Title

8th Semester Project

Project Report
Group 851

AALBORG UNIVERSITY
STUDENT REPORT

Aalborg University
The Technical Faculty for IT and Design

The Technical Faculty for IT and Design
Niels Jernes Vej 10, 9220 Aalborg Øst
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
"Title"

Theme:
Collaborative Robotics

Project Period:
ROB 7th Semester

Project Group:
Group 754

Participant(s):

Chathushka Jayashan 20230541
Daniel Doroshenko 20203904
Jakob Bøgh Larsen 20204794
Victor Hoffmann Risager 20203377

Supervisor(s):
Martin Alexander Garenfeld
Strahinja Dosen

Copies: 1

Page Numbers: 44

Date of Completion:
May 1, 2024

Abstract:

□

Preface

Aalborg University, May 1, 2024

Contents

Acronyms

1 Introduction **1**

2 Problem Analysis **2**
2.1 Human Hand 2
2.2 Conventional Prosthetics and Control 5
2.3 Modern Prosthetics and Control 8
2.4 Delimitation 17

3 Requirements **19**
3.1 Technical Requirements 19
3.2 Clinical Requirements 19

4 Solution Proposal **20**
4.1 Hardware Selection 20
4.2 Simulation 23

5 Modelling **25**
5.1 Trajectory Generation 25
5.2 Point Cloud Processing 30
5.3 Deep Reinforcement Learning 32

6 Implementation **36**
6.1 System Overview 36
6.2 Environment Setup 37
6.3 Model Training 37

7 Testing and Results **38**

8 Discussion **39**

9 Conclusion **40**

Bibliography **41**

Acronyms

ADL Activity of Daily Living
API Application Programming Interface

CNN Convolutional Neural Network

DIP Distal Inter-Phalangeal
DOF Degree of Freedom
DQN Deep Q-Network

EMG Electromyography

FMG Force Myography
FOV Field of View

HER Hindsight Experience Replay
HPO Hyper Parameter Optimisation

IMU Inertial Measurement Unit

MCP Metacarpal-phalangeal
MDP Markov Decision Process

PIP Proximal Inter-Phalangeal
POMDP Partially Observable Markov Decision Process
PPO Proximal Policy Optimisation

RGB Red, Green, Blue
RL Reinforcement Learning
ROS Robot Operating System
RTF Real Time Factor

SAC Soft Actor Critic
sEMG Surface Electromyography

1 Introduction

1.0.1 Initial Problem Statement

How can dexterous robotic hand prosthesis be semi-autonomously controlled using computer vision.

1.0.1.1 Directions of the project

- Focus on reinforcement learning
- Focus on different camera mounts.
- Focus on grasp quality with force control
- Focus on human intention interpretation (IMU integration for sensor fusion)
- Focus on more autonomous control (e.g. pregrasping)
-
-

2 Problem Analysis

In this chapter, the different aspects and intricacies referring to the problem are explored and analysed. Firstly, the physiological structure and practical functionalities of the human hand relating to activities of daily living (ADLs) are defined. Subsequently, a brief introduction to traditional prosthetics is included followed by the development towards more automated solutions that have been emerging in the last decades. Methods for solving the difficult problem of controlling the many DOFs pertaining to the human hand are differentiated, where one of the most promising solutions is the integration of computer vision.

2.1 Human Hand

Most humans find it effortless to articulate their hands in such a way as to grasp any object that can be encompassed within them. This is both a testament to the intertwined, sophisticated control of the human brain and body, and the physiologically complex nature of the human hand.

2.1.1 Functionality in Everyday Life

Hands are a large part of a person's everyday life, where they are used for exploring and manipulating the environment, using hand gestures during speech, and especially manipulating and grasping objects. Therefore, the loss of one or both hands is traumatic and requires a long adaptation process before physically and mentally accommodating properly to the new reality. Hands are central for practically all ADLs in areas that range between work, hobbies, and self-care. These activities include eating, drinking, writing, and donning and doffing, where most tasks require some form of dextrous manipulation of objects. [1]

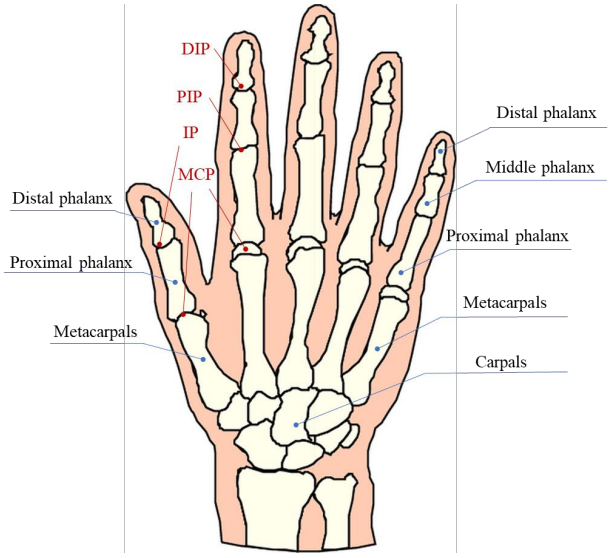
A comprehensive list of different objects commonly used in daily life is tabulated in [2], including the different weights and friction coefficients. They divided the objects into different task categories, some of them being *Food Preparation*, *Housekeeping*, *Office Tasks*, *Hobby/Sport*. The typical weight and dimension of each object is incorporated where the majority weigh less than 1 kg and has at least one dimension of less than 10 cm for the feasibility of grasping.

In the case of the simple task of drinking, the hand performs a series of fast and accurate adaptions to align the fingers for an encompassing grasp of the container with fluid. Along the path towards the object, the hands and eyes coordinate when to start closing the fingers to temporally align with the approach at the object. Subsequent to this, the object is lifted off the ground requiring the proper amount of force to avoid the object slip arising from grasping too softly or object damage arising from grasping too hard. When the object has been lifted, it must be smoothly directed toward the mouth, maintaining the orientation along the way, to avoid spilling. Finally, the hand must be gradually tilted at a regular rate before placing the container back at the same place it was picked from. As indicated by this simple task, a high amount of hand dexterity and control is required along with a tight interconnection between the hands and the eyes. Additionally, grasping the object is an essential part of the task; without grasping, almost all activities are rendered infeasible. To understand the kinematics and dynamics of these grasps, a deeper look into the hand anatomy is a central aspect to consider.

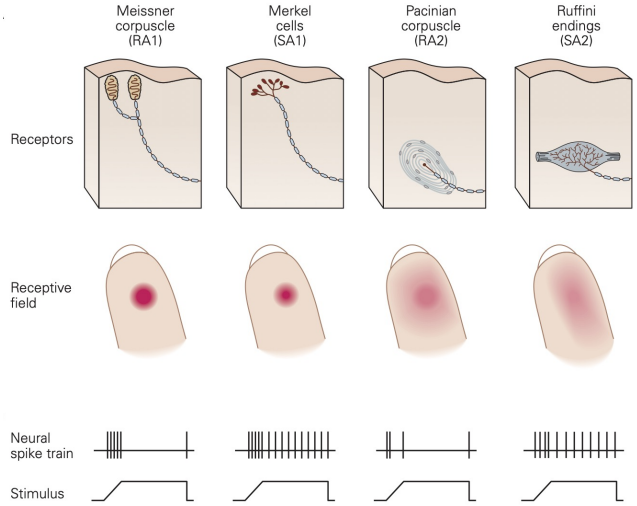
2.1.2 Anatomy

The human hand has a total of 24 degrees of freedom (DOFs) including three at the wrist. Each finger has four DOFs except the thumb which has five DOFs. The index-, middle-, ring-, and little finger contain three bones referred to as the proximal-, middle-, and distant phalanges, while the thumb only

has the two exterior phalanges, as seen in Figure 2.1a. The corresponding joints connecting these are denoted as the distal inter-phalangeal (DIP), the proximal inter-phalangeal (PIP), and the metacarpal-phalangeal (MCP), respectively. Fingers generally provide inward/outward motion, flexion/extension. The thumb is placed opposite to the other fingers and has therefore biologically evolved with larger range of motion and strength compared to the other fingers that are placed laterally with respect to each other. As such, the thumb is a key component in virtually all grasps to provide the opposing normal force. Each finger is connected to the wrist through the metacarpals that are placed within the palm of the hand. [3]



(a) Anatomy of the human hand. Acquired from [3].



(b) Mechanoreceptors and their individual properties. Acquired from [4].

Figure 2.1: The human hand structure.

In addition to its anatomical structure, the hand also consists of many small receptors in the skin to sense aspects such as pressure, vibration, texture, and heat and transmit them to the central nervous system. There are four of these receptors, known as mechanoreceptors, which are named Meissner's corpuscle, Pacinian corpuscle, Merkel's cells, and Ruffini's corpuscle. When these receptors are mechanically stimulated, they fire action potentials which is transmitted to the central nervous system and into the brain to generate the sense of touch. Each of these has high sensitivity, meaning they respond to a low amount of stimulation, however, they also have different properties with respect to the receptive field and adaptivity to stimulus, as seen in Figure 2.1b. The Meissner corpuscles and the Merkel cells are situated in the superficial layer of the skin, known as the dermis, entailing a concentrated receptive field. On the other hand, the Pacinian corpuscles and Ruffini endings are situated in the inner layer of the skin, known as the epidermis, entailing a diffuse receptive field. Furthermore, the differences in response to stimulus over time can be seen at the bottom of Figure 2.1b. Stimulus of these receptors provide tactile sensation in the hands. Tactile information is utilized during manipulation, and it is crucial for able bodied humans to perform effectual manipulation. It permits humans to form effective grasps without focusing on the object. [4]

2.1.3 Grasping

Human hands are the major prehensile of the human body, meaning that they are capable of grasping objects. When picking up an object, humans are capable of subconsciously determining which grasp obtains the most secure and firm grip of the object. The specific grasp posture can depend on various parameters, such as the shape, size, weight, surface properties, and structural integrity of the object,

although the main one is the goal of the specific task of which the object will be a part of [5]. It was proposed that human grasps could be categorised into two major categories, which are precision and power grasps. These differ based on the purpose of the grasp, which is either for security and stability for the former or dexterity and accuracy for the latter. These categories along with a tree like structure of more specific grasp types can be seen in Figure 2.2.

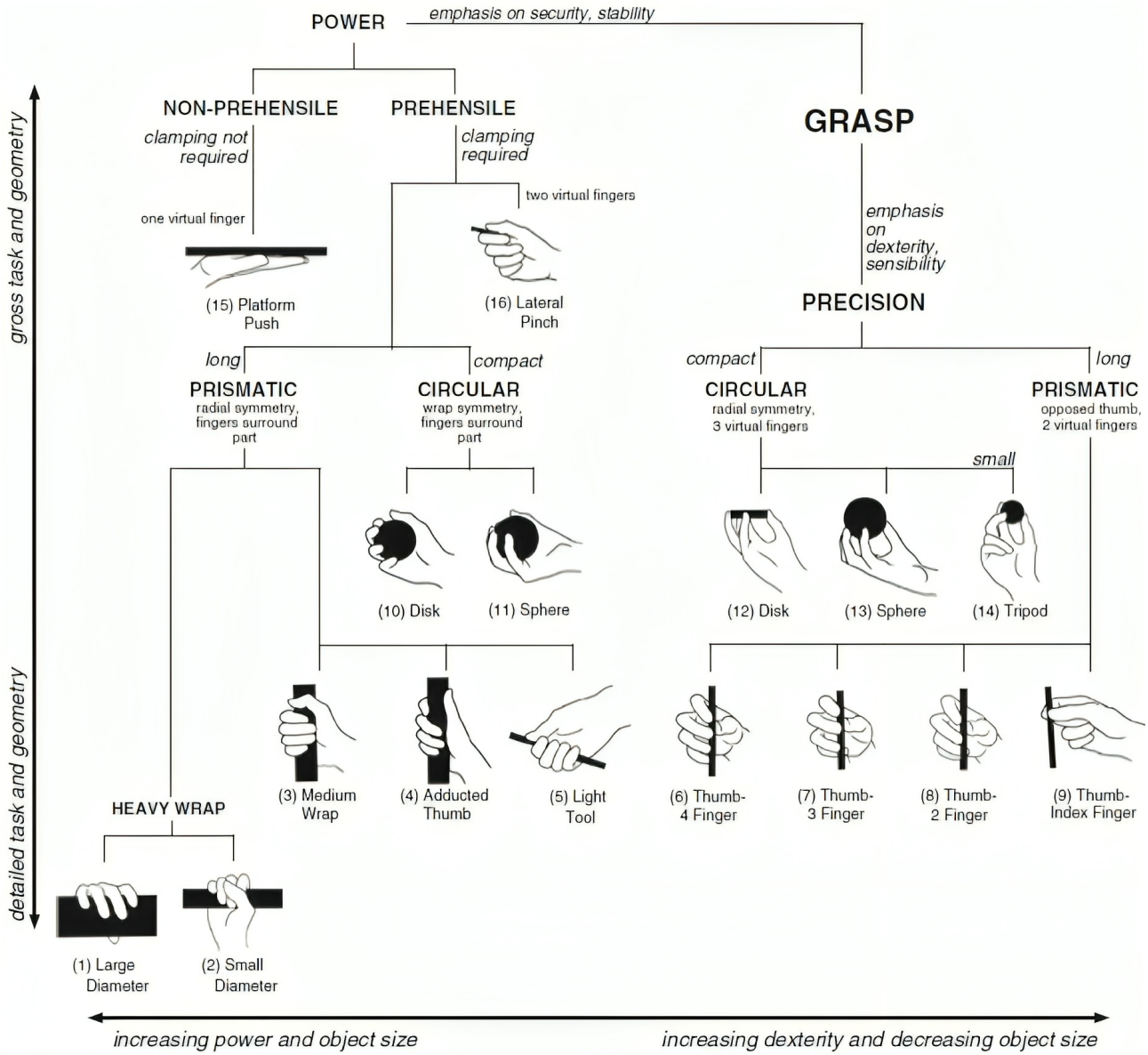


Figure 2.2: Grasp categorisation into power and precision grasps which differ based on object characteristics, motor control requirements, and task purpose. Acquired from [1].

The prehensile capability of the human hands is enabled by position of the four three-phalanx fingers and the thumb, which together can form an opposition space. The creation of an opposition space means that there is an equal and opposite force exerted from at least two sides on the object, thereby creating friction between it and the fingers, and subsequently enabling grasping as seen in Figure 2.3. If a proper opposition space is not formed or the friction coefficient of the object is low, the object may inadvertently slip from the fingers. In reality, the opposition space can be created by any two pairs of fingers as long as control of the object is maintained throughout the action, however, the flexibility, strength, and positioning of the thumb with respect to the other fingers makes it a key contribution in almost all grasps. [6, 7]

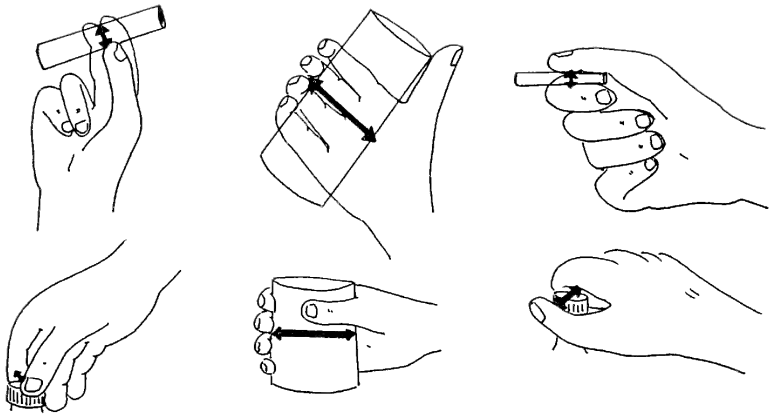


Figure 2.3: Opposition spaces of some basic grasps, where the arrows indicate the direction along which force is being exerted. Acquired from [7].

Prior to the actual grasping, the hand aligns the orientation and pre-shapes to the object geometry as it is approaching the object in question based on available visual information. Furthermore, the fingers commence closing at some point during the motion toward the object. As the individual fingers make contact with the object, tactile feedback is sent to the brain which in turn makes correcting actions to conform precisely to the object in the selected grasp posture. As such, several components of the prehension mechanism of the human hand occur in parallel and utilises exteroceptive sensing of the environment. [6]

2.2 Conventional Prosthetics and Control

In the recent years, technological advancements have continuously improved the dexterity of prosthetic hands and the performance of the embedded sensors. Although the electromechanics continue to reduce the gap to the ideal biological counterpart, the rejection rate of dexterous prostheses remains high and instead simple prostheses, such as the passive and body-powered, hold market dominance. While several factors contribute to this, the most prominent stems from control interfaces and human-machine interaction performance of prostheses that are insufficient in supporting dexterous operation. This can be attributed to the paradox of prosthesis control; the higher the amputation level, the more functions need to be replaced by less independent muscle signals. Contrary to the complex dexterous prostheses, the simple solutions offer simplicity, fast input response, and low cost, among other advantages. In this section, the functioning and control of the most common simple prostheses will be detailed, which can be categorised as passive, body-powered, and externally-powered. [8, 9, 10]

2.2.1 Passive Prostheses

This category constitutes prostheses that typically do not incorporate joint actuation or electronics and can be split into cosmetic prostheses and activity-specific prostheses. Cosmetic prostheses, such as the one seen in Figure 2.4a, are designed to look indistinguishable from the lost limb, and can either be used daily or occasionally. Because self-image is tightly related to physical appearance, this type of prosthesis can help with the user's social engagement and well-being. Besides appearance, the cosmetic prosthesis can also provide functionality, since some models allow one or multiple joint to be configured to, for instance, grasp or hold objects. This functionality still relies on the contralateral healthy limb to perform the primary manipulation task. Activity-specific prostheses are designed for specific tasks, as seen in Figure 2.4b, and have emerged since a single prosthesis cannot be ideal for all different tasks. These prostheses come with special termination tools that serve the same functionality

as the lost limb for the specific task. This allows users to participate in recreational or sporting activities, which contributes to their well-being and physical health. [9]

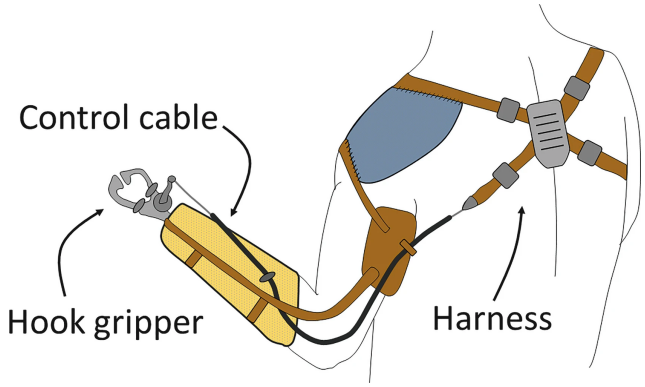
(a) Example of a cosmetic prosthesis. Acquired from [11].

(b) Examples of activity-specific prostheses. Acquired from [12].

Figure 2.4: Examples of passive prostheses.

2.2.2 Body-Powered Prostheses

This category of prostheses uses the proximal muscles of the user for simple actuation, such as flexion of a prosthetic elbow or opening and closing of a grasping mechanism. Such devices require the user to mount a harness through which cables or wires attached to the moveable joints of the prosthesis can be controlled, as illustrated in Figure 2.5a. As such, there is a direct mechanical link between the voluntary proximal muscle and the lost distal function that is being replaced. This solution is a common and popular choice due to its simplicity, light weight and low cost, but it also includes several shortcomings. It typically lacks simultaneous control of multiple joints, and requires dedication of the given muscle group for the prosthesis actuation. An additional drawback for these is that they can appear less cosmetically pleasing if a silicone or rubber cover is not used, and that they can be unnatural to those who are uninitiated. [9]



(a) Detailed view of body-powered prosthesis setup. Acquired from [13].

(b) Example of a simple 1 DOF gripper. Acquired from [14].

Figure 2.5: Body-powered prostheses.

2.2.3 Externally-Powered Prostheses

Externally-powered prosthesis are more advanced than the previous counterparts by including actuators and sensors. Unlike the body-powered prosthesis which dedicates some proximal muscles to replace the lost function, this prosthesis can take over some of the functions and allow more joints to be controlled, which in turn increases the number of DOFs and gives greater motion variety. Currently, the most common control interface for this prosthesis type is surface electromyography (sEMG) control. In

this method, two or more electrodes are placed on the skin of the individual from where they measure electrical potentials created as muscles are activated. With a higher amount of electrodes, a higher degree of control can be achieved as more muscles are recorded. Typically, these electrodes are placed on the forearm of the user as these muscles directly correlate to the movement of the hand. This enables the use of digital signal processing techniques to determine what action the user intends to perform. Alternatively, control can be achieved with intramuscular EMG, where the electrodes are injected into the human arm enabling higher selectivity and signal to noise ratio. However, since this approach is invasive, it is preferred less than the non-invasive sEMG. [9]

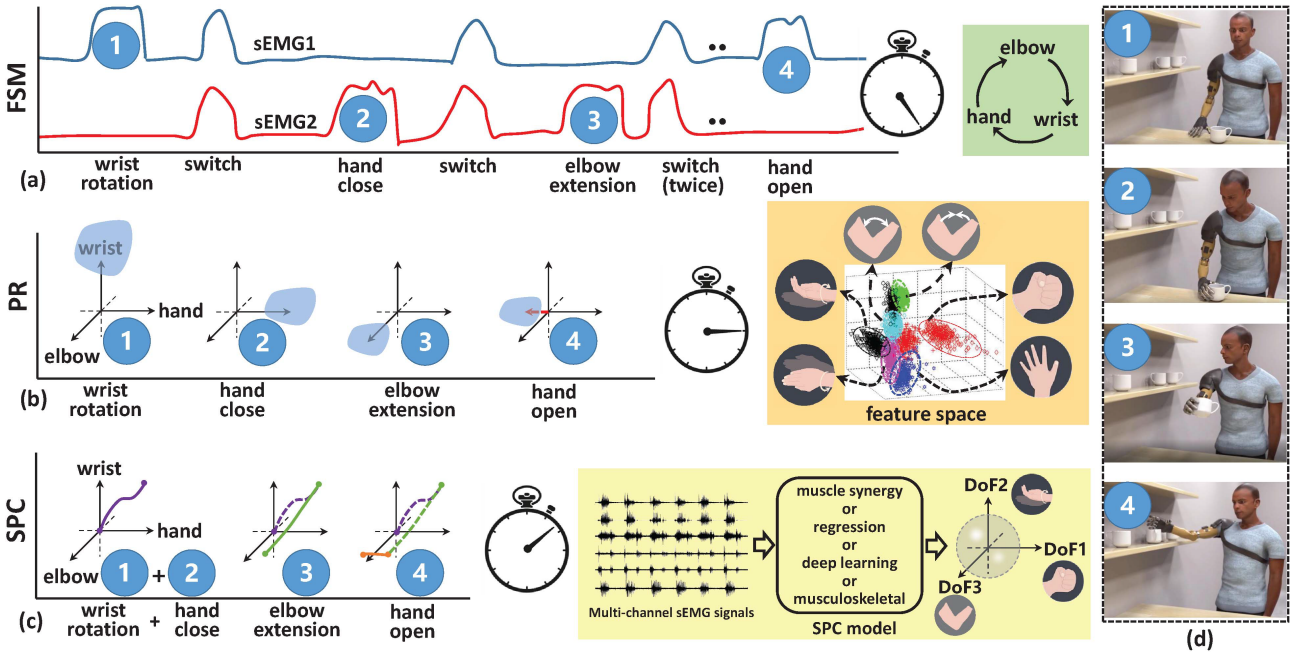


Figure 2.6: The three different control strategies for the sEMG interface. From top to bottom: Finite state machine, pattern recognition, simultaneous and proportional control. Acquired from [10].

There are in general three different control methods for sEMG, as seen in Figure 2.6, which are; 1) state machines, 2) pattern recognition, 3) simultaneous and proportional control. For the state machine approach, the control can be achieved with only two electrodes on either side of the forearm to capture data from two independent muscles. The basis of the method is to circle between states that correlate to some specific action or DOF by activating both electrodes simultaneously. When a given state is selected, each electrode controls a direction of the motion. This is a simple scheme but is not efficient for more complex tasks as only one action can be controlled at a time and circulating between states is time-consuming. The pattern recognition approach goes a step further and uses several electrodes placed around the arm and analyses the signals over small time windows to classify desired actions. The classification can be obtained with various machine learning methods, such as decision trees, support vector machines, or neural networks. Nevertheless, the control is still divided into a finite set of discrete actions and is subsequently still not a natural control architecture for humans. On the other hand, the last control method, simultaneous and proportional control, the movements are continuous and fluid as would be the case for an able-bodied human. In this strategy, a linear mapping between sEMG signals and human body kinematics is used to approximate the simultaneous motions of several DOFs. Optimising this mapping is still an active research area with some of the promising methods being deep learning, matrix factorisation, linear regression, and musculoskeletal modelling. [10]

While EMG is the most common, another similar, yet different solution, is the use of force myography (FMG). This approach uses force sensing resistors placed on the forearm to directly measure mechanical

muscle contraction instead of electric potentials. Here, a change in muscle stiffness is measured relative to a default relaxed state when the muscle is activated. While this has some advantages compared to sEMG, such as not being affected by sweat, there are also disadvantages, such as being sensitive to external forces and only being able to measure the larger muscles. [15]

2.3 Modern Prosthetics and Control

Modern prosthetics is here defined as the domain of prostheses that focuses on solving the problems of conventional practices. These problems are tackled by using higher degrees of autonomy and complex control methods to achieve a greater user experience for the amputees. However, they still lack behind in areas like multi-finger control, user intention recognition, and suffer from poor neural interaction performance. In addition, the amputees report that they experience an excessive cognitive burden due to the increased complexity of control. It is also stated that there remains inadequate technologies to achieve efficient anthropomorphic manipulation. Subsequently, proprioception and motion sensation of the prosthesis is rarely provided to amputees, resulting in the visual attention being completely preoccupied during the entire task, which in turn raises the cognitive burden further. Therefore, the increase in complexity and functionality with the intention of increasing ease of use of the prosthesis, ironically leads to an increase in cognitive burden. As a result, these problems constitute the foundation of the high rejection rate of modern day prostheses. [10]

The above mentioned problems resemble the inherent defects of insufficient and relatively bad information transmission between the human and the prosthesis. Consequently, there is a clear increase in the need for automating the control of prostheses, all while taking into account the cognitive burden associated with the device. In this section, different solutions that attempt to address these problems are explored, highlighting both advantages and disadvantages.

2.3.1 Data Modalities for Semi-Autonomous Prosthetics

The progress in the area of increased automated control in upper limb prosthetics has in recent years especially focused on tightly coupling vision into the control pipeline, as images have the potential to convey a large amount of information in a short amount of time. However, there are other exteroceptive modalities also capable of sensing the surrounding environment which have been used in successful implementations, such as proximity sensors and tactile sensors. For enhanced information extraction about the manipulation task at hand, these methods can be combined to utilise the complementary differences of the technologies. This sensor fusion is a key contributor in enabling the prostheses to obtain similar capabilities to humans.

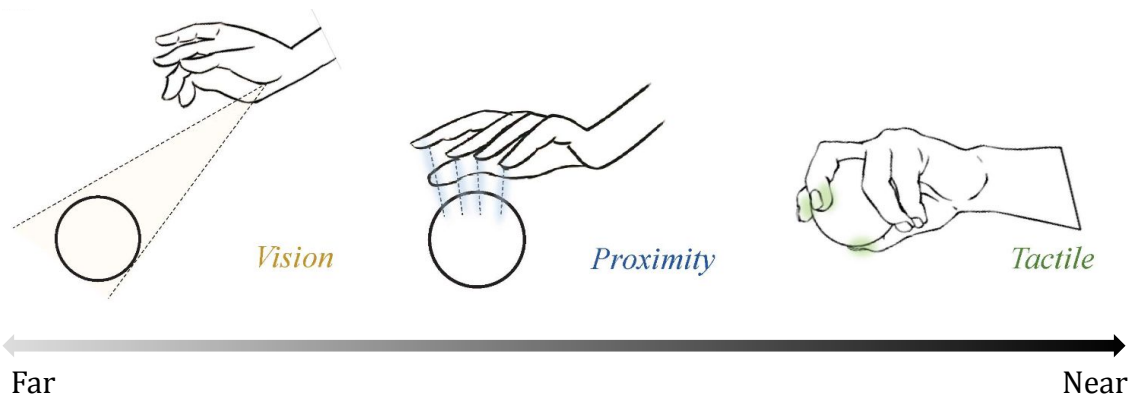


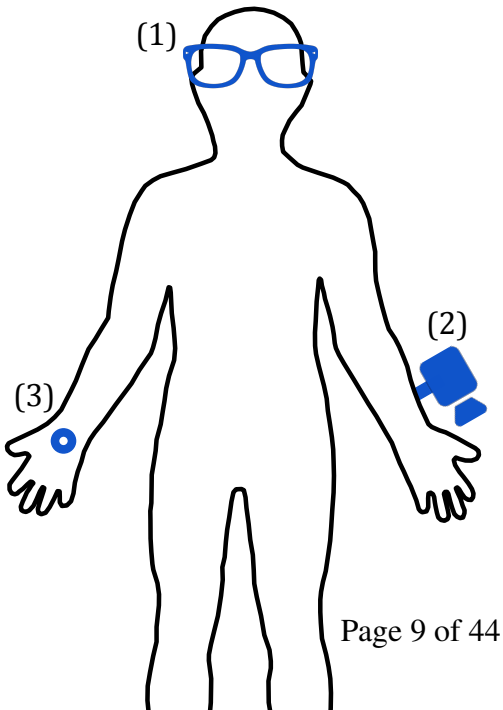
Figure 2.7: The three data modalities divided based on range. Adapted from [10].

- **Cameras:** Cameras offer a long range and information rich data extraction of the environment. They come in different variations but in general can be split into two major categories for prosthetic object manipulation, which are RGB and depth cameras. Based on the given purpose of the camera, one might prefer one over the other or even a combination. The RGB information can be combined with various CNN architectures to predict object pose, type, approach trajectory, and possibly grasp, however, adding the third dimension enables point cloud processing to aid in the prediction. Computer vision swiftly transforms modern prostheses by offering them the ability to foresee and interact with their surroundings. This has paved the way for the development of complicated control and data modalities by promoting higher degree of autonomy with the ultimate goal of providing a more natural and intuitive experience for users. Computer vision allows modern prosthetic limbs to understand their surroundings which aids in collision avoidance and navigation, object manipulation and grasping by recognising and classifying objects, and the interpretation of body movements and gestures allowing for more natural and intuitive control of the prostheses. [10]

- **Proximity sensors:** The middle range option is proximity sensors, which are simple devices that transmit electromagnetic radiation. When these encounter an object, they are reflected off it which alters the signal as it returns back into the receiver. This enables the sensor to detect when an object appears within some maximum distance to the sensor. As with tactile senors, the proximity sensors are most useful when aggregated in an array for more information extraction. These sensors can be utilised to determine when a given object is within grasping distance of the prosthetic hand and to pre-shape the hand to the object geometry for improved grasping quality. Since, a camera loses focus and light as the object becomes close, the proximity sensor is useful to bridge this gap, occurring in the hand closing phase of the grasp. [10]

- **Tactile sensors:** The tactile sensor is a short range modality which requires contact between it and a manipulated object. As this contact occurs, the sensor determines the interaction forces that arise between them. These sensors obtain contact information pertaining to the mechanical and physical properties of objects, such as shape, texture, temperature, and elasticity, which is not as easily available for e.g. proximity sensors and cameras. This information can be central to determining when contact with the object has been made or even for slip detection and grasp quality evaluation and adjustment. Tactile sensors are most suitable when combined, such as multi-dimensional force sensors or high-density tactile arrays, in order to obtain as many sensing points on the prosthetic hand as possible. [10]

As mentioned, some or all of these modalities can be combined to obtain a larger operating range, however, other interoceptive sensors can also be included in the sensor fusion. A common application is the use of inertial measurement units (IMUs) to determine hand pose and movement during grasping. This can be helpful to determine the user intention about which object to grasp in a cluttered environment. Based on the selection of a modality for data collection, a control strategy is required to utilise this information in performing tasks.



A significant consideration to make when attempting to utilise sensors for the prosthesis control use case is the placement of them. For tactile sensors, the placement is likely somewhere on the inside of the palm and on the tips

of the fingers, however, for cameras the placement is less apparent. The desired effect would be to obtain a large viewing area with minimal occlusion, while accounting for the required portability of the camera. This means that the camera needs to be a part of the individual with the prosthesis, which brings about two general placements, i.e. on the face or on the hand. For the face-mounted option, the camera could be part of a pair of augmented reality glasses, which could additionally display information about the current state and intention of the prosthesis [16]. However, this entails that separate glasses are needed in addition with the prosthesis and that the user always looks at the desired object to grasp. For the hand-mounted option, the camera can be placed somewhere around the wrist or in theory integrated inside the hand. With a camera inside the hand, occlusion would be minimised as the camera would always face in the same direction as the hand [17].

2.3.2 Semi-Autonomous Control Strategies

The efficient and natural control of prosthetic hands is likely the most pressing problem in successfully integrating semi-autonomous solutions into commercial use. The purpose of these methods is to offload some of the decision-making of everyday tasks to the prosthesis, and as such it is essential to obtain natural human behavior. This is not straightforward since the desired grasp of an object often depends on the intended task said object is used for. This is referred to as grasp affordances. As an example, when grasping a coffee mug, you likely want to grasp the handle instead of the hot cylindrical part for both practical and social reasons. Furthermore, the grasp needs to be of good quality to avoid slippage. The prosthesis should account for these aspects to have a low rejection rate in the amputee community. The control architecture is not easy to categorise into exact descriptions as they vary widely, although common for most approaches is the use of computer vision as suggested by the previous section. Here, some general methods that are compatible with computer vision are highlighted.

2.3.2.1 Point Cloud Processing

A point cloud consists of an unordered sequence of 3D points acquired through depth imaging from e.g. a stereo setup, structured light, or time-of-flight. For a given image, each pixel within some allowable distance range is converted to a 3D point to obtain the structure of the camera-facing side of the object in frame. Since point clouds differ from images in the sense that they do not have a clear spatial neighbourhood and are not unique due to permutations, different processing strategies are used. From the geometric information of objects that can be deduced from point clouds, an approximate grasping strategy can be inferred, with e.g. a simple decision tree. To realise this, the object in question can be isolated from the background, oftentimes a table, by means of segmentation algorithms. One common approach for table-top removal is to use RANSAC for finding the best-fitting plane in the point cloud and removing it.

2.3.2.2 Neural Networks

In recent years, CNNs have seen a large improvement for a variety of different applications, an example of which is object detection where Yolo, Faster R-CNN, or SSD are some of the leading algorithms in accuracy and speed [18]. For a grasping task, CNNs can be used to predict the 6D pose and shape of objects to aid in proper grasp selection. The grasp may be selected as being normal to the narrowest dimension of the object's oriented bounding box which is in alignment with typical human grasping

behaviour. The neural network approach is not exclusive to RGB images, but can equally be applied to point clouds. However, due to the differences in representation, different structures must be used, such as graph neural networks or recurrent neural networks [19]. Again, the point cloud data can be used to capture more accurate geometric data of the objects, which may enhance pre-shaping capabilities in the approach trajectory. These neural network structures can be combined with a secondary neural network to utilise haptic data when grasping the object to improve grasp quality and slip detection.

2.3.2.3 Reinforcement Learning

A similar approach to neural networks based on learning is reinforcement learning (RL), where an agent learns the optimal actions through experiences gained from iteratively exploring the environment. This Monte Carlo approach is required due to the stochastic nature of the environment, where the actions and observations are probabilistic. With RL, the agent may learn to pre-shape to an object and possibly even grasp it based on sensor data from the camera and hand to form the state and observation. While robotic grasping has been an active research topic with several solutions being based on RL, the leap to upper-limb control of a prosthetic device is still in its infancy. A large difficulty in the integration of RL into this field is the free base of hand which is purely based on user intention. This means that for a grasping task, the full state of the hand cannot be steered in a specific manner towards the object, but instead the hand needs to adjust for the human movement in real-time throughout the process.

2.3.2.4 Bidirectional Control

To close the loop in the interaction between human and robot, not only forward control signals need to be given from the user, but also feedback stimulation needs to be returned to the user as seen in Figure 2.9. This is in concordance with real life manipulation which is only made possible by continuous feedback from the human senses, predominantly vision, touch. Including this layer adds extra functionality and increases the natural feel of using the robot. For an upper-limb prosthesis, the visual feedback is already present, but the feeling of touch on the glove is missing. This can be introduced by means of sensory feedback on an intact limb of the individual by means of e.g. vibro-tactile or electro-tactile stimulation.

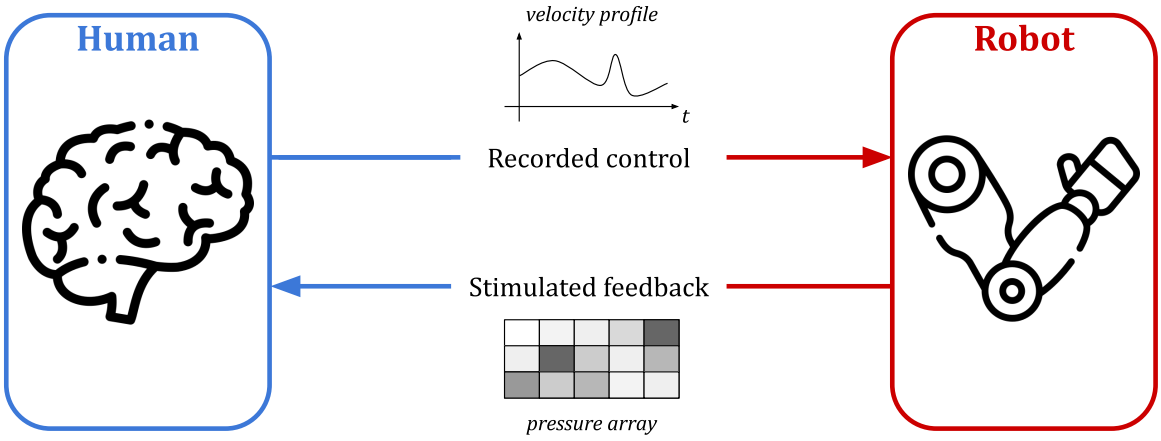


Figure 2.9: Example of the bidirectional control architecture between a human and a robot.

Achieving the closed loop control scheme enables a higher level of embodiment for the user, granted that the feedback has a natural feel and low latency. This can in turn aid in improving the perceived and actual control of the prosthesis.

2.3.3 Literature Overview of Semi-Autonomous Control

From the established methodologies for data extraction and hand control, some existing solutions for semi-autonomous upper-limb prostheses that to some extent incorporate one or more of these are here introduced. The solutions are intended to provide an overview of what has already been done and what is still yet to be explored. One of the less investigated issues with the most potential will ultimately be selected as the premise for the remainder of this project to contribute in the research area.

2.3.3.1 Prosthesis Control with Sensor Fusion

The process of human movement control and decision making for grasping is characterised by fusion of multisensory information, encompassing both exteroceptive and proprioceptive biological sensors. Such sensor fusion in the context of prosthetic device control could allow the high-level user processes to be assigned to the prosthesis controller, thereby automating tasks that the prosthesis user originally was responsible for. This observation was the basis for the approach taken by Markovic *et al.* [20], where the authors propose and test CASP, a context-aware controller for semi-autonomous and reactive control of a prosthetic hand using sensor fusion. The controller, depicted in Figure 2.10, constitutes two control units that the user voluntarily can switch between; the Automatic Control Unit (ACU), denoted by (1), and the Myoelectric Control Unit (MCU), denoted by (2). The ACU can automatically, simultaneously, and proportionally adjust multiple DOFs, which includes adjustment of grasp type, grasp size, and wrist angle. This is achieved by fusing information from two components, namely the Artificial Exteroception Module (AEM) and the Artificial Proprioception Module (APM), denoted by (3) and (4) respectively. While the former is based on a head-mounted RGB-D camera used to acquire shape, size, and orientation of objects at the centre of camera FOV, the latter constitutes an IMU to acquire the prosthesis' absolute orientation together with joint position sensors to determine wrist rotation and grasp aperture. The information from AEM and APM is subsequently fed into and fused by the Sensor Fusion Model (SFM), denoted by (5), to determine the optimal grasp pre-shape and orientation. The MCU is a manual controller that allows the user to trigger, supervise, correct, and fine-tune the ACU at any instance in time. This is achieved by a finite state machine, which enables sequential and proportional control by a myoelectric interface.

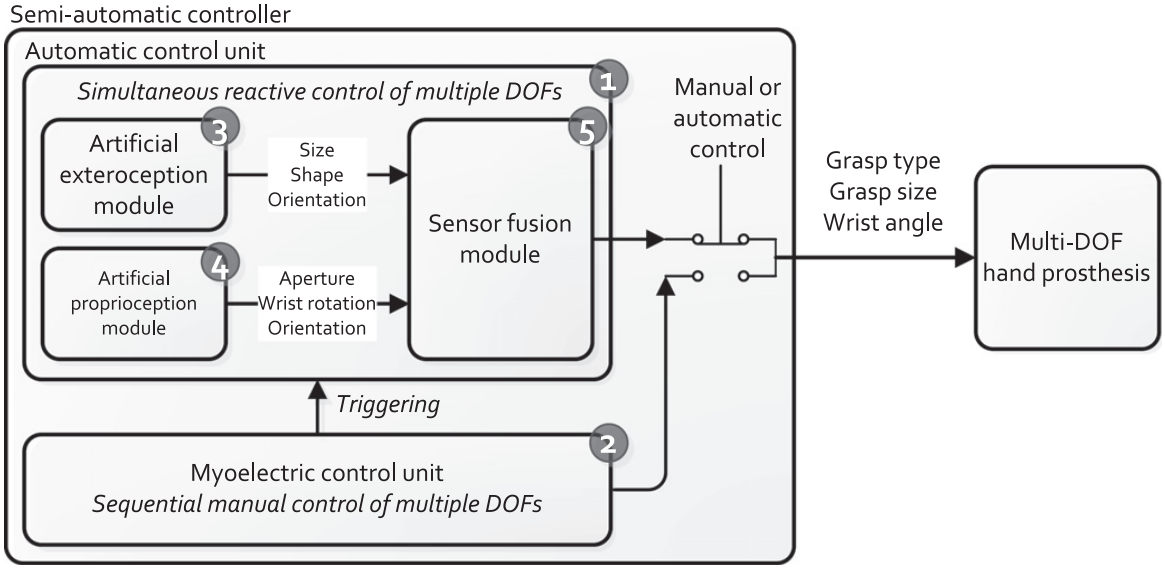


Figure 2.10: The control system proposed by Marko Markovic *et al.* [20].

The CASP system was tested with a 3-DOF Michelangelo Hand and compared to three classical manual control scenarios of varying complexity, which relied on a finite state machine. Some of the

notable findings of this comparison were that CASP was faster for complex scenarios, which in this case was manual control of 3 DOF, but slower than manual control of 1- and 2-DOF. Although these two simpler manual control scenarios were faster, the user employed more compensatory movement, as opposed to CASP or the 3-DOF manual scenario, due to the lack of wrist rotation by test design. Additionally, unlike manual scenarios, the preliminary training sessions did not achieve any notable improvement for the subsequent evaluation session.

2.3.3.2 Prosthesis Control with Vision and Electromyography

He *et al.* [21] developed a distributed control system for a vision-based Myoelectric Prosthetic Hand, which mainly consists of the Myo Armband from Thalmic Labs, a wearable armband retrofitted with eight electrodes for measuring EMG signals. The webcam was mounted on the top of the hand and was used for special key features of the vision system, such as object detection, recognition, grasp posture selection, and real-time feedback. The authors used an adapted YOLO architecture for real-time object detection. To grasp the target amount, because the number of detections was high, the grasp target was defined as the bounding box in the center of the image. This concept represents the likelihood that an object will become a target for grasping the object.

Handa *et al.* [22] proposed a vision-based teleoperation of a dexterous robotic hand-arm system, called DexPilot, which allows the control of 23 degree of actuation robotic hand system by observing only the bare human hand. This teleoperation set comprises an Allegro hand along with Syntouch Biotech tactile sensors fitted at the fingertips and the KUKA LBR iiwa7 R800 series robot arm. The area that observes and measures the motion of the human hand consists of four time-synchronised calibrated Intel RealSense D415 RGB-D cameras, which are spatially arranged to maximise the observation volume. This system relies heavily on the DART model-based tracker, which creates the backbone for tracking the joint angles and poses by utilising an articulated model of the hand against an input point cloud.

Wang *et al.* [23] discuss the integration of vision data with myoelectric prosthesis that promises restoration of hand abilities. This study utilised RetinaNet, a one-stage CNN for object detection, and a DcnnGrasp dual channel CNN for grasp classification using object category information, which improves grasp pattern recognition. This study investigated the deviation in grasp classification accuracy over the grasping period. The period capable of achieving the best grasp classification is introduced as the ”sweet period”. In addition, this study discussed the integration of the classification outcomes of sEMG and vision. This proposed method has increased the accuracy from 85.80 % to 90.06%.

Došen *et al.* [24] discussed a novel concept regrading the development of a CVS (Cognitive Vision System) for the control of grasping in dexterous prosthetic hands which is empowered by rule-based reasoning and computer vision that generates orientation and pre-shaping commands for control of prehension. The conceptual scheme is shown in Figure 2.11. It demonstrates when the user changes the orientation of the hand and issues the command using the EMG interface. CSV first estimates the object properties (shape and size) and autonomously selects the appropriate grasp type and aperture of the object using simple IF-Then rules, while the hand controller translates that grasp into desired force (for hand grasping) and finger positions (for hand pre-shaping) and sends to the low-level controller. This low-level controller implements force control and closed-loop position control during grasping and pre-shaping, respectively. After performing 612 grasp trials with 13 healthy subjects, the results showed that CSV correctly estimated size and grasp type in approximately 84% of trials and showed a 90% task accomplishment rate.

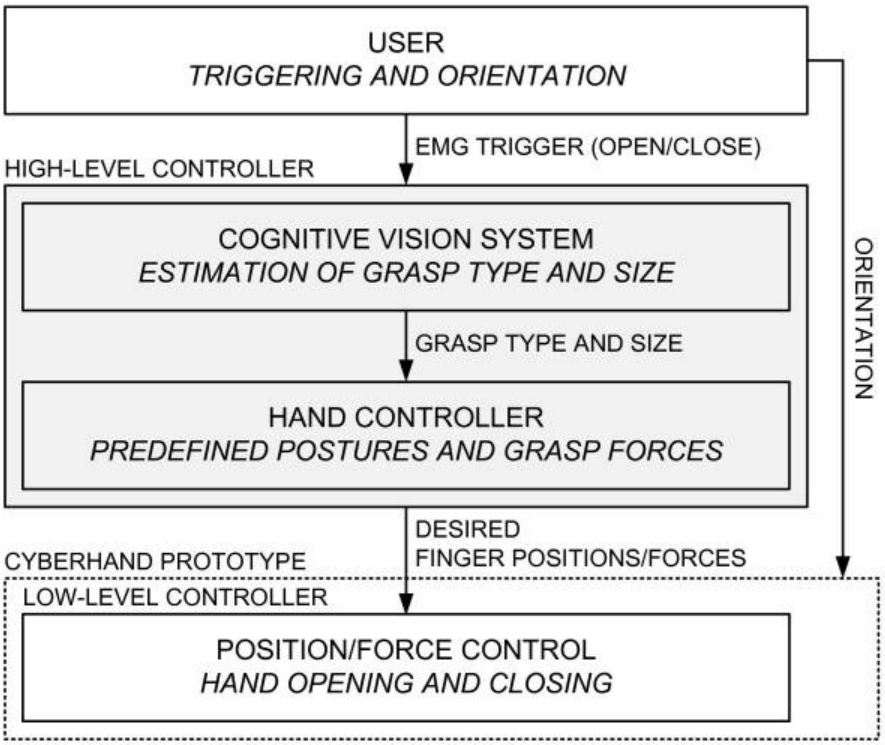


Figure 2.11: Control system architecture proposed by Došen *et al.* [24].

2.3.3.3 Prosthesis Control with Depth Sensors

Castro *et al.* [25] proposed a study that focuses on developing and evaluating a novel depth-embedded, semi-autonomous prosthesis control system using a single depth sensor placed on the dorsal side of the myoelectric prosthesis. The system aimed to automate the pre-shaping phase by adjusting the prosthesis configuration for users to effectively grasp objects in a variety of shapes, orientations, and sizes both in cluttered and isolated scenarios. The authors suggest a closed-loop proportional controller that generates velocity commands to continuously adjust the prosthesis configuration. It related to the targeted object orientation and dimension regardless of the approaching side of the target and the camera orientation based upon the desired state provided by computer vision. The authors implemented a depth-embedded semi-autonomous perception pipeline, as shown in Figure 2.12. To represent an object or part of an object, LCCP (Locally Convex Connected Patches) was used to segment distinct convex regions within the point cloud as shown in Figure 2.12(A) . The target was identified by projecting the point cloud centroids onto the camera plane and selecting the closest centroid. Then,the RANSAC algorithm was used to select the best-fit primitive model (sphere, cylinder, or cuboid) as probed in Figure 2.12(B). Based on the fitting model, grasp type and size were estimated using a decision tree (Figure 2.12(C)) that was created based on the object properties (length, width, diameter, and approaching angle for cylindrical models). Authors have used, The lateral grip for thin smaller objects and the palmar grip was used to grasp thick or long objects. In addition, the authors considered the cuboid model as a special case, and if two or three cuboid primitive faces were present from the aiming perspective, a ray-casting procedure was implemented. As revealed in Figure 2.12(D), the wrist orientation angle is defined by calculating the grasping direction vector from the model and projecting it onto the camera plane.

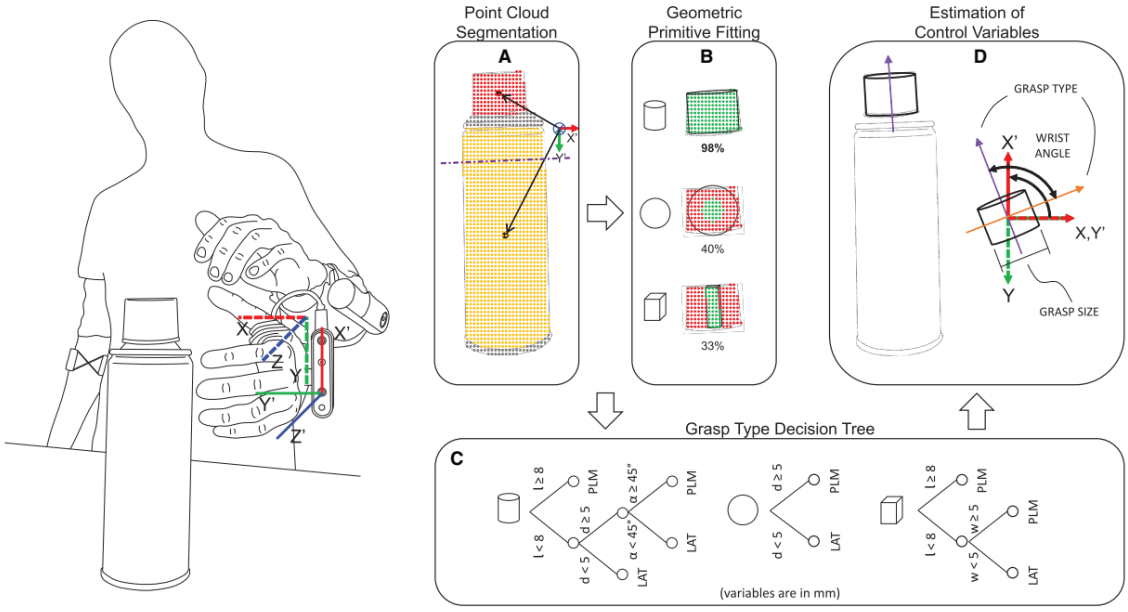


Figure 2.12: Depth perception pipeline proposed by Castro *et al.* [25].

2.3.3.4 Dexterous Manipulation with Point Cloud Reinforcement Learning

DexPoint is a framework proposed by Qin *et al.* [26] that enables dexterous manipulation of unseen objects in the same category as their corresponding training dataset. In addition, they propose a sim-to-real framework that focuses on generalisation of real-life deployment to their platform, which is a dexterous multi-finger robotic hand extending a robotic manipulator as seen in Figure 2.13. To facilitate the observation space of their reinforcement framework, they use an external depth RGB-D camera that observes the position of the object to be grasped in conjunction with the manipulator and robotic hand. As a consequence of using a single camera setup, parts of the hand will inevitably be occluded and thus not included in the point cloud. To combat this, they propose a technique to 'imagine' the occluded surfaces of the fingers and hand using forward kinematics of the kinematic profile of the hand. Subsequently this is augmented to the observed point cloud of the camera to collectively form an unoccluded observation of the environment. [26]

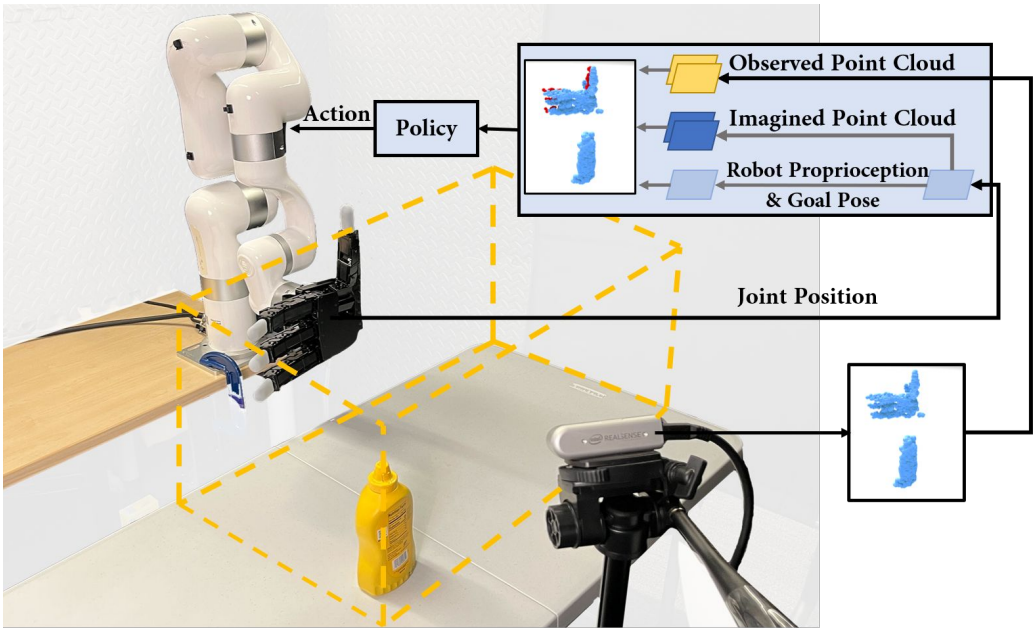


Figure 2.13: DexPoint physical setup with observations, actions, and policy illustrated.

In relation to the reward scheme in reinforcement learning, they also use contact pair information for reward design, without adding the contact directly to the observation, eliminating the need for a contact sensor [26]. This is crucial in highly dexterous robotic hands, as most of the volume of the hand is occupied by mechanical actuators, controllers, and even batteries, and therefore contact sensors are often not available on real robotic hands. In addition, they use a two-phase reward scheme, which is split up into reaching the object and then grasping the object. The reward scheme is based on a weighted combination of the following parameters:

- **Reach** is rewarded using the distance between the fingertips and the target.
- **Contact** is rewarded by ensuring that the thumb and at least two other fingers are in contact with the object.
- **Lift** is rewarded based on the distance between the object and the target, and will return a large reward upon task completion.
- **Control penalty** is given at each step.

Using this reward scheme, they uncover different findings and achieve great results. One of the findings include that the use of multiple objects and ablate parts of the algorithm during training, improves the robustness. The contact rewards proved crucial to have any convergence in learning, while the imagined point cloud reduces variance and improves initial learning rates. Collectively, these approaches in simulation resulted in $81 \pm 15\%$ and $68 \pm 9\%$ success rates for novel objects of bottles and cans, respectively. The current limitations of the solution includes the scope of the tasks, and a potential improvement of the solution could be to use a Recurrent Neural Network and temporal information for policy networks. [26]

2.3.3.5 Myoelectric Prosthesis Control with Reinforcement Learning

Conventional existing upper-limb prosthetics are limited in their ability to generate synergistic movement, and as was underlined in Section 2.2, there is a mismatch in prosthesis functionalities versus the available control signals. Vasan *et al.* [27] proposed to overcome these barriers with a controller that learns to predict prosthesis movement and thereby fills in the gap of limited control signals. By using a demonstration-guided reinforcement learning approach, as shown in Figure 2.14, a control policy for the prosthetic arm (control arm) can be trained by having it attempt to maximise position similarity to the contralateral intact biological limb (training arm). This training arm leverages the user's ability to create synergistic movement and will provide training motion that performs the same motion that simultaneously is being attempted with the residual limb. An actor-critic reinforcement learning agent assigned to each joint would receive negative rewards proportional to the difference between the desired joint angles of the training prosthesis and the current joint angles. EMG electrodes are located on the biceps and triceps on both upper limbs, and a motion capture glove with an IMU for tracking orientation is worn on the healthy contralateral limb to track the state of the training arm and compute rewards. Observations during training and testing consist of prosthesis joint states and EMG signals from the residual limb.

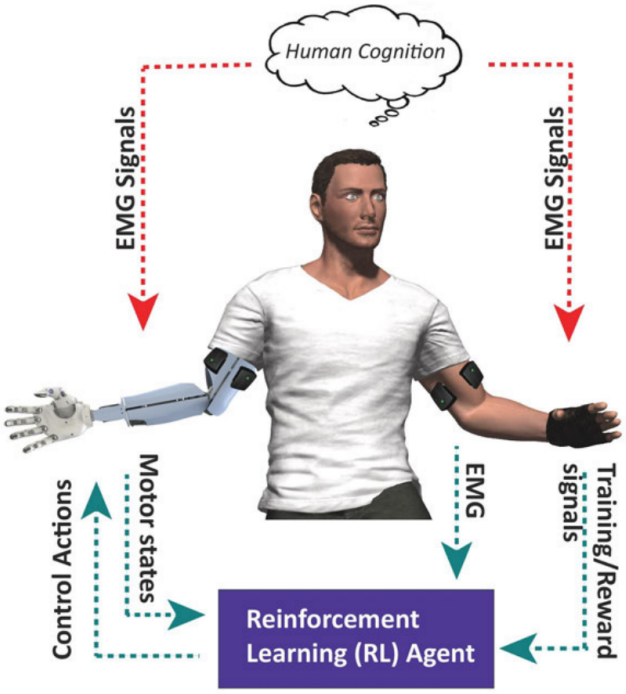


Figure 2.14: The demonstration-guided reinforcement learning approach proposed by Vasan *et al.* [27].

The results of this proposal showed that the joint agents continuously improved during training, and the performance during the last phases of training matched the performance during testing. Across all three test subjects, the angular mean absolute error (MAE) during testing remained below 0.2 radians for the gripper hand, below 0.3 radians for wrist rotation, and below 0.5 radians for wrist flexion/extension. When compared to a baseline consisting of reactive control, their proposed approach consistently showed higher MAE and exhibited occasional spikes in joint angles when a sudden transition occurred in the desired position.

2.3.3.6 Reinforcement Learning-Based Grasp Pattern Control

Nia *et al.* [28] proposed an approach to use deep reinforcement learning techniques to train a musculoskeletal hand model with 29 bones, 39 muscle-tendon units, and 23 joints to control achieving a set posture and grasping a ball in a virtual environment. The authors discussed the application of grasping a ball by controlling the musculoskeletal hand model in the MyoSuite framework with the selector-actor-critic deep reinforcement learning algorithm, which is an extended version of the actor-critic architecture that employs a continuous multivariate policy with maximum entropy. In the selector-actor-critic algorithm, the selector suggests the best action for each state by optimising the policy through the actor and examining each critic's approach. Furthermore, the associated hyperparameters for the selector-actor-critic algorithm training were the environment (MyoSuite framework using Gym environment), update factor, learning rate, gradient steps, discount factor, and selector-actor-critic policy. These are used to achieve precise grasping and orientation of the hand to perform the grasp by avoiding self-collision poses and coordinating finger activities.

2.4 Delimitation

Based on the explored solutions addressing the problem of automating upper-limb prosthesis control, a focused direction can be established for this project. As mentioned, point cloud data has been largely incorporated for gaining information about the size and shape of objects in the semi-autonomous

control paradigm enabling various control schemes. Furthermore, with the recent advancements in deep learning, RL has seen successful implementations of sim-to-real transferable models for dextrous robotic grasping such as DexPoint in [26]. However, these algorithms are trained on full robotic manipulators with the hand attached to the manipulator wrist, which differs from the problem of prosthesis control. Here, the major difference is that the base of the robot is free, meaning that it cannot be controlled by the robotic system and is completely dependent on the user. This has some consequences for the perspective of RL. Firstly, it entails that the system becomes less complex due to a lower amount of DOFs and consequently the action space is reduced. Secondly, it implicitly entails that to train the model, the approach trajectories to reach the objects must be generated either artificially or by recording actual human approaches using e.g. motion capture. While the latter might be the better approach, it requires more time, data, and setup than the former.

Having highlighted these aspects, there seems to be a gap in the research area regarding the utilisation of RL for semi-autonomous prosthesis control. Therefore, the project is here delimited to revolve around training an RL model for grasping of different objects with sim-to-real capabilities to provide a robust and generalisable solution. The environment comprises of a hand prosthesis with a depth camera for point cloud data that iteratively approaches target objects and attempts to grasp them. From a practical standpoint, there are different options as to how the semi-autonomous control occurs. The first option is to have the RL model always running and continuously adapting the pre-shape to faced objects, where the user then triggers the closing of the hand with EMG. Another option is to have the user initially trigger a grasping task with EMG by locking onto the currently faced object, whereafter the RL model is responsible for both pre-shaping and actually grasping the object. The user can cancel the task at any given moment to stop the lock on the object or let go of the object if it has already been grasped.

3 Requirements

In order to verify and validate the proposed solution, requirements are denoted to give a benchmark for what it should include from different perspectives. These requirements are inferred from the research completed in the problem analysis from Chapter 2. These requirements are divided into two categories for the solution, which are technical and clinical requirements. They are each divided into main requirements denoted by a number, and sub requirements denoted by a letter.

3.1 Technical Requirements

The technical requirements intend to verify the functionality of the proposed solution. As such, the solution should include the following technical aspects.

1. Provide pre-shaping and grasping of different objects to enable the capability of performing tasks of everyday life.
   (a) Framework includes a simulation interface
   (b) Human-realistic approach trajectories to the object
   (c) Modular to different prosthetic hands (structure, DOF)
   (d) Intention recognition in cluttered scenes
2. Provide sim-to-real capabilities that allows seamless integration for physical usage of the algorithm in real hand prostheses.
   (a) Framework includes a hardware interface
   (b) Generalisable and robust to objects not seen before
   (c) Real-time performance with low latency
   (d) Robust to noise in observations (camera data) and actions (actuator movements)
3. Provide comparable performance to state-of-the-art hand prosthesis solutions to ensure that the algorithm lives up to current standards in the field.
   (a) Average grasp success rate of 80% on seen objects
   (b) Average grasp success rate of 50% on unseen objects
   (c) Robust to Gaussian noise in observations and actions of up to XX%
   (d) Minimal time to complete grasp of $<$XX s
   (e) Minimal response time in actions performed of $<$200 ms
   (f) Minimal actuation effort of $<$XX W for grasping tasks

3.2 Clinical Requirements

The clinical requirements intend to validate the usability of the proposed solution. As such, the solution should include the following clinical aspects.

4. Be accepted by actual upper-limb amputees to improve their quality of life by aiding in ADLs.
   (a) Easy and natural control of the prosthesis
   (b) Reliable and predictable movements to allow safe operation
   (c) Low cognitive load during real-life tasks

Since the solution will only regard a simulation due to the limited time scope, these requirements will remain untested, but should in the future be incorporated in real tests with the actual target group for upper-limb prostheses.

4 Solution Proposal

For prosthetic control, it should always be the user that has the final if any undesired actions occur in the autonomous part of the control. To prevent user correction needing to happen too frequently, the user can be tightly integrated in the control scheme, although this comes at the expense of increased cognitive load. As such, there is a trade-off in the degree of automation for prosthesis control. In this solution, two levels of autonomy can be compared against each other as seen in Figure 4.1. Common for both methods is that the user initiates the autonomous grasping module, e.g. by inducing a spike in EMG for both extensor and flexor muscles on the forearm. As the user approaches the desired object, the prosthetic hand autonomously pre-shapes to enable a grasp on the given object. At this point, the two strategies differ. In the first one, the user has to perform the grasp with EMG to confirm that the action is desired and that the pre-shape is fitting; and in the second one, the algorithm itself is responsible for grasping the object with proper timing and exerted force when the user has reached within grasping distance of the object. In both approaches, the autonomous grasping can be cancelled by the user by issuing a certain command with EMG.

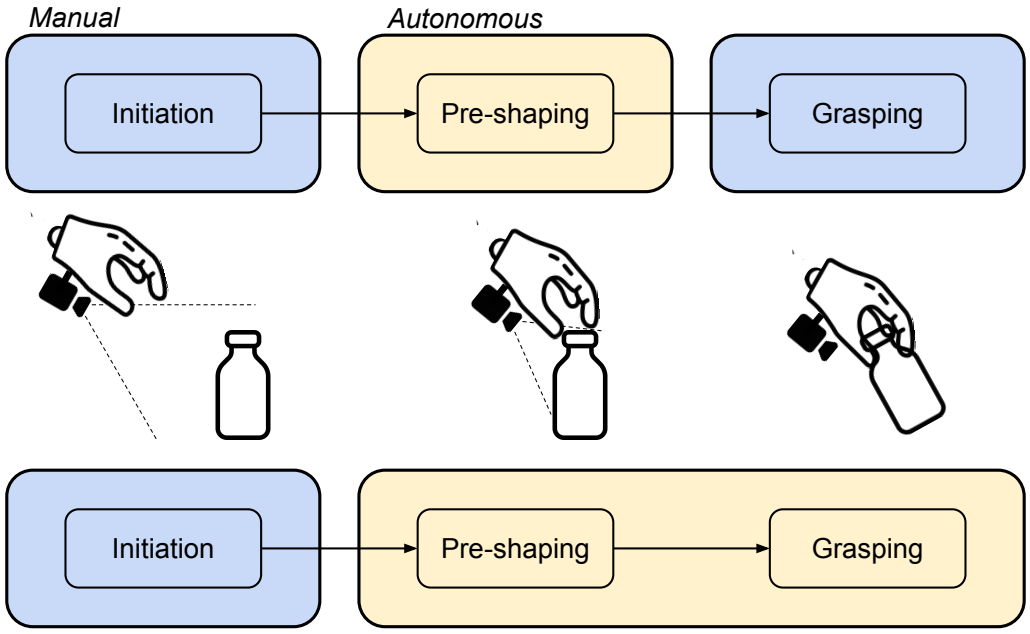


Figure 4.1: The two solution pipeline options available. On the top row, the top row represents the case where the actual grasping is done manually, whereas the bottom row is where the RL model performs the entire grasp.

This chapter delves into the different aspects of the solution to the posed problem statement in Section **??**, including the selection of hardware equipment for the physical implementation along with the simulation pipeline for training the reinforcement learning algorithm and verifying its applicability. The solution attempts to streamline the experience of prosthesis control by incorporating the user in the loop in a novel manner.

4.1 Hardware Selection

To facilitate the development of a simulated solution that is transferable to the real world, the choice of hardware must be designated. In this case, especially the mode of perception, i.e. camera, and mode of actuation, i.e. prosthetic hand, are essential aspects to consider. For these devices, there is a limitation to what is available for use, and consequently, the analysis and choice will only pertain to these. Additionally, a custom bracket for mounting the hand to the user and the camera to the hand

must be designed with several practical and technical aspects in mind.

4.1.1 Prosthetic Hand

For this project, two different anthropomorphic prosthetic hands are accessible, these being the Michelangelo hand from Ottobock, and the Mia hand from Prensilia. Both offer a durable and flexible solution with multiple DOFs and different possible grasps. While this gives them capabilities to mimic the functionalities of a real hand, they have some deficiencies with respect to highly dextrous movement incorporating independent finger motion. A comparison of the two hands can be seen in Table 4.1.

	 Michelangelo	 Mia R&D
Size	7.75 in	82×40×190 mm
Weight	510 g	480 g
Grasp Force	70 N	70 N
DOF	3	3/4
Grasps	7	5
Simulation	None	ROS support

Table 4.1: Comparison between available prosthetic hands.

Both hands are quite similar in the size, shape, weight, and number of DOFs. While the Michelangelo hand offers wrist rotation, the Mia Hand offers independent motion between the index finger and the middle, ring, and, little finger. The major difference between the two although for the sake of simulation is that the Mia Hand has integrated ROS support. For this reason, the Mia Hand will be selected as the main prosthetic hand for the development of this project.

4.1.1.1 Mia Hand

The Mia Hand is an anthropomorphic prosthesis hand created by Prensilia, which is used for amputees and in industrial applications. There are 3 different versions of the Mia Hand, which are the *Mia Hand Prosthesis*, *Mia Hand Industrial*, and *Mia Hand R&D*. Here, the *Mia Hand R&D* designed for research purposes and introduced in the previous comparison will be the point of focus.

The prosthetic hand exists in both a right and left hand version, and with three individually controlled finger groups, i.e. the index and thumb are individually actuated, while the middle, ring, and little fingers are coupled and thus controlled collectively. All the fingers comprise of a single phalanx, while the distal part of the fingers are covered with rubber, for higher friction and better stability during object manipulation. The hand provides 5 different pre-configured grasp types, which are a cylindrical, pinch, lateral, spherical, and tridigital grasp, some of which can be seen in Figure 4.2. Collectively, these grasp types cover around 80% of daily movements.

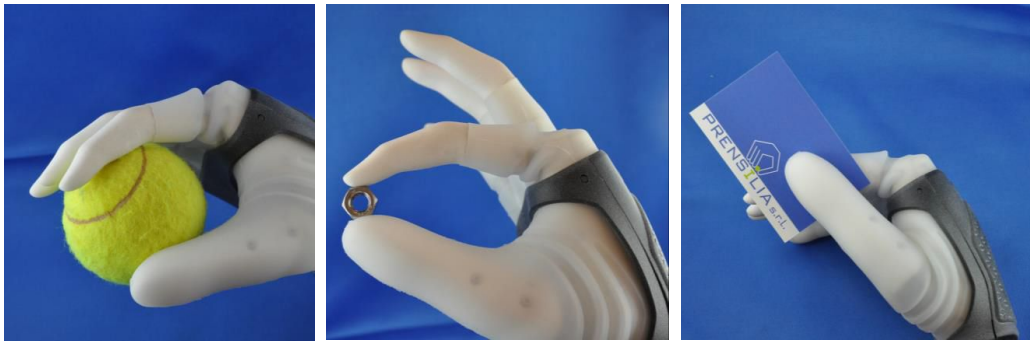


Figure 4.2: Three different grasps of the Mia hand. From left to right; cylindrical, pinch, lateral grasp.

4.1.2 3D Camera

For the 3D camera, two options are easily accessible, which are the PicoFlexx camera and Intel RealSense D415 camera. Both cameras are designed with an API for readily extracting the data in code, entailing that they are feasible solutions for the project. However, the cameras have some differences, which possibly makes one camera more compatible with the given problem than the other. A comparison of the two cameras can be seen in Table 4.2.

	 CamBoard pico flexx	 RealSense D415
Size	68×17×7.35 mm	99×20×23 mm
Weight	8 g	72 g
Resolution	224×171 px	1280×720 px
Frame rate	45 fps	90 fps
FOV (H×W)	62×45°	65×40°
Depth range	0.1–4.0 m	0.45–3.0 m
Depth accuracy (at 1 m)	<2%	<1%
RGB	No	Yes
Simulation	ROS support	ROS support

Table 4.2: Comparison between available 3D cameras.

The major trade-off between the two cameras is that the Intel RealSense D415 camera is physically larger and heavier but consequently offers more capabilities, such as the inclusion of RGB data and higher depth accuracy and resolution. If the solution were to be integrated in real life, the camera placement might play an essential role in the performance. Whereas, the smaller camera might be better for an in-the-hand prosthesis design, the Intel RealSense camera is more likely to produce useful results, which leads to it being the preferred choice in this project. This also enables possible RGB data utilisation in the point cloud or in case a CNN is to be incorporated in the pipeline.

4.2 Simulation

To train the RL model that should be used for the prosthetic control, a simulation environment is an essential tool due to the flexibility and speed compared to real-life training. This environment should enable training of a grasping task on different objects, which entails the incorporation of the following elements, as seen in Figure 4.3.

- **Approach trajectory:** To emulate a typical grasping task, the prosthetic hand should approach the object based on some user-realistic trajectory, which includes start-point, end-point, shape, and speed of the trajectory. To make the RL model generalisable, these properties must all be modified across different episodes. A considerable challenge setting especially the end-point of the trajectory is that it must ensure that a grasp is feasible, since otherwise the agent may receive a negative reward for a failing a task that was not possible in the first place.
- **Reinforcement learning:** While the approach trajectory only moves the absolute position of the prosthesis, the RL algorithm is responsible for selecting the finger actions that constitute the hand behavior under certain observations from the 3D camera and the imagined point cloud of the fingers. These actions are selected based on a trial-and-error where good actions are positively reinforced and bad actions are negatively reinforced based on an intricately devised reward function. The desired behavior is the grasping of the object when the hand reaches the end-point.
- **Object selection:** To ensure a generalisable model, the grasped object must be altered to accommodate different properties, such as shape, mass, surface, and centre of gravity. The object can either be spawned into the world when needed, or all the objects can be spawned in at the same time and simply moved to the desired position when needed. The generalisability can be show by training on specific categories of objects, such as bottles, and subsequently test on a different category, such as glasses.

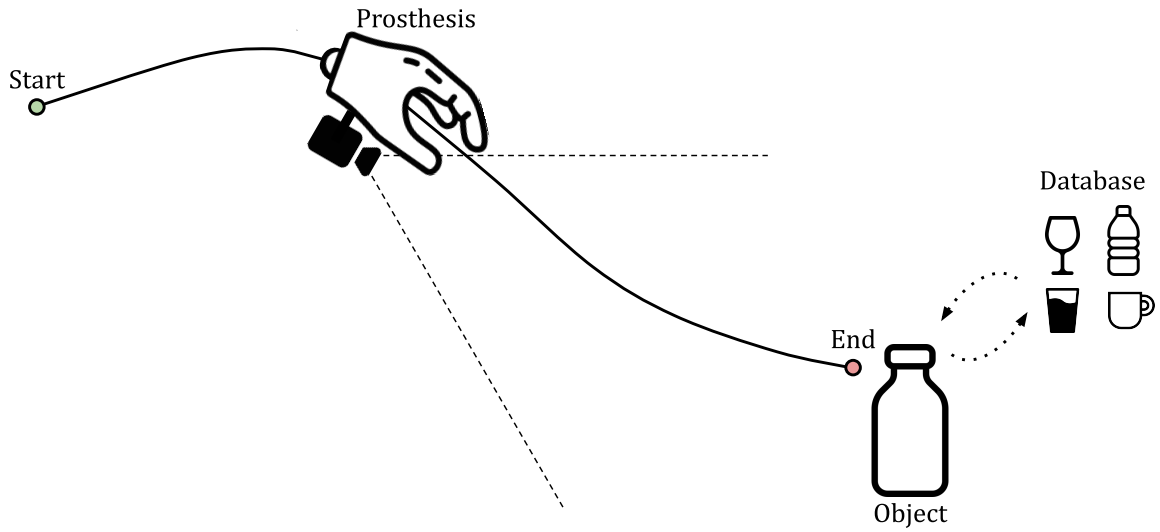


Figure 4.3: Illustration of an RL episode in the simulation environment. The database contains different objects that can be trained on for different episodes.

4.2.1 Simulation Environment

When selecting the simulation environment, several aspects should be taken into account, such as physics, speed, parallelisation, library support, usability, but also the compatibility with the given

problem at hand. Since the problem here includes a free hand with no base, the simulation environment should be able to accommodate this, e.g. by allowing direct positional movement of models. Additionally, the actual hand model, including kinematics, dynamics, and control, should be usable with the given environment, which for the Mia Hand is a URDF file encapsulated in a ROS package. Another important aspect to consider is the ability to simulate grasp interaction physics between the hand and other objects.

Five environments were considered, which were Gazebo, Mujoco, PyBullet, Webots, and Nvidia Isaac Gym. However, due to the high-end hardware requirements and steep learning curve of the Nvidia Isaac platform, this option was eventually disregarded. It can be difficult to compare the remaining options as they each have their own pros and cons, meaning a fair comparison may not be fully achievable. However, some of the general performance characteristic of each one can provide an initial foundation, which can be seen in Table 4.3. [29, 30]

	Gazebo	Mujoco	PyBullet	Webots
Real-time factor	**4.3**	2.4	0.7	1.8
CPU usage	273.0%	**116.3%**	119.3%	124.3%
Max time step	1 ms	3 ms	7 ms	**48 ms**

Table 4.3: Performance comparison of the four considered environments. The data is from the same robot manipulation task on a server in [29].

As seen from the table, Gazebo obtains the highest real-time factor (RTF) in headless mode, meaning it may be able to run faster though at the expense of lower precision and higher CPU usage. Similarly, the maximum time step where the simulation physics are stable is capped at 1 ms, which in turn alleviates the precision problem. An additional advantage of Gazebo is that it is highly integrated with ROS which means that the control interface for the simulation can quickly be adapted to a real-life hardware interface as introduced in Section 4.1. These considerations, along with familiarity with the software, has led to Gazebo being the simulation environment that is opted for in the RL task of this project.

5 Modelling

With the introduced solution proposal in mind, there are several elements that need to be modelled theoretically before being implemented in the actual simulation framework. This entails 1) the generation of the paths that the hand must move along in a given episode, 2) any point cloud processing that must take place in order to obtain the observation, both for actual camera data and the imagined point cloud, and 3) the deep RL pipeline along with the selected algorithm to be used.

5.1 Trajectory Generation

The approach path when grasping some object provides essential details towards the task in which the object is used in. This means that an object is grasped differently depending on how the hand approaches it. Additionally, human trajectories depend on the pose of the object relative to the human along with possible obstacles interfering in the trajectory. Therefore, the generation of realistic and diverse paths is an essential step if the RL model should be applicable in real life. The first step is to define a path planner from one pose to another, which includes both the time-varying position and orientation of the hand, and the second step is to deduce the start and goal poses of the trajectory. In reality, human motion intention is not easily modelled, especially with regards to a wide variety of object manipulation tasks where many motions are purely subconscious. Therefore, a path could beneficially be generated from recorded human motions over many trials, however, this is beyond the scope of this project. Instead, the paths are modelled using well-known methods from literature. In order to simplify trajectory generation, the process is split into a step for position generation and a step for orientation generation, thereby decoupling the two.

5.1.1 Position Generators

Position generation is a widely researched topic with a plethora of different options many of which pertain to the field of path planning where obstacles must be avoided as well. These various options are narrowed to only include continuous environment representations with the main ones considered being the following. All of these assume a known start and goal position.

- **Dynamic Window Approach:** This approach is a simple local or global planner that takes the kinematic constraints of the robot into account by searching the feasible velocity-space of linear and angular velocities. This means that the planner looks at an obstacle free dynamic window around the current linear and angular velocities that can be reached, and an objective function is then used to select the best combination based on heading, velocity, and distance to the goal. [31]
- **Navigation Function:** This approach is a type of artificial potential field method, where a mathematic function is used to introduce an attractive force towards the goal and a repulsive force away from obstacles. The motion is calculated using gradient descent leading the robot to the goal, and a special property of the navigation function is that there are no local minima. [31]
- **Rapidly Exploring Random Trees:** This approach is an online graph search method, where the search expands from the start configuration along randomised branches until reaching the goal. This entails that the path is not initially smooth, but this can be obtained by some post-processing. [31]
- **Dynamic Movement Primitives:** This approach is typically used to make robots emulate some recorded motion by introducing a forcing term in a regular PD controller. Alternatively, it can also be used exclusively as a PD controller to follow a reference trajectory, although this will

essentially lead to straight line paths between start and goal configuration if done in Cartesian space. [32]

- **Bezier Curves:** This approach is a very simple way of inducing curvatures in a path between two configurations with the use of parametric curves. It essentially introduces some control points which become part of a smooth trajectory to the goal by utilising a relatively simple mathematical function. [33]

The first three methods are complete path planners that take obstacles in the environment into consideration and attempt to seek the shortest path. This means that if they were to be used for this purpose, an artificial environment with obstacles would need to be included to make the paths more varied although this would come at the expense of computation. Furthermore, if using the dynamic movement primitives merely as a PD controller, it would also aim to achieve the shortest path based on some objective function. The only position generator of these examples that ignores obstacles and can easily be modified for a wide variety of different, continuous paths is the use of Bezier curves. This variety can be achieved by altering the amount and placement of the control points as seen in Figure 5.1. Additionally, its simplicity entails a very low computation time, even for a large number of control points.

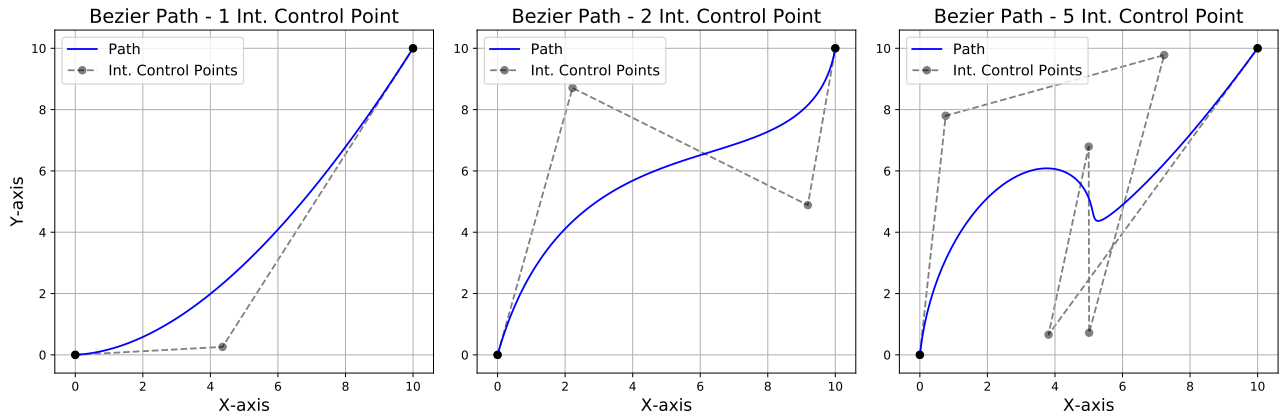


Figure 5.1: Examples of Bezier paths in 2D with different amounts of intermediate control points.

The general mathematical definition of a time-parameterised Bezier curve of order n and with control points, $[C_0, \ldots, C_n]$, is defined in Equation (5.1).

$$p(t) = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} C_i \tag{5.1}$$

where t is the time-variable and $\binom{\cdot}{\cdot}$ is the binomial operator. This means that the Bezier curves from Figure 5.1 are of order 2, 3, and 6, respectively, i.e. the order is one more than the the number of intermediate control points.

5.1.2 Orientation Generators

As for the case of position generation, an approach to compute the hand's orientation along a given trajectory is required. While the two steps are similar in several ways, orientation generation differs in that a trajectory of positions has been predetermined. In addition, the problem at hand imposes the constraint that the object must be within the camera's FOV throughout the whole trajectory, effectively reducing the space of applicable orientations. Orientation generators have been proposed in the following, and in common for them all is that they assume that the start and goal orientation is given.

- **Dynamic Movement Primitives:** Similar to the position generation, this approach can be used to either emulate a prerecorded orientation trajectory or reach a goal orientation with a non-linear trajectory in time by applying it as a PD-controller. By parameter tuning, both the time envelope, reconstruction accuracy, and transient response can be altered. A drawback of this approach is that the PD-controller gains must be tuned for differing distances between start and goal orientation if the same setting time is to be used. [32]
- **Linear interpolation:** This approach would derive the orientation along the trajectory by simple linear interpolation between the start and goal orientation. It offers the advantage of low computational overhead when compared to the other methods, along with a naturally fluid motion between the start and goal orientation.
- **Tangential orientation:** In this approach, the hand's orientation is based on the tangent of the positional trajectory. However, since the tangent cannot fully define a three-dimensional orientation, a way of determining the rotation around this tangential axis is required. Possible solutions could be to base the rotation on randomness, or linearly interpolate a rotation from the given start and goal orientation.
- **Facing target:** This approach is based on having the hand always be oriented towards the target object, defined by the line connecting the origin of the hand and the target object. Similarly to the tangential orientation, this constraint is not sufficient to fully define an orientation, and therefore the same solutions can be applied to fully constraint the orientation.

As emphasised above, the approach using dynamic movement primitive has the drawbacks of unrealistic non-linear shortest path to the target, due to the nature of the controller, and tuning for different path lengths. The three last controllers are simple and have low computational overhead, but linear interpolation stands out in particular as it requires less steps as a result of constraints being fully satisfied exclusively from interpolation. Since low computational overhead is desired, the linear interpolation approach is deemed suitable for generating the orientation.

5.1.3 Start Pose Sampling

For the start pose of the trajectory, the most desired characteristic is to approach the object from different angles so that the RL model learns to grasp the same object regardless of the initial approach. To do this, a semi-dome around the object with a given radius specifying the distance to the object is randomly sampled. This sampling is achieved by creating a random 3-dimensional unit vector and taking the absolute value of the z-component. Additionally, the vector is recomputed if the z-value is too low to the floor as this would entail a collision-state of the hand. An example of a start pose can be seen in Figure 5.2.

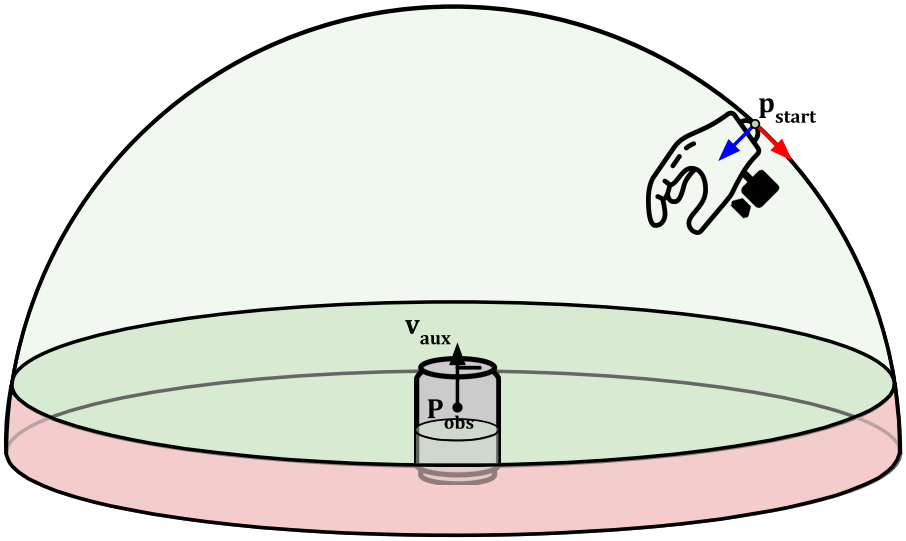


Figure 5.2: Visualisation of start pose sampling, where the green area indicates valid start poses, and the red area indicates a resampling due to too low height.

Given the 3D point, an initial orientation must be determined which can be practically defined by a rotation matrix. The desired characteristic is that the hand faces the object with the fingers as seen in the figure. To fully define a rotation matrix, two axes need to be found from which the third can be deduced using the right hand rule. The first axis, here denoted as the z-axis, can easily be determined as the normalised vector between the start position and the object centre. The x-axis is desired to point downwards, which is achieved by first creating the y-axis based on the cross product of an auxiliary vector with only a non-zero z-component and the z-axis. This is due to these two vectors describing a plane which the desired x-axis would lie in. Subsequently, the x-axis is obtained by following the right hand rule by utilising the cross product between the y-axis and z-axis. The calculations for all axes that can be readily combined into a rotation matrix can be seen in Equation (5.2) with reference to Figure 5.2.

$$\mathbf{z} = -\left(\mathbf{p_{start}} - \mathbf{p_{obs}}\right), \qquad \mathbf{y} = \mathbf{v_{aux}} \times \mathbf{z}, \qquad \mathbf{x} = \mathbf{y} \times \mathbf{z} \tag{5.2}$$

where $\mathbf{p_{start}}$ and $\mathbf{p_{obs}}$ are the start and object positions, respectively, and $\mathbf{v_{aux}} = [0,0,1]^T$ is the auxiliary vector. Notice that the order of the cross products are important for determining the correct orientation along with normalising the individual vectors afterwards. With the start pose and trajectory planner in place, only the goal pose remains unknown.

5.1.4 Goal Pose Sampling

Generally, special care must be taken when sampling the goal pose as opposed to the starting pose. The reason for this is that one must ensure that at least a single feasible grasp of the target object exists at this pose. Failure to do so would create a mismatch in the reward given to the RL model, which can compromise the performance of the algorithm depending on how frequent these poses with infeasible grasps occur. However, because the Mia hand has been augmented with three additional DOF, as was explained in Section **??**, the constraints imposed on goal pose sampling are significantly reduced due to the larger configuration space. As such, the goal position simply needs to be at an appropriate distance to the object, and then one can reasonably assume that a free hand configuration exists that produces an encompassing grasp.

Based on the assumption presented above, a goal sampling algorithm has been proposed that relies on the STL mesh of the object and the line connecting the hand and the centre of the object, as seen

in Figure 5.3a. The algorithm comprises of goal position sampling and goal orientation sampling, where the latter part relies on the former. For goal position sampling, the first step is to determine the intersection between the mesh and the connecting line. With a parametric expression for the connecting line, which consists of a point on the line and a vector parallel to it, and the normal vector of each triangle that constitutes the mesh, the intersection point between the plane coincident with a mesh triangle and the line can be determined by Equation (5.3).

$$t = -\left(\frac{n_t \bullet (p_l - v_0)}{n_t \bullet v_l}\right)$$
$$p_i = p_l + t\, v_l \tag{5.3}$$

where n_t is the normal vector of the triangle, p_l and v_l are the point and parallel vector of the connecting line, respectively, v_0 is a point on the triangle, and p_i is the intersection point.

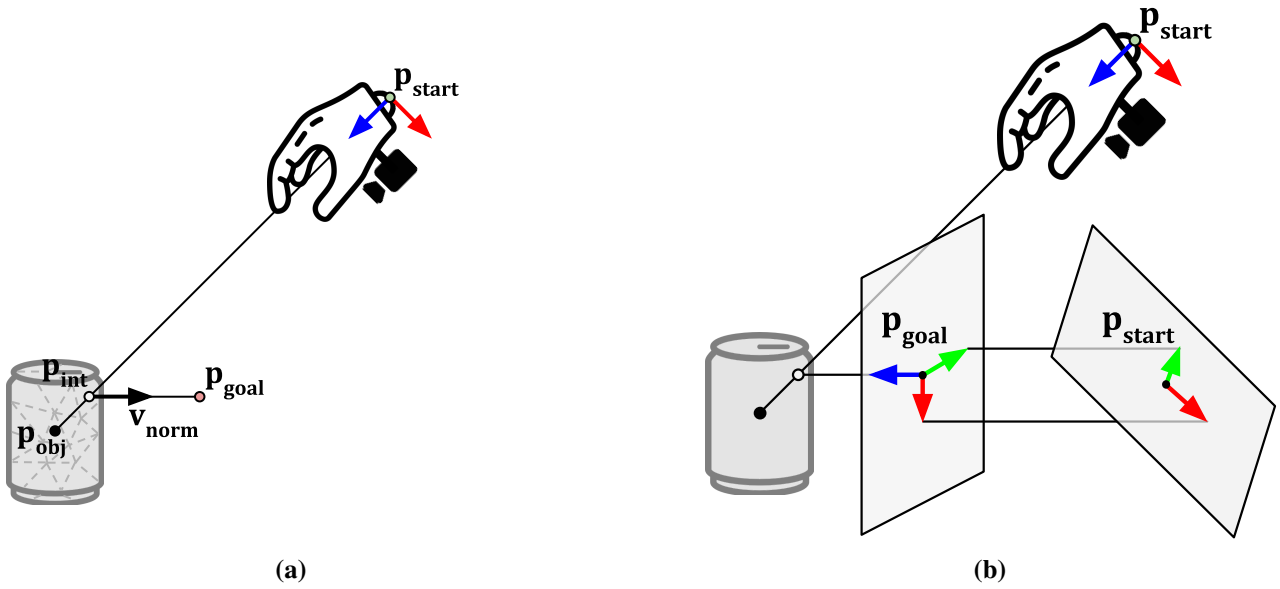


Figure 5.3: Visualisation of goal pose sampling.

After computing the plane intersection for all triangles, the candidates are filtered to only keep the ones that lie within the respective triangle such that the correct triangle normal can be extracted. Since multiple intersections might exist such as for objects with walls of non-zero width, the intersection point closest to the start pose is chosen. It should be noted that depending on the object, some connecting lines will not be intersecting any triangles if the mesh does not fully envelop the object's origin from all angles. An example of this could be mugs, where an unobstructed connecting line can be created at the mug's opening. Instead of resampling the start position, the intersection is taken to be the point in the mesh closest to the start position, and its normal vector will be parallel to the connecting line. After having determined the intersection point and normal vector, the goal position is derived by adding a predefined offset to the intersection point in the direction of the normal, as shown in Figure 5.3a.

For the goal orientation, an orientation is sought that resembles the start orientation to some degree. Assuming that the z-axis of the prosthesis goal frame is constrained to point towards the object along the intersection normal, a straightforward derivation of the remaining goal orientation axes could entail minimising the difference between these two orientations. Such a minimisation is arguably also one of the objectives in real human motion when considering task effort minimisation. To implement this, a simple approximate minimisation can be applied by orthogonally projecting the start orientation's x-

and y-axis onto the plane orthogonal to the goal frame's z-axis, as seen in Figure 5.3b. This projection is determined by Equation (5.4).

$$v_{pj} = v - \frac{v \bullet n}{n \bullet n} n \tag{5.4}$$

where v is the vector to be projected, n is a normal vector to the plane to be projected on, and v_{pj} is the resulting projected vector. Because the projection most likely will not result in a valid orientation, the projected axis with largest norm is chosen under the assumption that it approximates the starting orientation to a larger degree than the other. After normalising this new axis, the last remaining axis is determined as the cross-product of the two former.

5.2 Point Cloud Processing

The basis of the RL algorithm is the observations that it obtains along the trajectory which in this case primarily is point cloud data. To ensure desired behaviour, this point cloud data should be processed to include object in question and the fingers of the hand such that the model is able to draw parallels between the movement of the fingers and the manipulation of the object. As such, the point cloud from the RealSense camera and from the kinematic imagination of the hand must be extracted.

5.2.1 Depth Camera Processing

The RealSense camera uses stereo vision in which the baseline between the two built-in cameras is used to obtain the disparities in their entire overlapping FOV. These disparities are computed by comparing where the same physical point appears in the two rectified images. Knowing each disparity, $d = x_{left} - x_{right}$, can be converted to depth, Z, using simple geometry in Equation (5.5).

$$Z = \frac{f \cdot b}{d} \tag{5.5}$$

where f is the focal length of the cameras and b is the baseline between them. This data can be combined with the RGB data also recorded by the camera by performing image registration between the RGB and depth images to align them correctly, however, this likely does not provide much additional information in the RL algorithm. To emulate the real world, the camera can be included with a predefined noise model in Gazebo, typically Gaussian, which can be varied to test the robustness of the RL algorithm. Additionally, the clip plane of the depth data can be set to evaluate the impact of range capabilities along with disregarding spurious data that falls outside the range. [34]

To convert the depth data to a point cloud, the corresponding x- and y-positions to the depth must be determined as described in Equation (5.6).

$$X = \frac{Z(u - c_x)}{f}, \qquad Y = \frac{Z(v - c_y)}{f} \tag{5.6}$$

where (u, v) are the pixel coordinates, and (c_x, c_y) is the camera centre in pixels. As such, a sequence of n points corresponding to the number of pixels with valid depths can be obtained as $\mathbf{PC^{cam}} = [\mathbf{p_1}, \ldots, \mathbf{p_n}]^T \in \mathbb{R}^{n \times 3}$ with $\mathbf{p_i} = [X_i, Y_i, Z_i]^T$.

5.2.2 Imagined Hand Point Cloud

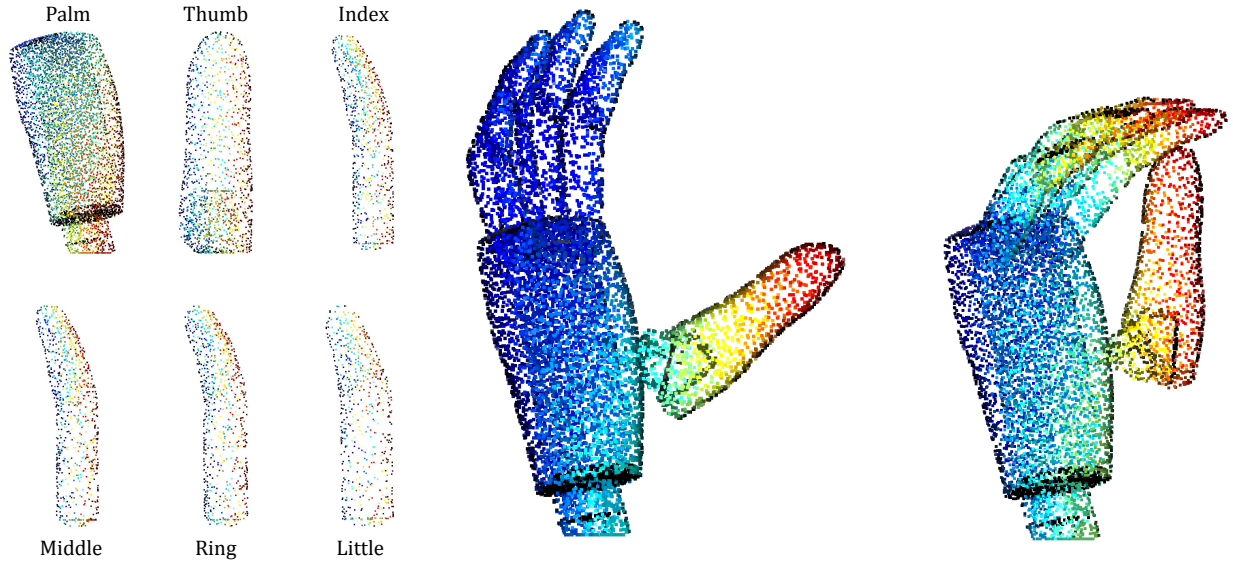
To mitigate issues of occlusion in the camera depth data, an additional point cloud is artificially generated based on the configuration of the prosthetic hand as explored in DexPoint [26]. More precisely, the point cloud is obtained by sampling the STL meshes that make up the hand while taking into account their relative transformations described by the hand kinematics. This sampling can be relatively simply done by selecting a triangle in the mesh at random and subsequently generating a point, $\mathbf{p_{rand}}$, within that triangle with the use of Barycentric coordinates and two uniformly distributed random numbers, $r_1, r_2 \sim \mathcal{U}(0,1)$, as seen in Equation (5.7). [35]

$$\begin{aligned} \mathbf{p_{rand}} &= a\,\mathbf{p_1} + b\,\mathbf{p_2} + c\,\mathbf{p_3} \\ a = 1 - \sqrt{r_1}, \qquad b &= \sqrt{r_1}\,(1 - r_2), \qquad c = \sqrt{r_1}\,r_2 \end{aligned} \tag{5.7}$$

where $\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}$ are the three vertices of the triangle. The point, $\mathbf{p_{rand}}$, is guaranteed to lie within the triangle since it follows the Barycentric constraints $a, b, c >= 0$ and $a + b + c = 1$. Repeating this n times, one can obtain a point cloud that resembles the given mesh. Further, to obtain the full hand, this must be done for each individual mesh such that the total number of desired points is $N = n_1, \ldots, n_m$ with m being the number of meshes. To account for the different sizes of the meshes, the amount of points n_i that is sampled for a given mesh should correspond to the proportion of the total surface area that it contributes to in the full hand as per Equation (5.8).

$$n_i = N\frac{A_i}{A}, \qquad i = 1, \ldots, m \tag{5.8}$$

where A_i is the surface area of mesh i and A is the total surface area. The effect of this evenly distributed sampling can be seen in Figure 5.4, where both the individual and combined sampled meshes are visualised.



(a) Individual hand point cloud groups. **(b)** Open hand point cloud. **(c)** Closed hand point cloud.

Figure 5.4: The sampled point cloud from the meshes of the fingers. The individual fingers can be transformed in their local frame and subsequently combined to modify the shape of the resulting hand point cloud.

The individual sampled point clouds from Figure 5.4a need to be correctly transformed to combine them into the full hand. These transformations along with corresponding scalings are included in the URDF[1] file of the hand, which can be utilised to form the hand in Figure 5.4b. However, since

[1] Unified Robot Description Format (URDF) is an XML format for representing robots in simulation.

these individual components are sampled in a fixed configuration, the resulting combined point cloud will consequently be static, which is said open hand configuration. To enable tracking of the hand configuration as the fingers physically move, a rotation about the z-axis of each finger joint must be applied to the individual point clouds before combining them as described in Equation (5.9).

$$\mathbf{PC}_{\mathbf{new}}^{\mathbf{imag}} = \mathbf{PC}_{\mathbf{old}}^{\mathbf{imag}} RotZ(\theta) \tag{5.9}$$

where $\mathbf{PC}_{\mathbf{new}}^{\mathbf{imag}} \in \mathbb{R}^{n\times 3}$ and $\mathbf{PC}_{\mathbf{old}}^{\mathbf{imag}} \in \mathbb{R}^{n\times 3}$ are the new and old imagined point clouds, respectively, and $RotZ(\theta) \in SO(3) \subset \mathbb{R}^{3\times 3}$ is a 3D rotation about the z-axis by θ degrees. This results in a new hand configuration as exemplified in Figure 5.4c, where each joint has been rotated 45° to close the hand. Since the joint positions at any given time are described with respect to the original open hand configuration (zero-configuration), the corresponding old point cloud is saved indefinitely to accommodate new angle rotations at each given step.

5.2.3 Post-Processing Steps

As shown in Table 4.1, the resolution of the RealSense camera is 1280×720 px, which corresponds to a total of 921,600 pixels[2]. This is evidently infeasible as an input to the RL algorithm, meaning the input size should be downsampled.

5.3 Deep Reinforcement Learning

Before diving into deep RL, understanding RL in general is a prerequisite, which is typically described by Markov Decision Processes (MDPs). MDPs are a way of describing the interaction process that occurs between and agent and its environment. Given an agent state S_t and a reward R_t, the agent takes an action A_t that modifies the given environment. As such, the environment undergoes a transition to a new state S_{t+1} which in turn results in a new reward R_{t+1} based on how desirable this transition was. This is described by the probabilistic transition model $p(s'|s,a)$, which gives the probability distribution over the next state s' given the state-action pair (s,a). The final goal in this alternating interaction between agent and environment is to find the optimal policy, $\pi(a,s)$, that maximises the expected reward. The rewards are given based on a function that maps an environment transition into a real non-negative number, $r(s,a,s')$, and the selection of this function is described in Section 5.3.2. To realise this maximisation of expected rewards, the state value function, $V(s)$, must be introduced, which essentially quantifies the desire to be in a given state formulated by the sum of future discounted rewards from said state. The state-action value function, otherwise known as the Q-function, $Q(s,a)$, further parametrises over the immediately next action which proves useful in RL. In essence, the problem of RL is to find the state-action value function or simply an approximation of it. [36]

Until this point, it is assumed that the environment state is known at any given time, which is typically not the case in robotics. As such, an additional variable known as the observation O_t must be included when the agent makes a decision. This is known as a Partially Observable Markov Decision Process (POMDP) which can be seen in Figure 5.5. The variables pertaining to the POMDP are each restricted to a mathematical space which are the state space, $S \in \mathcal{S} \subseteq \mathbb{R}^n$, action space, $A \in \mathcal{A}(\mathcal{S}) \subseteq \mathbb{R}^m$, observation space, $O \in \mathcal{O} \subseteq \mathbb{R}^p$, and reward space, $R \in \mathcal{R} \subseteq \mathbb{R}_{\geq 0}$. In this project, the state includes the hand joint positions and velocities, the observation comprises of the point cloud data from Section 5.2, and the action space consists of the individual joint velocities of the controlled fingers.

[2] For the depth image, this would be lower due to the non-overlapping regions of the cameras.

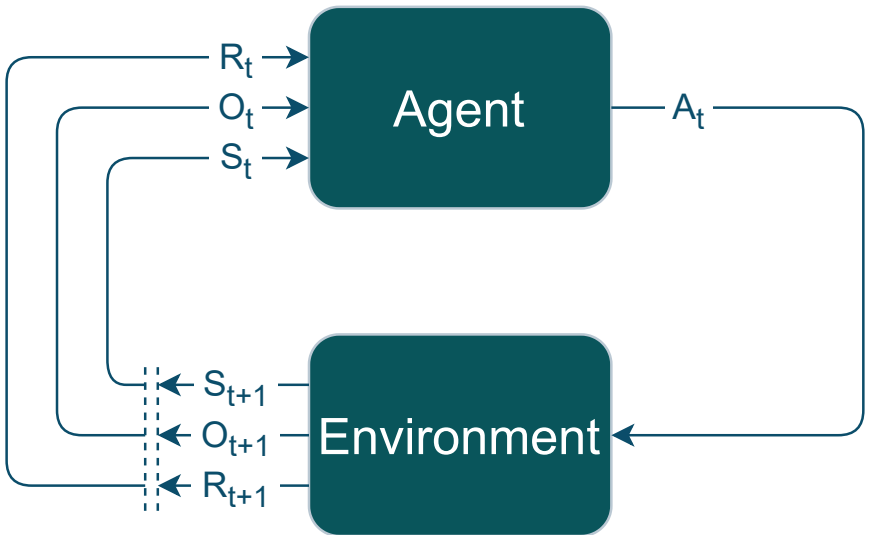


Figure 5.5: Illustration of the the POMDP framework. Here, *S* is the state, *O* is the observation, and *R* is the reward. Adapted from [36].

Although the above explains the general idea behind RL, it has not yet introduced the extension to deep RL. As the name suggests, deep RL is the combination of reinforcement learning with deep learning, which has proven to be highly useful for high-dimensional state and observation spaces, making it ideal for robotics scenarios. This is done by utilising the exceptional capability of deep neural networks as function approximators, where the notion of deep neural networks means that it consists of an input layer, output layer, and at least two hidden layers along with non-linear activations between successive layers. This can be applied either to estimate the value function or directly to determine the optimal policy. The method to approach this optimal policy depends on the specific deep RL algorithm that is employed. [37]

5.3.1 Algorithms for Deep Reinforcement Learning

Since there are unfathomably many deep RL implementations that differ substantially or slightly, only some of the most popular algorithms considered state-of-the-art within robotics are introduced here. Only the specific branch of model-free algorithms are included since robotics deals with transition models with noise. These algorithms each have their own pros and cons, and the specific context in which they are used has a substantial impact on their applicability. A description of these are given in the following and the division is shown in Figure 5.6. The policy-based methods typically use on-policy learning, meaning the agent updates the same model that generates the actions, whereas the value-based methods typically use off-policy learning, meaning the agent chooses actions from separate network than the one it uses to update the model.

- **Deep Q-Network (DQN):** In DQN, a deep neural network is used to approximate the Q-function which is applied in connection with a modified version of Q-learning. These modifications are used to counteract the instability that is typically introduced in the learning process when using the neural networks in this way. Firstly, it utilises experience replay which randomises over the previous data to minimise correlations between subsequent observations as these tend to be very similar. Secondly, it only periodically updates the target policy over a batch of transitions to further reduce correlations with the target. [38]
- **Proximal-Policy Optimisation (PPO):** PPO is policy-gradient method, meaning it directly attempts to learn the policy using stochastic gradient descent. This has the benefit that the state, action, and observation spaces need not be discrete, as they would to in value-based

methods where the Q-function is approximated. PPO is a data efficient and robust method by introducing two features. Firstly, it uses clipped trust regions with pessimistic estimates of the objective function to avoid large steps in the policy gradient that result in non-converging behavior. Secondly, it alternates between policy sampling and objective optimisation in batches to reduce data correlation similar to DQN. The PPO algorithm is usually integrated with an actor-critic structure. [39]

- **Soft Actor Critic (SAC):** SAC is based on the actor-critic framework in RL where the actor chooses the actions that the agent should take and the critic evaluates how good these actions are. SAC differs from original strategies of simply maximising the expected reward by additionally maximising the entropy and thereby trying to circumvent the issues of purely on-policy and purely off-policy methods. As such, it attempts to succeed the given task while acting as randomly as possible to ensure sufficient exploration of the decision space and robustness in learning. This algorithm also allows continuous state, action, and observation spaces like PPO. [40]

- **Hindsight Experience Replay (HER):** HER is method that deals with the recurring problem in RL of sample efficiency of sparse rewards thereby relaxing the necessity of careful reward design. The general idea is to include the concept of sub-goals to the model such that it can still learn from partially achieving a larger goal. This is achieved by adding a goal state as an input to the model and replaying each episode with the sub-goals that was reached as this input. Although HER is not in itself a deep RL algorithm, it can be readily combined with any off-policy algorithm such as DQN or SAC. [41]

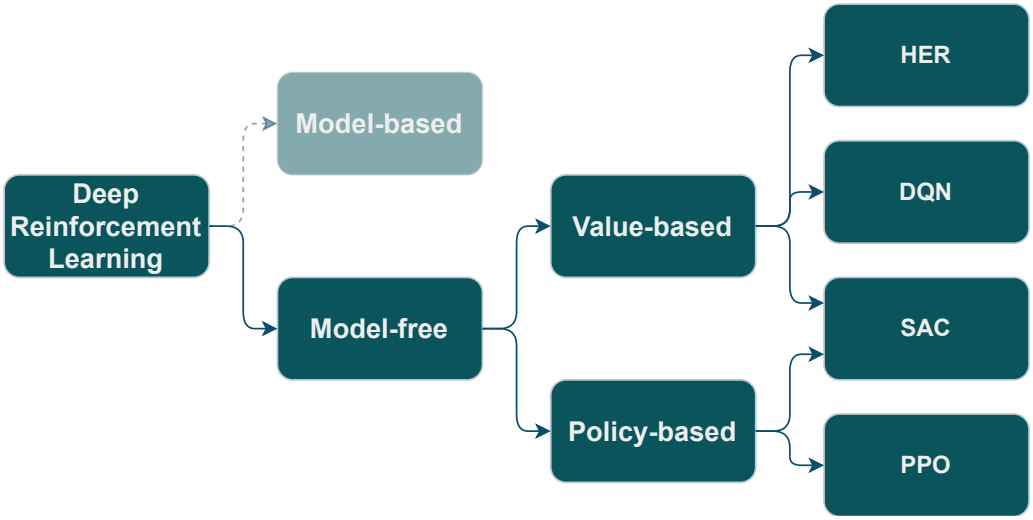


Figure 5.6: Categorisation of the selected deep RL algorithm.

With these algorithms briefly explained, the different advantages and disadvantages can be highlighted, which can be seen in Table 5.1. Evidently, the algorithms each offer their own pros and cons, and it can be difficult in general to select one optimal algorithm for a given purpose. However, for this project, the PPO algorithm is deemed the most suitable based on it being suitable for continuous environments, having low complexity, and it being the preferred choice in DexPoint [26].

	Advantages	Disadvantages
DQN	High-dimensional state space Learns directly from sensory data	Difficult to use continuous action spaces Overestimates Q-values
PPO	Handles continuous action spaces Optimises clipped trust region objectives Low complexity	Limited exploration Sensitive to hyperparameters
SAC	Handles continuous action spaces High-dimensional state spaces Optimises entropy-regularised objectives	Computationally expensive Sensitive to hyperparameters
HER	Handles sparse rewards Simplifies reward design using sub-goals	Only compatible with off-policy methods Increased complexity

Table 5.1: Overview of the pros and cons of the explained deep RL algorithms. Adapted from [42].

5.3.2 Reward Shaping

5.3.3 Hyperparameter Tuning

Hyperparameter tuning is one of the most difficult yet crucial parts of whether the RL algorithm learns something useful or not. For clarification, hyperparameters are all the free parameters of the RL algorithm that can be varied to modify the learning process. Typically, hyperparameters depend on each other, entailing that in order to ensure an optimal configuration, the entire space of allowed hyperparameters must be explored. However, this is almost never feasible, entailing that only a subset can be explored. This can be done by hand-tuning with grid search methods to discretise over the hyperparameter space. Unfortunately, this can be time-consuming and does not ensure good performance, thereby opening an opportunity for alternative methods for hyperparameter selection, i.e. hyperparameter optimisation (HPO). This has in general shown to improve the performance as tested in [43], though at the expense of introducing an additional layer of complexity. Apart from the dependence on each other, the specific hyperparameter configuration performance also depends significantly on the random seed, which means that overfitting can easily occur.

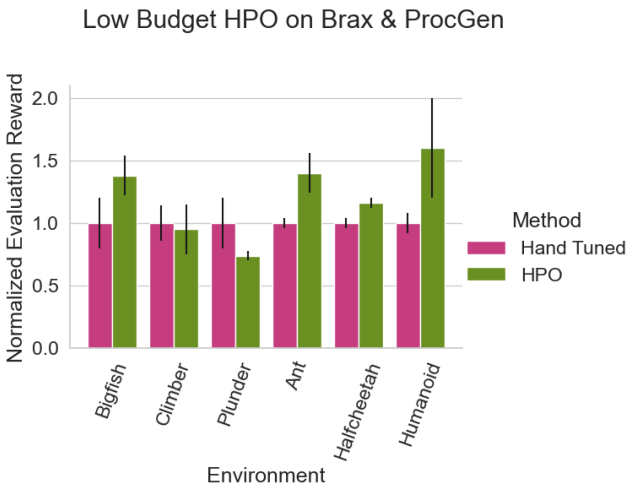


Figure 5.7: Comparison in performance between hand-tuning and HPO for different environments. Taken from [43].

[43, 44]

5.3.4 Open Source Libraries

6 Implementation

With a deeper understanding of the different elements required of the solution, an actual implementation of the RL hand grasping pipeline will here be established. The implementation is split into two parts, where each part will respectively detail the setup of the environment and the training of the RL model. However, firstly a system overview is included to make the structure of the system clear to the reader.

6.1 System Overview

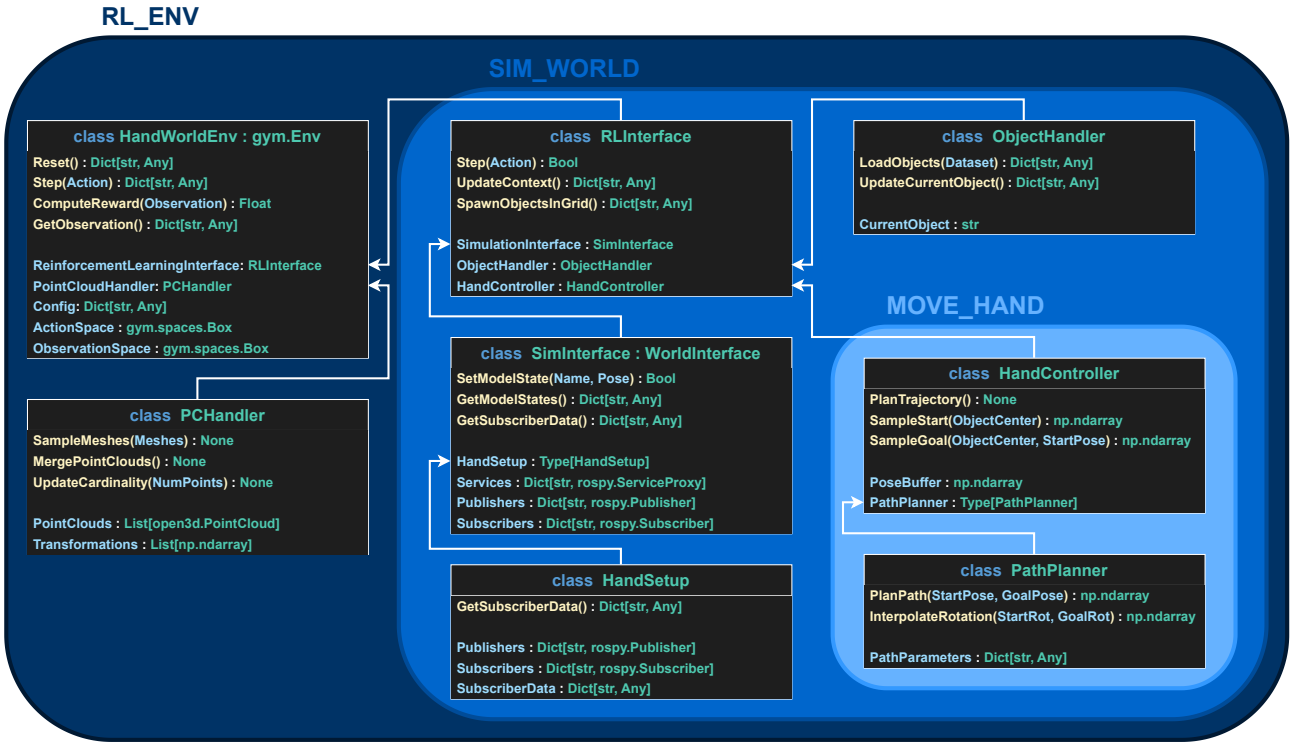


Figure 6.1: Class diagram of the system structure.

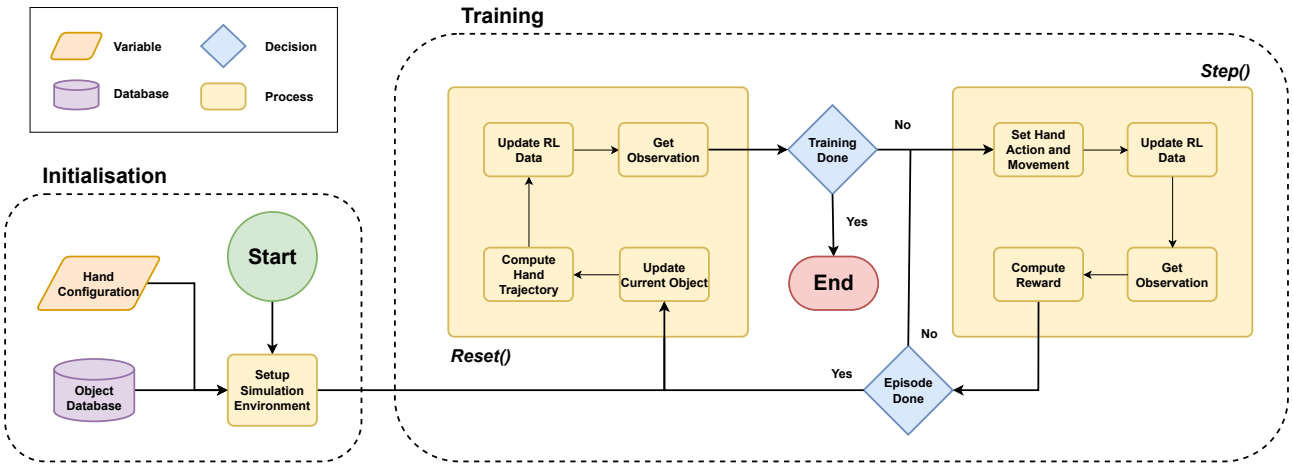


Figure 6.2: Flowchart of the system structure.

6.2 Environment Setup

6.3 Model Training

7 Testing and Results

8 Discussion

9 Conclusion

Bibliography

[1] Aaron M. Dollar. "Classifying Human Hand Use and the Activities of Daily Living". In: *The Human Hand as an Inspiration for Robot Hand Development*. Ed. by Ravi Balasubramanian and Veronica J. Santos. Cham: Springer International Publishing, 2014, pp. 201–216. ISBN: 978-3-319-03017-3. DOI: 10.1007/978-3-319-03017-3_10.

[2] Kayla Matheus and Aaron M. Dollar. "Benchmarking grasping and manipulation: Properties of the Objects of Daily Living". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010, pp. 5020–5027. DOI: 10.1109/IROS.2010.5649517.

[3] Wei Zheng, Ning Guo, Baohua Zhang, Jun Zhou, Guangzhao Tian, and Yingjun Xiong. "Human Grasp Mechanism Understanding, Human-Inspired Grasp Control and Robotic Grasping Planning for Agricultural Robots". In: *Sensors* 22.14 (2022). ISSN: 1424-8220. DOI: 10.3390/s22145240.

[4] Dale Purves, David Fitzpatrick George J Augustine, Lawrence C Katz, Anthony-Samuel LaMantia, James O McNamara, and S Mark Williams. "Neuroscience, 2nd edition". In: Sinauer Associates, 2001. Chap. 8.

[5] J. R. and Napier. "The Prehensile Movements of the Human Hand". In: *The Journal of Bone and Joint Surgery* (Nov. 1956). DOI: 10.1302/0301-620X.38B4.902.

[6] Brett Douville, Libby Levison, and Norman I. Badler. "Task-Level Object Grasping for Simulated Agents". In: *Presence: Teleoperators and Virtual Environments* 5.4 (Nov. 1996), pp. 416–430. DOI: 10.1162/pres.1996.5.4.416.

[7] T. Iberall, G. Bingham, and M. A. Arbib. "Opposition Space as a Structuring Concept for the Analysis of Skilled Hand Movements". In: *Generation and Modulation of Action Patterns*. Ed. by H. Heuer and C. Fromm. Springer-Verlag Berlin Heidelberg, 1986, pp. 158–174. ISBN: 3-540-16941-5.

[8] Ivan Vujaklija, Dario Farina, and Oskar C. Aszmann. "New developments in prosthetic arm systems". In: *Orthopedic Research and Reviews* 8 (2016), pp. 31–39.

[9] Brian Lee, Frank J. Attenello, Charles Y. Liu, Michael P. McLoughlin, and Michael L.J. Apuzzo. "Recapitulating Flesh with Silicon and Steel: Advancements in Upper Extremity Robotic Prosthetics". In: *World Neurosurgery* 81.5 (2014), pp. 730–741. ISSN: 1878-8750. DOI: https://doi.org/10.1016/j.wneu.2014.03.012.

[10] Weichao Guo, Wei Xu, Yanchao Zhao, Xu Shi, Xinjun Sheng, and Xiangyang Zhu. "Toward Human-in-the-Loop Shared Control for Upper-Limb Prostheses: A Systematic Analysis of State-of-the-Art Technologies". eng. In: *IEEE transactions on medical robotics and bionics* 5.3 (2023), pp. 563–579. ISSN: 2576-3202.

[11] IndiaMart. *Below Elbow Prosthesis*. Accessed on March 1, 2024.
URL: https://m.indiamart.com/proddetail/below-elbow-prosthesis-21277931233.html.

[12] Britt H. Young. *The Prosthetic Hand Arms Race*. 2022.
URL: https://spectrum.ieee.org/the-prosthetic-hand-arms-race.

[13] Ivan Vujaklija and Dario Farina. "Mechatronic Design of Functional Prosthetic Systems". In: *Bionic Limb Reconstruction*. Ed. by Oskar C. Aszmann and Dario Farina. Cham: Springer International Publishing, 2021, pp. 37–43. ISBN: 978-3-030-60746-3. DOI: 10.1007/978-3-030-60746-3_5.
URL: https://doi.org/10.1007/978-3-030-60746-3_5.

[14] Steeper Group. *Mechanical Hand: Technical Information*. Accessed: 2024-03-01. 2021. URL: https://www.steepergroup.com/media/10502/stppr272-mechanical-hand-technical-information-issue-5.pdf.

[15] Zhen Gang Xiao and Carlo Menon. "A Review of Force Myography Research and Development". In: *Sensors* 19.20 (2019). ISSN: 1424-8220. DOI: 10.3390/s19204557. URL: https://www.mdpi.com/1424-8220/19/20/4557.

[16] Simon Hazubski, Harald Hoppe, and Andreas Otte. "Non-contact visual control of personalized hand prostheses/exoskeletons by tracking using augmented reality glasses". In: *3D Printing in Medicine* 6 (Feb. 2020), p. 6. ISSN: 2365-6271. DOI: 10.1186/s41205-020-00059-4.

[17] Jonathan DeGol, Aadeel Akhtar, Bryan Manja, and Timothy Bretl. "Automatic Grasp Selection using a Camera in a Hand Prosthesis". In: Aug. 2016, pp. 431–434. DOI: 10.1109/EMBC.2016.7590732.

[18] Shrey Srivastava, Amit Vishvas Divekar, Chandu Anilkumar, Ishika Naik, Ved Kulkarni, and V. Pattabiraman. "Comparative analysis of deep learning image detection algorithms". In: *Journal of Big Data* 8.1 (May 2021), p. 66. ISSN: 2196-1115. DOI: 10.1186/s40537-021-00434-w.

[19] Kan Luo, Hongshan Yu, Xieyuanli Chen, Zhengeng Yang, Jingwen Wang, Panfei Cheng, and Ajmal Mian. "3D point cloud-based place recognition: a survey". In: *Artificial Intelligence Review* 57.4 (Mar. 2024), p. 83. ISSN: 1573-7462. DOI: 10.1007/s10462-024-10713-6.

[20] M. Markovic, S. Dosen, D. Popovic, B. Graimann, and D. Farina. "Sensor fusion and computer vision for context-aware control of a multi degree-of-freedom prosthesis". In: *J Neural Eng.* (Dec. 2015). DOI: 10.1088/1741-2560/12/6/066022.

[21] Yunan He, Ryusei Shima, Osamu Fukuda, Nan Bu, Nobuhiko Yamaguchi, and Hiroshi Okumura. "Development of Distributed Control System for Vision-Based Myoelectric Prosthetic Hand". eng. In: *IEEE access* 7 (2019), pp. 54542–54549. ISSN: 2169-3536.

[22] Ankur Handa, Karl Van Wyk, Wei Yang, Jacky Liang, Chao Yu-Wei, Qian Wan, Stan Birchfield, Nathan Ratliff, and Dieter Fox. "DexPilot: Vision Based Teleoperation of Dexterous Robotic Hand-Arm System". eng. In: *arXiv.org* (2019). ISSN: 2331-8422.

[23] Shuo Wang, Jingjing Zheng, Ziwei Huang, Xiaoqin Zhang, Vinicius Prado da Fonseca, Bin Zheng, and Xianta Jiang. "Integrating computer vision to prosthetic hand control with sEMG: Preliminary results in grasp classification". In: *Frontiers in Robotics and AI* 9 (2022). ISSN: 2296-9144. DOI: 10.3389/frobt.2022.948238. URL: https://www.frontiersin.org/articles/10.3389/frobt.2022.948238.

[24] Strahinja Došen, Christian Cipriani, Miloš Kostić, Marco Controzzi, Maria C. Carrozza, and Dejan B. Popovič. "Cognitive vision system for control of dexterous prosthetic hands: Experimental evaluation". eng. In: *Journal of neuroengineering and rehabilitation* 7.1 (2010), pp. 42–42. ISSN: 1743-0003.

[25] Miguel Nobre Castro and Strahinja Dosen. "Continuous Semi-autonomous Prosthesis Control Using a Depth Sensor on the Hand". eng. In: *Frontiers in neurorobotics* 16 (2022), pp. 814973–814973. ISSN: 1662-5218.

[26] Yuzhe Qin, Binghao Huang, Zhao-Heng Yin, Hao Su, and Xiaolong Wang. *DexPoint: Generalizable Point Cloud Reinforcement Learning for Sim-to-Real Dexterous Manipulation*. 2022. arXiv: 2211.09423.

[27] Gautham Vasan and Patrick M. Pilarski. “Learning from demonstration: Teaching a myoelectric prosthesis with an intact limb via reinforcement learning”. eng. In: *2017 International Conference on Rehabilitation Robotics (ICORR)*. Vol. 2017. United States: IEEE, 2017, pp. 1457–1464. ISBN: 1538622963.

[28] Nafiseh Ghaffar Nia, Ahad Nasab, and Erkan Kaplanoglu. “Reinforcement Learning-Based Grasp Pattern Control of Upper Limb Prosthetics in an AI Platform”. eng. In: *2022 3rd International Informatics and Software Engineering Conference (IISEC)*. Piscataway: IEEE, 2022, pp. 1–4. ISBN: 9781665459952.

[29] Marian Körber, Johann Lange, Stephan Rediske, Simon Steinmann, and Roland Glück. *Comparing Popular Simulation Environments in the Scope of Robotics and Reinforcement Learning*. 2021. DOI: 10.48550/arXiv.2103.04616. arXiv: 2103.04616.

[30] Jack Collins, Shelvin Chand, Anthony Vanderkop, and David Howard. “A Review of Physics Simulators for Robotic Applications”. In: *IEEE Access* 9 (2021), pp. 51416–51431. DOI: 10.1109/ACCESS.2021.3068769.

[31] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to Autonomous Mobile Robots*. 2nd edition. MIT Press, 2011. ISBN: 978-0-262-29532-1.

[32] Matteo Saveriano, Fares J. Abu-Dakka, Aljaz Kramberger, and Luka Peternel. “Dynamic Movement Primitives in Robotics: A Tutorial Survey”. In: *CoRR* abs/2102.03861 (2021). DOI: 10.1177/02783649231201196. arXiv: 2102.03861.

[33] Senay Baydas and Bulent Karakas. “Defining a curve as a Bezier curve”. In: *Journal of Taibah University for Science* 13.1 (2019), pp. 522–528. DOI: 10.1080/16583655.2019.1601913.

[34] Yonghuai Liu, Nick Pears, Paul Rosin, and Patrik Huber. “Two-View Geometry”. In: *3D Imaging, Analysis and Applications*. 2nd edition. Jan. 2020, pp. 62–72. ISBN: 978-3-030-44069-5. DOI: 10.1007/978-3-030-44070-1.

[35] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. “Shape Distributions”. In: *ACM Transactions on Graphics* 21.4 (Oct. 2002), pp. 807–832.

[36] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2nd edition. MIT Press, 2018. ISBN: 978-0-262-19398-6.

[37] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. “An Introduction to Deep Reinforcement Learning”. In: *Foundations and Trends® in Machine Learning* 11.3–4 (2018), pp. 219–354. ISSN: 1935-8245. DOI: 10.1561/2200000071.

[38] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. “Playing Atari with Deep Reinforcement Learning”. In: *CoRR* abs/1312.5602 (2013). DOI: 10.48550/arXiv.1312.5602.

[39] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. “Proximal Policy Optimization Algorithms”. In: *CoRR* abs/1707.06347 (2017). DOI: 10.48550/arXiv.1707.06347.

[40] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor”. In: *CoRR* abs/1801.01290 (2018). DOI: 10.48550/arXiv.1801.01290.

[41] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. “Hindsight Experience Replay”. In: *CoRR* abs/1707.01495 (2018). DOI: 10.48550/arXiv.1707.01495.

[42] Dong Han, Beni Mulyana, Vladimir Stankovic, and Samuel Cheng. “A Survey on Deep Reinforcement Learning Algorithms for Robotic Manipulation”. In: *Sensors* 23.7 (2023). ISSN: 1424-8220. DOI: 10.3390/s23073762.

[43] Theresa Eimer, Marius Lindauer, and Roberta Raileanu. “Hyperparameters in Reinforcement Learning and How To Tune Them”. In: *CoRR* abs/1707.01495 (2023). DOI: 10.48550/arXiv.2306.01324.

[44] Hadi S. Jomaa, Josif Grabocka, and Lars Schmidt-Thieme. “Hyp-RL : Hyperparameter Optimization by Reinforcement Learning”. In: *CoRR* abs/1906.11527 (2019). DOI: 10.48550/arXiv.1906.11527.