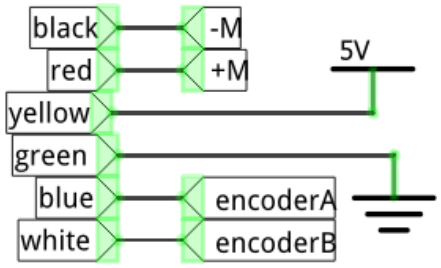


Figure 5.6 Motor - encoder wiring

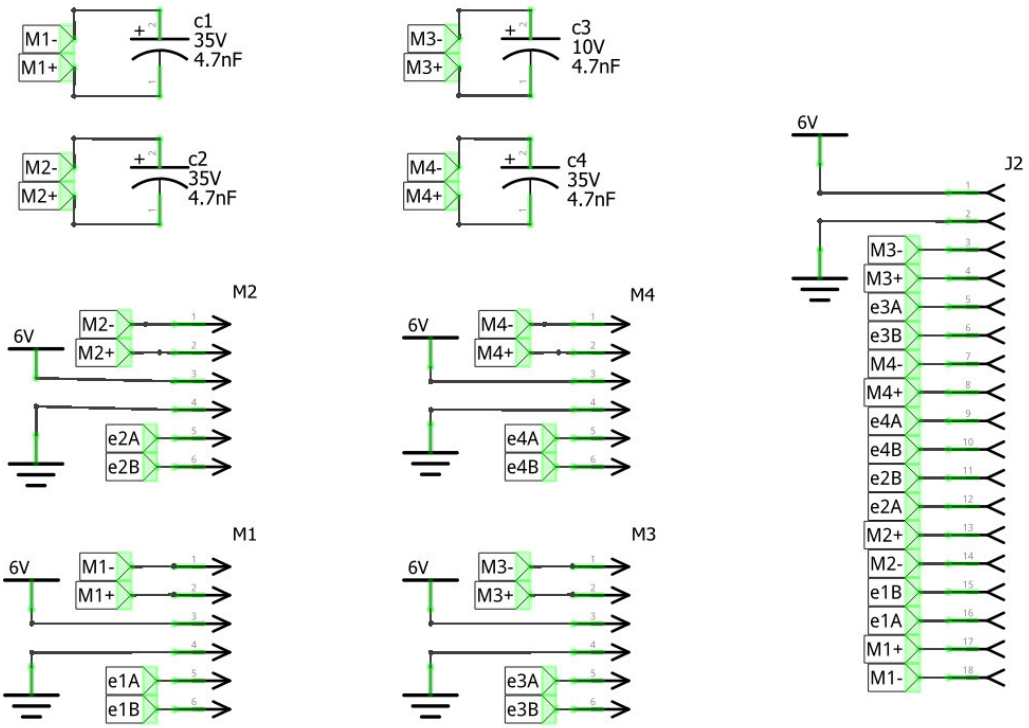


Figure 5.7 Connections board schematic

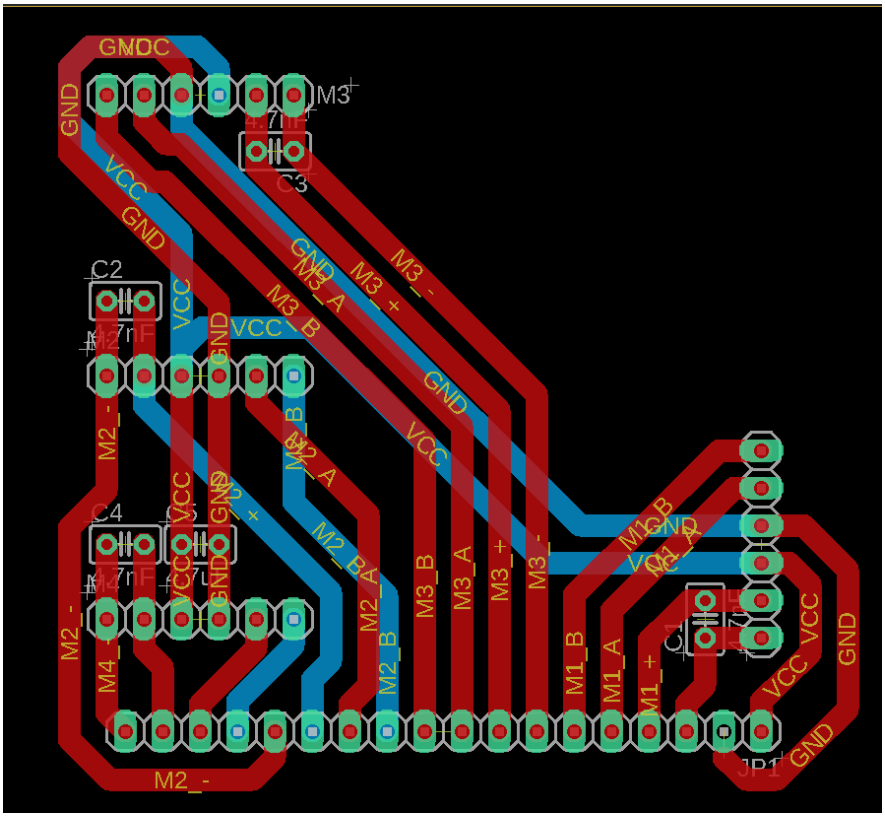


Figure 5.8 PCB design

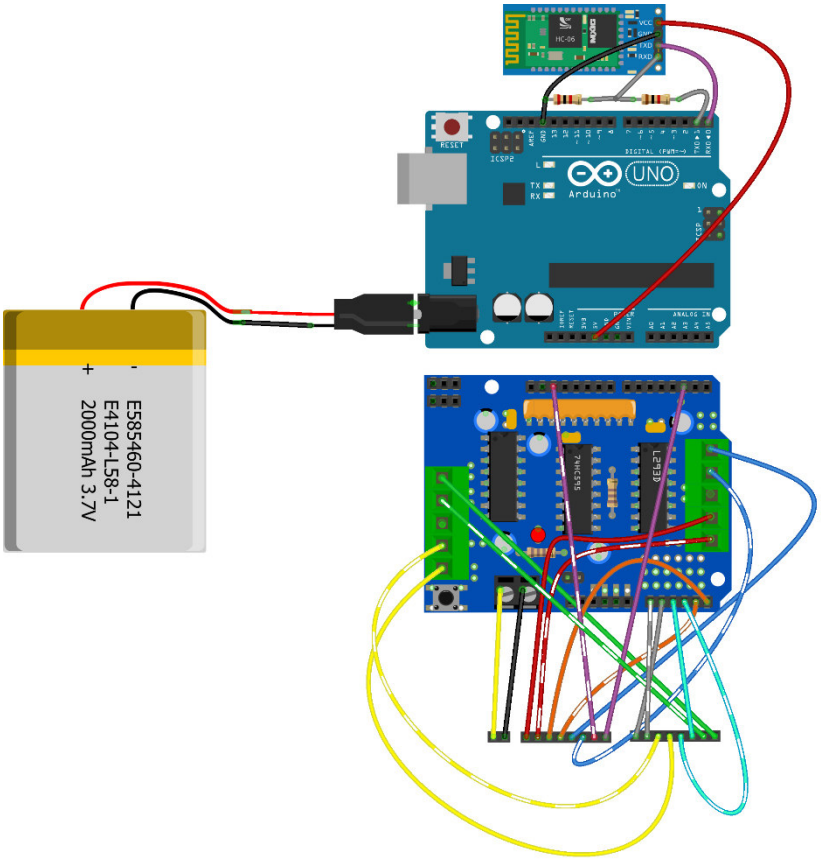


Figure 5.9 Fritzing hardware connection diagram