EXPLORING THE POTENTIAL OF COMPUTER VISION AND MACHINE LEARNING IN ENHANCING THE FUNCTIONALITY OF AN EMG-CONTROLLED PROSTHETIC HAND

A Thesis Submitted to the

College of Graduate and Postdoctoral Studies

In Partial Fulfillment of the Requirements

For the Degree of Master of Science

In the Division of Biomedical Engineering

University of Saskatchewan

Saskatoon, Saskatchewan

Canada

By

Jethro Odeyemi

PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master of Science degree from the University of Saskatchewan, the author agrees that the Libraries of this University may make it freely available for inspection. The author further agrees that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised the thesis work or, in their absence, by the Head of the Department or the Dean of the College in which the thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without the author's written permission. It is also understood that due recognition shall be given to the author and to the University of Saskatchewan in any scholarly use which may be made of any material in this thesis.

DISCLAIMER

Requests for permission to copy or to make other uses of materials in this thesis in whole or part should be addressed to:

Chair of the Division of Biomedical Engineering
57 Campus Drive,
University of Saskatchewan
Saskatoon, Saskatchewan, Canada S7N 5A9

OR

Dean
College of Graduate and Postdoctoral Studies
University of Saskatchewan
116 Thorvaldson Building, 110 Science Place
Saskatoon, Saskatchewan S7N 5C9
Canada

ABSTRACT

The potential of using machine learning techniques to develop prosthetic arms that can automatically perform hand gestures and grasp objects is very important in healthcare systems. Hands are an important part of the body for all vertebras, animals use theirs for locomotion, however, because of our bipedal nature as humans, we use our hands majorly for gripping and general manipulation. Humans without hands must live with prostheses, as a result, prosthetic hands must be well-sophisticated to perform the functions of a regular human hand. Many electronic prosthetics available in the market come with sophisticated control methods. It may take several months of continuous training for the human to learn how to accurately control the prosthetic fingers and perform tasks like picking up objects. This research aims to alleviate this problem by proposing an automated method for performing hand gestures and grasping objects using computer vision-based techniques and machine learning. This research demonstrates the feasibility of this approach by training tree-based classifiers to interpret EMG signals because they offer a direct measure of feature importance. Of the two tree-based classifiers implemented, the results show that the decision tree classifier outperforms the random forest classifier in terms of precision, recall, and F1-score, with EMG signals from Channel 2 being the most important feature for both models. Using an RGBD camera mounted at the base of the gripper which records observation in discrete steps, this research demonstrated the effectiveness of machine learning in automated object gripping. Agents were trained using the existing Soft Actor-Critic (SAC), Deep Q-Network (DQN) and Proximal Policy Optimization (PPO). This research shows that the SAC algorithm is the most effective approach for training agents to perform automated grasping tasks, outperforming other algorithms such as DQN and PPO in terms of their success rate. The agents were trained on three different types of objects (remote controller, soap bar, and mug), and the results show that the two factors (object shape and size) affect the agent's ability to converge to an optimal

policy. The SAC algorithm demonstrated a remarkable resilience when tested on diverse environments and objects, and at varied hyperparameters. While the PPO algorithm demonstrated greater adaptability than DQN, it did not perform as well as the SAC algorithm in terms of overall success rate and ability to handle diverse scenes and objects. Discussions of the reason behind these results are provided. The contribution of this thesis is the conclusion that the second channel of an 8-channel EMG device is the most significant when using Decision Tree Classifiers to interpret EMG signals. Also, the SAC algorithm has a great potential in developing intelligent prosthetic arms with the automatic object gripping capabilities, paving the way for more advanced prosthetics in the future.

ACKNOWLEDGEMENTS

It was a privilege to undertake this research under the guidance of my supervisor, Prof. Chris Zhang, for whom I hold deep gratitude. His patience, systematic approach to problem-solving, and scholarly advice were invaluable in the success of my program. Furthermore, I am thankful for his professional contributions, which greatly enhanced my understanding of the fields of artificial intelligence. During my first advisory committee meeting, Prof. Kelvin Wong, a member of my advisory committee, provided insightful recommendations in Machine Learning that significantly impacted this study's success. I am grateful for his time and valuable contributions.

I am also deeply appreciative of my parents, Dr. Femi Odeyemi and Mrs. Nike Odeyemi, for their unrelenting support, encouragement, and financial assistance throughout my academic pursuits. They made significant sacrifices to ensure my success, and their unwavering dedication is an inspiration to me.

Thanks to Baris Yazici for his work on branch duelling q-networks and making his codebase open sourced.

DEDICATION

This research work is dedicated to the All-Knowing God, who has guided me throughout my academic career and is the source of all knowledge.

Also, to my supporting family for their continuous spiritual and financial assistance during my study

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

HCI	Human-computer interaction
EMG	Electromyography
AI	Artificial intelligence
CNN	Convolutional neural network
EEG	Electroencephalography
SFAP	Single fiber action potential
MAP	Muscle Action Potential
sEMG	Surface-EMG
iEMG	Intramuscular-EMG
SFE	Single fiber EMG electrodes
FEA	Finite element analysis
BCI	Brain-computer interface
DC	Direct current
SMA	Shape memory alloy
DOF	Degree of Freedom
MCP	Metacarpophalangeal
QTC	Quantum Tunnelling Composite
MDP	Markov Decision Process
DQN	Deep Q Networks
RL	Reinforcement learning
SAC	Soft Actor-Critic
PPO	Proximal Policy Optimization

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

Hands are an important part of the body for all vertebras, animals use theirs for locomotion, however, because of our bipedal nature as humans, we use our hands majorly for gripping or grasping. Performing simple activities with the human hands requires a series of complex commands sent from the brain to the hand muscles. There is no doubt that the human hands play a very vital role in our daily activities such as opening doors, picking up objects, and general manipulation. Losing our hands means we would have lots of difficulty carrying out these activities and would most likely have to depend on an aide for assistance. Amputees have to live with prostheses as a substitute for their missing hands, as a result, prosthetic hands must be well sophisticated to perform the functions of a regular human hand.

There are different types of prosthetic hands on the market today (Hook, 2021; Össur, 2021; Ottobock, 2021). Cosmetic prosthetic hands otherwise referred to as passive prostheses (Henson, 2021), which do not have any active component and as such cannot be manipulated for gripping objects. Though passive, cosmetic prostheses help to provide balance and general stability. There are also body-powered prostheses (Prigge, 2021), which are operated by the use of cables and strings. Motion transmitted from the upper arm is captured by the cable and used to open or close the prosthetic hand. As the field of Human-Computer Interaction (HCI) grew, prostheses were being developed which made use of Electromyography (EMG) signals for control as an alternative to cables and harnesses. EMG signal is that signal that measures electrical activities during muscle contraction (Raez, et al., 2006). EMG-controlled prostheses have embedded motors that move the fingers based on the electrical activities recorded during muscle contraction (Prigge, 2021). These types of prostheses offer much more advantages than cosmetics or body-powered prostheses such as the ability to define multiple grip patterns, user-

friendly, and less effort required to grip objects. Other types of hand prostheses include hybrid prostheses which make combine multiple control methods such as EMG and body-powered control or EMG and Artificial Intelligence (Ottobock, 2021), and brain-controlled prosthetics that use brain transmission for finger control (Bright, et al.,), and Artificial Intelligence (AI) in prosthetics (Nayak & Das, 2020).

AI has many applications in prostheses particularly when concerned about improving accuracy and speed, without negatively trading off on the weight of the system. With various advances in deep learning techniques and Convolutional Neural Networks (CNN) (Zhang, et al., 2018; Alzubaidi, et al., 2021), modern prostheses being developed now make use of some form of AI technology. AI has been applied to prosthetic hands in different ways. Many researchers make use of computer vision technology and CNN to accurately predict grip patterns (Castro & Rigolin, 2022; Slade, et al., 2015; Ghazaei, et al., 2017). This technique offers improved speed and accuracy than the traditional EMG methods. The control system for EMG-based prostheses is largely pre-defined based on a set sequence and does not offer as much flexibility as AI-based prostheses. CNN methods used in prostheses include object recognition and pattern recognition. For object detection, cameras embedded on the palm or fingers of the prosthetic hand capture the object within the region of interest. The object type and appropriate gripping pattern are then determined by the CNN. CNN is also used for pattern recognition of EMG signals. It translates the electrical signals of the muscles into patterns, and the prosthetic fingers move following the predicted patterns.

This thesis explores the potential of using machine learning techniques to develop prosthetic arms that can automatically perform hand gestures and grasping objects. Modern prosthetic hands are designed to function like a normal human hand, as such, the primary function of a hand prosthesis is to offer seamless manipulation. Prosthesis fingers are manipulated to pick up objects, create hand gestures, type, and make use of tools. When handling objects like an

egg, or glass cups, it is important to ensure that the prosthetic hand has the capabilities to handle these delicate objects. Many prosthetics available do not offer a means to automatically control the grip force of prosthetics. Some require the amputee to undergo several months of training to be able to accurately control their muscles in other to manipulate the opening and closing of the prosthesis fingers. The prosthetic hands that offer an automatic grip control make use of sensors embedded in the fingers and palm of the prosthetic hand. However, the additional sensors tend to increase the weight of the prosthesis making it less comfortable for the user. The knowledge gap addressed in this thesis is the lack of prosthetic arms that can automatically grip objects or perform hand gestures, without requiring the user to undergo months of training or adding excessive weight to the prosthesis.

To achieve this knowledge gap, the study examines the the use of machine learning algorithms to enhance accuracy in estimating hand gestures and detecting objects. The study also examines the impact of object shape and size on the effectiveness of machine learning algorithms, the optimization policies such as Soft Actor-Critic (SAC), Proximal Policy Optimization (PPO), Deep Q Networks (DQN), for training agents in automated grasping tasks. The use of machine learning to enhance the robustness, fault tolerance, and resilience of prosthetic arms to different environments and objects (Wang, et al., 2020; Zhang, et al., 2017; Zhang, et al., 2011) is also covered in this thesis. The study employs various techniques and methodologies such as classification models trained on electromyography (EMG) data, evaluation of the performance of trained models and agents using classification metrics, and training and testing of agents on different types of objects. The results obtained from these experiments provide insights into the feasibility and effectiveness of using machine learning techniques to develop advanced prosthetic arms.

1.2 Problem Description

It has been noted that many times, objects slip or are damaged when grasped by a prosthetic hand (Abul-Haj & Hogan, 1990; Kyberd, et al., 2007). There are many intelligent prosthetics available in the market today. These prosthetics come with different methods to control the finger movement of the robotic hand such as through EMG signals, Electroencephalogram (EEG), eye tracking, transfer learning, and facial recognition. While these control methods allow the amputee to manipulate the prosthesis fingers, a seamless method to automate the grip control and hand gestures of the prosthetic hand is still largely unexplored. Some of the methods of grip control that have been applied in prosthetic hands include:

1. Manual Control: In the method, the control of the grip force falls solely on the human. In an EMG-controlled prosthetics, the amputee must accurately control his muscles using his discretion to ensure that the grip force is just right enough to pick up an object. This method is highly inefficient, a little lapse in concentration and the prosthetic hand could damage the object if the force exerted is too much.
2. Human control using haptic feedback that returns feedback on touch, pressure, stress, or temperature of the object (Clemente, et al., 2019; Kim & Colgate, 2012; Wang, et al., 2020; Ruyi, et al., 2022).
3. Human control using a force feedback system that exerts a force on the upper limb of the amputee proportional to the magnitude of grip force applied by the prosthetic hand (Brockdorff, et al., 2022).
4. Through electrical pulses fed to the skin which corresponds to the exerted grip force by the prosthetic hand (Chortos, et al., 2016).

All these methods require extra sensors or other components to function which inherently increases the weight of the prosthetic hand. Thus, the user's experience is compromised when

applying the grip force control methods discussed above. Rather than adding more sensors and hardware to the system, this thesis will approach the problem of grip control from a software perspective.

1.3 Objectives

The overall objective of this thesis is to explore the application of computer vision and machine learning technologies in prosthetic hands to automate the process of performing hand gestures and grip control of the prosthetic hand.

The following specific objectives will be pursued to achieve this goal:

- **Objective 1**: Train tree-based classification models in accurately interpreting EMG signals
- **Objective 2**: Train an agent using existing SAC, PPO and DQN algorithms to enable the prosthetic hand's end effector to automatically grasp objects and to examine how each of the mentioned models affects the hand’s ability to effectively grasp objects.
- **Objective 3**: Test the optimal model on varying objects to examine how object’s physical properties affect prosthetic’s hand ability to effectively grasp objects.

1.4 Research Scope

The scope of this research will be to investigate the potential of machine learning techniques in developing advanced prosthetic arms with automatic gesture performance and object gripping capabilities. To achieve this goal, the study will employ various techniques and methodologies, including training classification models on electromyography (EMG) data to estimate hand gestures, and using optimization policies to train agents to perform automated grasping tasks. The study will also investigate the impact of the two factors, namely object shape and size, on the effectiveness of machine learning techniques for training agents to converge to an optimal policy. The research will be conducted using simulation environments

and publicly available datasets, and the results will be evaluated in terms of various classification metrics. The scope of the research will be limited to the development of a proof-of-concept framework, and the feasibility and effectiveness of the developed framework will be discussed in the context of potential future applications in prosthetic hand design.

1.5 Organization of the Thesis

The thesis is organized into seven chapters as follows:

Chapter 1 gives a background overview of the planned study and discusses the research gap and motivation to carry out this thesis. This chapter also sheds light on the project objectives and the research questions to be answered.

Chapter 2 investigates the anatomy of the normal human hand and explains the concepts of EMG and its application in prosthesis development. A comprehensive review of pieces of literature was done on the various control methods of a prosthetic hand to ascertain the pros and cons of these control methods in other to decide the best control method to use for this study. This chapter also discusses the major problem associated with prosthetic hands from the reviews conducted.

Chapter 3 of this thesis emphasizes the significance of selecting appropriate tools and libraries for robotics simulation. The chapter discusses how robotics simulation facilitates the testing and refinement of control algorithms, hardware components, and system designs before deploying them in the real world. It also provides a detailed analysis of the most popular options available for robotics simulation.

In Chapter 4, the first experiment is detailed, which aims to create a model capable of precisely predicting the user's intended gesture through analysing the EMG signal data.

Chapter 5 discusses the second experiment. This experiment examines the application of reinforcement learning to train the prosthetic arm gripper to be able to grip an object in minimum time, with optimal force and at the right contact points.

Chapter 6 discusses the results obtained from the experiments 1 and 2.

Chapter 7 presents conclusions drawn from the study, limitations, and recommendation of future work.

.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter gives a thorough explanation of the background necessary to comprehend the planned research as well as a review of relevant literature. Section 2.2 discusses the human hand anatomy and the usefulness of Electromyography in prosthetic hand control. Section 2.3 gives a background into reinforcement learning. An overview of the literature on the various control techniques used for hand prosthetics is presented in Section 2.4. The chapter's conclusion, Section 2.5, provides an explanation of why the planned study is necessary.

2.2 Background

2.2.1 Electromyography

When the human muscles contract or expands, electrical activities are transmitted. These electrical activities can be measured by a technique known as Electromyography (EMG). EMG helps us to better understand the biomechanical functioning of the human body and as such, it has lots of applications in clinical and biomedical research. EMG when applied in prosthetic arms, is used to generate finger movements based on the pattern of electrical activities transmitted from the muscles in the upper limb of the amputee. A Single Fiber Action Potential (SFAP) is the electrical activity that regulates the responses of the skeletal muscles to perform any of the following functions: movement of the body, maintenance of the body posture, joint stabilization, and body heat generation (Göker, 2014). Electrodes are embedded in EMG to accurately map and visualize these SFAPs which can be done in different ways. A computer-assisted method that made use of an array of six electrodes was implemented by (Okajima, et al., 1995): two electrodes were used for stimulation while the other four were used to record the Muscle Action Potential (MAP). An iterative method of estimation performed on the recorded MAP is then used to calculate the SFAP. EMG is divided into two types based on

how it is positioned to record muscle data. Surface-EMG (sEMG) is non-invasive and is used in most prosthetic hands while Intramuscular-EMG (iEMG) requires the skin to be pierced iEMG offers more stable data of the muscle's electrical activities than sEMG (Smidstrup, et al., 2011). There are five types of electrodes used in EMG (Göker, 2014):

1. Surface EMG Electrode: Surface electrodes are non-invasive and only placed on the surface of the human skin. It has a large surface area, meaning it can pick up a very wide range of electrical signals. They are non-selective and required a little amount of time to set up.
2. Macro EMG Electrode: These are used with iEMG to analyze the aggregate of electrical activity in the same motor unit.
3. Concentric EMG Electrode: This is the traditional method used in many clinical or biomedical examinations. It consists of a cannular and core which act as the reference electrode and active electrode respectively. To measure the muscle's electrical activity, a needle is inserted into the skin which contains the core and cannular, while a ground electrode is placed at a distance from the needle.
4. Monopolar EMG Electrode: This is also a needle electrode similar to the Concentric electrode but is less expensive and offers higher sensitivity.
5. Single Fiber EMG Electrode (SFE): SFE is used in situations when data is required from only a single muscle fiber. As a result, it generally offers better sensitivity than other EMG electrodes.

2.2.1.1 Processing EMG Signals

Typically, raw EMG signals are recorded and processed through several steps shown in Figure 2-1 (Altimari, et al., 2012)

1. Filteration: The first step in EMG signal preprocessing is signal filtering. This process is applied to obtain measurable values for the signal and it involves passing the raw signal through a band stop filter (Dobra & Susca, 2016) of 50 to 60 Hz to remove interferences.
2. Rectification: Initially, the filtered signal cycles through the positive and negative baseline, rectification helps to cut off the negative part of the signal making it flow in one direction only. Two types of signal rectification methods can be performed: half-wave rectification which completely deletes the negative part of the EMG signal and full-wave rectification which adds the part of the signal below the baseline to the positive side of the signal. Half-wave rectification is however not recommended because it eliminates useful parts of the signal.
3. Smoothing: This process further removes noise in the signal creating a clean signal and making it easier to identify points and peaks in the EMG signal.

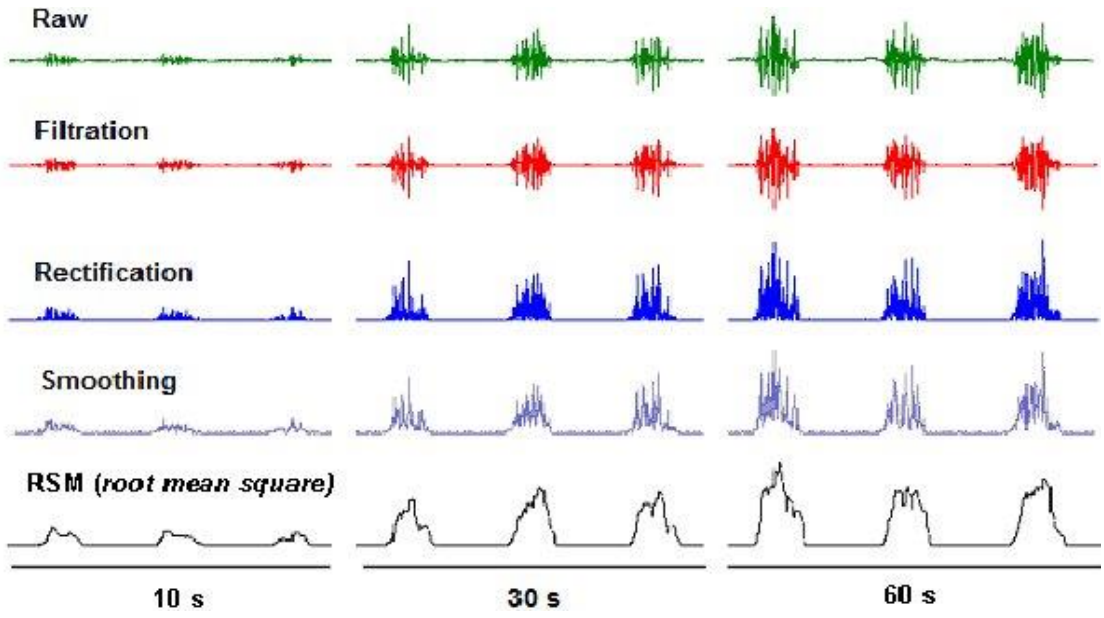


Figure 2-1 EMG Signals at various stages of processing (Altimari, et al., 2012)

2.2.2 The Human Hand Anatomy

To create an anthropomorphic prosthetic hand, we need to understand the mechanics and anatomy of a functioning human hand. The human hand consists of eight carpals, five metacarpals, and fourteen phalanges which adds up to 27 bones as (Maw, et al., 2016) illustrated in Figure 2-2. The carpals are irregular bones that connect the metacarpals to the ulna and radius and allow for flexibility in the wrist. The metacarpals contain the muscles and tendons (Britannica, 2018), the 5 metacarpals allow us to manipulate individual fingers while the phalanges are responsible for gripping or grasping objects.

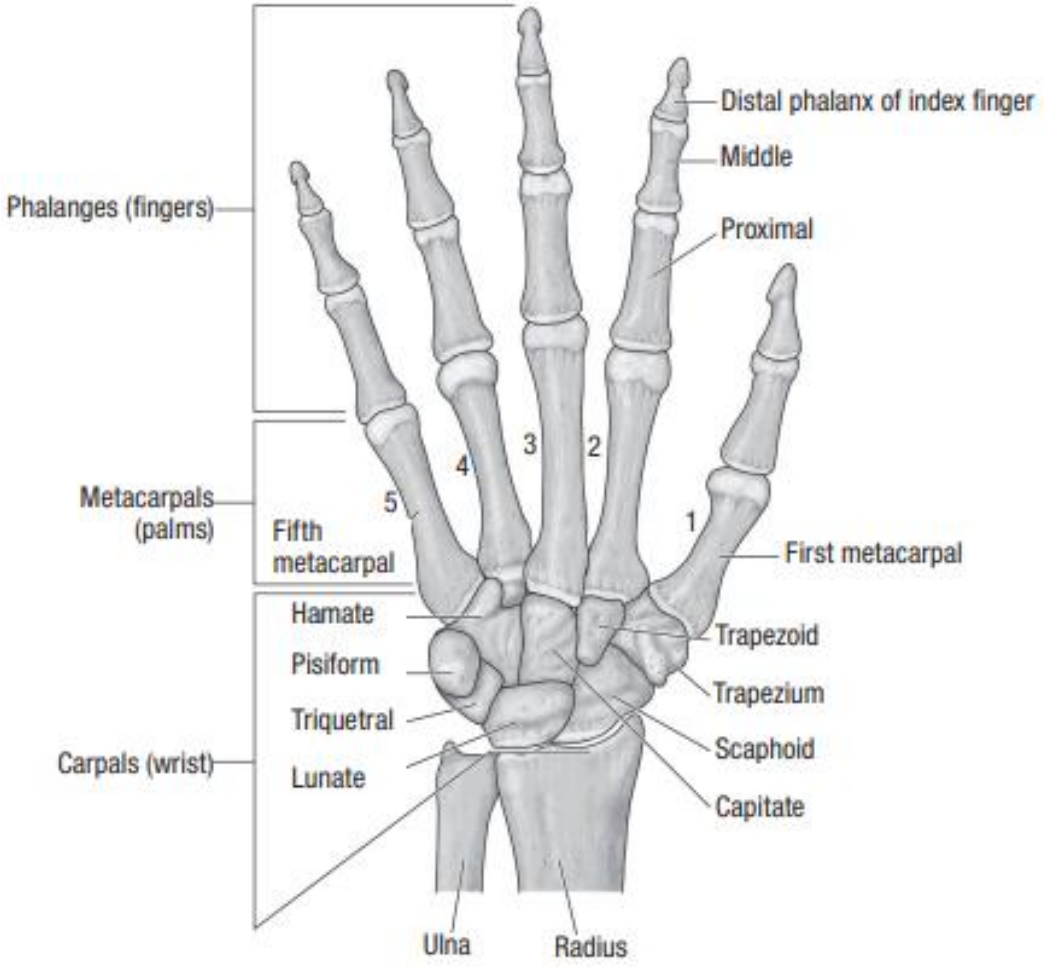


Figure 2-2 The Hand Anatomy (Maw, et al., 2016)

(Dunai, et al., 2021) created a prosthetic hand based on the exact anatomy of a real human hand. In the study, they investigated the metric dimensions of the metacarpal bones and phalanges of the average human hand. The results are shown in Table 2-1.

Table 2-1 Bones in the human hand – Metric dimensions

Bone	Thumb	Index finger	Middle finger	Ring finger	Little finger
Metacarpal	1.3567	2.049	1.906	1.719	1.578
Proximal Phalange	1.134	1.489	1.683	1.563	1.254
Intermediate Phalanges		0.864	1	0.994	0.719
Distal Phalanges	0.74	0.757	0.798	0.778	0.698

A similar study was performed by (Astin, 1999), where the maximum finger force was studied that can be applied by the human hand for different age groups for different finger patterns to aid anthropomorphic prosthetic hand development. The finger force of the human hand was measured by strain gauge transducers. The results of the experiments are summarized in Table 2-2.

Table 2-2 Maximum finger strength attainable for different finger patterns based on the person's age group (Astin, 1999)

Age groups	Lateral pattern	Chuck pattern	Palmar pattern	Grasp
18-29	78.93	77.88	51.95	359.91
30-39	85.49	86.06	59.32	449.45
40+	82.69	79.45	56.01	328.64

From Table 2-2 overall, people between the age group of 30 – 39 exert the most grip force for any finger pattern while those between the ages of 18 – 29 on average exert the least. The importance of this finding to the proposed framework in this thesis will be discussed later.

2.3 Reinforcement Learning

Reinforcement learning (RL) forms a major part of this thesis study. In this study, the prosthetic hand end-effector learns to grip objects of different shapes with the right amount of force in various environments using different reward models and policies. Reinforcement learning is a subfield of machine learning that deals with how intelligent agents can learn to make decisions in dynamic environments by taking actions that maximize a cumulative reward (Kaelbling, et al., 1996). It is based on the idea of trial-and-error learning, where the agent learns from its experiences with the environment, rather than relying on pre-programmed rules. Rewards and penalties are used in reinforcement learning to communicate appropriate and inappropriate behaviour.

The goal of reinforcement learning is to form algorithms that can learn how to make good decisions in complex, uncertain, and dynamic environments. This makes it well-suited for a wide range of applications, such as game playing, robotics, autonomous driving, and resource management, among others.

At the heart of reinforcement learning is the concept of an agent that interacts with an environment. The environment is typically modelled as a Markov Decision Process (MDP), which is a mathematical framework that describes the dynamics of the environment in terms of states, actions, rewards, and transitions, and is discussed next.

2.3.1 Markov Decision Process

Markov decision process (MDP) (Markov decision processes, 1989) is a mathematical framework used to model decision-making in situations where outcomes are uncertain and influenced by both random chance and agent actions. MDPs are widely used in the fields such as artificial intelligence, operations research, economics, and control theory. In this thesis, we

will provide an in-depth explanation of MDPs, including their components, formal definition, and applications.

2.3.1.1 Components of MDPs

An MDP consists of five main components represented in Figure 2-3: a set of states S, a set of actions A, a transition function T, a reward function R, and a discount factor γ. Let us consider each of these components in more detail.

1. State Space (S) - The state space is the set of all possible states that the agent can be with. It represents the information that the agent has about the world at a particular time. For example, in a game of chess, the state space includes all the possible configurations of the board.
2. Action Space (A) - The action space is the set of all possible actions that the agent can take. An action is a decision that the agent makes based on its current state. For example, in a game of chess, the action space includes all the possible moves that the agent can make.
3. Transition Function (T) - The transition function defines the probability of moving from one state to another state after taking a particular action. Mathematically, the transition function is denoted by T(s,a,s'), where s is the current state, a is the action taken, and s' is the next state. The transition function captures the dynamics of the environment, which may be stochastic, deterministic, or a combination of both.
4. Reward Function (R) - The reward function defines the immediate reward that the agent receives after taking an action in a particular state. The reward function is denoted by R(s,a,s'), where s is the current state, a is the action taken, and s' is the next state. The reward function is used to guide the agent towards the optimal policy that maximizes the expected cumulative reward.

5. Discount Factor (γ) - The discount factor is a value between 0 and 1 that determines the relative importance of immediate and future rewards. A discount factor of 0 means that only immediate rewards are considered, whereas a discount factor of 1 means that future rewards are given equal weight as immediate rewards. The discount factor is used to ensure that the expected cumulative reward is finite.

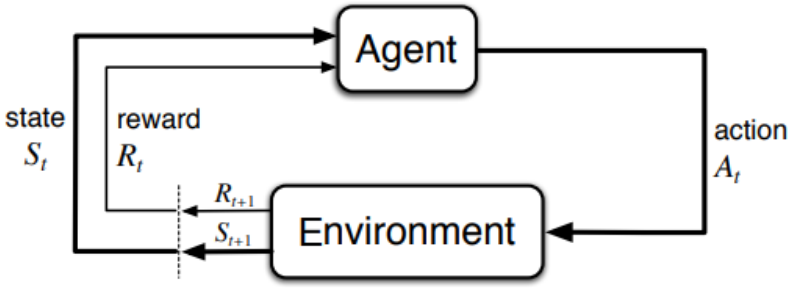


Figure 2-3 MDP Structure

A Markov decision process is formally defined as a tuple (S, A, T, R, γ), where S is the state space, A is the action space, T is the transition function, R is the reward function, and γ is the discount factor. An MDP is a stochastic process, where the state of the environment evolves over time according to the transition function T. At each time step, the agent observes the current state s, selects an action a from the action space A, and receives a reward r from the reward function R. The goal of the agent is to learn a policy π: S → A that maps each state to an action, such that the expected cumulative reward is maximized.

In an MDP, the probability that an agent can get to a state s' from a state s at time t by taking action a can be represented by the equation 2.1.

$$P_{\propto}(s, s') = P_r(s_{t+1} = s' \mid s_t = s, a_t = a) \tag{2.1}$$

If the agent was to receive a reward for each transition between states, the equation 2.1 can be extended as shown in equation 2.2.

$$P_{\propto}(r|s,s') = P_r(s_{t+1} = s', R_t = r \mid s_t = s, a_t = a) \quad (2.2)$$

2.3.1.2 Solving an MDP problem

Value iteration (Antos, Szepesvarf, & Munos, 2007) and policy iteration (Dankert, et al., 2005) are two algorithms used to solve the MDP problem and to find the optimal policy for an agent in a given environment.

Value iteration is an iterative algorithm that starts with an initial estimate of the value function and improves it in each iteration. The algorithm iteratively applies the Bellman optimality equation until the values converge to the true optimal values. The Bellman optimality equation for the value function is defined in Equation 2.3

$$V^*(s) = \max\_a \sum\_s' P(s'|s,a) [R(s,a,s') + \gamma V^*(s')] \quad (2.3)$$

where V*(s) is the optimal value function for state s, P(s'|s,a) is the probability of transitioning from state s to state s' given action a, R(s,a,s') is the reward for transitioning from state s to state s' with action a, and γ is the discount factor.

Policy iteration, on the other hand, is a more direct approach that involves improving the policy and the value function alternately until convergence. In each iteration, the algorithm evaluates the value of the policy and improves the policy based on the value (Figure 2-4). The value function is then updated using the new policy, and the process repeats until convergence. The steps involved in policy iteration are as follows:

1. Initialization: Start with an initial policy π and value function V.
2. Policy Evaluation: Evaluate the value function Vπ for the current policy π shown in equation 2.4. This equation calculates the expected value of being in state s under the current policy π and then taking action a, followed by transitioning to state s' with probability P(s^',r|s,π(s)) and receiving the immediate reward r.

$$V^{\pi}(s) = \sum s', rP(s', r|s, \pi(s))[r + \gamma V^{\pi}(s')] \quad (2.4)$$

3. Policy Improvement: Improve the policy by selecting the best action in each state based on the value function (equation 2.5). This equation selects the action that maximizes the expected value of the next state for each state s, based on the value function V^{π} obtained in step 2. The resulting policy π_1 is guaranteed to be better than or equal to π.

$$\pi_1(s) = argmax_a \sum s', rP(s', r|s, a)[r + \gamma V^{\pi}(s')] \quad (2.5)$$

4. Repeat steps 2-3 until convergence.

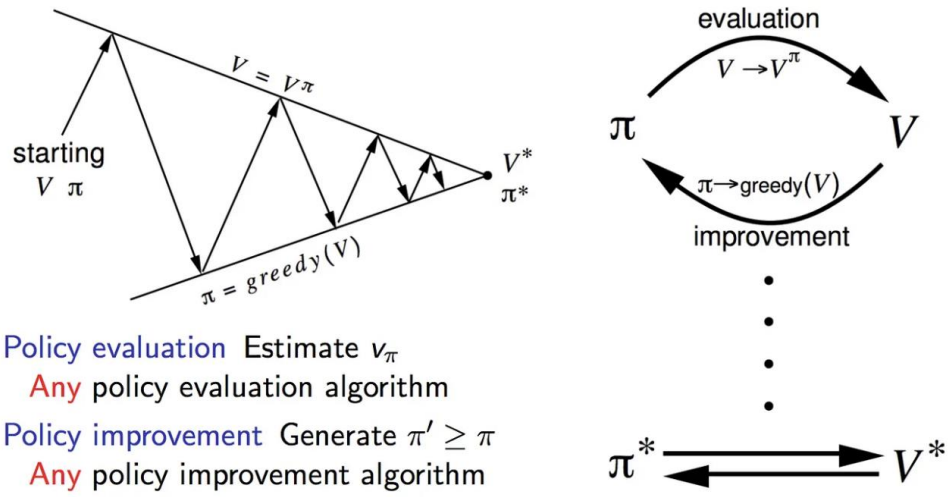


Figure 2-4 Policy Iteration (Kung-Hsiang, 2018)

2.3.2 Q-Learning

Q-learning (Matiisen, 2015) is a popular reinforcement learning algorithm that learns the optimal action-value function $Q(S_{t+1}, a)$ directly. The algorithm updates the action-value function using the following update rule:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \underbrace{max}_{a} Q(S_{t+1}, a) - Q(S_t, A_t))] \quad (2.6)$$

where $Q(S_t, A_t)$ is the estimate of the action-value function at time t, R_{t+1} is the reward received at time t, $\propto$ is the learning rate, γ is the discount factor, and $\underbrace{max}_{a} Q(S_{t+1}, a)$ is the maximum value over all actions that can be taken in the next state.

Q-learning: Learn function $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$

Require:
Sates $\mathcal{X} = \{1, \ldots, n_x\}$
Actions $\mathcal{A} = \{1, \ldots, n_a\}$, $A : \mathcal{X} \Rightarrow \mathcal{A}$
Reward function $R : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$
Black-box (probabilistic) transition function $T : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$
Learning rate $\alpha \in [0, 1]$, typically $\alpha = 0.1$
Discounting factor $\gamma \in [0, 1]$
procedure QLEARNING($\mathcal{X}$, A, R, T, α, γ)
  Initialize $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ arbitrarily
  while Q is not converged **do**
    Start in state $s \in \mathcal{X}$
    while s is not terminal **do**
      Calculate π according to Q and exploration strategy (e.g. $\pi(x) \leftarrow \arg\max_a Q(x, a)$)
      $a \leftarrow \pi(s)$
      $r \leftarrow R(s, a)$ ▷ Receive the reward
      $s' \leftarrow T(s, a)$ ▷ Receive the new state
      $Q(s', a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \cdot \max_{a'} Q(s', a'))$
      $s \leftarrow s'$
  return Q

Figure 2-5 Q-Learning pseudocode (Kung-Hsiang, 2018)

2.3.3 Soft Actor-Critic

The Soft Actor-Critic (SAC) algorithm is a practical approximation to soft policy iteration that uses function approximators for both the Q-function and the policy (Haarnoja, et al., 2018). It alternates between optimizing both networks with a stochastic gradient descent. The parameters of these networks are ψ, θ, and φ. The state value function approximates the soft value, which can be estimated from a single action sample from the current policy. However, including a separate function approximator for the soft value can stabilize training and is convenient to train simultaneously with the other networks. The soft value function is trained to minimize the squared residual error (Equation 2.7).

$$J_V(\varphi) = \in_{s_t} \sim D\left[\frac{1}{2}\left(V_\varphi(S_t) - \in_{a_{t\sim\pi_\varphi}}\left[Q_\theta(S_t, a_t) - log\pi_{\emptyset(a_t|S_t)}\right]\right)^2\right] \quad (2.7)$$

The soft Q-function parameters can be trained to minimize the soft Bellman residual (equation 2.8), which can be optimized with stochastic gradients (Equation 2.9).

$$J_Q(\theta) = \in_{(s_{t,a_t})} \sim D\left[\frac{1}{2}\left(Q_\theta(S_t - a_t) - \hat{Q}(s_t, a_t)\right)^2\right] \quad (2.8)$$

$$\hat{V}_\theta J_Q\ (\theta) = \nabla Q_\theta\ (a_t - s_t\ )\left(\hat{Q}(s_t, a_t) - r(s_t, a_t) - \gamma V_\varphi(s_{t+1})\right) \quad (2.9)$$

The policy parameters can be learned by directly minimizing the expected KL-divergence (Equation 2.10).

$$J_{\pi(\varphi)} = \in_{s_t} \sim D[D_{KL}(\pi_\varphi(.|s_t)||\left(\frac{\exp(Q_\theta(s_t,.))}{Z_\theta(s_t)}\right)] \quad (2.10)$$

A typical solution for policy gradient methods is to use the likelihood ratio gradient estimator. In SAC, the target density is the Q-function, which is represented by a neural network and can be differentiated. It is thus convenient to apply the reparameterization trick instead, resulting in a lower variance estimator. The algorithm uses a neural network transformation to reparametrize the policy (Equation 2.11), and we can approximate the gradient of the objective with an unbiased gradient estimator that extends the DDPG style policy gradients to any tractable stochastic policy (Equation 2.12).

$$a_t = f_\varphi(\in_t; s_t) \quad (2.11)$$

$$\hat{V}_\theta J_\pi\ (\emptyset) = \nabla Q_\theta\ (a_t - s_t\ )\left(\nabla_{a_t} log\pi_\emptyset(a_t|s_t\right) - \nabla_{a_t} Q(s_t, a_t))\nabla\_\emptyset\ f\_\emptyset(\in_t; s_t) \quad (2.12)$$

Algorithm 1 Soft Actor-Critic

Initialize parameter vectors ψ, $\bar{\psi}$, θ, ϕ.
for each iteration **do**
  for each environment step **do**
    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$
    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$
  end for
  for each gradient step **do**
    $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
    $\bar{\psi} \leftarrow \tau\psi + (1 - \tau)\bar{\psi}$
  end for
end for

Figure 2-6 SAC pseudocode (Haarnoja, et al., 2018)

2.3.4 Proximal Policy Optimization

The goal of PPO to find a policy (a set of actions to take in different states) that maximizes the expected reward. PPO is based on the idea of a surrogate loss function that is used instead of the policy gradient, and it uses a value function to estimate the advantage of taking an action in a given state (Schulman, et al., 2017).

To implement PPO, one first constructs a surrogate loss function that can be differentiated and optimized using stochastic gradient ascent. This surrogate loss function includes a clipping term or a KL penalty term, depending on the specific implementation. Additionally, a value function is used to estimate the advantage of taking an action in a given state. The policy and value function can share parameters, which helps to reduce the number of parameters that need to be learned.

To optimize the surrogate loss function, PPO uses multiple steps of stochastic gradient ascent, typically using Adam optimization. The algorithm collects data by running the policy for a fixed number of timesteps, and then uses this data to optimize the surrogate loss function. The

data is typically collected by running multiple parallel actors, each of which collects a fixed number of timesteps.

Algorithm 1 PPO, Actor-Critic Style

for iteration=$1, 2, \ldots$ **do**
  for actor=$1, 2, \ldots, N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
end for

Figure 2-7 PPO pseudocode (Schulman, et al., 2017)

2.3.5 Deep Q-Networks

Deep Q-Networks (DQN) is a variant of Q-learning that uses deep neural networks to represent the action-value function. DQN was first introduced by (Mnih, 2015), and has since been extended and improved in many ways.

DQN works by using a neural network (Figure 2-8) to approximate the action-value function. The algorithm uses a technique called experience replay to store and randomly sample from a replay buffer of past experiences, which helps to stabilize the learning process and reduce the effects of correlations between consecutive samples.

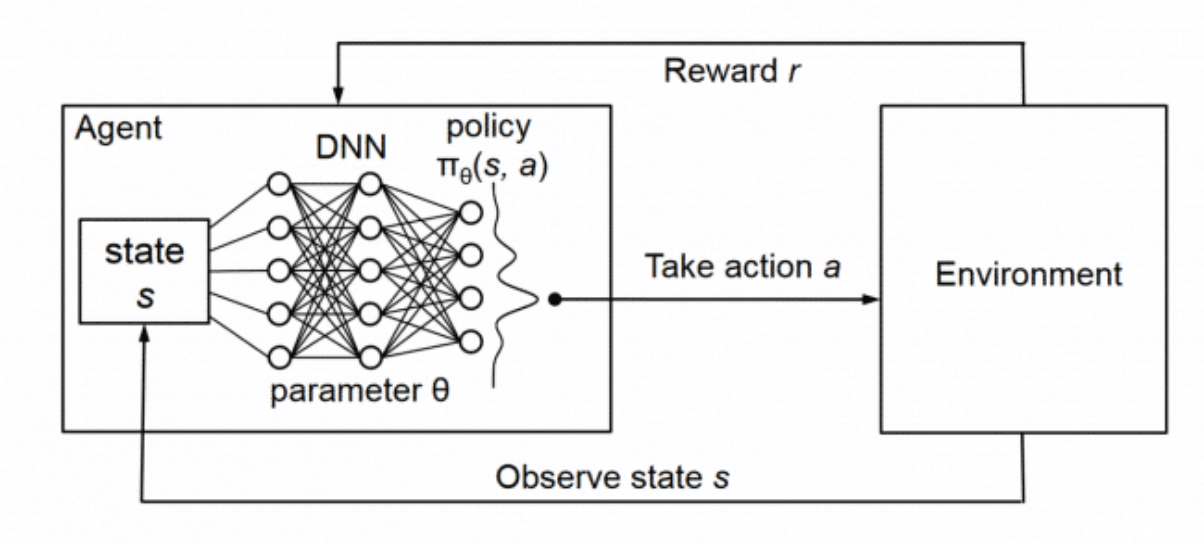


Figure 2-8 DQN RL process

DQN also uses a technique called target network to stabilize the learning process. The target network is a separate copy of the neural network that is used to compute the target values in the update rule. The weights of the target network are updated slowly, typically after a fixed number of iterations, to track the changes in the action-value function.

Algorithm 1 Deep Q-learning with experience replay

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
for episode 1, M **do** Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \arg\max_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in the emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store experience $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of experiences $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the weights θ
    Every C steps reset $\hat{Q} = Q$
  end for
end for

Figure 2-9 DQN pseudocode (Roderick, et al., 2017

2.4 Related Work

2.4.1 Cosmesis

There is not a lot of recent literature on cosmesis, however, only 1 in every 3 amputees makes use of a cosmetic prosthetic hand according to a study in (Maciel, et al., 2018). This type of prosthetic hand does not have any manipulative function, however, it helps to provide the user with stability and general body balance. Cosmesis offers some psychological advantages by eliminating trauma that may arise from looking at the amputated limb and plays some functional roles by allowing the person to make the best possible use of his amputated hand (Pillet, 1981). Highsmith, et al. (2009) developed a cosmesis with a cylindrical grasp function. The cosmesis is made up of an aluminum cylindrical inner sleeve split into two halves across the long axis. The complete structure includes two steel washers welded into a pip which acted as a transverse stop and a closure strap at the tip for attachment to the upper limb. This device was tested by an amputee to use in paddling a boat, who comfortably completed the activity. The major flaw of this device is that it has a steel frame that is heavy and can easily corrode when in contact with water.

2.4.2 Harness-Controlled Prosthetic Hands

A study in (Bustamante, et al., 2018) explores the use of 3D printing technology to prosthetic hand development. A volumetric size scaling approach was used during the computer-aided design of the hand, in such a way that downscaling or upscaling the prosthesis hand based on the size of the amputee's upper limb would not pose a problem. Additionally, a new finger movement was implemented which made use of a four-bar linkage mechanism (Difonzo, et al., 2020) such that when a cable is tensioned, it pulls on the first of the four-bar linkage, this motion is continually transmitted through the phalanges. This approach is novel and requires fewer tools than the traditional cable-controlled prosthetic hands. Body-powered prostheses are considerably much cheaper than EMG-controlled alternatives because of the absence of

electronic parts. However, they are also larger in weight and generally less efficient, as such, various research is ongoing on improving the efficiency of body-powered prostheses. A study carried out in (Smit & Plettenburg, 2010) reviewed five commercially available body-powered prostheses and proposed two methods to improve their efficiency. In terms of controllability, the mechanism and glove hysteresis should be reduced. To ensure that the amputee's muscle is not fatigued on continuous activation of the prosthesis, the activation force should also be lowered. Damodar, et al. (2019) designed a prosthetic with a complete mechanical transmission system actuated by gears. By making use of a spring-gear system to transmit motion, the prosthetic hand could deliver a very large amount of force while requiring little effort from the amputee. A prosthetic arm emulator was designed in (Poddar, et al., 2021) for amputees without a lower limb. The complete system is made up of four potentiometers, linear actuators, and a pulley system.

2.4.3 Myoelectric-Controlled Prostheses

A new approach to topology optimization for a compliant robotic finger was explored by Zheng, et al. (2016). Their design was novel and was successfully able to combine the monolithic structure of the finger with a distributed compliant method for topology optimization. The method employed for topology optimization involved parameterization, optimization formulation, genetic algorithm, and finite element analysis (FEA). A prototype was built and compared to the FEA results for a 30 degrees rotation in a link, the difference between both models was found to be only 15% which is explained by errors during the fabrication of the prototype. Hao, et al. (2021) developed a low-cost soft prosthetic hand with embedded actuation and soft sensors. Similar to (Zheng, et al., 2016), this study also employed a monolithic structure because of its lightweight properties. The embedded sensors made it possible to account for an initial contact detection with the object to be gripped, this method ensured that the fingers stopped bending immediately it touched the object so as not to damage

it. This study however has many drawbacks. First, the extra sensors in the arm made the prosthetic hand heavier which should not be the case for an anthropomorphic prosthetic hand. Senors also tend to degrade whenever used, so for every contact that the fingers make with an object, the sensors' inherent property decreases which can cause a malfunction in the prosthetic hand. Palermo (2017) developed a framework to improve EMG-based control for a prosthetic hand. Surface electrodes were placed on the user's arm and signal data were generated based on how the person contracts his muscles. This framework uses these signals to determine the best finger movement for the prosthesis in four steps: movement replication, feature extraction to remove noise from the data, movement classification to determine the best possible movements based on the extracted features, and finally, the prosthesis actuates the result. Compliance is very important in prosthetic hand technology, this concept is investigated in the literature (Liu, et al., 2018; Mutlu, et al., 2016). (Liu, et al., 2018) designed and developed a compliant prosthesis, the difference between this study and traditional compliant prosthetic hands is that the flexure hinges were defined as 3D objects during design as opposed to treating it as a 2D object. Mutlu, et al. (2016) demonstrated the efficiency of compliant fingers in gripping fragile objects or objects of irregular shapes. They developed a compliant prosthesis finger that made use of asymmetrical flexure hinges. It was discovered that these types of flexure hinges offer a larger mechanical load when compared to other types under exact loading conditions. Mohammadi, et al. (2020) were peculiar about prosthetic hands for children. In their study, they made sure to account for both the cost-effectiveness and weight of this prosthesis by making use of a monolithic structure. The outcome of the study is an anthropomorphic prosthesis with a total weight of 230 gr which is ideal for children between five to seven years of age. The prosthesis is controlled by two surface-EMG and defines two grip patterns: pinch and spherical. Mohammadi, et al. (2020) is another piece of literature that addresses the issue of weight while not compromising the general standard. Described in the

research is a prosthesis with independent control for each finger. The prosthetic arm is multiple actuated and impressively achieves a flexion speed of 1.3 seconds.

2.4.4 An Artificial Intelligence Approach for Seamless Prosthesis Control

A hand prosthesis prototype that incorporates EMG-based control and computer vision was discussed by Castro, et al. (2022). The prototype provides for seamless control of a prosthetic hand that makes use of image processing techniques to define one of five grip patterns (Tripod, Neutral, Pronated, Key grip, and Index finger extension) for the prosthetic hand. This paper however does not investigate a means to control the amount of grip pressure which is an important factor when grasping very delicate objects. Wu, et al. (2020) described a resilient robot that uses embedded computer vision techniques for self-docking. In the docking stage, a camera installed on the robotic devices uses a Yolov3 (Redmon, et al., 2016) architecture for tracking and aligning the end-effectors. Zhiqin Qian, et al. (2022) proposed a modified version of the Yolov4 architecture for real-time detection and classification by connection skipping and parsing a dilation rate. This model was built on eleven convolutional layers and showed results that outperformed the traditional Yolov4 (Alexey, et al., 2020) in terms of accuracy and robustness. To understand how CNNs can be applied for detection and classification tasks, Rawat & Wang (2017) comparatively reviewed over 300 papers on CNNs. Their review focused on three main factors: the early success of CNNs which began in late 1980, how they rejuvenated the field of deep learning, and the challenges of several developed CNN architectures. Similarly, Czimmermann, et al. (2020) studied over 200 scholarly articles and discussed several methodologies for defect detection in materials like textiles, ceramics, and metals. They concluded that when working with powerful techniques like neural networks, a large amount of resources is required to effectively train the network. The same also applies to the field of robotics where systems learn from real-world data. However, no solutions were discussed to overcome parallelization problems that occur when dealing with large amounts of

data. Their review of supervised and unsupervised learning methods shows that the former is time expensive and requires a larger database than the latter. (Sturm & Ramalingam, 2004) developed a black box concept for proper camera calibration. Their study looked at how views of known objects can be gotten from unknown views for calibrating the general image model which is the typical project model used in image processing. The theory of the concept was developed and supported with experiments but was not practically demonstrated. A study by (Abbasi, et al., 2019) used unsupervised learning techniques to understand patterns in different grasp types. The grasp data were collected by fitting a custom glove on a human hand that had sensors to measure pressure from different areas of the hand. This learning technique was able to aggregate over twenty grip types into nine clusters. This concept is however too expensive and not practical in the real world, it would be much better to restrict the number of grip patterns to a few numbers and use a particular pattern for different gripping operations. A prosthetic hand with an automatic grasp selection feature was created by DeGol, et al. (2016). A computer was embedded in a modified version of the open-sourced EMG prosthetic arm created by (Slade, et al., 2015) which uses CNN to define five gripping techniques: pinch, key, power, three-jaw chuck, and tool. Like the prototype developed by (Castro, et al., 2022), this system also does not factor in the grip pressure required for each pattern. Both systems will fail when picking very delicate objects like an egg which can withstand only about 3 PSI of pressure from its shorter axis (Link, 2001). A study carried out Ghazaei, et al. (2017) explored a deep learning-based control for a prosthetic hand also incorporating computer vision techniques. A two-layer CNN architecture was implemented and trained using 1000 common objects to classify hand grips. The prosthetic arm was controlled from two sources: myoelectric signals gotten from sensors attached to the user's muscles and the convolutional neural network which receives image data from the embedded camera and determines the object within the region of interest, then draws an inference on the specific grip pattern to define. A major difference

between this paper and previously discussed ones is that, unlike others that use some sort of object classification algorithm and define the grip based on the object detected, this system uses a grip classification approach. This inherently is a more efficient method because there is no need to train the CNN using a vast amount of computation resources, rather objects can simply be grouped into clusters and an ideal gripping type is then defined. Another research by Shi, et al. (2020) looked at how artificial intelligence can help improve the control of prosthetic hands. This study achieved up to 98% accuracy when classifying everyday objects that can be found in homes. The peculiar point of the study is the method of data collection employed. For good training, the dataset collected was thoroughly augmented by taking images of the same object but with different shapes, and sizes, at different camera distances, and with varied illumination.

2.4.5 Neuro-Prostheses

Researchers have explored various means to improve the control of a prosthetic hand to provide amputees with a more efficient and effortless means to manipulate these prostheses. Neuroprostheses have been developed which make use of neural activity to actuate the prosthetic hand, this functions the same way as the human hand since our hands are effectively controlled by the muscles acting on signals from the brain. One of such is the mind-controlled prosthetic hand designed by Beyrouthy, et al. (2016). This was designed to be of low cost by making use of an Electroencephalography (EEG) headset (LaRocco, et al., 2020) which also delivers very accurate results. Unlike other methods of recording mind data which require some form of surgical implantation, the EEG headset is non-invasive. When a person thinks, ionic currents in the brain neurons cause some form of voltage oscillations which can be recorded by the electrodes mounted on an EEG headset. This data is then analyzed and used to effectively determine corresponding hand movements. The prosthetic finger actuation is controlled by motors based on signals sent from a Raspberry Pi and Arduino microcontroller.

One of the disadvantages of using an EEG for prosthesis control is the difficulty involved in analyzing EEG signals. The human brain is complex and has various parts to it. The EEG method does not effectively give information on what part of the brain the signals are coming from, so it is difficult to make sense of the EEG data. To make better sense of the EEG data for prosthetic hand control, Li, et al. (2018) came up with a method that uses a facial expression-based Brain-Computer Interface (BCI) system. They investigated four types of facial expressions: Raising Brow, Furrowing Brow, Left Smirking, and Right Smirking, and a novel filtering algorithm was developed for EEG data processing. This system was tested on 18 subjects and showed a better performance than traditional mind-controlled prostheses. Most EEG devices function by measuring the frequency generated based on real-time brain activity. The frequencies are classed as follows (Devashish, et al., 2015): Delta (0 to 3 Hz), Theta (7 Hz to 7 Hz), Alpha (8 Hz to 12 Hz), Low Beta (13 Hz to 15 Hz), Midrange Beta (16 Hz to 20 Hz) and High Beta (21 Hz to 30 Hz). Each of these frequencies corresponds to the particular mental state of a human being. A brain-controlled anthropomorphic prosthetic hand was developed by Hasan, et al. (2018) which mimics not just the functioning of a human hand, but also how we react to pain when our hands meet something hot. A temperature sensor was embedded in the prosthesis that provided feedback on the temperature of the object being gripped by the prosthetic hand. If the temperature was above a threshold such that a normal human hand cannot handle, the prosthetic hand would detect this and the motors are actuated causing the prosthetic fingers to open and drop the object immediately. Many of the approaches to neuro-prostheses discussed so far have been non-invasive. Invasive methods such as surgical implants in a patient's scalp have also been used to control bionics. (Warwick, et al., 2003) studied the applications of implant technology for the control of prosthetic hands. An experiment was performed by surgically implanting a nerve fiber in the arm of a healthy man. By making use of a neural interface, brain signals were transmitted to a computer through a transmitter

attached to a connector pad. The disadvantage of this invasive technique is that the fiber implants tend to degrade over time, as such, amputees would need to be subject to surgeries at intervals to replace these implants, which is expensive and ultimately dangerous. Overall, while EEG response is earlier and faster than EMG (Zhang, et al., 2021), EMG-based prostheses are recommended for better finger control.

2.4.6 A hybrid control method for a prosthetic hand

A hybrid-controlled prosthesis uses more than one method to control finger movement. A recent study by Shi, et al. (2022) combined augmented reality, eye tracking, and myoelectric signals to improve the control of prosthetic hands. Eye-tracking functionality is implemented using eyewear which has nine GazeButton (Rivu, et al., 2019). The GazeButton is designed using augmented reality to collect information when the user gazes on a button for more than 120 ms. Three of the GazeButtons are for wrist control while the other six are used to define finger grip types which include the lateral, hook, cylindrical, spherical, tripod, and pinch. EMG signals are measured by sensors on the prosthetic hand, which are used to ascertain the user's intention. A major flaw of this system is that the eyewear uses Wi-Fi to transmit data to the prosthetic hand. In a Wi-Fi dead zone, the system would fail, and the amputee would not be able to control the prosthetic fingers. An intelligent prosthesis named HyPro was developed by (Semasinghe, et al., 2018). The HyPro implements the power, tip, lateral, hook, and index point grasp types using a combination of EMG and body-powered control. Unlike most prophetic hands, the HyPro also offers wrist control in 2 Degree of Freedom (DoF). The HyPro also has a method to control the fingers' grip force using body power when the amputee protracts or retracts his shoulder. This method of grip force control however requires more effort from the amputee, and continuous shoulder movement, in the long run, could cause a strain on the amputee's shoulder. Most prosthetic hands available in the market have the same problem of weight due to the presence of motors for finger actuation. This problem was recognized by

(Lau, 2009) who came up with a hybrid actuation for prosthetic hands using Direct Current (DC) and Shape Memory Alloy (SMA) actuators. This allowed for more active DoF movement for the fingers without compromising the weight of the overall system. The DC actuator controls the flexion-extension movement of the metacarpophalangeal (MCP) joint while the SMA actuator allows the fingers to produce a large amount of grasping force for large linear displacement of the fingers. A hybrid control strategy for a robotic prosthesis was implemented by (Chen, et al., 2009) which made use of an adaptive neuro system and hard computing to cause a two-dimensional movement of the thumb and index finger. The neuro interference system was designed using a regression matrix, the results when compared with other controllers showed higher performance in reducing disturbances and general interference which would lead to e better prosthetic hand control. Similar to the study carried out by (Lau, 2009), (Low, et al., 2013) developed a hybrid actuated prosthetic hand with embedded tactile sensing feedback. The tactile sensor was developed using a Quantum Tunnelling Composite (QTC) for the purpose of implementing a closed-loop control system such that feedback is sent so the prosthetic finger can grasp hands placed on the prosthetic hand.

2.5 Discussion

In this section, over sixty-five literatures were systematically reviewed and summarized in Table 2-3. The papers reviewed in this section were collated based on 5 popular type of prosthetics in the market which are listed as follows:

1. Cosmesis: The literature review in this category focused on the concept of cosmesis in prosthetics. While not extensively discussed in recent literature, cosmesis plays a significant role in the lives of individuals with limb loss. These papers shed light on the importance of prosthetic devices that not only provide functionality but also prioritize aesthetics and appearance. The inclusion of the cosmesis category in this review

emphasizes the need to consider both functional and psychological aspects when designing prosthetic hands.

2. Harness-Controlled Prosthetic Hands: This category of literature review explored the advancements in harness-controlled prosthetic hands, specifically focusing on the utilization of 3D printing technology.
3. Myoelectric-Controlled Prostheses: The literature review in this category focused on myoelectric-controlled prostheses, which utilize electromyography (EMG) signals to enable users to control their prosthetic hands.
4. Artificial Intelligence based Prostheses: This category of literature review discussed the integration of artificial intelligence (AI) techniques, including computer vision and EMG-based control, to achieve seamless control of prosthetic hands.
5. Neuro-Prostheses: The reviewed literature in this category explored the use of neural activity to control prosthetic hands, mimicking the natural control mechanisms of the human hand. These neuro-prostheses utilize signals from the user's brain and muscles to actuate the prosthetic hand. Despite not being extensively covered in this section, the findings highlighted the ongoing advancements in this area, aiming to provide amputees with more efficient and effortless means of manipulating their prostheses.

By categorizing the literature into these distinct categories, the review has shown the wide range of research and advancements in the field of prosthetic hand development. Each category contributes unique insights and approaches to address the challenges associated with designing functional, aesthetically pleasing, and user-friendly prosthetic hands.

Table 2-3 Review at a glance

Literature	Control Method	Pros	Cons	Notes
(Highsmith, et al., 2009)	No control (cosmesis)	Low cost	• Can perform only a single operation.	

			• Heavy	
(Poddar, et al., 2021)	Potentiometers and linear actuators	Can be used as a prosthesis testing framework	• Cannot perform high-speed activities. • Not mobile	
(Hao, et al., 2021)	EMG	• Monolithic • Grip force control using extra sensors	• Extra sensors make it heavier	Similar to (Zheng, Cao, Qian, Chen, & Zhang, 2016)
(Palermo, 2017)	sEMG	Non-invasive	• Long training time • A limited number of finger movements	
(Liu, et al., 2018)		Compliant		Similar to (Mutlu, Alici, Panhuis, & Spinks, 2016)
(Mohammadi, et al., 2020)	EMG	• Low weight (230 gr) • Ideal for children aged 5 – 7 years. • Independent finger actuation	Limited functionality (Can only perform two grip types)	Similar to (Mohammadi, et al., 2020)
(Castro, et al., 2022)	• EMG • CNN (Computer vision)	High software accuracy (99%)	No means to control grip force	
(Abbasi, et al., 2019)	Unsupervised learning	• Good software performance	• Too expensive • Not practical	

		(aggregated 20 grasps into 9 clusters)		
(Slade, et al., 2015)	• EMG • CNN	Good grip type determination accuracy	No grip control, it fails when picking delicate objects.	
(Ghazaei, et al., 2017)	• EMG • Neural Network	• Uses grip classification . • Low training time		
(Beyrouthy, et al., 2016)	• EEG		Difficulty in analyzing EEG data	
(Li, et al., 2018)	• Facial Expression • EEG	High accuracy		
(Warwick, et al., 2003)	• EEG	✓	Invasive (amputee requires surgery)	
(Shi, et al., 2022)	• Augmented reality • Eye-tracking • EMG	• High efficiency and accuracy	Eyewear transmits data through Wi-Fi, so the system will fail in a Wi-Fi dead zone.	
(Lau, 2009)	• DC • SMA	• More active finger DoF movement • High grip force		Similar to (Low, et al., 2013)

2.6 Conclusion

In conclusion, this literature review has revealed the importance and significance of this study in addressing the knowledge gap in prosthetic hand development. The findings from the reviewed literature demonstrate that while there have been notable advancements in prosthetic hand technology, there still exists a lack of the automated gripping and hand gesture capabilities without extensive training or added weight.

This research not only contributes to the advancement of prosthetic hand technology but also has broader implications in the field of human-machine interaction and assistive technologies. The methodologies and findings presented in this thesis can serve as a foundation for future advancements in the design and control of robotic systems, with potential applications in healthcare, rehabilitation, and industrial automation.

CHAPTER 3

METHODOLOGY

3.1 Introduction

Chapter 3 of this thesis focuses on the tools and libraries used for the development and testing of robotic systems. Robotics simulation plays a crucial role in the development of robotic systems by allowing researchers and developers to test and refine control algorithms, hardware components, and overall system designs before deploying them in the real world. This chapter highlights the importance of selecting the appropriate tools and libraries for robotics simulation and presents an in-depth analysis of the most popular options available.

3.2 Python Programming Language

Python is a high-level, interpreted programming language (Python Software Foundation, 2023) that has become increasingly popular in recent years for use in robotics simulation. Its simplicity, readability, and flexibility make it an ideal choice for developing and testing algorithms and control systems in simulated environments.

Python is also one of the best and most popular tools used in the field of AI. One of the main advantages of using Python for robotics simulation is its ease of use. Python is known for its simple syntax and intuitive structure, which makes it easy to write and read code. This means that developers can spend less time debugging code and more time focusing on developing algorithms and control systems. Python's simplicity and use make it the perfect language for creating AI programs. Python's syntax is simple to read and write, which facilitates the creation and upkeep of sophisticated AI algorithms. Additionally, Python offers a large selection of modules and frameworks that facilitate the execution of numerous AI tasks, including deep learning, machine learning, and natural language processing.

Python's adaptability makes it a good choice for robotics simulation. Python is a general-purpose programming language, so it may be used for a variety of projects, including web development and data processing. This makes it simple to incorporate many software programs and libraries into the robotics simulation environment, such as NumPy (NumPy Homepage, 2005) for scientific computing, Pandas (Beazley, 2012) for data analysis, and OpenCV (Corporation, 2001) for computer vision.

3.3 PyBullet

PyBullet (Collins, et al., 2019) is a physics simulation engine (Figure 3-1) designed for use in robotics, machine learning, and computer graphics applications. It is a popular open-source alternative to other physics engines such as PhysX (Maciel, et al., 2009) and Havok (Filipczuk & Nikiel, 2008), and is widely used in both academic and industrial settings. PyBullet is built on top of the Bullet physics engine (Chung & Pollard, 2016), which provides accurate and efficient physics simulations for a wide range of applications.

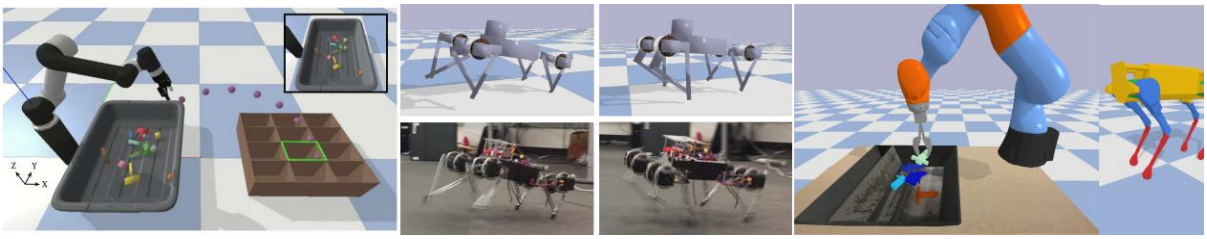


Figure 3-1 Pybullet simulation environment (Bullet, 2022)

The choice of simulation engine is a critical decision for anyone working in robotics, as it can significantly impact the development and testing of robotic systems. A simulation engine is essentially a software platform that models the behavior of physical systems in a virtual environment, allowing researchers and developers to test and refine control algorithms, hardware components, and overall system designs before deploying them in the real world. Different simulation engines have different strengths and weaknesses, and the choice of engine

can have a significant impact on the accuracy, speed, and realism of the simulation, as well as the cost and complexity of the development process.

For this research, we initially compared the performance of the three most popular simulation engines - Mujoco, Gazebo, and PyBullet. Mujoco (DeepMind Technologies Limited, 2021) is a proprietary physics engine that is particularly well-suited for simulating articulated bodies, such as humanoids and robotic arms. It provides high-fidelity simulations (Figure 3-2) that are accurate and computationally efficient, making it a popular choice for research in fields such as biomechanics and robotics. However, its proprietary nature means that it can be expensive to use and limits the ability of researchers to modify or extend the engine to meet their specific needs.

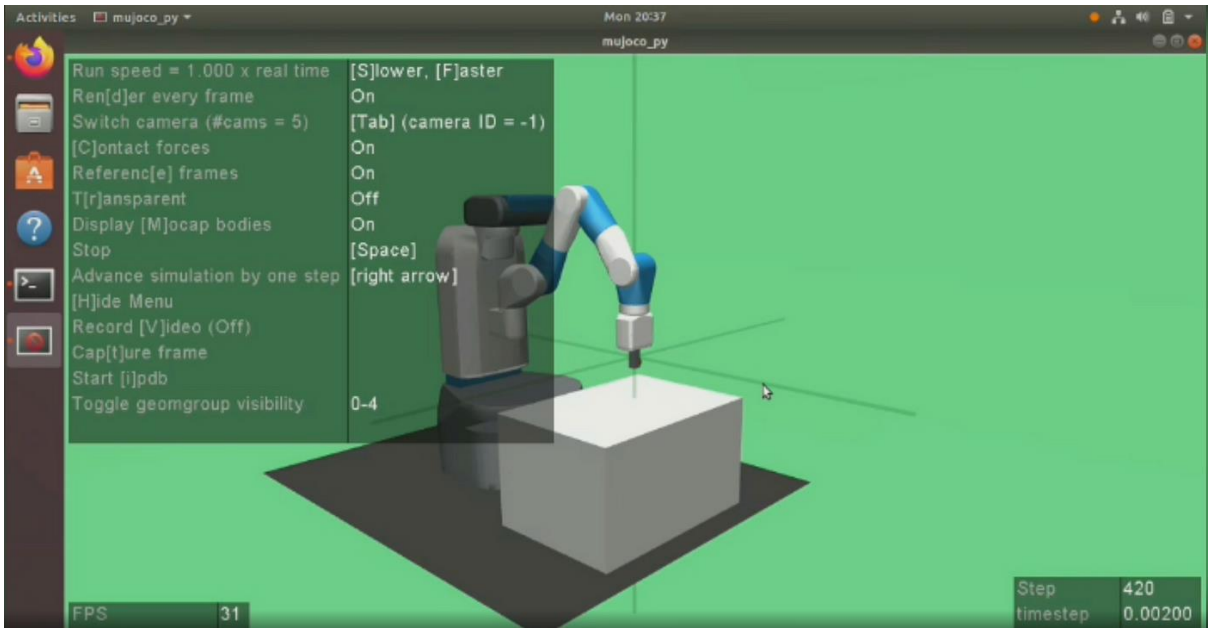


Figure 3-2 MuJoCo simulation environment (Dixit, 2020)

Gazebo (Meyer, et al., 2012), on the other hand, is an open-source simulation engine that is particularly well-suited for simulating robots in dynamic environments. It provides a range of sensors and actuators, as well as tools for visualizing and analyzing simulation data, making it a popular choice for testing and validating control algorithms and overall system designs.

Additionally, because it is open source, it is highly customizable and can be modified and extended to meet specific research needs. However, because it is designed to simulate complex environments (Figure 3-3), Gazebo can be computationally intensive, making it less suitable for certain types of simulations.

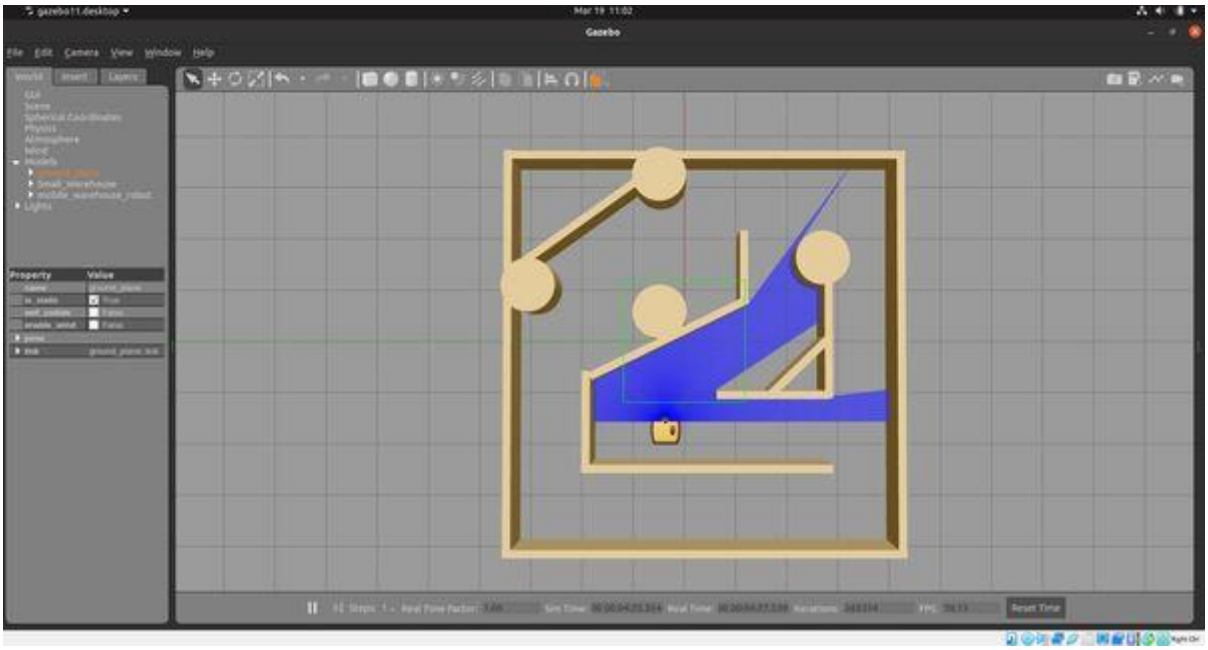


Figure 3-3 Gazebo simulation of a warehouse robot (Automatic Addison, 2021)

3.4 Stable Baselines

Stable Baselines is a set of reinforcement learning algorithms that are designed to be reliable, easy to use, and scalable. The library provides a range of algorithms, including Soft Actor-Critic (SAC), Proximal Policy Optimization (PPO) and Deep Q-Networks (DQN) which were compared in this study. These algorithms are implemented in a modular and extensible way, allowing users to customize them for their specific needs. Stable Baselines also provides a range of utilities for working with reinforcement learning environments, such as wrappers for Gym environments and utilities for visualizing training progress.

3.4.1 Key Features of Stable Baselines

Stable Baselines has several key features that make it a popular choice for reinforcement learning research and development.

1. Modular and Extensible Architecture: Stable Baselines is built on top of Tensorflow (Schrimpf, 2016), allowing users to easily customize the algorithms for their specific needs. The library provides a modular and extensible architecture, making it easy to add new algorithms, wrappers, and utilities.
2. Easy to Use: Stable Baselines provides a simple and intuitive interface for training and evaluating reinforcement learning models. The library provides a range of pre-trained models that can be easily loaded and used for different applications.
3. High Performance: Stable Baselines is designed to be efficient and scalable, making it suitable for large-scale reinforcement learning problems. The algorithms are optimized for multi-core CPUs and GPUs, allowing for fast training times.
4. Stable and Reliable: Stable Baselines algorithms are designed to be stable and reliable, providing consistent performance on a wide range of reinforcement learning problems. The library provides several features for improving stability, such as reward scaling, gradient clipping, and early stopping.
5. Comprehensive Documentation and Support: Stable Baselines has comprehensive documentation and a large community of users and developers, providing support and guidance for users at all levels.

3.5 OpenAI Gym

OpenAI Gym is a toolkit for developing and comparing reinforcement learning (RL) algorithms. It provides a standardized interface for RL environments, which allows researchers

and developers to focus on the algorithmic aspects of RL without having to worry about the specifics of individual environments.

The Gym interface consists of a set of standard methods for interacting with the environment, including methods for getting the current state of the environment, taking actions, and receiving rewards. The interface also provides a way to reset the environment to its initial state and to retrieve information about the available actions and observations.

3.6 Summary

Chapter 3 discussed the tools and libraries used in robotics simulation, specifically Python programming language, PyBullet simulation engine, and Stable Baselines reinforcement learning algorithms. Python's simplicity, readability, and flexibility make it an ideal choice for developing and testing algorithms and control systems in simulated environments. PyBullet is a physics simulation engine designed for use in robotics, machine learning, and computer graphics applications. It is widely used in both academic and industrial settings due to its accurate and efficient physics simulations. Stable Baselines is a set of reinforcement learning algorithms that are designed to be reliable, easy to use, and scalable. The library provides a range of algorithms, including SAC, PPO, and DQN. The chapter compares the performance of different simulation engines and concludes that PyBullet works best for the research's use case, which is why it was chosen as the simulation engine. The chapter's focus is on the tools and libraries used in robotics simulation, emphasizing the advantages and limitations of each.

CHAPTER 4

EXPERIMENT SETUP 1 – FINGER PATTERN PREDICTION

4.1 Introduction

The goal of this simulated experiment is to train tree-based classification models that can accurately predict the gesture that the user is intending to make based on the EMG data measured from the electrical activities or the user's muscles.

4.2 Dataset Description

The dataset used in this experiment was collected from an open source (Nasri, et al., 2019), freely available for use under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 License. The EMG device used in this open source is an 8-channel EMG armband shown in Figure 4-1. The Myo armband is a gestural interface which allows researchers to access raw Inertial Measurement Unit (IMU) and 8-channels of 8-bit EMG data. It communicates raw data via Bluetooth and streams IMU data at 50Hz and EMG data at 200Hz. Data is taken from the Myo library within the Software Development Kit (Thalmic Labs, 2015).

The MYO Armband stands as a reliable and versatile tool for capturing EMG data and enabling a wide range of applications. Researchers have successfully utilized the MYO Armband to control a robotic arm with 6 Degrees of Freedom, achieving comparable speed and precision to the user's own movements (Widodo et al., 2018). Furthermore, studies have reported high classification accuracy in gesture recognition tasks using the MYO Armband with its practical applications in the areas such as physiotherapy, rehabilitation, and sign language classification (Kaur et al., 2016; Abreu et al., 2016). The device has also proven to be a cost-effective alternative to more expensive non-invasive electromyography methods in the field of amputation patient rehabilitation (Abduo and Galster, 2015).

Figure 4-1 8-Channel EMG band (Thalmic Labs, 2015)

The participants had varying physical characteristics such as height and weight and were in the age range of 20 to 35 years old. The dataset was designed to have a diverse range of participants to ensure that the models can generalize well to different individuals. Although the dataset is small, the combination of careful pre-processing and participant diversity contributes to the validity of our findings.

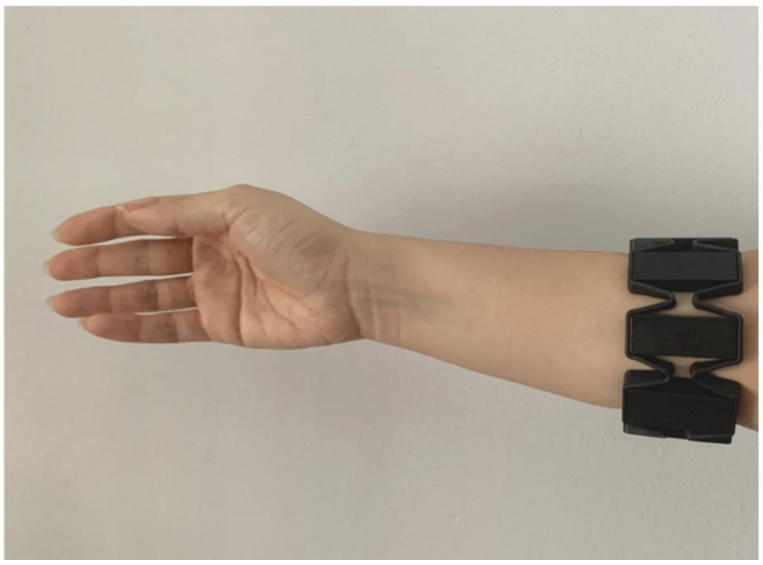


Figure 4-2 EMG signal data collection through electrodes placed on the forearm (Jiang, et al., 2022)

The dataset in this study consists of the Myo EMG data collected from 15 participants who performed 7 different hand gestures. The dataset was structured as a collection of folders with

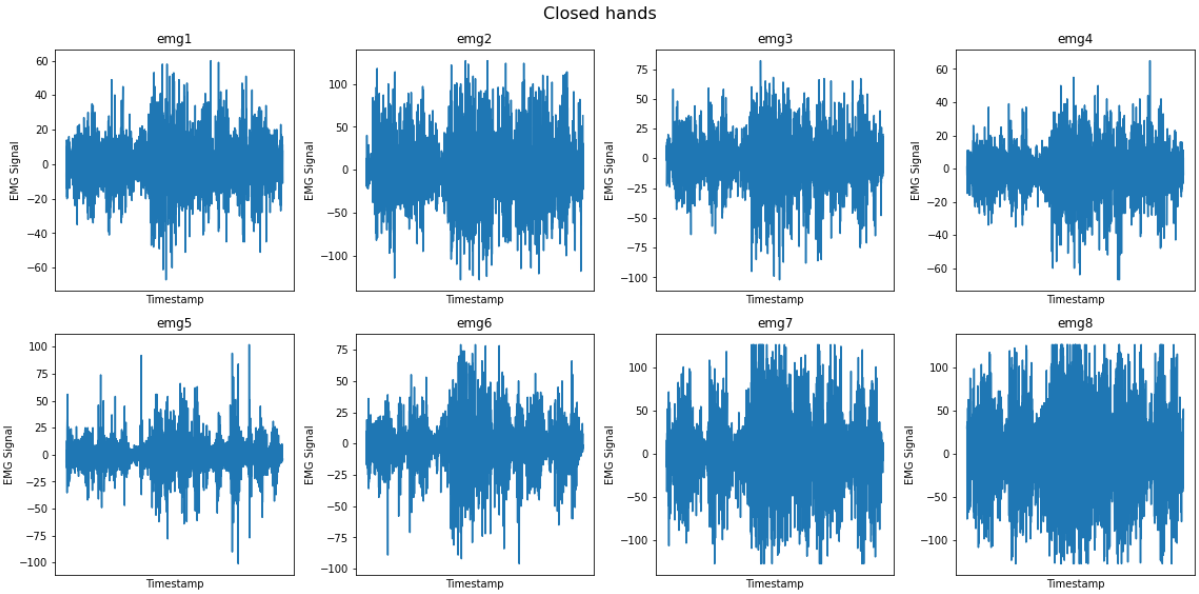
each folder containing Comma-Separated Values (CSV) files that contain the EMG signals recorded from an individual performing a particular gesture. In some cases, there were two CSV files for a single gesture, indicating that the gesture was performed twice by the same participant. The signals from the publicly available dataset were on the gestures shown in Figure 4-2.

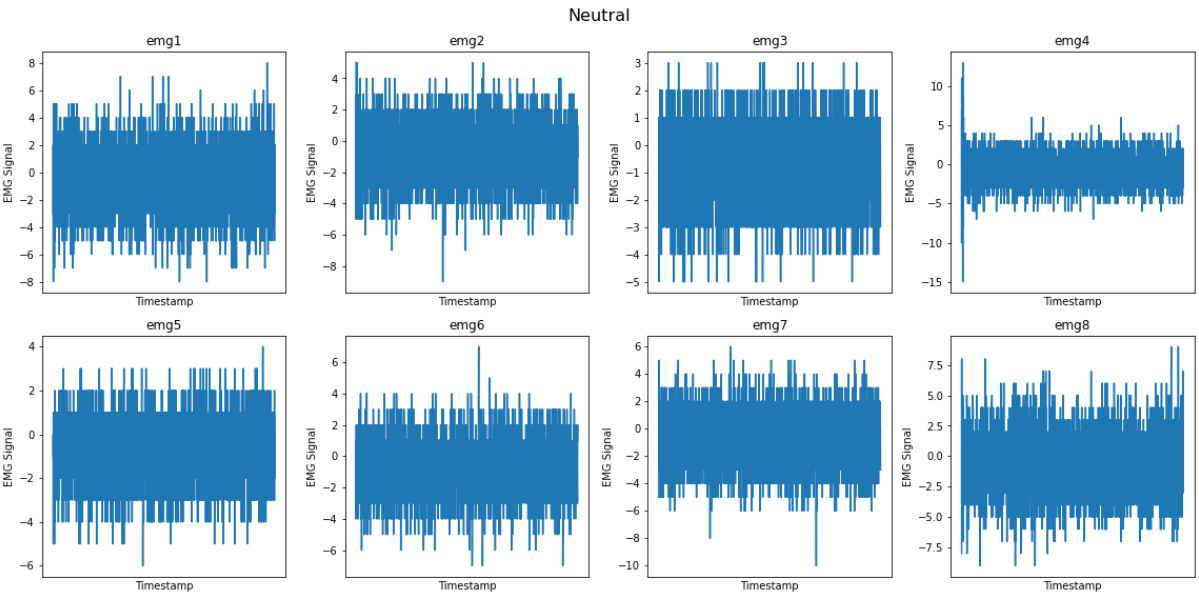
Figure 4-3 List of gestures

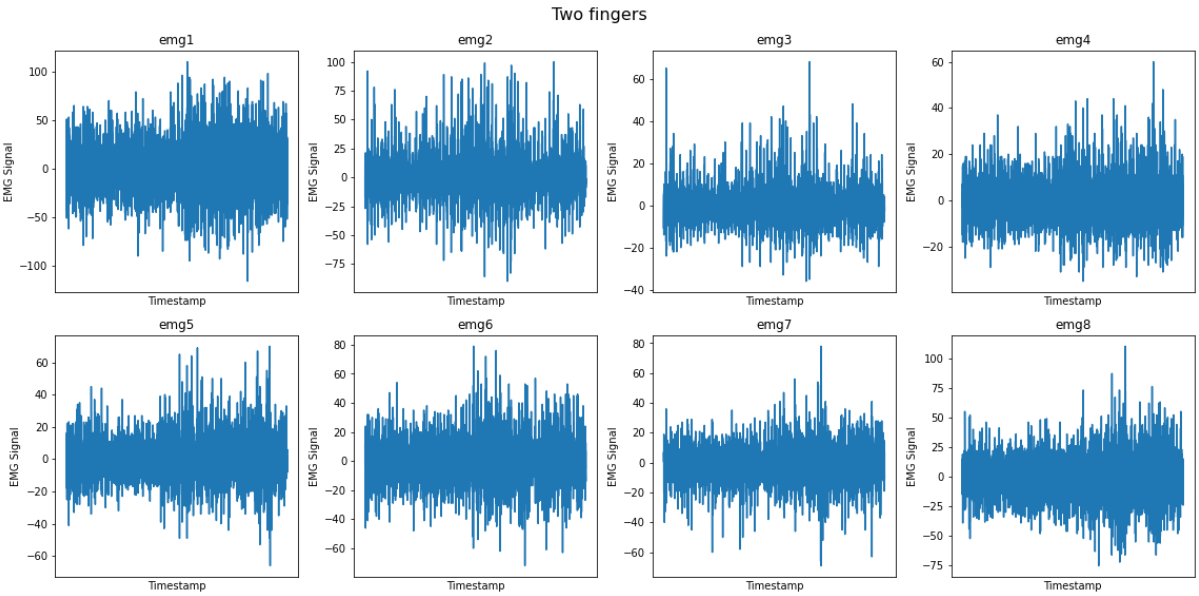
The signal information for all gestures for the participant is shown in Figure 4-3. The signal is represented as a range of values from 128 to -128, which are the raw EMG signal values captured by the Myo armband. The signals were collected in 8 different locations on the wrist. The signals were collected at times in the precision of nanoseconds.

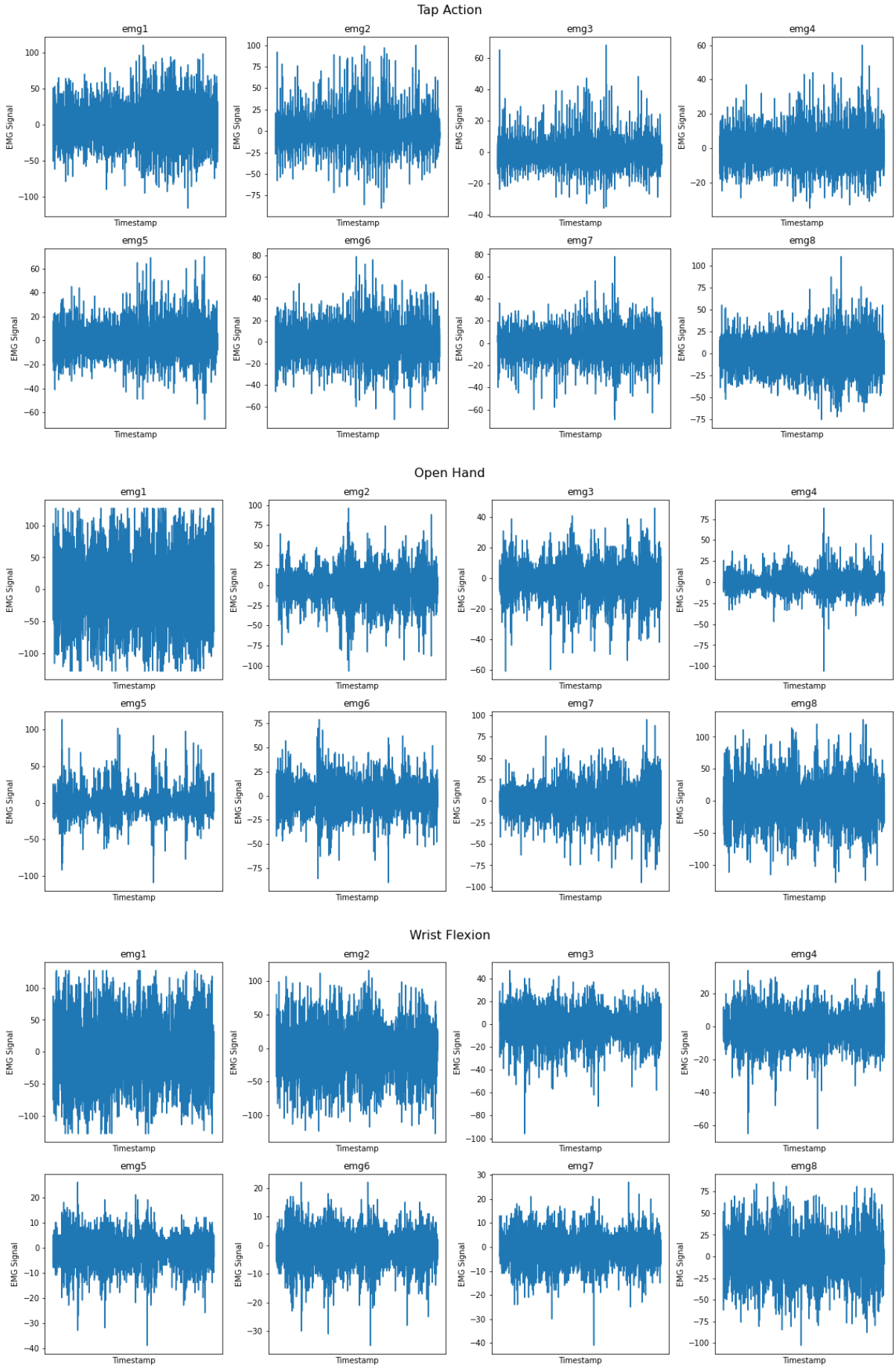
The issue of imbalanced data in supervised learning is a common challenge that needs to be taken note of to ensure the reliability and validity of research findings when searching for the right data. Imbalanced data refers to a situation where the distribution of classes in the dataset is uneven, leading to a potential bias in the trained model. To mitigate this issue, this thesis has taken several steps. First, the dataset used in this study was carefully evaluated of the diversity of participants with varying physical characteristics, thereby increasing the representation of different classes. This approach aims to reduce the bias towards the majority classes and improve the model's ability to generalize to the minority classes. Additionally, the performance metrics were employed beyond simple accuracy, such as precision, recall and F1-score, to evaluate the model's performance comprehensively. By considering these metrics, we a more robust assessment of the model's effectiveness in correctly classifying both the majority and minority classes can be achieved.

In the context of training tree-based classifiers to predict hand gestures using EMG data, it is essential to consider the suitability of the dataset size for achieving reliable and accurate results. While a small dataset of 15 participants might be considered limited in other contexts, it can still provide valuable insights and yield meaningful outcomes in the specific domain of hand gesture recognition in this study. The effectiveness of tree-based classifiers lies in their ability to capture complex patterns and relationships in the data, often requiring less training data compared to other machine learning algorithms. Additionally, the Myo armband is well sophisticated to capture clean muscle activities by minimizing noises and artifacts in the EMG data, further enhancing the reliability of the dataset.


Closed hands
emg1
emg2
emg3
emg4
emg5
emg6
emg7
emg8
EMG Signal
Timestamp


Neutral
emg1
emg2
emg3
emg4
emg5
emg6
emg7
emg8
EMG Signal
Timestamp


Two fingers
emg1
emg2
emg3
emg4
emg5
emg6
emg7
emg8
EMG Signal
Timestamp


Tap Action
emg1
emg2
emg3
emg4
emg5
emg6
emg7
emg8
EMG Signal
Timestamp
Open Hand
emg1
emg2
emg3
emg4
emg5
emg6
emg7
emg8
EMG Signal
Timestamp
Wrist Flexion
emg1
emg2
emg3
emg4
emg5
emg6
emg7
emg8
EMG Signal
Timestamp

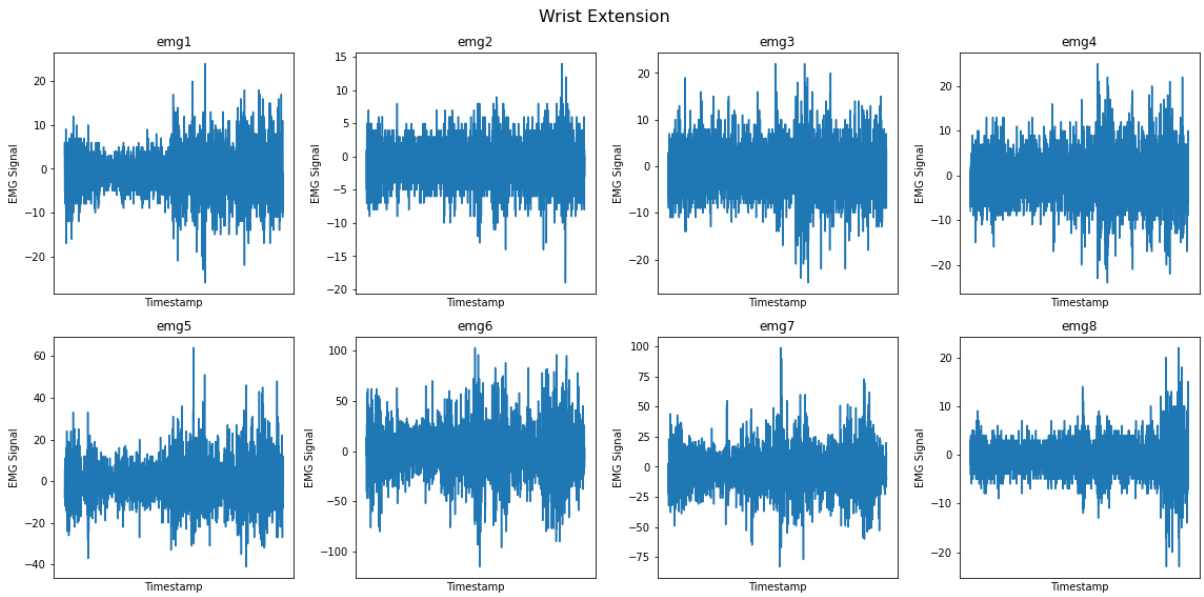


Figure 4-4 EMG signal data for the first participant

4.3 Data Pre-processing

Data preprocessing is an essential step in data analysis that involves cleaning, transforming, and preparing the data before it is used for analysis. The goal of data preprocessing is to ensure that the data is in a format that can be easily analyzed by statistical and machine learning algorithms. The data is preprocessed to improve the quality of the data and make it more suitable for analysis or modeling. It also helps to reduce the risk of errors or biases in the analysis, leading to more accurate and reliable results. In this experiment, the preprocessing steps included splitting the dataset into training and testing sets and scaling the features using the standard scaler.

Splitting the dataset (Figure 4-4) is crucial to evaluate the performance of the model on unseen data. The data was randomly split into training and test sets with a splitting ratio of 80:20, where 80% of the dataset was used for training and 20% for testing.

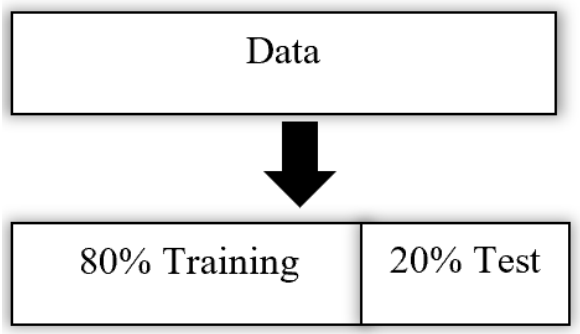


Figure 4-5 Data Splitting

After splitting the dataset, the next step was to scale the features. Feature scaling is necessary because some machine learning algorithms are sensitive to the scale of the input features.

4.3.1 Standard Scaling

The feature scaling technique used for this experiment is Standard Scaling (Pedregosa, 2011). Standard Scaling is a preprocessing technique that is used to transform data so that it has a mean of zero and a standard deviation of one. It is an essential step in many machine learning algorithms that involve numerical data, as it can help improve the performance of the algorithm and make the data more interpretable.

The standard scaling works by subtracting the mean of each feature and then dividing by its standard deviation. This transformation can be expressed mathematically in Equation 1:

$$z = \frac{x-\mu}{s} \tag{4.1}$$

where z is the transformed value, x is the original value, μ is the mean of the feature, and s is the standard deviation of the feature.

The main idea behind the Standard Scaling is to ensure that all features are on the same scale, so that one feature does not dominate over the others in the analysis or modeling. By standardizing the data, we ensure that each feature has equal importance in the analysis or modeling, and that the algorithm is less sensitive to the scale of the data. This can lead to more

accurate and robust results, as well as a better understanding of the relationships between the features.

4.4 Feature Selection

Feature selection is a crucial step in any machine learning pipeline. The purpose of feature selection is to identify the most relevant features from the dataset, which can improve the model's performance and reduce its complexity. Feature selection methods help to identify a subset of features that are most informative for the classification or regression task at hand. In this experiment, we used feature selection to identify the most informative features from the dataset, which can help to improve the accuracy of the classification model.

In this experiment, we used the f_classif method (Pedregosa, 2011), which is a statistical technique that evaluates the relationship between each input feature and the target variable. This method is based on analysis of variance (ANOVA), which is a statistical method used to determine whether the means of two or more groups are significantly different. In this case, the groups are the different hand gestures, and the mean is the value of the EMG signal for each feature.

The f_classif method calculates the F-value for each feature, which is a measure of the ratio of the variance between the groups to the variance within the groups. The F-value is then converted to a p-value, which is a measure of the probability of observing the F-value under the null hypothesis that the means of the groups are equal. A low p-value indicates that the feature is highly significant and is likely to be informative for the classification task.

The feature selection step is performed separately on the training and test sets to avoid data leakage. Data leakage occurs when information from the test set is used to select the features, which can lead to overfitting and poor generalization performance. Therefore, we fit the feature selector on the training set and transform both the training and test sets using the same selector.

This ensures that the model is only using the selected features for classification and not any additional information from the test set.

4.5 Hyperparameter Tuning

Hyperparameter tuning is a crucial step in machine learning, especially in building models that can achieve optimal performance. The goal of hyperparameter tuning is to find the best hyperparameters that result in the highest accuracy and lowest error. Hyperparameters are parameters of the machine learning model that are set prior to training, such as the number of trees in a random forest or the learning rate in a neural network.

In this experiment, a grid search cross-validation (Pedregosa, 2011) was used to tune hyperparameters. Grid search is an exhaustive search of all possible hyperparameter combinations within a given range or set of values. Cross-validation (Figure 4-5) is a technique for assessing the performance of a machine learning model by training and testing on different subsets of the data.

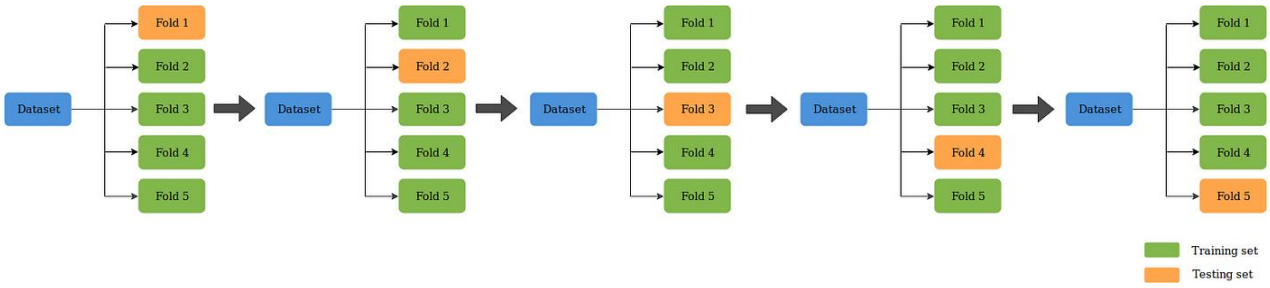


Figure 4-6 5 folds cross validation (Krishni, 2018)

The following parameters were tuned (Table 4-1):

1. Maximum depth of the decision tree: This is the maximum number of levels allowed in the decision tree. A deeper tree can fit the training data better, but it can also overfit the data and perform poorly on new data.

2. Maximum number of features considered for splitting each node: When splitting a node in the decision tree, this parameter determines the maximum number of features to consider. This can help prevent overfitting and improve generalization performance.
3. Minimum number of samples required to split an internal node: This parameter specifies the minimum number of samples required to split an internal node. If a node has fewer samples than this value, it will not be split.
4. Minimum number of samples required to be at a leaf node: This parameter specifies the minimum number of samples required to be at a leaf node. If a leaf node has fewer samples than this value, it will be considered as a noise point and ignored.

Table 4-1 Parameter grid

Parameters				
Maximum depth	3	5	10	None
Maximum features	None	Square root function	Log 2	
Minimum samples split	2	5	10	
Minimum samples leaf	1	2	4	

4.6 Model Training

The model was trained using two popular tree-based models: Decision Tree classifiers and Random Forest classifiers. These non-linear models have shown very high accuracy in identifying relevant features as well as providing a clear and intuitive decision rule making them very easy to interpret. Asides being easier to train, they have shown better performance than deep learning in several domain (Grinsztajn, et al., 2022; Lundberg, et al., 2020).

1. Decision Trees: Decision trees are a type of supervised learning algorithm used for classification and regression analysis (Kingsford & Salzberg, 2008). They work by recursively partitioning the data set into smaller subsets based on the values of a selected attribute or feature. The process starts with the root node, which represents the entire data set, and then splits into child nodes based on a certain attribute's value (Figure 4-6).

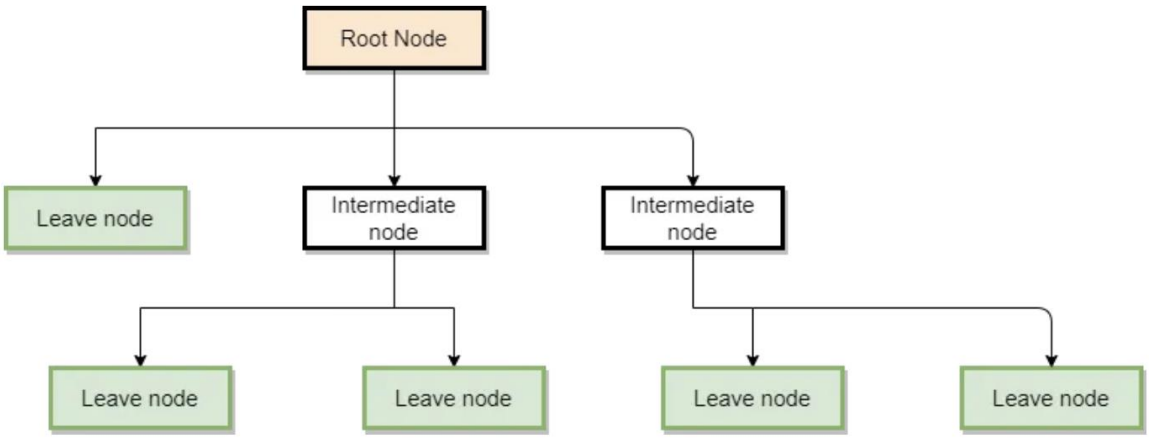


Figure 4-7 Decision tree (Thorn, 2020)

The algorithm selects the best attribute to split the node based on an impurity measure like Entropy (Equation 4.2).

$$E = -\sum_i^C p_i \log_2 P_i \tag{4.2}$$

where p_i is the probability of randomly picking an element of class i. The splitting continues recursively until the leaf nodes represent a homogeneous class, i.e., all the samples in the node have the same label. The final decision tree is constructed by connecting the nodes and their splits.

2. Random Forest: Random Forest is an ensemble learning algorithm that constructs multiple decision trees and aggregates their results to make a final prediction (Ali, et al., 2012). Each tree is constructed using a randomly selected subset of the training data set and a randomly selected subset of the features. This randomness introduces diversity among the trees, which helps to reduce overfitting and improve the accuracy of the

predictions. During prediction, the algorithm takes the majority vote of all the trees to make the final prediction.

4.7 Model Integration with Prosthetic Gripper

As stated in the previous chapter, all simulations for this experiment were performed in the pybullet simulation environment. Once the classifier has been trained, it can be integrated with the PyBullet simulation. The output from the classifier can be used to control the movement of the robotic arm, allowing it to replicate the natural movements of the user's limb. This is achieved by mapping the predicted gesture to a set of movements of the arm's joints.

During the simulation, the arm's movements are controlled in a cyclical manner using a series of decision statements. The current state of the arm is determined by the duration of the previous state and the time elapsed since the start of the simulation. By adjusting the duration of each state and the movements associated with each state, a more intuitive movement pattern for the arm can be simulated.

The gripper used for this simulation is PyBullet's wsg50 two finger gripper (Figure 4-6).

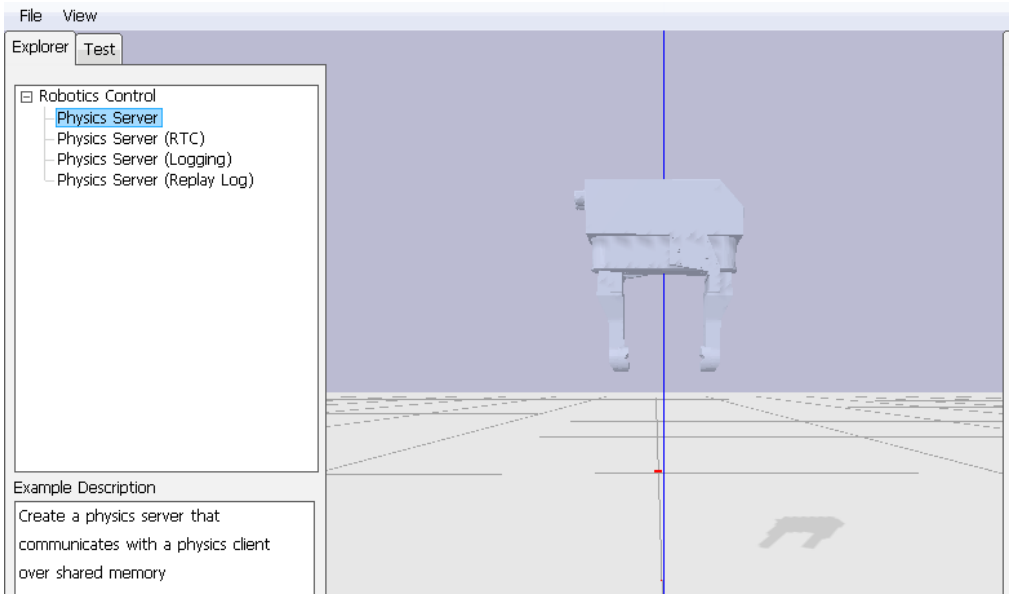


Figure 4-8 Wsg50 two finger gripper

4.8 Summary

Chapter 4 discussed the method of developing a model that accurately predicts the gesture the user intends to make based on EMG data from their muscles. The dataset consists of Myo EMG data from 15 participants, including seven different hand gestures, with the signals collected at eight different locations on the wrist. The data was preprocessed by splitting it into training and testing sets and scaling the features using the standard scaler. The feature selection method used was f_classif, which evaluated the relationship between each input feature and the target variable based on analysis of variance. The feature selection was performed separately on the training and test sets to avoid data leakage. This chapter covers methods to improve the accuracy of the classification model and reduce its complexity.

CHAPTER 5

EXPERIMENT SETUP 2 – GRIP CONTROL

5.1 Introduction

This section covers the experimental setup in using reinforcement learning to train the prosthetic arm gripper to be able to grip an object in minimum time and with optimal force. This experiment builds on the codebase provided by (Baris, 2020), the changes to the code made from this study are provided in the Appendices section.

5.2 Experiment Process

The gripper was simulated and visualized in a simple PyBullet environment which includes a plane to set the world and a table at the center as shown in Figure 5-1. Objects are randomly spawned on the table from a defined height with the goal of having the gripper locate these objects and grip them. In a real-world scenario, the gripper would be mounted as the end effector of a prosthetic arm. The human would need to place the end effector over an object to be grasped while the gripping action by the end effector will be automatically controlled by the gripper.

An RGBD camera is mounted on the gripper which returns information on the observed state. As seen in the figure below, the camera returns information to the gripper from three perspectives: RGB data, depth data and the segmented mask

The gripper positions itself based on actions it receives in the form of a collection of values (dx, dy, dz, ∅, open/close gripper). These values are in the range of -1 to +1 where dx, dy, dz represent the step translation the gripper needs to make in the x, y and z direction respectively. ∅ represents the yaw rotation amd the last value signifies whether to open or close the gripper. Once again, these actions are in play here to run the simulation. In a real-world scenario, the

gripper would need information on just the yaw and gripper open/close action since the human would have to translate the gripper in the x, y or z directions.

Rewards or Punishments are given in every episode depending on different factors like whether the gripper accomplished its task or how long time it took to accomplish that task. An episode ends when the gripper successfully picks up an object or if unable to after a set number of steps.

A shaped reward function was used tailored to the specific gripping task as opposed to using a regular binary reward function.

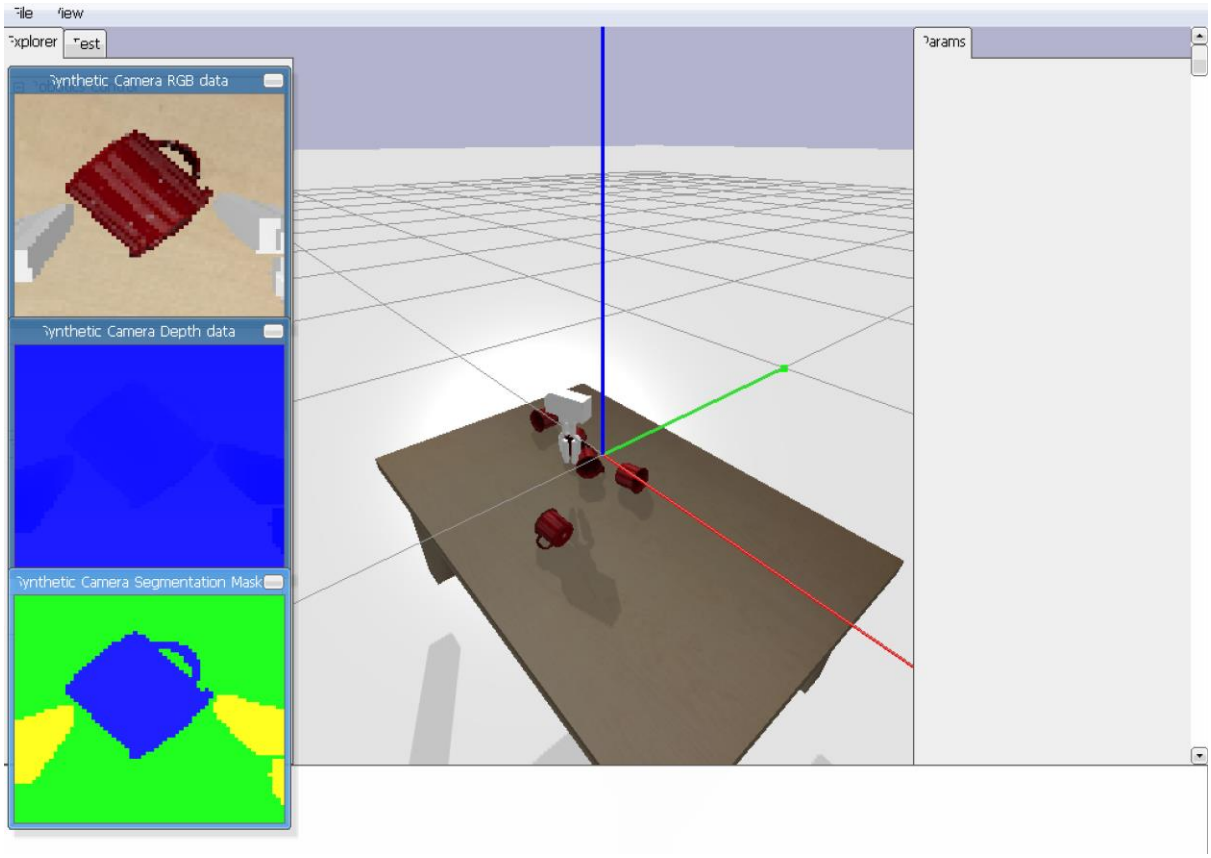


Figure 5-1 Simulation scene

5.2.1 Environment Assumptions

1. The simulation environment is deterministic, i.e., given the same initial state and actions, the environment will always produce the same outcome.

2. The action space and observation space of the environment are well-defined and consistent throughout the training and testing phases.
3. No external condition can cause the objects to move other than the gripper or other spawned objects in the environment.

The environment parameters are summarized in Table 5-1.

Table 5-1 Environment parameters

	Environment
Action space	5
Observation	RGBD and Depth data
Reward	Shaped reward
Discount factor	0.99

5.3 Gripper Model

The gripper used for this simulation is the WSG50 gripper (Figure 5-2, Figure 5-3) was used. The WSG50 is a high-precision, two-finger parallel gripper designed for a wide range of automation applications.

Figure 5-2 WSG50 gripper (Weiss Robotics, 2015)

The WSG50 gripper is provided as a built-in robot in the PyBullet library files making it off the box compatible with the PyBullet simulation environment. The gripper is made up of 8 links and 8 joints (Table 5-2). A two-finger gripper was used to make it easier to set up the simulation.

Table 5-2 WSG50 gripper links and joints

Links	Joint	Joint type
Base link	Base joint	Prismatic
Motor	Base joint motor	Prismatic
Left hinge	Motor left hinge joint	Revolute
Right hinge	Motor right hinge joint	Revolute
Gripper left	Gripper left hinge joint	Revolute
Gripper right	Gripper right hinge joint	Revolute
Finger left	Gripper finger left	Fixed
Finger right	Gripper finger right	Fixed

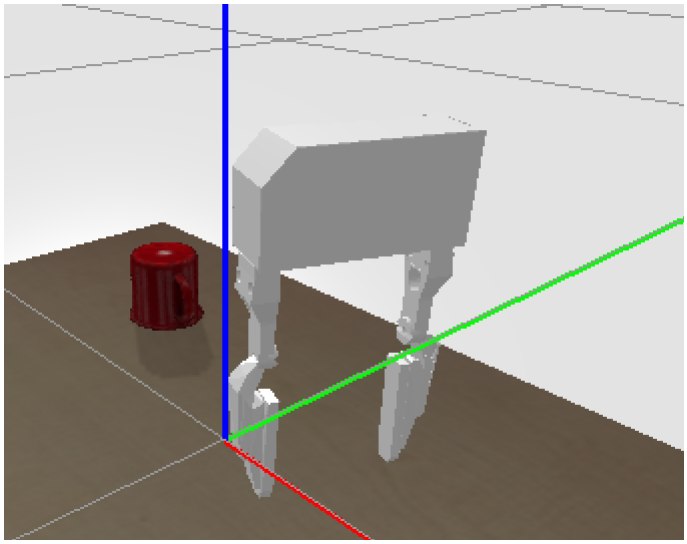


Figure 5-3 WSG50 model

5.4 Objects Dataset

One of the critical factors that determine the success of reinforcement learning for grasping and manipulation tasks is the quality of the training data. The training data must be diverse enough to cover the variability in object shapes, sizes, and textures that the gripper is likely to encounter in the household environment.

(Baris, 2020) made use of random wooden blocks for the training and testing the RL models which do not fully represent the use case of the gripper. For our experiment, we made use of the Pybullet URDF models library (Liu Z. , 2021) which contains a wide variety of household objects, including cups, plates, bowls, bottles, cans, and other everyday items (Figure 5-4). The models were sourced from various sources, including the YCB dataset, which is a collection of 3D models of household objects.

The models in the Pybullet URDF models library have been carefully designed to accurately represent the physical properties of the real-world objects they are based on. This means that when they are used in robot grasping simulations, the results will be realistic and representative of what would happen if the robot were interacting with the real-world object.

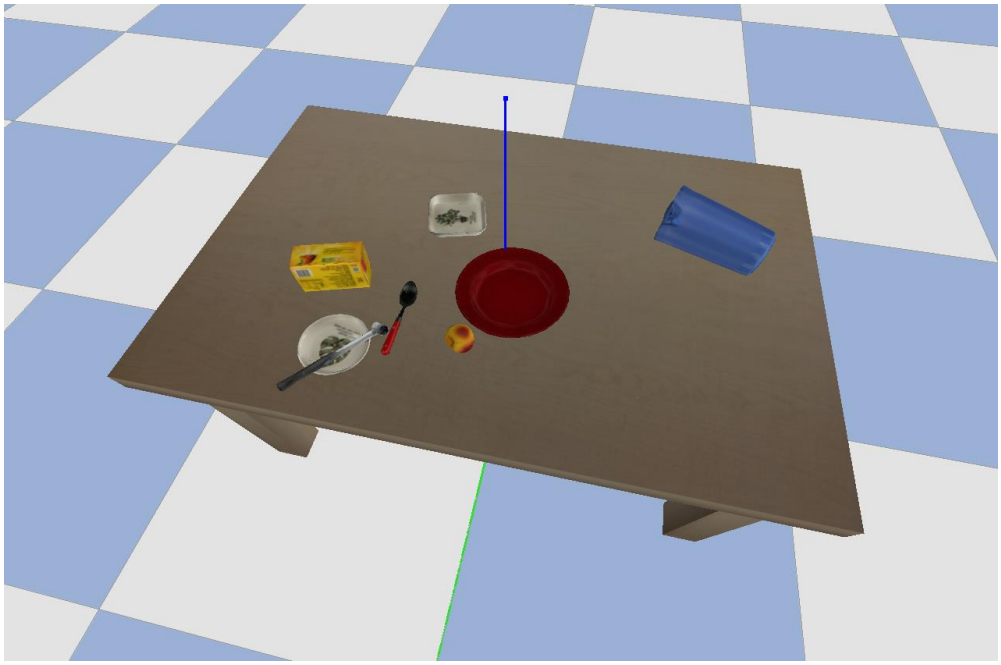


Figure 5-4 Some Pybullet URDF models

Due to gripper constraint such as:

1. Maximum width of the gripper: This refers to the maximum distance between the gripper fingers when they are fully opened. It is important to know this value to ensure that the gripper can grasp objects of the desired size.
2. Minimum width of the gripper: This refers to the minimum distance between the gripper fingers when they are fully closed. It is important to know this value to ensure that the gripper can grasp small objects without dropping them.
3. Gripper shape: This refers to the overall shape of the gripper, including the shape of the fingers and the angle at which they open and close. The gripper shape can affect the types of objects the gripper can grasp and the force it can apply to them.

Some household objects the gripper was trained on includes remote controller, mug (Figure 5-5) and soap bar.

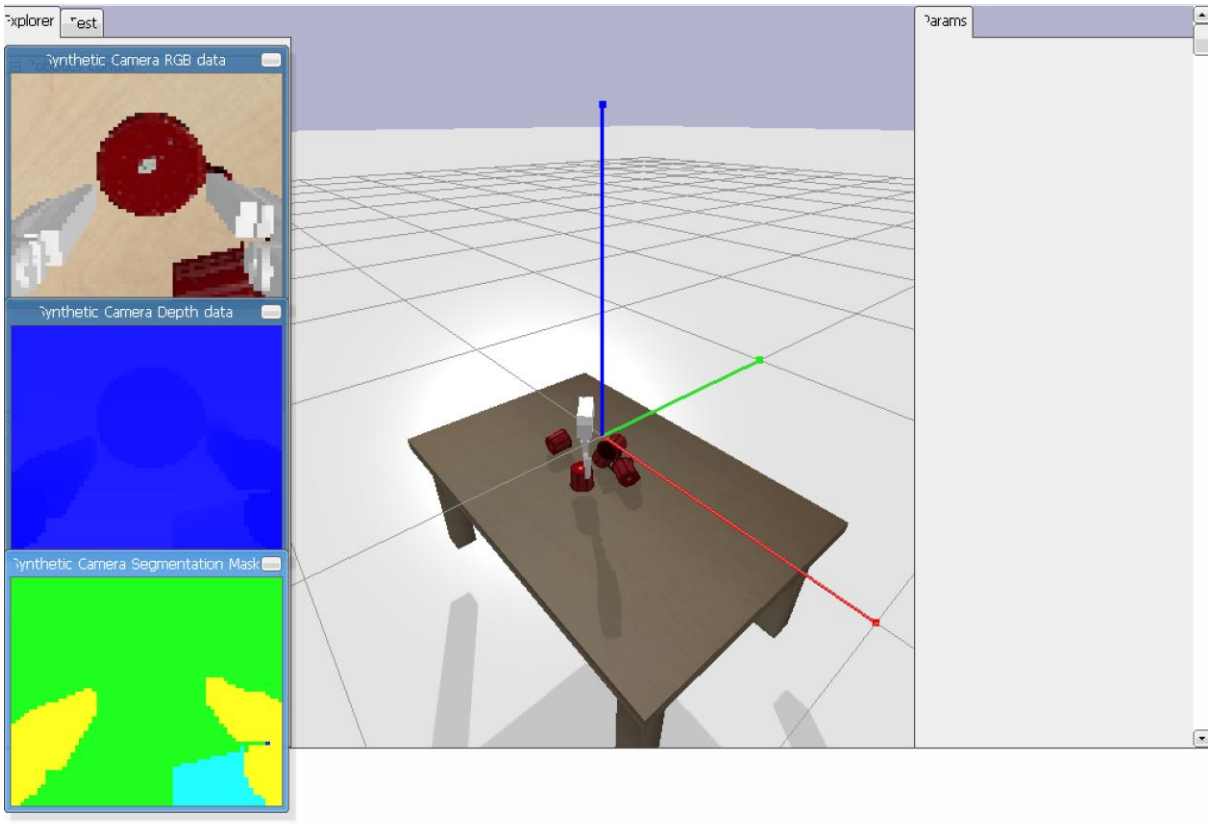


Figure 5-5 Gripping a red mug.

5.5 Sensor

The sensor in the task represents a camera mounted at the midpoint of the gripper's base which records observations captured in discrete steps. The observations captured by the sensor represents the state space in the Markov decision process. In an approach like (Baris, 2020), we implemented a perception pipeline using RGBD observations captured by the camera (Figure 5-6). RGBD stands for Red, Green, Blue, and Depth and refers to a type of imaging technology that combines traditional RGB color data with depth information. We used this to capture both color and depth information simultaneously. The depth information is represented as a gray-scale image, where each pixel represents the distance between the camera and the corresponding point in the scene.

Figure 5-6 RGBD data at specific time

The method to preprocess the RGBD data before feeding it into the convolutional neural network is like the method implemented by (Baris, 2020). Since our task involves a gripper with a certain width, we need to include this information in our processing of the sensor output.

To do this, we need to add the gripper width information to the sensor data we collect, without changing the shape of the sensor data. This means we can't just add the actuator width information as a separate piece of data. Instead, we need to add it in a way that the sensor data keeps its original shape.

To achieve this, we pad the gripper width information into a three-dimensional array that has the same shape as the sensor data (i.e., 64x64x1 or 64x64x4). This extra information is added as an extra "layer" to the sensor data, which we call a "channel". By doing this, we can make sure that the actuator width information is included in the processing of the sensor data without changing the way we process it.

During training, we remove this extra channel so that the robot can learn how to use the sensor data to control its movements, considering the width of the gripper. By including the gripper

width information in this way, we can ensure that the robot can use this information to perform the task we want it to do.

The RGBD data is then fed into a convolutional neural network to learn meaningful features and representations of the scene, which is used as input to the agent's decision-making process. the output is passed through one or more fully connected layers, which are also sometimes called dense layers. The output of the last fully connected layer can then be flattened into a one-dimensional array. The difference in the CNN architecture from the work of (Breyer, et., 2019) and the one used in this study, is the introduction of dropout layers (Srivastava, et al., 2014) which determines the fraction of the input units to drop out during the training of a neural network. The importance of this is to avoid overfitting which is a reoccurring problem in machine learning. We also made a switch from Leaky Rectified Linear Unit (ReLU) to Exponential Linear Unit because of its success in speeding up learning compared to other LUs. After the tensor output has been flattened into a one-dimensional array, the unpadded gripper width information is then concatenated to the RGBD tensor array, to form the observation vector. This process is represented with the schematic shown in Figure 5-7.

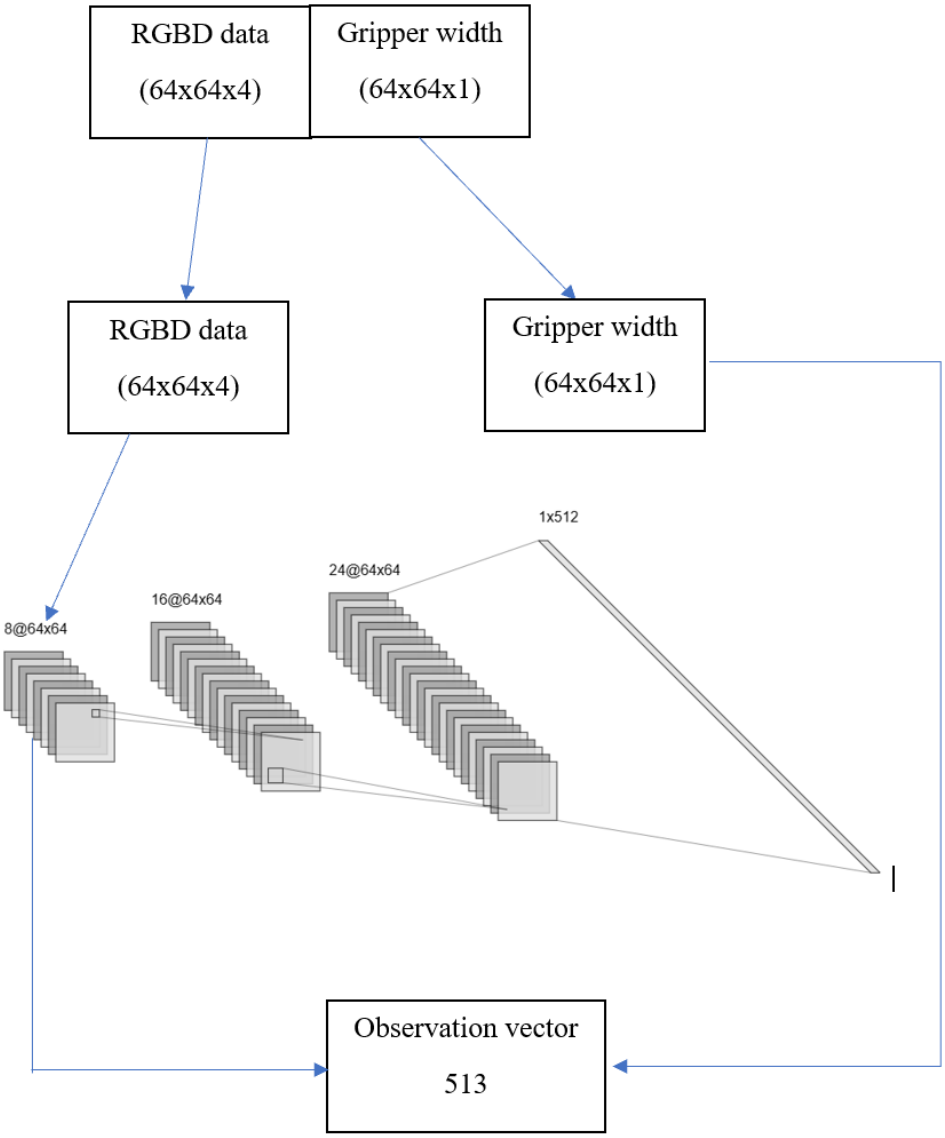


Figure 5-7 RGBD data to CNN process

5.6 Reward Function

The reward function implemented for this task is a custom-shaped reward function designed to incentivize the gripper's behaviour in the environment. The aim of the reward function is to incentivizes the agent to grasp the object efficiently and to lift it to the desired height quickly, while penalizing the agent for taking too long to complete the task. The reward function consists of several parameters that influence the reward value and can be summarized with the Table 5-3. Whether an object has been grasped is detected by checking the gripper width after the robot runs a close gripper event. The reward function parameters (Table 5-3) used are the

same with the work of (Baris, 2020), however, there were few problems with his implementation of the reward function which we attempted to solve.

1. Lack of exploration: The reward function does not provide any incentives for the agent to explore the environment. This can lead to the agent getting stuck in local optima and failing to discover better solutions. To solve this, we introduced a random exploration component to the reward signal (APPENDIX A). This is important because it encourages the agent to take actions that it hasn't taken before, which can help it discover new ways to interact with the environment and achieve the task.
2. Sparse rewards: The reward function is designed to provide a large reward only when the arm lifts the object by the desired amount. This can make learning difficult for the agent, as it may take a long time to receive any useful feedback. By adding a distance penalty, we encourage the agent to move the object towards the target position (APPENDIX A). This helps to alleviate the issue of sparse rewards, as the agent can receive a small reward for making progress towards the goal, even if it hasn't fully achieved it yet.

Table 5-3 Reward function parameters

	Non-terminal state	Terminal state
Object grasped?	$r_g - r_{tp}$	$r_t - r_{tp}$
Not grasped?	$-r_{tp}$	@timeout $-r_{tp}$

The terminal reward r_t is a large positive reward given to the agent upon successful completion of the task, and it is set to 10000. This reward acts as a powerful incentive to encourage the agent to complete the task efficiently and effectively.

The grasping reward r_g is a reward given to the agent upon detection of a successful grasp, and it is set to 100. This reward is intended to incentivize the agent to focus on grasping the object effectively and efficiently.

The time penalty r_{tp} is a negative reward given to the agent for taking too long to complete the task, and it is set to 200. This penalty encourages the agent to complete the task as quickly as possible, without sacrificing effectiveness.

5.7 Training Algorithms and Procedure

The agent was trained using three existing policies, SAC, DQN and PPO. The choice of these algorithms was based on their proven effectiveness and wide usage in the field (Breyer, et al., 2019). By leveraging the strengths of SAC, DQN, and PPO, it was anticipated that the agent could benefit from their respective advantages in terms of stability, sample efficiency, and performance. The hyperparameter configuration shown in Table 5-4, 5-5, 5-6 for each of these agents were selected on the premise of a similar work by (Baris, 2020)

Table 5-4 DQN Hyperparameters

Learning rate	Batch size	Prioritized replay
0.001	32	True

Table 5-5 SAC Hyperparameters

Max iteration	Batch size	Layers	Buffer size	Step size
400	64	[64, 64]	1000000	0.0003

Table 5-6 PPO Hyperparameters

Learning rate	Layers	Number of steps
0.003	[64, 64]	2000

Training begins at the reset state where the objects have the spawned on the table and the gripper also spawned mid air. For each of the algorithms tried during training, we defined the reward function that provide positive feedback when the agent takes actions that move it closer to the goal, and negative feedback when it takes actions that move it further away from the goal.

Next, we run the agent through a series of episodes in timesteps of 1000 in the environment. Each episode consists of the agent taking actions and receiving rewards. The agent's goal is to maximize the cumulative reward over all episodes. During each episode, the agent observes the current state of the environment, chooses an action based on its current policy, and receives a reward based on the outcome of the action. The agent then updates its policy based on the observed rewards and the current state of the environment.

The process of updating the policy varies depending on which algorithm was being tried. During training, the agent explores the environment to learn the optimal policy. The training process continues until the agent has the maximum number of episodes which is predefined.

5.8 Summary

This chapter describes the experimental setup for using reinforcement learning to train a prosthetic arm gripper to grip objects in minimal time and with optimal force. The gripper is simulated and visualized in a simple PyBullet environment and an RGBD camera is mounted on it to return information on the observed state. Rewards or punishments are given in every episode depending on different factors, such as whether the gripper accomplished its task or how long it took to accomplish the task. A shaped reward function was used tailored to the specific gripping task. The WSG50 gripper was used for this simulation, which is a high-precision, two-finger parallel gripper designed for a wide range of industrial automation

applications. The pybullet_URDF_models library was used to source household objects to train the model. Finally, the gripper's constraints such as maximum and minimum width were taken into consideration when selecting the objects.

CHAPTER 6

RESULTS AND DISCUSSIONS

6.1 Introduction

This chapter discusses the results obtained from experiment 1 – gesture control and experiment 2 – automatic object grasp control using computer vision and reinforcement learning.

6.2 Results of Experiment 1 – Automatic Gesture Estimation and Control

Two models were developed for the gesture prediction task using decision tree classifier (Figure 6-1) and a random forest classifier.

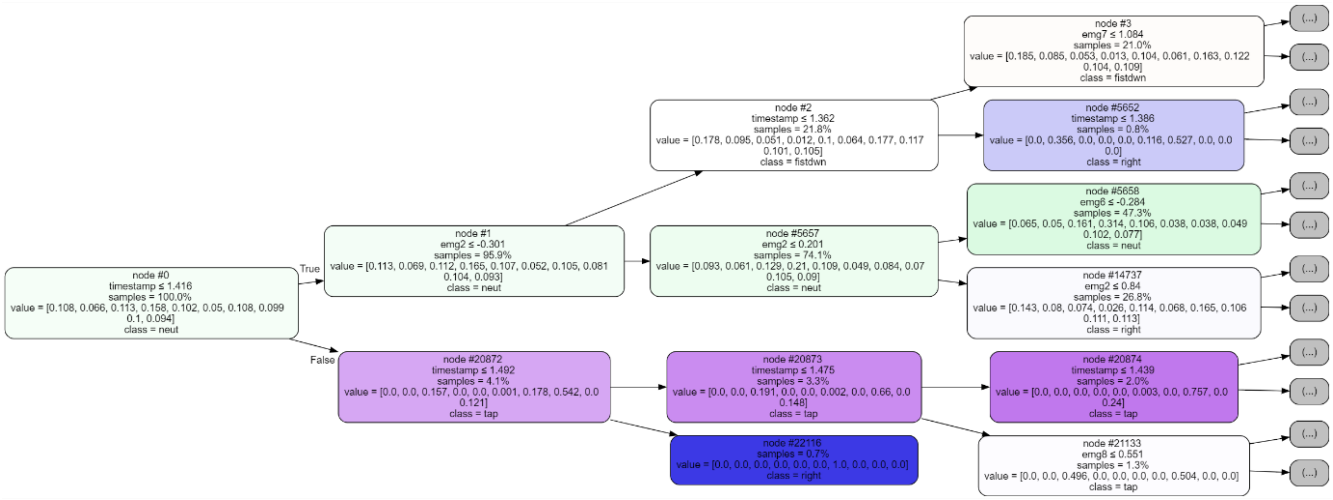


Figure 6-1 Decision tree architecture

The classification reports (Table 6-1, Table 6-2) show the performance of the two classifiers, Decision Tree, and Random Forest, on the task of estimating user gestures based on EMG data. The reports include precision, recall, F1-score, and support metrics for each class, as well as macro and weighted average metrics.

Table 6-1 Decision tree classification report

Parameters	precision	recall	f1-score	support
1	0.94	0.95	0.95	9744
2	0.93	0.94	0.93	5768
3	0.92	0.93	0.93	10116
4	0.98	0.98	0.98	14105
5	0.93	0.93	0.93	9144

6	0.91	0.91	0.91	4455
7	0.94	0.94	0.94	9642
8	0.92	0.92	0.92	8637
9	0.94	0.94	0.94	8819
10	0.94	0.93	0.93	8551
Accuracy			0.94	88981
Macro avg	0.94	0.94	0.94	88981
Weighted avg	0.94	0.94	0.94	88981

Table 6-2 Random Forest classification report

Parameters	precision	recall	f1-score	support
1	0.84	0.89	0.86	9744
2	0.91	0.88	0.85	5768
3	0.84	0.89	0.86	10116
4	0.97	0.99	0.98	14105
5	0.84	0.85	0.84	9144
6	0.90	0.79	0.84	4455
7	0.87	0.91	0.89	9642
8	0.85	0.79	0.82	8637
9	0.87	0.86	0.87	8819
10	0.88	0.83	0.85	8551
Accuracy			0.88	88981
Macro avg	0.88	0.87	0.87	88981
Weighted avg	0.88	0.88	0.88	88981

Precision is a measure of the accuracy of the classifier in predicting positive samples, while recall is a measure of the classifier's ability to correctly identify positive samples. F1-score is a combination of precision and recall, which provides a balanced measure of the classifier's performance. Support represents the number of samples for each class. The parameters number corresponds to each gesture as shown in Figure 4-3.

For the Decision Tree classifier, the precision, recall, and F1-score are high for all classes, ranging from 0.91 to 0.98. The support is also high for most classes, indicating that the classifier is effective at predicting all the gesture types. The macro average F1-score is 0.94, indicating good overall performance, while the weighted average F1-score is also 0.94, indicating that the classifier performs well across all classes.

For the Random Forest classifier, the precision, recall, and F1-score are also high for most classes, ranging from 0.84 to 0.97. However, the precision, recall, and F1-score for some classes, such as gestures 1 and 3, are lower than those of the Decision Tree classifier. The support is high for most classes, indicating that the classifier is effective at predicting all classes. The macro average F1-score is 0.87, which is lower than that of the Decision Tree classifier, while the weighted average F1-score is 0.88, which is slightly lower than that of the Decision Tree classifier.

We can see, the Decision Tree classifier performed better than the Random Forest classifier on this task. The Decision Tree classifier achieved higher precision, recall, and F1-score for most classes, resulting in higher macro and weighted average F1-scores. This performance difference may be because Decision Trees are better suited for tasks with a smaller number of classes and a smaller number of features, such as the EMG data used in this experiment. In contrast, Random Forest is better suited for tasks with a larger number of features, where individual decision trees may overfit the data.

6.2.1 Examining the feature importance of each EMG signal channel

The feature importance is a measure of the relative importance of each input feature in predicting the target variable, which in this case is the gesture being performed. In this case, we examined the relative importance of each feature used for the random forest classifier and decision tree classifier (Table 6-3, Figure 6-2, Figure 6-3).

Table 6-3 Classifiers feature importance.

Features	Decision tree	Random forest
Timestamp	0.76347229	0.22290544
Emg1	0.0412258	0.22076942
Emg2	0.05053453	0.23331555
Emg3	0.01173471	0.0206179

Emg4	0.02075155	0.00979559
Emg5	0.02532406	0.05200387
Emg6	0.04182738	0.05985259
Emg7	0.0243001	0.1304108
Emg8	0.02082958	0.05032884

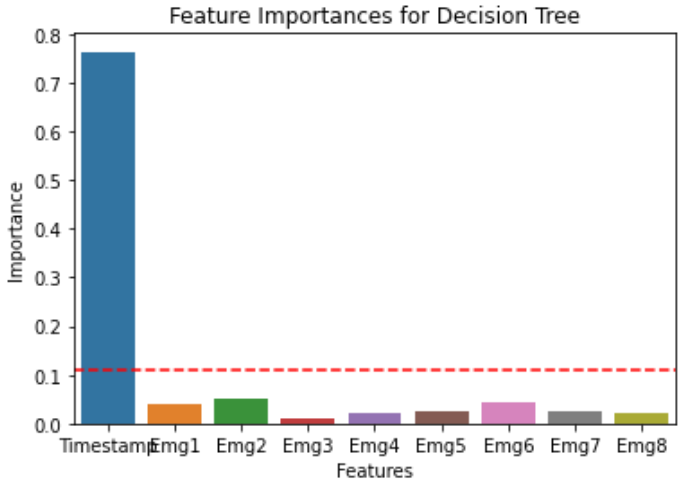


Figure 6-2 Feature importance for the Decision tree model

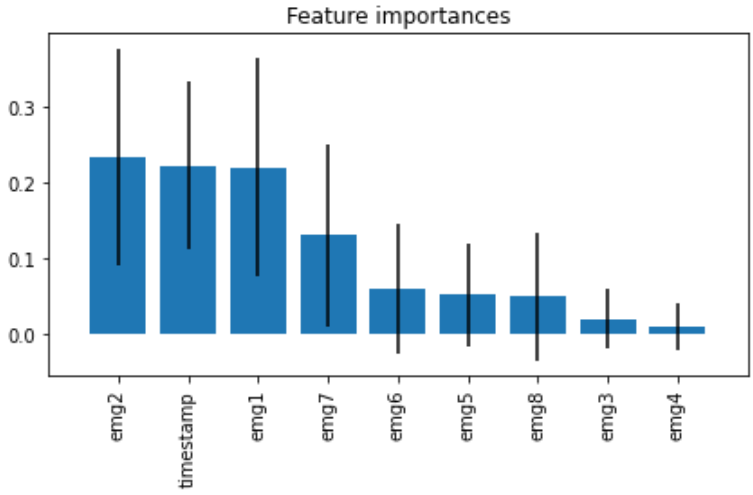


Figure 6-3 Feature importance for the Random Forest model

Looking at the decision tree model, excluding the timestamps, we can see that the EMG signals gotten from channel 2 is the most important feature, with an importance score of 0.0505. This

same conclusion also applies with the random forest classifier which shows the EMG signal on channel 2 as the greatest influence of the model's predictions.

It is interesting to note that the importance of timestamp is relatively high in both models. This suggests that constant use of the device is a critical factor to the model's performance in accurately estimating the gesture being performed than the actual EMG signals themselves.

6.2.2 Hyperparameter tuning.

The best parameters after a 5-folds cross validation are shown in Table 6-4.

Table 6-4 Best parameters after 5 folds cross validation

Best Parameters	Decision Tree	Random Forest
Max depth	None	5
Max features	None	
Min samples leaf	4	1
Min samples split	10	1
Num estimators		300

For the decision tree model, the best set of hyperparameters were identified as having a maximum depth and maximum number of features that were not limited, while requiring a minimum of 4 samples per leaf and a minimum of 10 samples per split. This resulted in a high training set score of 0.9759 and a test set score of 0.9394, indicating that the model was able to generalize well to unseen data.

In contrast, the random forest model had lower training and test set scores compared to the decision tree model. The best set of hyperparameters for the random forest model included a maximum depth of 5, a minimum of 1 sample per leaf, a minimum of 4 samples per split, and 300 estimators. The training set score for the random forest model was 0.3975, which indicates that the model did not fit the training data well. The test set score was 0.3947, which indicates that the model did not generalize well to unseen data.

6.3 Results of Experiment 2 – Grip control

6.3.1 Algorithm-specific success rate

Three different algorithms were tested during training: DQN, SAC and PPO. The training went on for 1000000 timesteps and the performance of each algorithm by success rate are shown in Figure 6-4, 6-5 and 6-6.

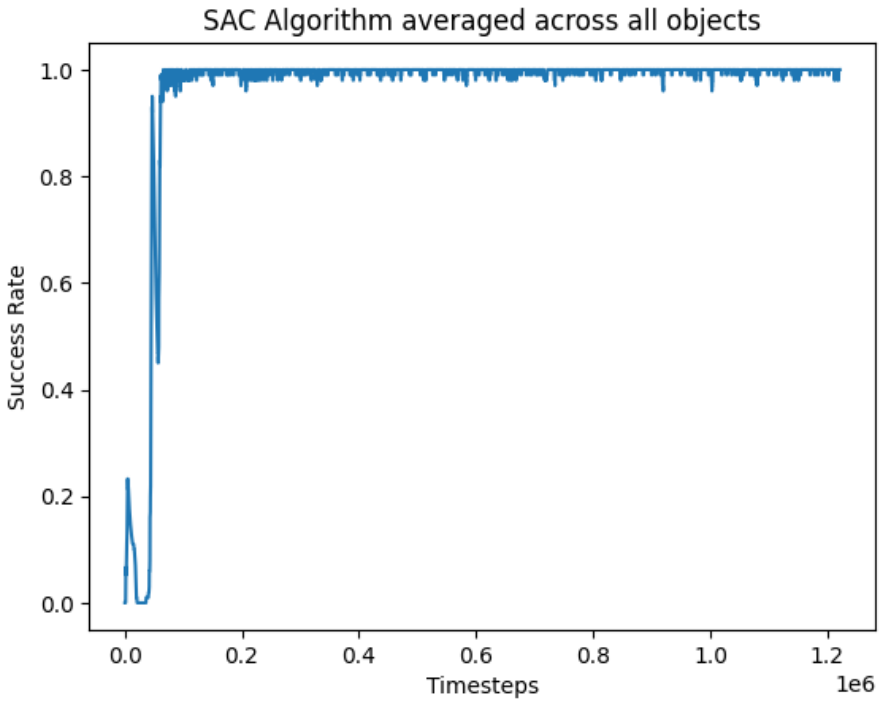


Figure 6-4 SAC success rate by timesteps averaged across all objects.

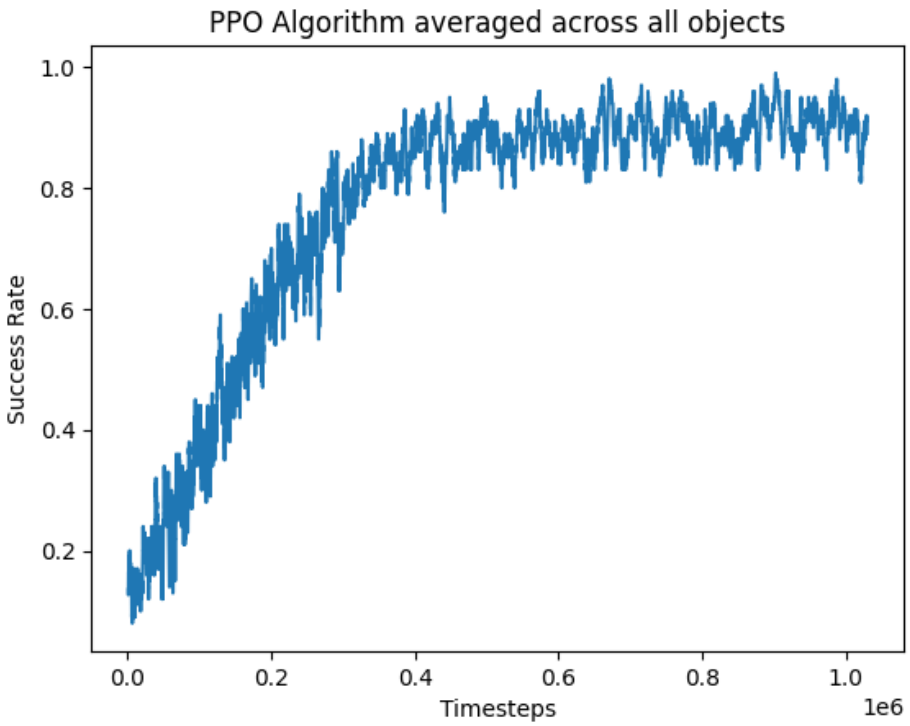


Figure 6-5 PPO success rate by timesteps averaged across all objects

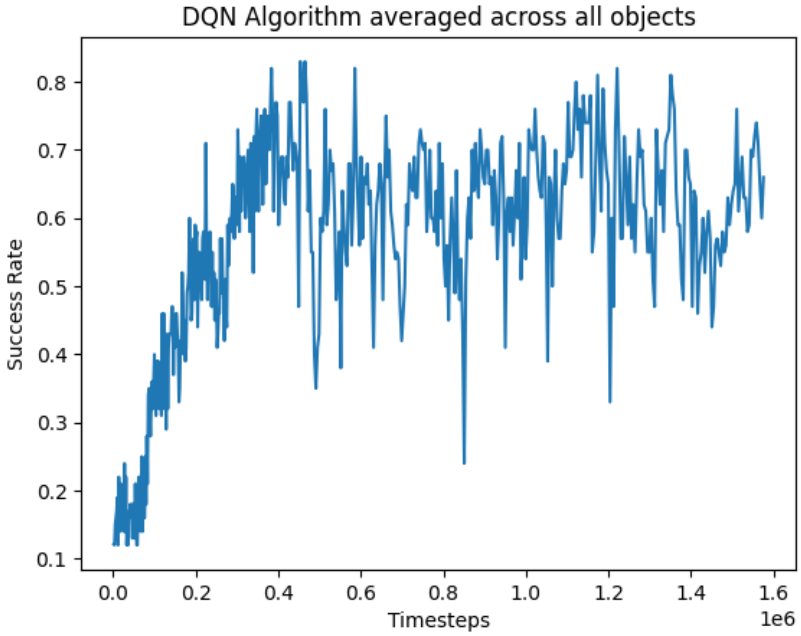


Figure 6-6 DQN success rate by timesteps averaged across all objects.

From the results above, we can see that SAC outperformed the other two algorithms in terms of success rate by converging the quickest in just under 200000 timesteps. On the DQN algorithm, the agent never fully learns the optimal policy required for the grasping task. We can see it reaches a peak success rate of 80% in about 450000 timesteps then stays at

equilibrium at that point. There are several possible reasons why SAC performed better than the other two algorithms. First, SAC is an off-policy actor-critic algorithm that uses a soft value function to estimate the Q-value of actions. This soft value function provides a smoother estimate of the Q-value than the hard value function used in DQN and PPO, which can lead to better performance in the action spaces. SAC also uses a stochastic policy that is updated using a combination of maximum entropy reinforcement learning and entropy regularization. This encourages the policy to explore the state space more thoroughly, which can lead to better performance in environments with sparse rewards, such as the object picking task in our study. DQN might have performed badly because it is an on-policy algorithm that suffers from instability when learning from action spaces.

Table 6-5 Mean Success rate for each algorithm.

Summary	Result
Mean success rate for DQN	0.6021689086910577
Mean success rate for SAC	0.9903811107807406
Mean success rate for PPO	0.821488216618748

From the results, the SAC algorithm performed the best, with a mean success rate of 0.990. This is significantly higher than the mean success rate of the DQN algorithm (0.602) and the PPO algorithm (0.821). The high success rate achieved by the SAC algorithm can be attributed to its inherent advantages in handling high-dimensional action spaces.

On the other hand, the DQN algorithm performed the worst, with a mean success rate of 0.602. The lower success rate achieved by the DQN algorithm can be attributed to the fact that the Q-learning architecture employed by the DQN algorithm is known to have stability issues when applied to continuous control tasks, as it tends to overestimate the Q-values and leads to suboptimal policies.

MANOVA test was further conducted on the success rates of the three algorithms: DQN, SAC, and PPO, to investigate whether there were significant differences in their performance. The results are shown in Figure 6-7:

```
             Df  Pillai approx F num Df den Df    Pr(>F)
algorithms    2 0.57982   2040.7      4  19994 < 2.2e-16 ***
Residuals  9997
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 6-7 MANOVA analysis investigating the significant differences between the three different algorithms.

Based on the results of the MANOVA analysis, there is strong evidence to suggest that there are significant differences between the three different algorithms in terms of their performance. The Pillai's trace statistic of 0.57982 and the highly significant p-value (< 2.2e-16) indicate that the different algorithms have a substantial impact on the success rate of the gripper. The approximate F-value of 2040.7 further supports the presence of a significant effect. These findings indicate that the choice of algorithm significantly influences the observed differences in performance of the prosthetic gripper.

6.3.2 Comparative Analysis of SAC, PPO and DQN for Prosthetic Hand Grasping: Hyperparameter Exploration

We performed further tests to examine how the algorithms robustness to varying hyperparameters.

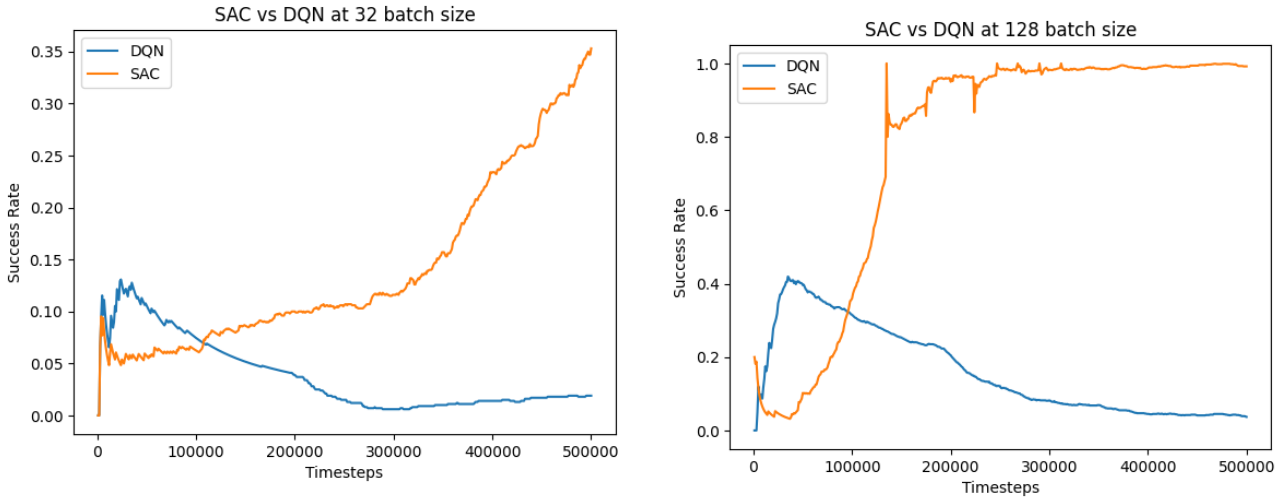


Figure 6-8 SAC vs PPO at 128 batch size and 32 batch size

Increasing the batch size to 128 had a positive effect on the convergence speed of the SAC algorithm, enabling it to converge quickly after just 100,000 time-steps (Figure 6-8). This performs much better than the SAC with RGBD presented in (Baris, 2020).

With a larger batch size, the SAC algorithm benefits from increased sample efficiency. More data is available for each update step, allowing for better estimation of the policy gradient and reducing the impact of noisy or outlier experiences. This increased data diversity enhances the stability of the learning process and facilitates faster convergence towards an optimal policy. Since increasing the batch size increases the amount of information available for each update, this significantly increases the agent's training time. The performance of the DQN algorithm drops at both batch size 32 and 128, compared to the initial analysis at batch size 64 (Figure 6-6). We can infer that at batch size 64, there is an optimal balance between sample efficiency and computational overhead. Deviations from this batch size, either lower or higher, lead to diminished performance due to limitations in data availability or computational inefficiencies.

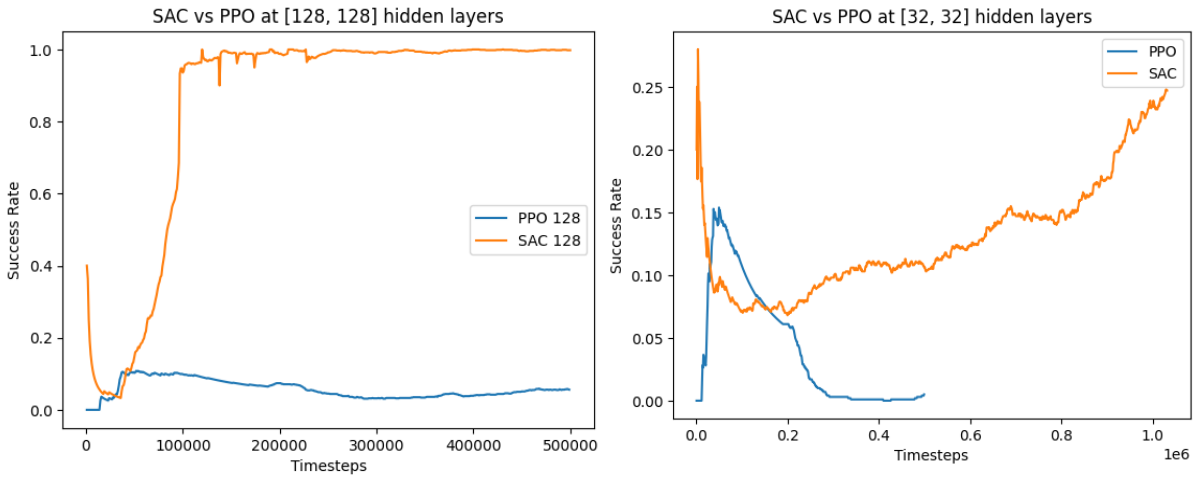


Figure 6-9 SAC vs PPO at varied number of hidden layers

We observe a similar pattern as Figure 6-8 when varied the number of hidden layers for the SAC and PPO algorithms. The SAC algorithms benefit from an increase in the number of hidden layers while the PPO algorithm performs poorly in both cases at 500000 timesteps and would possibly take a longer time to converge.

Based on these results, we can make an inferred comparison of these three algorithms.

Table 6-6 Comparison of SAC, PPO and DQN

Algorithm	SAC	PPO	DQN
Convergence Speed	Quickest to convergence	Slow convergence	Slow convergence
Hyperparameter Sensitivity	Robust to hyperparameters, relatively easier to tune	Performance is heavily affected when deviating slightly from an optimal hyperparameter	Performance is also heavily affected

Training time	Takes the longest time to train	Lower training time than SAC	Requires the least amount of training time

6.3.3 Object-specific success rate

Using SAC as the preferred algorithm, the agent was once again trained to grasp three different types of objects – remote controller, soap bar and mug (Figure 6-11, 6-12, 6-13). This is to determine how object shape and size affects the agent convergence at optimal policy.

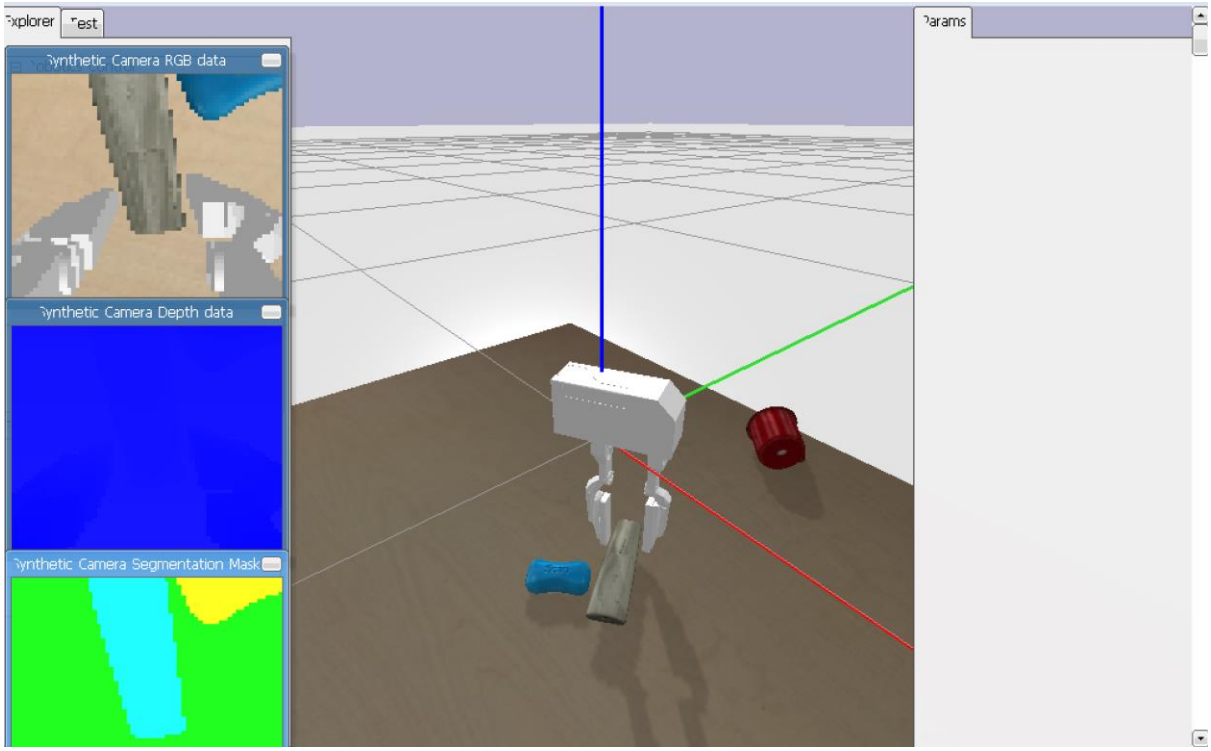


Figure 6-10 WSG50 gripper successfully grips the remote controller.

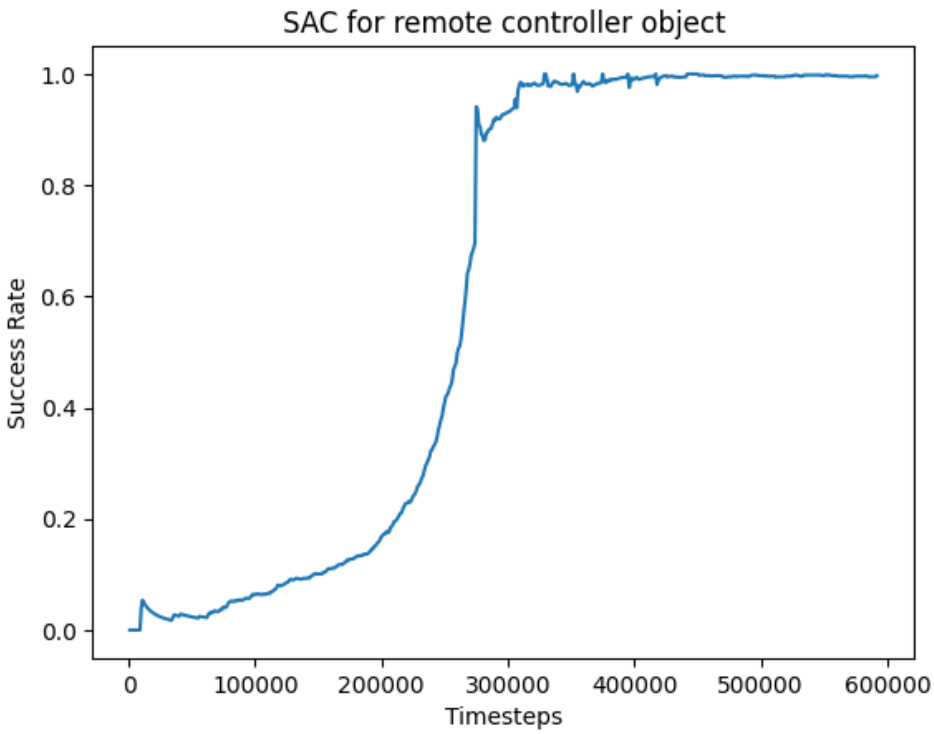


Figure 6-11 SAC success rate by timesteps in grasping the remote controller object.

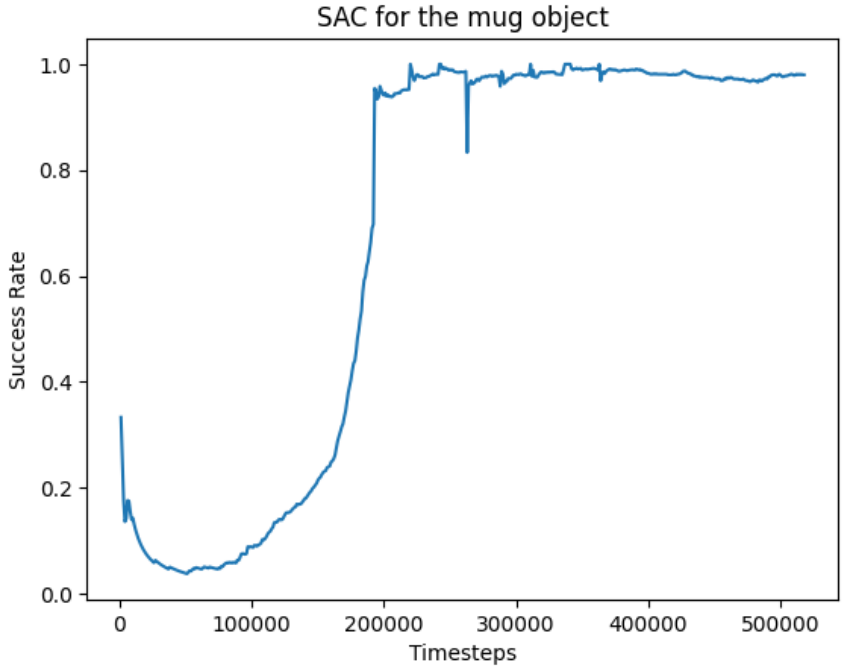


Figure 6-12 SAC success rate by timesteps in grasping the mug object.

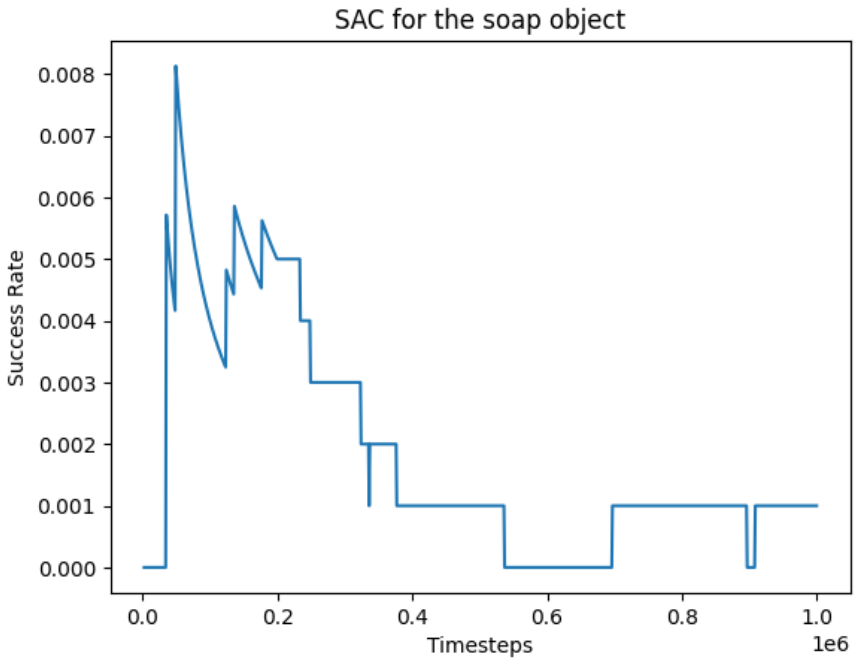


Figure 6-13 SAC success rate by timesteps in grasping the soap bar object.

The results showed that the mug was the easiest object for the agent to grasp, as it converged the fastest, achieving a success rate of over 0.9 after 200,000 timesteps. The remote controller was the second easiest object to grasp, with a success rate of over 0.8 after 250,000 timesteps. However, the soap bar was the most challenging object for the agent, as it failed to converge, with a success rate that remained between 0.0 and 0.008 throughout the training process.

These results are consistent with our understanding of the properties of these objects. The remote controller and mug are both easy to grip, with well-defined shapes and surfaces that the agent can easily detect and grasp. On the other hand, the soap bar has a smooth surface and is difficult to grip, making it challenging for the agent to learn a successful grasping policy.

These results can also be explained in terms of the exploration-exploitation trade-off. The agent's task is to maximize its reward, which in this case is the success rate of grasping an object. To do this, the agent must explore different grasping strategies to find the one that maximizes its reward. However, if the exploration process is too extensive, the agent may fail

to converge to an optimal policy. On the other hand, if the agent exploits the same grasping strategy without exploring others, it may miss out on better strategies.

In the case of the soap bar, the slippery surface and difficult-to-grip nature of the object may have made it more challenging for the agent to explore different grasping strategies. As a result, the agent may have become stuck in a suboptimal policy that did not lead to successful grasping.

In contrast, the well-defined edges and surface of the remote controller and mug may have made it easier for the agent to explore different grasping strategies, leading to faster convergence to an optimal policy. Also, the fact that the success rate of these objects continued to improve over time suggests that the agent was able to find better grasping strategies through exploration without getting stuck in a suboptimal policy.

MANOVA test was further conducted on the success rates of the three objects: mug, remote controller, and bar of soap, to investigate whether the type of object significantly affects the performance of the SAC algorithm. The results are shown in Figure 6-14.

```
            Df Pillai approx F num Df den Df    Pr(>F)
objects      2 0.6888    552.9      4   4210 < 2.2e-16 ***
Residuals 2105
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 6-14 MANOVA analysis investigating the significant differences between gripping the three different objects using the SAC policy.

Based on the results of the MANOVA analysis, there is strong evidence to suggest that the type of object (remote controller, soap, mug) has a significant effect on the success rates of the prosthetic gripper. The Pillai's trace statistic of 0.6888 and the highly significant p-value (< 2.2e-16) indicate that the different object types contribute significantly to the variation in the

success rate of the agent. The approximate F-value of 552.9 further supports the presence of a substantial effect. These findings suggest that different objects cause the RL agent to exhibit distinct performance levels in task completion.

6.4 Conclusion

This chapter presented the experimental findings related to the prediction of user gestures based on EMG data and the training of an agent to grasp objects using different algorithms. The results provide valuable insights into the performance of the models and algorithms employed in this study.

Regarding the prediction of user gestures, two models were developed using decision tree and random forest classifiers. The decision tree classifier outperformed the random forest classifier, demonstrating higher precision, recall, and F1-score for most gesture classes. The analysis of feature importance highlighted the significance of EMG signals from channel 2 and the timing of gestures. The decision tree model exhibited excellent performance on both the training and test sets, with a high generalization ability to unseen data. These findings suggest that the decision tree classifier is well-suited for predicting user gestures based on EMG data.

Moving on to the training of an agent for grasping objects, three different algorithms (DQN, SAC, and PPO) were evaluated. The SAC algorithm demonstrated superior performance in terms of success rate, converging the quickest and achieving a high mean success rate of 99%. In contrast, the DQN algorithm performed the worst, with a mean success rate of 60%. Therefore, SAC was identified as the preferred algorithm for training the agent in grasping tasks.

Furthermore, the agent's performance was assessed when grasping three different types of objects: a remote controller, a soap bar, and a mug. The results revealed that object shape and size significantly influenced the agent's ability to converge to an optimal policy. The mug was

the easiest object to grasp, demonstrating a success rate of over 90% after 200,000 timesteps. The remote controller exhibited the second highest success rate, exceeding 80% after 250,000 timesteps. Conversely, the soap bar proved to be the most challenging object for the agent, with a success rate remaining low throughout the training process.

The MANOVA test confirmed the statistical significance of the object type on the performance of the SAC algorithm, with the remote controller and mug outperforming the soap bar. These findings emphasize the importance of considering object shape and surface characteristics when training agents to perform successful grasping actions. Additionally, the results highlight the effectiveness of the SAC algorithm in training agents for grasping tasks.

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS

7.1 Overview and Conclusions

This thesis conducted two experiments aimed at enhancing the functionality of prosthetic arms. The first experiment developed an automatic gesture prediction and control system using EMG signals, and the second experiment aimed to use reinforcement learning to train a prosthetic arm gripper to grasp objects efficiently. Both experiments achieved promising results using machine learning techniques. The first experiment trained two models using decision tree and random forest classifiers to predict user gestures based on EMG data. The decision tree classifier outperformed the random forest classifier, achieving higher precision, recall, and F1-score for most classes. The results suggest that the decision tree classifier is better suited for estimating user gestures based on EMG data.

The second experiment aimed to train a prosthetic arm gripper to grasp objects in minimal time and optimal force using reinforcement learning. Three algorithms (DQN, SAC, and PPO) were tested, and SAC was found to be the most effective in training agents to perform grasping tasks. Object shape and surface characteristics were found to play a crucial role in the agent's ability to learn successful grasping strategies. The SAC algorithm showed unprecedented robustness to varying scenes and objects, while the PPO algorithm showed increased robustness compared to DQN but did not perform as well as the SAC algorithm in terms of overall success rate and generalizability. Further research is needed to improve the overall success rate and generalizability of the PPO algorithm.

This thesis has made significant progress towards achieving the overall research goal of developing a software framework for prosthetic hands using computer vision and machine learning technologies to automate hand gestures and grip control. The objectives pursed in this research are revisited and shown as follows:

- **Objective 1**: Train tree-based classification models in accurately interpreting EMG signals
- **Objective 2**: Train an agent using existing SAC, PPO and DQN algorithms to enable the prosthetic hand's end effector to automatically grasp objects and to examine how each of the mentioned models affects the hand's ability to effectively grasp objects.
- **Objective 3**: Test the optimal model on varying objects to examine how object's physical properties affect prosthetic's hand ability to effectively grasp objects.

The specific objectives outlined above were successfully pursued, leading to valuable findings and contributions in the field of prosthetics.

The following conclusions can be drawn from this study:

1. With respect to Objective 1, the trained decision tree classifier reports more accurate results in interpreting EMG signals and estimating hand gestures when compared with the Random Forest classifier.
2. With respect to Objective 2, the SAC algorithm excels the most in comparison with DQN and PPO when training a prosthetic hand to automatically grasp objects with appropriate force and at the correct contact point.
3. With respect to Objective 3, the object physical properties such as the shape and texture are a major determinant of how effectively the prosthetic hand can grip objects without it slipping or getting damaged.

By combining the findings from the various objectives, this research paves the way for the development of prosthetic hand systems that can accurately interpret user intentions, perform complex hand gestures, and provide precise and adaptive grasping capabilities. Such advancements have the potential to significantly improve the lives of individuals with limb loss, empowering them with enhanced control, dexterity, and independence.

7.2 Contributions

This thesis makes significant contributions to the field of Myo prosthetic hand development and advances the use of computer vision and machine learning technologies in this domain. The contributions can be summarized as follows:

1. For an 8-channel EMG device, experiments from this thesis were able to infer that the second channel is the most significant when interpreting the user's muscle activities. The identification of the channel 2 as a key driver in gesture recognition offers practical implications for designing and implementing robust and efficient gesture recognition systems, as it provides guidance for feature selection and prioritization in real-world applications.
2. The thesis emphasizes the development of smarter prosthetics by exploring the use of reinforcement learning that enables the prosthetic hand's to automatically grasp objects with appropriate force and at the correct contact point. The results from analysis shed light into the practicability of this approach and opens the possibility for the development of more advanced prosthetics in the future.
3. The study found that the success of the grasping policy is dependent on the properties of the object, with well-defined shapes and surfaces being easier to grasp. This finding can inform the development of more efficient and effective prosthetic grasping systems robust enough to accurately grip objects of varying shapes and textures.
4. Improved performance of autoencoder model originally implemented by (Breyer, et al., 2019) appropriately scaling the learning rate for each parameter for efficient handling of sparse data.

7.3 Limitations

A major limitation of our study is related to the simulation environment used for the experiments. While the simulation environments offer several advantages over real-world experiments, such as repeatability, scalability, and cost-effectiveness, they may not always be a true representation of the real world. In our case, the simulation environment does not capture all the complexities and variations of the real-world grasping task, such as variations in object appearance, lighting conditions, lack of gravity on the objects and sensor noise. Thus, the generalizability of our findings to real-world scenarios may be limited.

Another limitation of our study is related to the choice of the grasping task and the objects used for the experiments. We selected three objects with distinct shapes and sizes, but these objects may not represent the full range of objects that a robotic agent may encounter in the real world. Furthermore, the grasping task itself is simplified and does not account for other aspects of manipulation, such as pushing, pulling, or placing the objects in specific locations.

This research is also limited based on the assumption of a static environment in our experiments. In the real world, the environment may change dynamically, and the agent needs to adapt to these changes to perform the task successfully. Our study does not address this aspect of the grasping task, and future research could explore how to incorporate dynamic environments into the training process.

Another limitation is related to the evaluation metric used for the experiments. In our study, we used the success rate of the grasping task as the primary evaluation metric. However, this metric does not capture other aspects of the grasping task, such as the time taken to complete the task or the efficiency of the grasping motion. Future research is expected to explore alternative evaluation metrics that better capture these aspects of the grasping task.

7.4 Recommendations and Future Work

First, incorporating a scale factor to the reward function like the curriculum learning method could enhance the agent's ability to generalize to different objects and environments. By gradually increasing the complexity of the objects and scenes, the agent can gradually learn to generalize to more challenging scenarios.

Incorporating haptic feedback to the agent can potentially improve the agent's performance in grasping tasks. Recent advancements in deep learning and computer vision can enable the agent to better perceive and understand the object's shape, texture, and other physical properties.

Exploring the use of reinforcement learning algorithms that can handle continuous and high-dimensional action spaces can further improve the agent's performance in grasping tasks. Recently, some researchers (Finn, et al., 2017) have proposed using meta-learning techniques to enable agents to adapt quickly to new tasks and environments. Meta-learning techniques here is used to refer to a type of machine learning approach where an algorithm learns from the output of other machine learning algorithms.

Also, a future work could include training the agent in a more realistic simulated environment that closely mimics real-world scenarios can improve the agent's ability to transfer learned skills to the real world. This can involve incorporating more realistic physics, lighting, and object textures into the simulation.

We also recommend exploring the use of multi-agent reinforcement learning for grasping tasks, where multiple agents can collaborate to perform a task.

REFERENCES

Abbasi, B., Sharifzadeh, M., Noohi, E., Parastegari, S., & Žefran, M. (2019). Grasp Taxonomy for Robot Assistants Inferred from Finger Pressure and Flexion. 2019 International Symposium on Medical Robotics (ISMR), 1-7.

Abduo M, Galster M (2015) Myo gesture control armband for medicalapplications. https://www.semanticscholar.org/paper/Myo-Gesture-Control-Armband-for-Medical-Abduo-Galster/3b5ed355b09beecb7b2b6bbd23fead44b50374c6

Abreu JG, Teixeira JM, Figueiredo LS, Teichrieb V (2016) Evaluatingsign language recognition using the myo armband. In: 2016 XVIIISymposium on Virtual and Augmented Reality (SVR), IEEE, pp64–70

Abul-Haj, C., & Hogan, N. (1990). Functional assessment of control systems for cybernetic elbow prostheses. II. Application of the technique. IEEE Transactions on Biomedical Engineering, 37(11), 1037-1047.

Alexey, B., Chien-Yao, W., & Hong-Yuan, M. L. (2020, April 23). YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv. doi:10.48550/arXiv.2004.10934

Ali, J., Khan, R., Ahmad, N., & Maqsood, I. (2012). Random Forests and Decision Trees. Retrieved 4 15, 2023, from http://ijcsi.org/papers/ijcsi-9-5-3-272-278.pdf

Altimari, L., Dantas, J., Bigliassi, M., Kanthack, T., Moraes, A., & Abrao, T. (2012, 10). Influence of Different Strategies of Treatment Muscle Contraction and Relaxation Phases on EMG Signal Processing and Analysis During Cyclic Exercise.

Alzubaidi, L., Zhang, J., Humaidi, A., Al-Dujaili, A., Duan, Y., Al-Shamma, O., . . . Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges,

applications, future directions. Retrieved from https://doi.org/10.1186/s40537-021-00444-8

Antos, A., Szepesvarf, C., & Munos, R. (2007). Value-Iteration Based Fitted Policy Iteration: Learning with a Single Trajectory. Retrieved 4 15, 2023, from http://researchers.lille.inria.fr/~munos/papers/files/fpi_adprl.pdf

Astin, A. D. (1999, December 16). Finger force capability: measurement and prediction using anthropometric and myoelectric measures. Masters Thesis, Virginia Polytechnic Institute and State University. Retrieved from http://hdl.handle.net/10919/30923

Automatic Addison. (2021, March 20). How to Simulate a Robot Using Gazebo and ROS 2.

Baris, Y. (2020). Branch Dueling Deep Q-Networks for Robotics Applications. Masters Thesis.

Beazley, D. M. (2012). Data processing with Pandas. Log in, 37(6), 76-81. Retrieved 4 15, 2023, from https://usenix.org/publications/login/december-2012-volume-37-number-6/data-processing-pandas

Beyrouthy, T., Al Kork, S., Korbane, J., & Abdulmonem, A. (2016, 8). EEG Mind controlled Smart Prosthetic Arm.

Bright, D., Nair, A., Salvekar, D., & Bhisikar, S. (2016). EEG-based brain controlled prosthetic arm.

Brockdorff, C. v., Aquilina, Y., Cauchi, R., Salib, M. A., Attard, J., & Camilleri, K. P. (2022). ,An Integrated Force Feedback System for a Prosthetic Hand. 48th Annual Conference of the IEEE Industrial Electronics Society (pp. 1-6). Brussels, Belgium: IECON . doi:10.1109/IECON49645.2022.9969077.

Bullet. (2022, April 24). Bullet Physics SDK. Retrieved from GitHub: https://github.com/bulletphysics/bullet3

Bustamante, M., Vega-Centeno, R., Sánchez, M., & Mio, R. (2018). A Parametric 3D-Printed Body-Powered Hand Prosthesis Based on the Four-Bar Linkage Mechanism. 2018 IEEE 18th International Conference on Bioinformatics and Bioengineering (BIBE), 79-85.

Castro, M., Pinheiro, W., & Rigolin, G. (2022, January 24). A Hybrid 3D Printed Hand Prosthesis Prototype Based on sEMG and a Fully Embedded Computer Vision System. Frontiers in Neurorobotics, 15. doi:10.3389/fnbot.2021.751282

Chen, C.-H., Naidu, D., Perez-Gracia, A., & Schoen, M. (2009, 10). A Hybrid Adaptive Control Strategy for a Smart Prosthetic Hand. Conference proceedings: ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference, 2009.

Chortos, A., Liu, J., & Bao, Z. (2016). Pursuing prosthetic electronic skin. Nature Mater, 15, 937–950. doi:https://doi.org/10.1038/nmat4671

Chung, S.-J., & Pollard, N. S. (2016). Predictable behavior during contact simulation: a comparison of selected physics engines. Computer Animation and Virtual Worlds, 27(3), 262-270. Retrieved 4 15, 2023, from https://onlinelibrary.wiley.com/doi/abs/10.1002/cav.1712

Clemente, F., Valle, G., Controzzi, M., Strauss, I., Iberite, F., Stieglitz, T., . . . Cipriani, C. (2019, 2). Intraneural sensory feedback restores grip force control and motor coordination while using a prosthetic hand. Journal of Neural Engineering, 16(2), 26034. Retrieved from https://doi.org/10.1088/1741-2552/ab059b

Collins, J., McVicar, J., Wedlock, D., Brown, R. A., Howard, D., & Leitner, J. (2019). Benchmarking Simulated Robotic Manipulation through a Real World Dataset. arXiv: Robotics. Retrieved 4 15, 2023, from https://arxiv.org/abs/1911.01557

Corporation, I. (2001). Open Source Computer Vision Library Reference Manual. Retrieved 4 15, 2023, from http://www.cs.unc.edu/~stc/FAQs/OpenCV/OpenCVReferenceManual.pdf

Czimmermann, T., Ciuti, G., Milazzo, M., Chiurazzi, M., Roccella, S., Oddo, C., & Dario, P. (2020). Visual-Based Defect Detection and Classification Approaches for Industrial Applications—A SURVEY. Sensors, 20(5). Retrieved from https://www.mdpi.com/1424-8220/20/5/1459.

Damodar, D., Patel, C., Patel, S., & Patel, R. (2019, 1). Design of Gear Driven Cable Operated Mechanical Prosthetic Hand.

Dankert, J., Yang, L., & Si, J. (2005). An analysis of gradient-based policy iteration. Retrieved 4 15, 2023, from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1556399

DeepMind Technologies Limited. (2021). Mujoco. Retrieved April 15, 2015

DeGol, J., Akhtar, A., Manja, B., & Bretl, T. (2016). Automatic grasp selection using a camera in a hand prosthesis. 2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 431-434.

Devashish, S., Amrita, N., Dany, B., & Bhisikar, S. A. (2015). Mind Controlled Robotic Arm. IOSR Journal of Electronics and Communication Engineering, 36-44.

Difonzo, E., Zappatore, G., Mantriota, G., & Reina, G. (2020). Advances in Finger and Partial Hand Prosthetic Mechanisms. Robotics, 9(4). Retrieved from https://www.mdpi.com/2218-6581/9/4/80

Dixit, K. (2020, June 2). Getting Started With Reinforcement Learning(MuJoCo and OpenAI Gym).

Dobra, P., & Susca, M. (2016, 5). Analog and Digital Notch Filter Implementation.

Dunai, L., Novak, M., & García Espert, C. (2021). Human Hand Anatomy-Based Prosthetic Hand. Sensors, 21(1). Retrieved from https://www.mdpi.com/1424-8220/21/1/137

Fei Wang, Zhiqin Qian, Zhiguang Yan, Chenwang Yuan, Wenjun Zhang, 2020. A Novel Resilient Robot: Kinematic Analysis and Experimentation. IEEE Access, vol. 8, pp. 2885-2892, doi: 10.1109/ACCESS.2019.2962058.

Filipczuk, P., & Nikiel, S. (2008). Real-time simulation of a sailboat. Retrieved 4 15, 2023, from http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.ieee-000004581575

Finn, C., Abbeel, P., & Levine, S. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. arXiv.

Grinsztajn, L., Oyallon, E., & Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on tabular data? *arXiv*.

Ghazaei, G., Alameer, A., Degenaar, P., Morgan, G., & Nazarpour, K. (2017, 5). Deep learning-based artificial vision for grasp classification in myoelectric hands. Retrieved from https://doi.org/10.1088/1741-2552/aa6802

Göker, I. (2014, 5). Detection and Conditioning of EMG. In G. Naik, Applications, Challenges, and Advancements in Electromyography Signal Processing (pp. 58-85). doi:10.4018/978-1-4666-6090-8.ch003

Hao, Z., Charbel, T., & Gursel, A. (2021). A 3D Printed Soft Prosthetic Hand with Embedded Actuation and Soft Sensing Capabilities for Directly and Seamlessly Switching Between Various Hand Gestures. IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), 75-80.

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *arXiv*.

Hasan, S., Al-Kandari, K., Al-Awadhi, E., Jaafar, A., Al-Farhan, B., Hassan, M., . . . AlKork, S. (2018). Wearable Mind Thoughts Controlled Open Source 3D Printed Arm with Embedded Sensor Feedback System. Proceedings of the 2nd International Conference on Computer-Human Interaction Research and Applications (CHIRA 2018), 141-149. doi:10.5220/0006929701410149

Henson, A. (2021, March 2). An Introduction to Cosmetic Prostheses. Retrieved June 12, 2022, from Arm Dynamics - Redefining Possibility: https://www.armdynamics.com/upper-limb-library/an-introduction-to-cosmetic-prostheses

Highsmith, M., Carey, S., Koelsch, K., Lusk, C., & Maitland, M. (2009, 12). Design and Fabrication of a Passive-Function, Cylindrical Grasp Terminal Device. Prosthetics and Orthotics International, 33(4), 391-398. Retrieved from https://doi.org/10.3109/03093640903241771

Hook, O. K. (2021). Medical Center Orthotics & Prosthetics. Retrieved June 12, 2022, from https://mcopro.com/prosthetics/technology/ottobock-kids-hook/

Jackson, B. Thalmic Labs Myo Armband Hits Consumer Release, for Sale on Amazon.https://www.itbusiness.ca/news/thalmic-labs-myo-armband-hits-consumer-release-for-sale-on-amazon/54056, last accessed 03/06/2023.

Jiang, N., Pradhan, A., & He, J. (2022). Gesture Recognition and Biometrics ElectroMyogram (GRABMyo). PhysioNet. doi:https://doi.org/10.13026/dzhr-nj10.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research, 4, 237–285. doi:10.1613/jair.301

Kaur M, Singh S, Shaw D (2016) Advancements in soft comput-ing methods for emg classification. Int J Biomed Eng Technol20(3):253–271

Kim, K., & Colgate, J. (2012). Haptic Feedback Enhances Grip Force Control of sEMG-Controlled Prosthetic Hands in Targeted Reinnervation Amputees. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 20(6), 798-805.

Kingsford, C., & Salzberg, S. L. (2008). What are decision trees. Nature Biotechnology, 26(9), 1011-1013. Retrieved 4 15, 2023, from https://ncbi.nlm.nih.gov/pmc/articles/pmc2701298

Krishni. (2018, December 16). K-Fold Cross Validation. Retrieved from Medium: https://medium.datadriveninvestor.com/k-fold-cross-validation-6b8518070833

Kung-Hsiang, H. (2018, January 11). *Introduction to Various Reinforcement Learning Algorithms. Part I (Q-Learning, SARSA, DQN, DDPG)*. Retrieved from Towards Data Science: https://towardsdatascience.com/introduction-to-various-reinforcement-learning-algorithms-i-q-learning-sarsa-dqn-ddpg-72a5e0cb6287

Kyberd, P., Wartenberg, C., Sandsjö, L., Jönsson, S., Gow, D., Frid, J., . . . Sperling, L. (2007). Survey of Upper-Extremity Prosthesis Users in Sweden and the United Kingdom. JPO: Journal of Prosthetics and Orthotics, 19(2). Retrieved from https://journals.lww.com/jpojournal/Fulltext/2007/04000/Survey_of_Upper_Extremity_Prosthesis_Users_in.6.aspx

LaRocco, J., Le, M., & Paeng, D.-G. (2020). A Systemic Review of Available Low-Cost EEG Headsets Used for Drowsiness Detection. Frontiers in Neuroinformatics, 14. Retrieved from https://www.frontiersin.org/article/10.3389/fninf.2020.553352

Lau, B. (2009, 1). An intelligent prosthetic hand using hybrid actuation and myoelectric control.

Li, R., Zhang, X., Lu, Z., Liu, C., Li, H., Sheng, W., & Odekhe, R. (2018). An Approach for Brain-Controlled Prostheses Based on a Facial Expression Paradigm. Frontiers in Neuroscience, 12. Retrieved from https://www.frontiersin.org/article/10.3389/fnins.2018.00943

Link, J. (2001, February 14). How much pressure can an egg withstand? (MadSci Network) Retrieved June 06, 2022, from http://www.madsci.org/posts/archives/2001-02/982342741.Ph.r.html

Liu, S., Zhang, H., Yin, R., Chen, A., & Zhang, W. (2018). Flexure Hinge Based Fully Compliant Prosthetic Finger. (Y. Bi, S. Kapoor, & R. Bhatia, Eds.) Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016, 839-849.

Liu, Z. (2021). OCRTOC: A Cloud-Based Competition and Benchmark for Robotic grasping and manipulation. doi:https://arxiv.org/pdf/2104.11446.pdf

Lundberg, Scott & Erion, Gabriel & Chen, Hugh & DeGrave, Alex & Prutkin, Jordan & Nair, Bala & Katz, Ronit & Himmelfarb, Jonathan & Bansal, Nisha & Lee, Su-In. (2020). From Local Explanations to Global Understanding with Explainable AI for Trees. Nature Machine Intelligence. 2. 10.1038/s42256-019-0138-9.

Low, C., Kasim, M., Koch, T., Dumitrescu, R., Yussof, H., Jaafar, R., . . . Ng, K. (2013, 1). Hybrid-Actuated Finger Prosthesis with Tactile Sensing. International Journal of Advanced Robotic Systems, 10(10), 351. Retrieved from https://doi.org/10.5772/56802

M. Breyer, F. Furrer, T. Novkovic, R. Siegwart and J. Nieto, "Comparing Task Simplifications to Learn Closed-Loop Object Picking Using Deep Reinforcement Learning," in *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1549-1556, April 2019, doi: 10.1109/LRA.2019.2896467.Britannica, T. E. (2018, June 20). Metacarpal. Retrieved from Encyclopedia Britannica: https://www.britannica.com/science/metacarpal

Maat, B., Smit, G., Plettenburg, D., & Breedveld, P. (2018, 2). Passive prosthetic hands and tools: A literature review. Prosthetics and orthotics international, 42(1), 66-74. Retrieved from https://pubmed.ncbi.nlm.nih.gov/28190380

Maciel, A., Halic, T., Lu, Z., Nedel, L., & De, S. (2009). Using the PhysX engine for Physics-based Virtual Surgery with Force Feedback. International Journal of Medical Robotics and Computer Assisted Surgery, 5(3), 341-353. Retrieved 4 15, 2023, from https://ncbi.nlm.nih.gov/pmc/articles/pmc2810833

Roderick, M., MacGlashan, J., & Tellex, S. (2017). Implementing the Deep Q-Network. *arXiv*.

Ruyi He, Bing Zhang, Zhuming Bi, Wenjun Zhang, 2022. "Development of a Hybrid Haptic Device with a High Degree of Motion Decoupling." M2VIP 2022, November 16-18, USA.

Matiisen, T. (2015, December 19). Demystifying Deep Reinforcement Learning. Computational Neuroscience Lab.

Maw, J., Wong, K., & Gillespie, P. (2016, 3). Hand anatomy. British Journal of Hospital Medicine, 77, C34-C40.

Meyer, J., Sendobry, A., Kohlbrecher, S., Klingauf, U., & Stryk, O. v. (2012). Comprehensive simulation of quadrotor UAVs using ROS and gazebo. Retrieved 4 15, 2023, from http://gkmm.tu-darmstadt.de/publications/files/meyer2012quadrotorsimulation.pdf

Mnih, V. K. (2015). Human-level control through deep reinforcement learning. Nature 518, 529-533. doi:https://doi.org/10.1038/nature14236

Mohammadi, A., Lavranos, J., Tan, Y., Choong, P., & Oetomo, D. (2020). A Paediatric 3D-Printed Soft Robotic Hand Prosthesis for Children with Upper Limb Loss. 2020 42nd

Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), 3310-3313.

Mohammadi, A., Lavranos, J., Zhou, H., Mutlu, R., Alici, G., Tan, Y., . . . Oetomo, D. (2020, 5). A practical 3D-printed soft robotic prosthetic hand with multi-articulating capabilities. PLOS ONE, 15(5), e0232766-. Retrieved from https://doi.org/10.1371/journal.pone.0232766

Mutlu, R., Alici, G., Panhuis, M., & Spinks, G. M. (2016). 3D printed flexure hinges for soft monolithic prosthetic fingers. University of Wollongong Research Online, 3(3), 120-133.

Nasri, N., Orts-Escolano, s., Gomez-Donoso, F., & Cazorla, M. (2019). Inferring static hand poses from a low-cost non-intrusive semg sensor. Sensors.

Nayak, S., & Das, R. (2020). Application of Artificial Intelligence (AI) in Prosthetic and Orthotic Rehabilitation. (V. Sezer, S. Öncü, & P. Baykas, Editors) Retrieved from https://doi.org/10.5772/intechopen.93903

NumPy Homepage. (2005). Retrieved 4 15, 2023, from http://www.numpy.org/

Okajima, Y., Tsubahara, A., Kondo, K., Chino, N., Noda, Y., & Tomita, Y. (1995). A new method of estimating the distribution of muscle fiber conduction velocities. Electroencephalography and Clinical Neurophysiology/Electromyography and Motor Control, 97(6), 310-317. Retrieved from https://www.sciencedirect.com/science/article/pii/0924980X9500158H

Össur. (2021). i-Limb Ultra. Retrieved June 12, 2022, from https://www.ossur.com/en-ca/prosthetics/arms/i-limb-ultra

Ottobock. (2021). Bebionic Hand EQD. Retrieved June 12, 2022, from https://www.ottobock.com/en-ca/product/8E70

Palermo, F. (2017). Repeatability analysis of hand movement recognition for control of robotic prosthesis based on sEMG data. Sapienza Universita di Roma.

Pedregosa, F. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

Pillet, J. (1981, October). The aesthetic hand prosthesis. Orthop Clin North Am, 4, 961-969.

Poddar, S., Cummiskey, D., & Kang, J. (2021, 11). A Cable-actuated Prosthetic Emulator for Transradial Amputees (Vol. 2021).

Prigge, P. (2021, January 11). Introduction to Body-Powered Prostheses. (Arm Dynamics Redefining Possibility) Retrieved June 12, 2022, from https://www.armdynamics.com/upper-limb-library/introduction-to-body-powered-prostheses

Prigge, P. (2021). Introduction to Myoelectric Prostheses. (Arm Dynamics - Redefining Possibility) Retrieved June 6, 2022, from https://www.armdynamics.com/upper-limb-library/introduction-to-myoelectric-prostheses#:~:text=Myoelectric%20prostheses%20have%20motors%20and,give%20off%20an%20electrical%20signal

Python Software Foundation. (2023, April 15). Python Software Foundation. Retrieved from Python Software Foundation: https://docs.python.org/3/

Raez, M., Hussain, M., & Mohd-Yasin, F. (2006). Techniques of EMG signal analysis: detection, processing, classification and applications. Retrieved from https://pubmed.ncbi.nlm.nih.gov/16799694

Rawat, W., & Wang, Z. (2017, 9). Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. Neural Computation, 29(9), 2352-2449. Retrieved from https://doi.org/10.1162/neco_a_00990

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. Proceedings of the IEEE conference on computer vision and pattern recognition.

Rivu, S., Abdrabou, Y., Mayer, T., Pfeuffer, K., & Alt, F. (2019). GazeButton: Enhancing Buttons with Eye Gaze Interactions. Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications. Retrieved from https://doi.org/10.1145/3317956.3318154

Schrimpf, M. (2016). Should I use TensorFlow. arXiv: Learning. Retrieved 4 15, 2023, from https://arxiv.org/abs/1611.08903

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv*.

Semasinghe, C., Ranaweera, R., Prasanna, J., Kandamby, H., Madusanka, D., & Gopura, R. (2018). HyPro: A Multi-DoF Hybrid-Powered Transradial Robotic Prosthesis. (Y. Li, Ed.) Journal of Robotics, 2018, 8491073. Retrieved from https://doi.org/10.1155/2018/8491073

Shi, C., Yang, D., Zhao, J., & Liu, H. (2020). Computer Vision-Based Grasp Pattern Recognition With Application to Myoelectric Control of Dexterous Hand Prosthesis.

Shi, Chunyuan; Yang, Dapeng; Zhao, Jingdong; Jiang, Li. (2022). i-MYO: A Hybrid Prosthetic Hand Control System based on Eye-tracking, Augmented Reality and Myoelectric signal. arXiv. doi:10.48550/arxiv.2205.08948

Slade, P., Akhtar, A., Nguyen, M., & Bretl, T. (2015). Tact: Design and performance of an open-source, affordable, myoelectric prosthetic hand. 2015 IEEE International Conference on Robotics and Automation (ICRA), 6451-6456.

Smidstrup, A., Erkocevic, E., Niemeier, M., Bøg, M., Rosenvang, J., & Kamavuako, E. (2011). A COMPARISON STUDY OF EMG FEATURES FOR FORCE PREDICTION. Proceedings of the MEC'11 conference, UNB. Retrieved from https://hdl.handle.net/10161/4723

Smit, G., & Plettenburg, D. (2010, 12). Efficiency of Voluntary Closing Hand and Hook Prostheses. Prosthetics and orthotics international, 34, 411-427.

Srivastava, Nitish & Hinton, Geoffrey & Krizhevsky, Alex & Sutskever, Ilya & Salakhutdinov, Ruslan. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 15. 1929-1958.

Sturm, P., & Ramalingam, S. (2004). A Generic Concept for Camera Calibration. (T. Pajdla, & J. Matas, Eds.) Computer Vision - ECCV 2004, 1-13.

Tan Zhang, WJ Zhang, MM Gupta, 2017. Resilient Robots: Concept, Review and Future Directions. Robotics. 6, 22; doi:10.3390/robotics6040022.

Thorn, J. (2020, March 8). Decision trees explained. Retrieved from Towards data science: https://towardsdatascience.com/decision-trees-explained-3ec41632ceb6

Wang F., Qian Z.Q, Lin Y., Zhang WJ, 2020. Design and Rapid Construction of a Cost-Effective Virtual Haptic Device. IEEE transactions on mechatronics. doi: 10.1109/TMECH.2020.3001205.

Warwick, K., Gasson, M., Hutt, B., Goodhew, I., Kyberd, P., Andrews, B., . . . Shad, A. (2003, 10). The Application of Implant Technology for Cybernetic Systems. Archives of

Neurology, 60(10), 1369-1373. Retrieved from https://doi.org/10.1001/archneur.60.10.1369

Weiss Robotics. (2015). WSG 50-110. Retrieved April 15, 2023, from https://weiss-robotics.com/servo-electric/wsg-series/product/wsg-series/selectVariant/wsg-50-110/

Widodo MS, Zikky M, Nurindiyani AK (2018) Guide gesture applica-tion of hand exercises for post-stroke rehabilitation using myoarmband. In: 2018 international electronics symposium on knowl-edge creation and intelligent computing (IES-KCIC), IEEE, pp120–124

Wu, J. Q., Yuan, C. W., Yin, R. X., Sun, W., & Zhang, W. J. (2020). A Novel Self-Docking and Undocking Approach for Self-Changeable Robots. 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), 1, 689-693. doi:10.1109/ITNEC48623.2020.9085076

W. J. Zhang, G. Yang, Y. Lin, C. Ji and M. M. Gupta, "On Definition of Deep Learning," 2018 World Automation Congress (WAC), Stevenson, WA, USA, 2018, pp. 1-5, doi: 10.23919/WAC.2018.8430387.

Zhang, W.J., Sun, Z.H., Zhang, B., 2011. On the concept of the resilient machine. The 6th IEEE Conference on Industrial Electronics and Applications (ICIEA 2011), Beijing, June 21-23.

Zhang, X., Li, H., Lu, Z., & Yin, G. (2021). Homology Characteristics of EEG and EMG for Lower Limb Voluntary Movement Intention. Frontiers in Neurorobotics, 15. Retrieved from https://www.frontiersin.org/article/10.3389/fnbot.2021.642607

Zheng, Y., Cao, L., Qian, Z., Chen, A., & Zhang, W. (2016). Topology optimization of a fully compliant prosthetic finger: Design and testing.

Zhiqin Qian, Weiji Jing, Yi Lv, & Wenjun Zhang (2022, June 1). Automatic Polyp Detection by Combining Conditional Generative Adversarial Network and Modified You-Only-Look-Once. IEEE SENSORS JOURNAL, 22(11), 10841-10849.

APPENDIX A: Code Snippets Showing the Introduction of Exploration and Distance Penalty to the Reward Function Originally Implemented by (Breyer, at al., 2019) (Baris, 2020)

```
class Reward:
    """Simple reward function reinforcing upwards movement of grasped objects."""
    def __init__(self, config, robot):
        self._robot = robot
        self._shaped = config.get('shaped', True)
        self._max_delta_z = robot._actuator._max_translation
        self._terminal_reward = config['terminal_reward']
        self._grasp_reward = config['grasp_reward']
        self._delta_z_scale = config['delta_z_scale']
        self._lift_success = config.get('lift_success', self._terminal_reward)
        self._time_penalty = config.get('time_penalty', False)
        self._table_clearing = config.get('table_clearing', False)
        self.lift_dist = None
        self._lifting = False
        self._start_height = None
        self._old_robot_height = None
        # Exploration parameters
        self._explore_weight = config.get('explore_weight', 0.1)
        # Object distance penalty parameters
        self._distance_penalty = config.get('distance_penalty', False)
        self._target_pos = np.array(config.get('target_pos', [0., 0., 0.]))
    def __call__(self, obs, action, new_obs):
        position, _ = self._robot.get_pose()
        object_pos = self._robot.get_object_position()
        robot_height = position[2]
        reward = 0.

        if self._robot.object_detected():
```

```
            if not self._lifting:
                self._start_height = robot_height
                self._lifting = True
            if robot_height - self._start_height > self.lift_dist:
                # Object was lifted by the desired amount
                return self._terminal_reward, robot.RobotEnv.Status.SUCCESS
            if self._shaped:
                # Intermediate rewards for grasping and lifting
                delta_z = robot_height - self._old_robot_height
                reward = self._grasp_reward + self._delta_z_scale * delta_z
        else:
            self._lifting = False
        # Time penalty
        if self._shaped:
            reward -= self._grasp_reward + self._delta_z_scale * self._max_delta_z
        else:
            reward -= 0.01
        # Exploration component
        if np.random.rand() < self._explore_weight:
            reward += np.random.normal()
        # Object distance penalty component
        if self._distance_penalty:
            target_dist = np.linalg.norm(object_pos - self._target_pos)
            reward -= target_dist
        self._old_robot_height = robot_height
        return reward, robot.RobotEnv.Status.RUNNING
    def reset(self):
        position, _ = self._robot.get_pose()
        self._old_robot_height = position[2]
```

APPENDIX B: Code Snippets Including Changes made to the AutoEncoder Object Originally Implemented by (Breyer, at al., 2019) (Baris, 2020)

```
class SimpleAutoEncoder(Encoder):
    def _build(self, config):
        dropout_rate = config.get('dropout_rate', 0.5)
        activity_regularizer = config.get('activity_regularizer', l1(l=0.01))
        learning_rate = config.get('learning_rate', 0.01)
        # Encoder
        h = inputs
        for layer in network:
            h = Conv2D(filters=layer['filters'],
                       kernel_size=layer['kernel_size'],
                       strides=layer['strides'],
                       padding='same')(h)
            h = ELU(alpha)(h)
            h = Dropout(dropout_rate)(h)
        shape = h._keras_shape[1:]
        h = Flatten()(h)
        h = Dense(encoding_dim, activity_regularizer=activity_regularizer)(h)
        z = ELU(alpha)(h)
        self._encoder = Model(inputs, z, name='encoder')
        self._encoder._make_predict_function()
        # Decoder
        latent_inputs = Input(shape=(encoding_dim,))
        h = Dense(np.prod(shape), activity_regularizer=activity_regularizer)(latent_inputs)
        h = ELU(alpha)(h)
        h = Reshape(shape)(h)
        for i in reversed(range(1, len(network))):
            h = UpSampling2D(size=network[i]['strides'])(h)
            h = Conv2D(filters=network[i - 1]['filters'],
```

```
            kernel_size=network[i]['kernel_size'],
            padding='same')(h)
    h = ELU(alpha)(h)
    h = Dropout(dropout_rate)(h)
h = UpSampling2D(network[0]['strides'])(h)
outputs = Conv2D(1, network[0]['kernel_size'], padding='same')(h)
self._decoder = Model(latent_inputs, outputs, name='decoder')
# Loss function
loss = 'binary_crossentropy'
# Optimizer
optimizer = Adagrad(lr=config['learning_rate'])
# Autoencoder
self._model = Model(inputs, self._decoder(self._encoder(inputs)))
self._model.compile(optimizer=optimizer, loss=loss)
return self._model
```

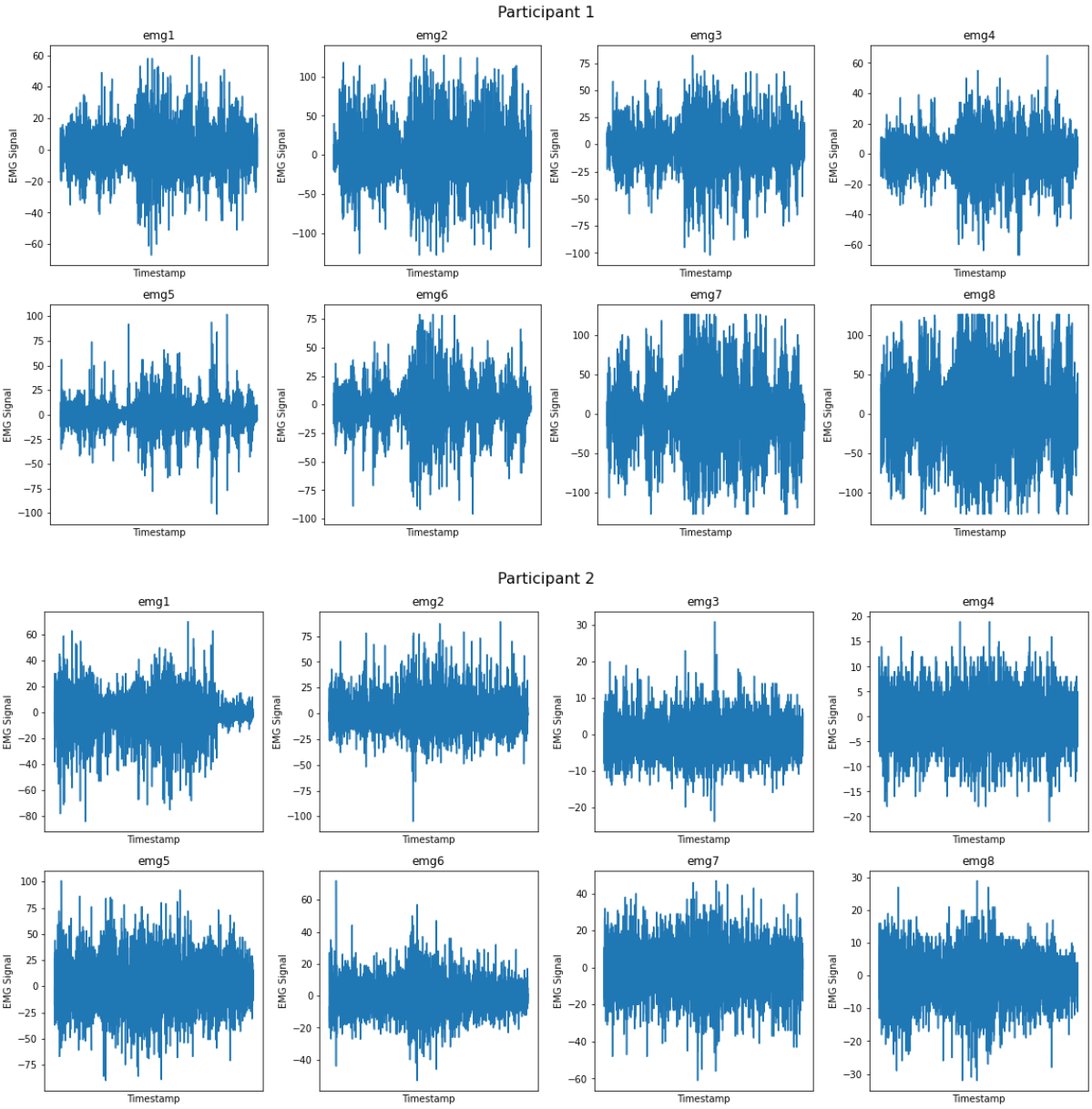
APPENDIX C: Pseudocode for a Generalized Grid Search Cross Validation

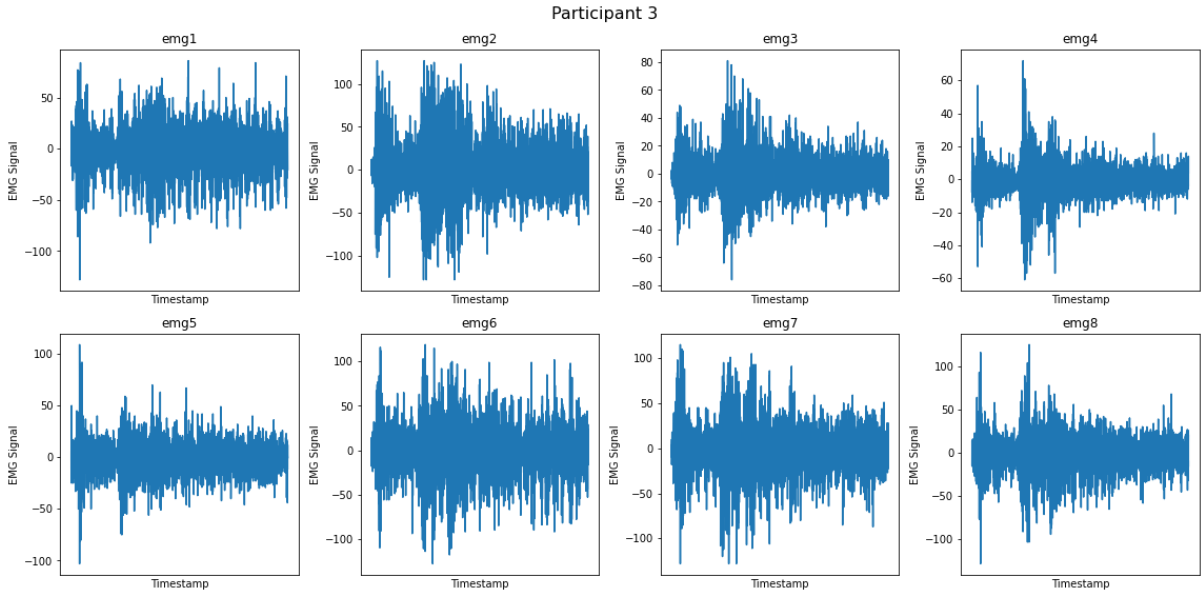
```
BestHyperparameters = argmax(PerformanceMetric)
for each combination of hyperparameter values:
    θ = (θ_1, θ_2, ..., θ_H)  # Hyperparameter values combination
    performance_sum = 0
    for each fold:
        D_train = D \ D_k  # Training set excluding the k-th fold
        D_k = k-th fold  # Validation set
        model = train_model(D_train, θ)  # Train the model on the training set
        performance_sum += evaluate_model(model, D_k)  # Evaluate the model on the validation set
    performance_avg = performance_sum / K  # Average performance across all folds
    if performance_avg > BestPerformance:
        BestPerformance = performance_avg
        BestHyperparameters = θ
return BestHyperparameters
```

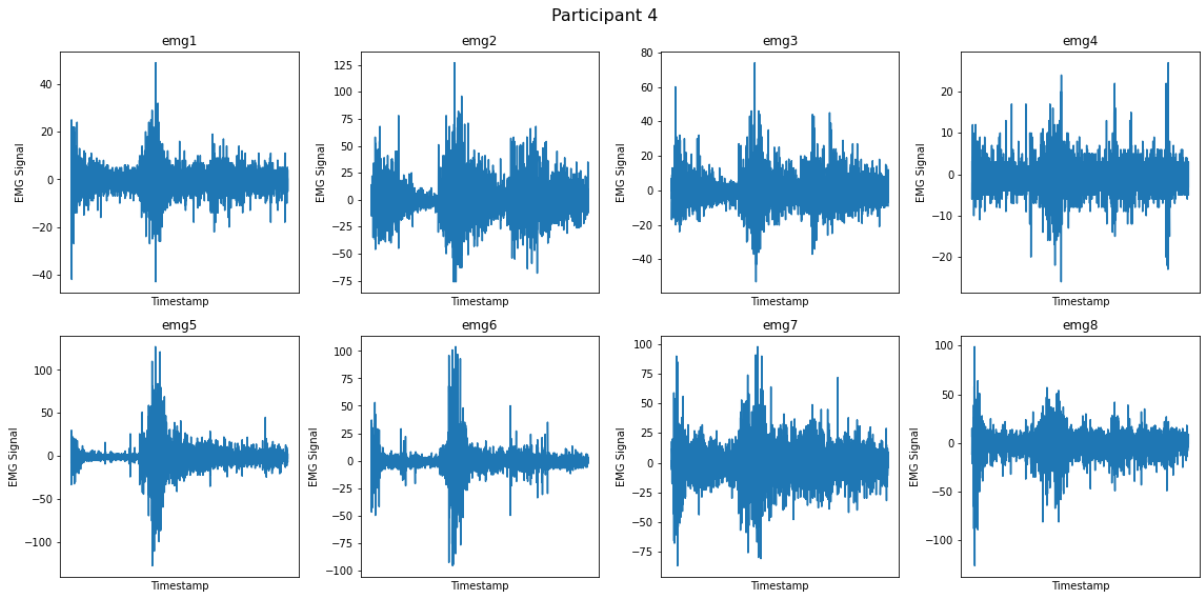
APPENDIX D: Loading the Objects to the Simulation Environment Using the PyBullet URDF Model Library

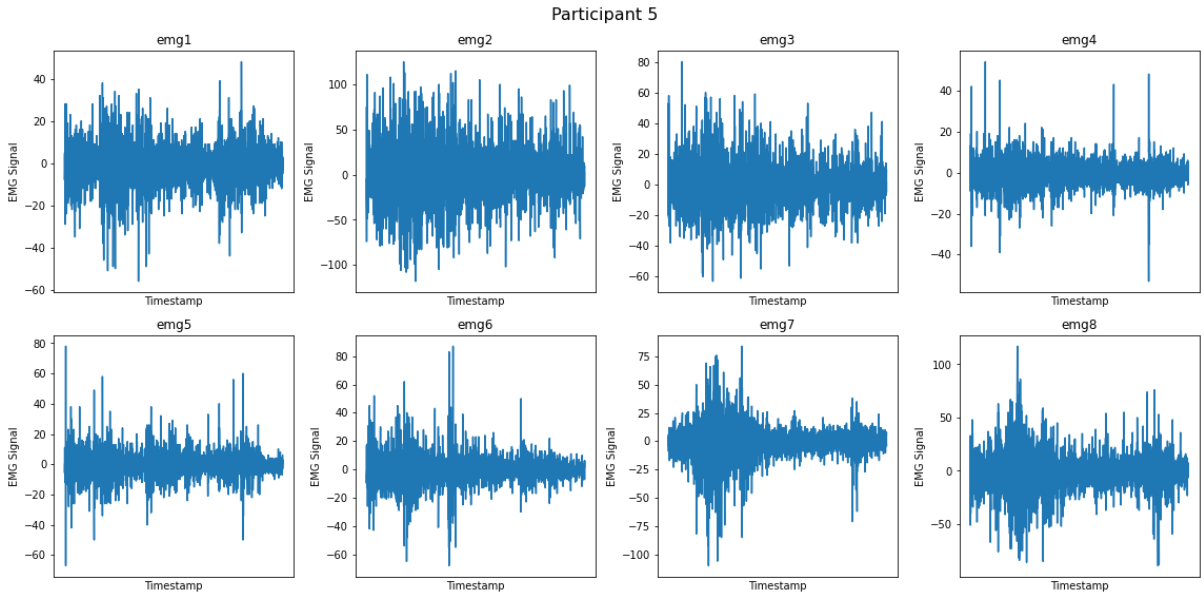
```
import urdf_models.models_data as md
class Scene(BaseScene):
    def reset(self):
        models_lib = md.model_lib()
        object_names = [‘soap’,’mug’,’remote_controller_1’]
       # Spawn objects
       for name in object_names:
          model_urdf_path = models_lib[name]
          self._world.add_model(model_urdf_path)
          self._world.run(.2) #wait for object to rest before spawning the next

       self._world.run(1.)
```

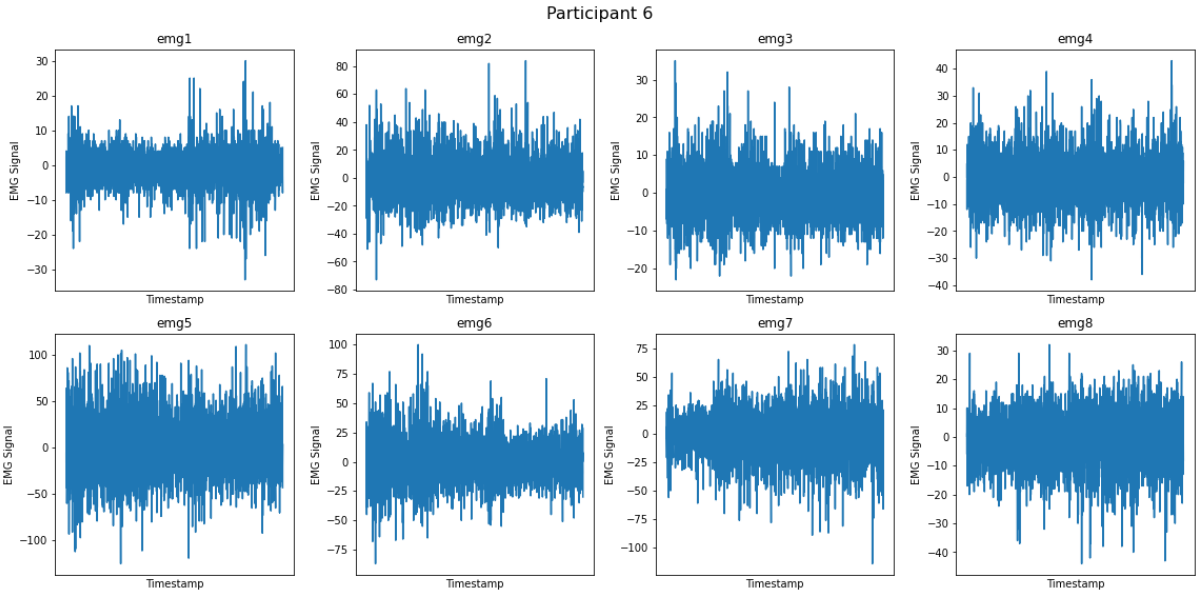
APPENDIX E: EMG Data Acquisition (Closed Hands Gesture)

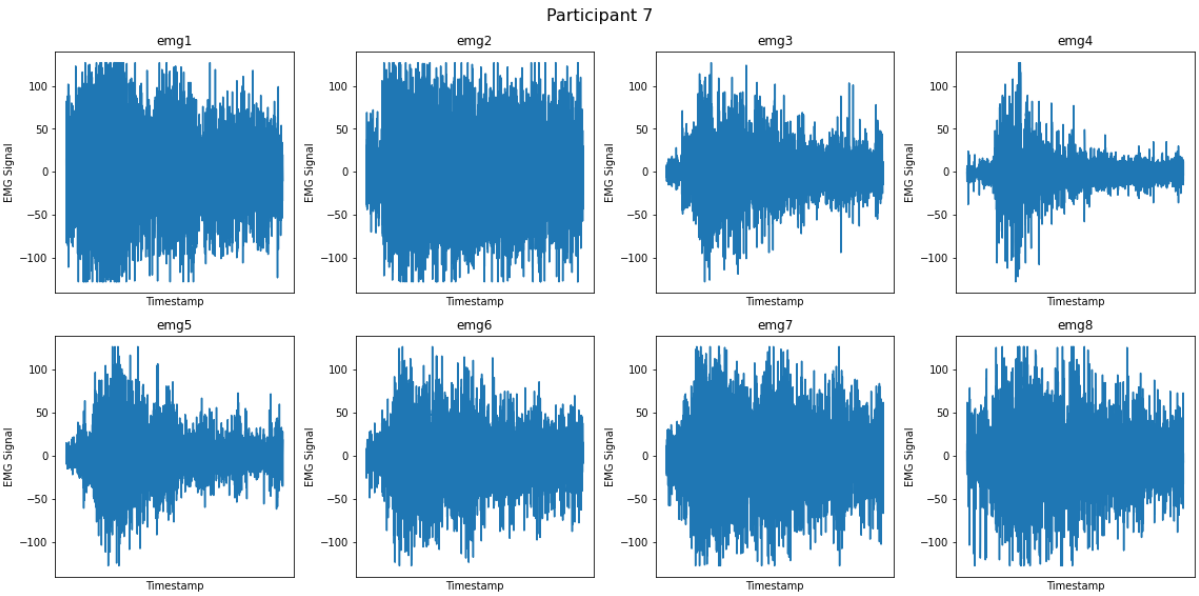


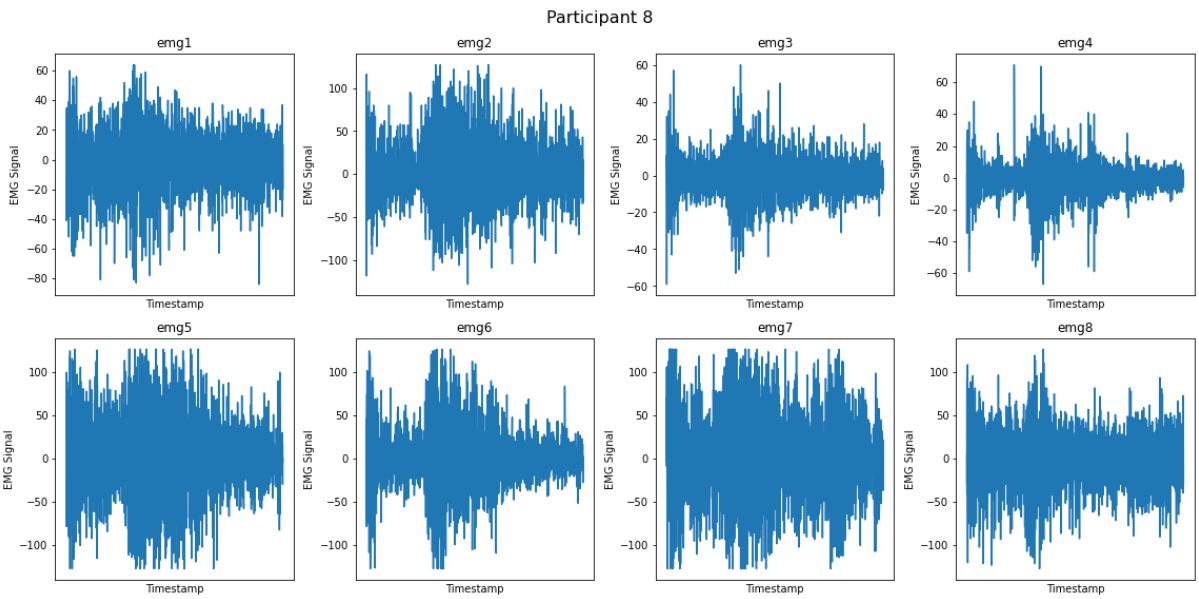

Participant 3
emg1
emg2
emg3
emg4
emg5
emg6
emg7
emg8
EMG Signal
Timestamp

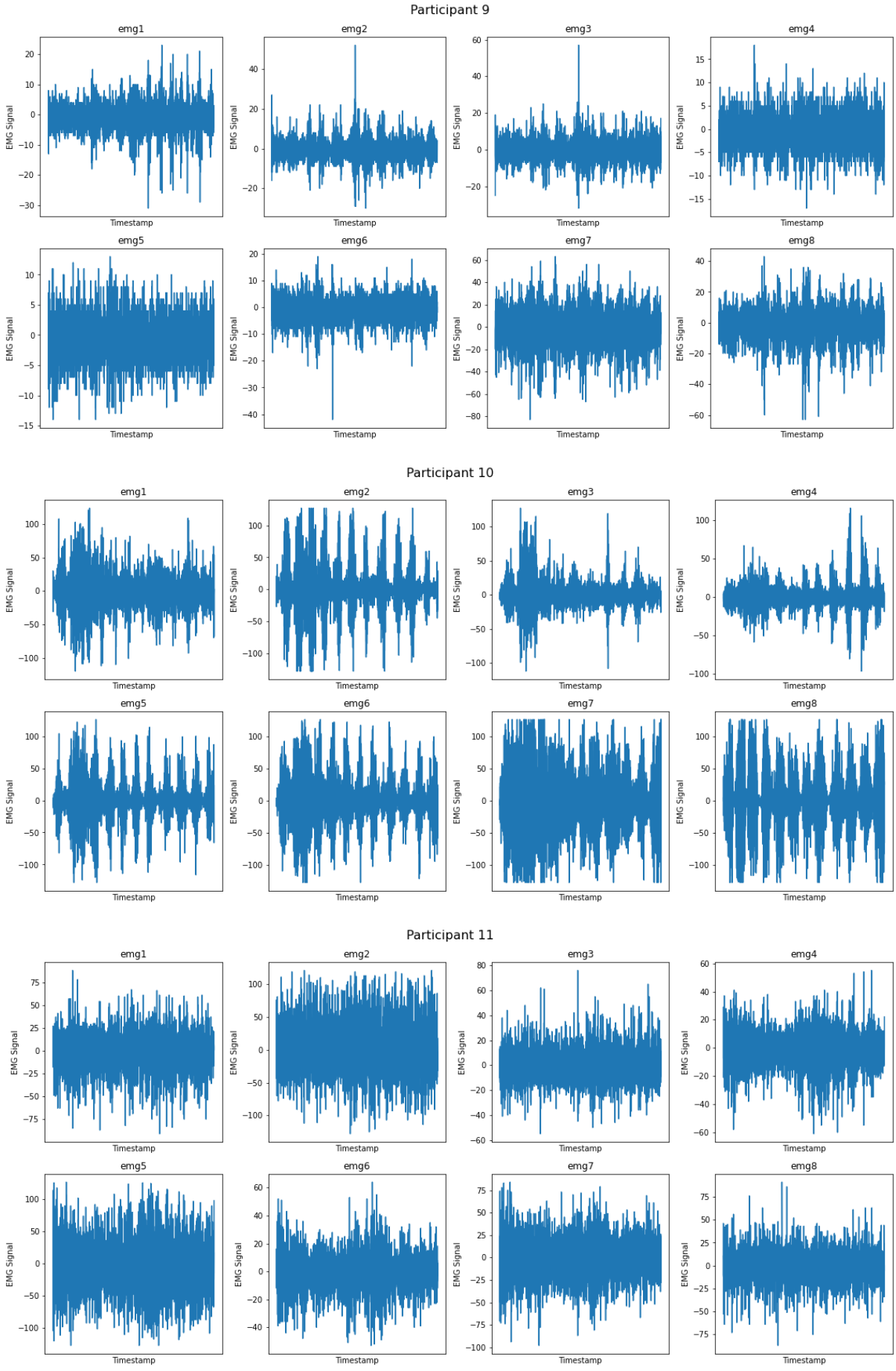

Participant 4
emg1
emg2
emg3
emg4
emg5
emg6
emg7
emg8
EMG Signal
Timestamp

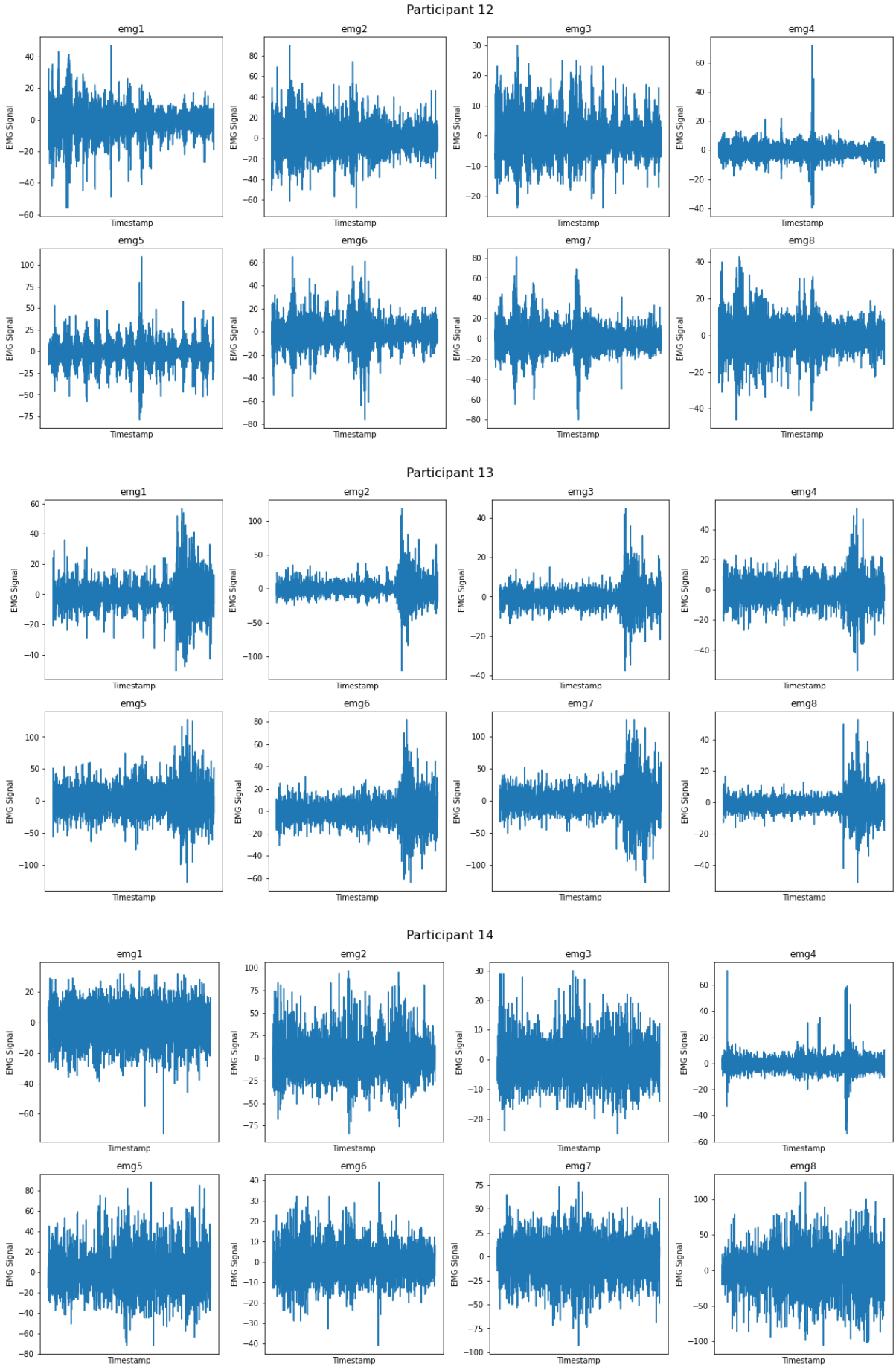

Participant 5
emg1
emg2
emg3
emg4
emg5
emg6
emg7
emg8
EMG Signal
Timestamp

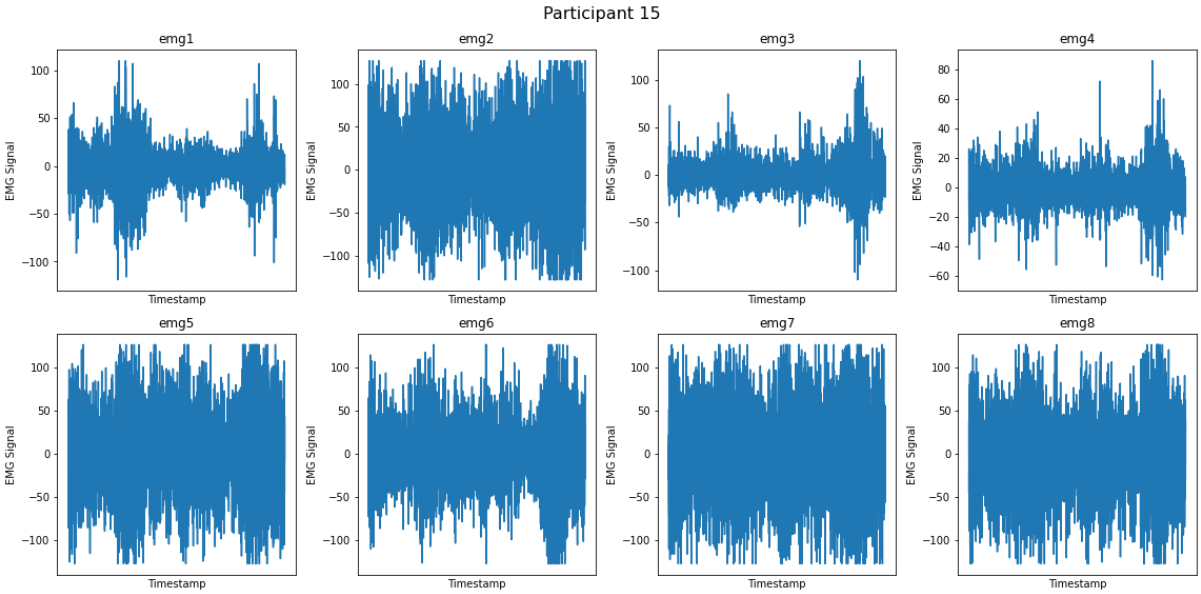

Participant 6
emg1
emg2
emg3
emg4
emg5
emg6
emg7
emg8
EMG Signal
Timestamp


Participant 7
emg1
emg2
emg3
emg4
emg5
emg6
emg7
emg8
EMG Signal
Timestamp


Participant 8
emg1
emg2
emg3
emg4
emg5
emg6
emg7
emg8
EMG Signal
Timestamp


Participant 9
emg1
emg2
emg3
emg4
emg5
emg6
emg7
emg8
EMG Signal
Timestamp
Participant 10
emg1
emg2
emg3
emg4
emg5
emg6
emg7
emg8
EMG Signal
Timestamp
Participant 11
emg1
emg2
emg3
emg4
emg5
emg6
emg7
emg8
EMG Signal
Timestamp


Participant 12
emg1
emg2
emg3
emg4
emg5
emg6
emg7
emg8
EMG Signal
Timestamp
Participant 13
emg1
emg2
emg3
emg4
emg5
emg6
emg7
emg8
EMG Signal
Timestamp
Participant 14
emg1
emg2
emg3
emg4
emg5
emg6
emg7
emg8
EMG Signal
Timestamp


Participant 15
emg1
emg2
emg3
emg4
emg5
emg6
emg7
emg8
EMG Signal
Timestamp