POLITECNICO DI TORINO

UNIVERSIDAD POLITECNICA DE MADRID

Double Degree Master in
Mechatronic Engineering || Automatica y Robotica

Double Degree Master's Thesis

Ensuring Safety in Upper-Limb Prostheses: Tactile Sensor and Machine Learning for Risk Prediction

Supervisors

Prof. Alessandro RIZZO (PoliTo)

Prof. Manuel FERRE (UPM)

Prof. Cristina PIAZZA (TUM)

Dr. Patricia CAPSI MORALES (TUM)

Candidate

Martina COLUMBARO

April 2024

Abstract

Upper limbs play an important role in everyday life, enabling a variety of activities beyond mere object manipulation or grasping, as allowing communication, productivity, creativity, and physical health through writing, drawing, sports and recreation. It is evident, then, that the loss of upper limb functions deeply impacts individuals' daily lives and quality of life.

Despite extensive research to replicate upper limb capabilities with prosthetic devices, users often face challenges in adapting to their prosthesis, leading to high rejection rates. Addressing this challenge requires prosthetics that not only restore functionality but also offer natural control and autonomy for daily activities.

In particular, prostheses often demonstrate inadequate sensory feedback and limited proprioceptive information. This deficiency in sensory perception leads to poor slip control, difficulties in adjusting grip force, complications with object manipulation, and decreased dexterity. These factors, together with a heavy reliance on visual cues, prevent an easy integration of prosthetic devices into daily routines and challenge users' acceptance.

In line with these goals, this master's thesis seeks to enhance grasping safety by introducing a machine-learning algorithm capable of interpreting sensory information from tactile sensors embedded within the prosthetic hand. The work involves a comprehensive overview of existing non-invasive feedback mechanisms and tactile sensor technologies, alongside an in-depth exploration of experimental methodologies for detecting and predicting slippage.

Afterwards, to build and train the machine learning algorithm, a comprehensive dataset was collected, incorporating various actions categorized into three main groups: grasp, risky and non-risky. These actions involved interacting with objects of different shapes and textures. The data was gathered using a commercially available prosthesis, specifically the Michelangelo hand, equipped with six tactile sensors embedded on its fingers.

Finally, an experimental validation was conducted, involving external participants interacting with the prosthetic hand. This validation served to evaluate the accuracy of the algorithm's predictions and gather feedback for potential enhancements. Through this iterative process of data collection, algorithm development, and experimental validation, this thesis aims to predict slippage to ensure grasping safety.

Resumen

Los miembros superiores juegan un papel importante en la vida cotidiana, permitiendo una variedad de actividades más allá de la mera manipulación u agarre de objetos, al posibilitar la comunicación, productividad, creatividad y salud física a través de la escritura, dibujo, deportes y recreación. Es evidente, entonces, que la pérdida de funciones de los miembros superiores impacta profundamente en la vida diaria y calidad de vida de los individuos.

A pesar de la extensa investigación para replicar las capacidades de los miembros superiores con dispositivos protésicos, los usuarios a menudo enfrentan desafíos en la adaptación a su prótesis, lo que lleva a altas tasas de rechazo. Abordar este desafío requiere prótesis que no solo restauren la funcionalidad, sino que también ofrezcan control natural y autonomía para las actividades diarias.

En particular, las prótesis a menudo muestran retroalimentación sensorial inadecuada e información propioceptiva limitada. Esta deficiencia en la percepción sensorial conduce a un control insuficiente del deslizamiento, dificultades para ajustar la fuerza de agarre, complicaciones en la manipulación de objetos y disminución de la destreza. Estos factores, junto con una gran dependencia de las señales visuales, dificultan la integración fácil de los dispositivos protésicos en las rutinas diarias y desafían la aceptación de los usuarios.

En línea con estos objetivos, esta tesis de Master busca mejorar la seguridad en el agarre al introducir un algoritmo de aprendizaje automático capaz de interpretar información sensorial de sensores táctiles integrados dentro de la mano protésica. El trabajo implica una visión general exhaustiva de los mecanismos de retroalimentación no invasivos existentes y las tecnologías de sensores táctiles, junto con una exploración en profundidad de metodologías experimentales para detectar y predecir el deslizamiento.

Posteriormente, para construir y entrenar el algoritmo de aprendizaje automático, se recopiló un conjunto de datos exhaustivo, que incorpora varias acciones categorizadas en tres grupos principales: agarre, riesgoso y no riesgoso. Estas acciones implicaron interactuar con objetos de diferentes formas y texturas. Los datos se recopilaron utilizando una prótesis disponible comercialmente, específicamente la mano Michelangelo, equipada con seis sensores táctiles integrados en sus dedos.

Finalmente, se llevó a cabo una validación experimental, que involucró a participantes externos interactuando con la mano protésica. Esta validación sirvió para evaluar la precisión de las predicciones del algoritmo y recopilar comentarios para posibles mejoras. A través de este proceso iterativo de recopilación de datos, desarrollo de algoritmos y validación experimental, esta tesis tiene como objetivo predecir el deslizamiento para garantizar la seguridad en el agarre.

Acknowledgements

Sono trascorsi ben due anni e mezzo dall'ultima volta che mi sono ritrovata qui a ringraziare tutte le persone che hanno contribuito al mio percorso. Sembrava ieri, eppure ne sono successe di cose in questi due anni e mezzo! Per prima cosa, Maps mi ha notificato che ho raggiunto quota 251 città visitate... ah no? non era questo lo scopo dell'università? Forse avrei dovuto puntare ad un tour delle aule studio nel mondo? Beh, neanche quella cifra è bassa. Io e il mio fedele computer abbiamo girato il mondo insieme, e quando avevo bisogno di una pausa tra un 20 km di camminata e l'altro, quale migliore cosa che mettersi a studiare in un bar con vista?

Questa volta, però, è stata un'esperienza diversa. Ho avuto il privilegio di chiamare casa ben quattro di queste città: Ortona, Torino, Madrid e Monaco.

A Ortona ho lasciato la mia famiglia, in senso geografico eh, perché anche se lontani chilometri loro trovano sempre il modo di essermi vicini. Come papà, che con i suoi "Ciao bimba, come stai?" arriva sempre al momento giusto, grazie ad un uccellino che gli sussurra sempre le informazioni giuste, o come mamma (l'uccellino), che con la sua capacità di inventare sempre qualcosa di speciale, sa come regalarmi un sorriso anche nelle giornate più difficili. Sapete quanto è difficile consolare una persona che si fa prendere spesso da attacchi di panico? Ve lo dico io, molto, eppure mamma ci riesce sempre anche a km e km di distanza. Poi c'è Simone, il mio fratellino, da quando ti sei traferito a Torino il nostro legame si è rafforzato, e non vedo l'ora di rimangiare le tue polpettine al sugo! Fin dal giorno zero sei stato il regalo più bello, anche se sappiamo come è andato il nostro primo incontro. Infine, non per importanza, c'è Milo, il mio cagnolino. Una peste è dire poco, iperattivo e instancabile, ma quando era il momento di studiare, tu eri sempre lì sotto la mia sedia ad aspettare che finissi, probabilmente per assicurarti che non piangessi, chissà.

Grazie per tutto quello che avete fatto in questi anni, anzi, tutta la vita. Grazie per la presenza, per i sacrifici, perché senza di voi oggi non sarei qui, banale ma vero. Grazie per avermi insegnato quali sono le cose veramente importati nella vita, per avermi sempre permesso di scegliere e per aver fatto crescere in me la determinazione di poter ottenere tutto ciò che voglio.

Poi è arrivata Torino. Questa città, oltre alle tante giornate di pioggia, mi ha omaggiato di una fantastica persona, Andrea, il mio fidanzato. Un santo e un coraggioso a detta di qualcuno, per avermi sopportato durante le settimane conclusive di questa tesi. C'è una parola tedesca impronunciabile "lebenslangerschickschalsshots" che dicono significhi "lifelong treasure of destiny". Non sono un'esperta di tedesco, quindi, potrebbe anche significare "svuota-frigo", direi che in entrambi i casi ti si addica alla perfezione. Per tutte le volte che mi hai detto "Vedi! Lo sapevo che ce l'avresti fatta", Andrea, è risaputo che bisogna farsi un piantino prima di conquistare il mondo! Non so dove ci porteranno le nostre vite, difficile prevederlo dato che cambiamo città ogni 2x3, ma io sarò sempre lì al tuo fianco per vederti vincere.

Quando Torino ha iniziato a starmi un po' stretta, ecco che arriva Madrid. Le emozioni e le esperienze che ho vissuto in questa città sono indescrivibili. Qui ho ritrovato la spensieratezza che avevo perso, la voglia di esplorare il mondo e di conquistarlo. Ha senso ringraziare una città? Per me, sì.

Ma purtroppo, anche le esperienze più belle giungono al termine, e così è stato quando ho dovuto lasciare Madrid per approdare a Monaco di Baviera. Non è scattata la scintilla, lo ammetto, ma questa città sicuramente mi ha fatto capire quanto io sia forte, sia perché ho resistito a temperature di -15°C, sia per le sfide che sono riuscita a superare durante lo sviluppo di questa tesi.

Ovviamente, durante questo percorso ho fatto nuove amicizie e conservato quelle di vecchia data. Anche se non posso ringraziarvi uno per uno, visto che la consegna della tesi è tra sole quattro ore, sappiate che vi voglio bene. Grazie di tutto!

Tra queste persone, vorrei fare due menzioni speciali. La prima a Stefano, sei stato una manna dal cielo, il fratello di cui non sapevo di aver bisogno. Con te ho trovato svago, serenità e anche una spalla su cui piangere. Ti conservo nel mio cuoricino insieme a tutti i momenti belli e i video dei corgi. Grazie!

La seconda menzione va a Pachi, l'amico di sempre. Anche se a km di distanza, questa volta abbiamo condiviso insieme le gioie e i dolori dello scrivere una tesi. Grazie per i messaggi di incoraggiamento e grazie per non avermi bloccato dopo che ti ho ghostato non so quante volte, scusa ho dei problemi con la tecnologia.

Vivere lontano da casa non è per tutti. Serve un cuore grande abbastanza da contenere tutto ciò che si lascia dietro.

Eppure, mi sento così fortunata ad avere delle persone meravigliose al mio fianco.

"Rare sono le persone che usano la mente,
poche coloro che usano il cuore
e uniche coloro che usano entrambi."

Rita Levi-Montalcini

Questo traguardo
è dedicato a tutte le donne
e in particolare a mia mamma,
la donna più forte che abbia mai conosciuto.

Table of Contents

List of Tables

List of Figures

Acronyms

ADLs
: Daily Life Activities

ULP
: Upper Limb Prosthesis

EMG
: Electromyography

SiNR
: Silicon NanoRibbon

FEA
: Finite Element Analysis

(F)PCB
: (FFlexible) Printed Circuit Board

CDC
: Capacitance to Digital Converter

I2C/IIC
: Inter Integrated Circuit

(I)SMSP
: (Integral) Sliding Mode Slip Prevention

PD
: Proportional Derivative

PI
: Proportional Integrative

FSR
: Force Sensing Resistor

DWT
: Discrete Wavelet Transform

(H or L)LC
: (High or Low) Level Control

LMG
: Lightmyography

FC
: Friction Cone

PCA
: Principal Component Analysis

BP(F)
: Bandpass (Filtering)

SNR
: Signal to Noise Ratio

IC
: Integrated Circuit

Chapter 1

Introduction

In everyday life, as individuals interface with the external environment, the remarkable capabilities of the human senses are often overlooked. From simple tasks like shaking hands with a friend, zipping up or catching a ball, to more complex actions such as putting shoes on in the dark; beneath the surface of these seemingly ordinary interactions, the ability to perceive and manipulate objects relies heavily on sensory feedback. [1]

It is this combination of tactile perception and proprioception, that is the basis of what is called haptic feedback, a concept that is gaining great importance in the prosthetic sector. The term 'haptic' finds its origins in the Greek word meaning "related to the sense of touch" [2]; enclosing not only the perception of objects through tactile sensations but also proprioceptive manipulation. This definition is consistent with the one coined by Gibson in 1966 as "the individual's sensitivity to the world adjacent to his body" [3], underlining so the enduring importance of this concept.

In the human experience, sensorimotor feedback serves as a constant source of information about the surrounding environment, directing actions and interactions. Consequently, it can be deduced that the absence of this information, even during the most basic daily activities (ADLs), poses a significant challenge for prosthetic users, and presents individuals with a range of obstacles, encompassing physical limitations and psychological impacts. [4] [5].

To this end, in recent decades, prosthetic devices have played a pivotal role in restoring both mobility and functionality to individuals who have undergone limb loss due to a variety of factors, such as traumatic injury, congenital conditions, or medical amputations. Moreover, the development and implementation of prosthetic solutions have been able to adapt to the severity of these amputations, which can vary greatly, ranging from partial to complete loss of limb, each presenting its own set of unique challenges.

As prosthetic technology has evolved over time, there has been a notable transition from rudimentary wooden structures to highly sophisticated devices incorporating cutting-edge materials and technologies. [6]. However, despite these advancements, a significant proportion of existing prosthetic systems still lack adequate somatosensory feedback, making prosthetic users often rely heavily on visual cues for functionality. This deficiency in sensory information, coupled with the over-reliance on visual feedback, is one of the main causes of rejection, as individuals may find the prosthetic experience unnatural or uncomfortable, and has contributed to rejection rates for prosthetic hands reaching as high as 40% [7].

Recognizing this problem, researchers have explored various approaches to enhance user experience and improve functionality of prosthetic limbs, and one promising approach that has garnered increasing attention is the one of shared control. Shared control refers to a symbiotic interaction between the user and the prosthetic device, where both parties contribute to the control of movement and manipulation tasks [8]. By integrating elements of both user intention and automated control algorithms, shared control seeks to bridge the gap between human intuition and machine precision, ultimately enhancing the user's sense of agency and control over the device.

While shared control holds immense promise in revolutionizing prosthetic technology, it is not without its challenges. One notable limitation is the potential for delays in information processing, which can arise due to factors such as sensor latency, computational complexity, and communication bandwidth constraints [9]. Moreover, achieving seamless integration between user intention and automated control algorithms poses a considerable technical and algorithmic challenge. These problems can impact the real-time responsiveness of the prosthetic system, leading to suboptimal user experiences and reduced overall performance.

1.1 Thesis' objectives

Despite the challenges, the potential benefits of shared control in enhancing prosthetic functionality and user satisfaction are undeniable. With this objective in mind, by addressing the limitations of existing prosthetic systems, this work seeks to develop a novel method that combines user intention with a machine-learning algorithm to improve grip stability and object manipulation in real-world scenarios.

Central to this endeavour is the introduction of an unexplored approach that goes beyond traditional prosthetic control paradigms. Specifically, this research focuses on distinguishing between safe and risky slips in everyday actions, offering a deeper understanding of grasping dynamics. Later on, through the integration of an advanced machine learning algorithm capable of interpreting tactile sensory data, the system aims to recognize, and potentially prevent, risky slip occurrences, thereby elevating grasping safety and user confidence.

By prioritizing the development of sensory feedback mechanisms through advanced sensor technology and the implementation of a machine learning algorithm, this thesis project aims to enhance the functionality and safety of prosthetic limbs. While current myoelectric prostheses have focused on motor function, integrating advanced sensor technology has the potential to revolutionize the user-prosthesis experience. Such integration is not an option but an essential requirement in prosthetic design, representing a pivotal element in improving overall quality of life and fostering independence among amputees.

1.2 Thesis' outline

The thesis is structured as follows:

Chapter 2:
This chapter introduces the theoretical foundation of slip detection. Initially, it discusses the significance of non-invasive sensory feedback methods and outlines the required sensors. Subsequently, it directs attention towards the primary objectives of this thesis project: slip detection and shared control with an exploration of the latest methodologies in these fields.

Chapter 3:
The chapter introduces the proposed Shared-Autonomy Control Method, examining the Friction Cone and Bandpass Filter Methods' advantages and limitations. Later on, it suggests the approach used to enhance grasping stability and finally, a Random Forest Classifier for machine learning-based slip detection is implemented.

Chapter 4:
This chapter specifically explores the initial phase of this work, concentrating on the examination of pre-recorded data to achieve a thorough comprehension of the subject matter.

Chapter 5:
This chapter introduces and explains the hand and sensors used in the project's development, also offering a brief overview of the final structure before addressing sensor calibration. Following, once confirmed the feasibility of this study in the previous chapter, attention shifts to implementing and refining the chosen method and creating a dataset aligned with the study's scope. Finally, two participants will interact with the robotic hand to validate the model.

Chapter 6:
This chapter presents the comprehensive results of the offline analysis, online analysis, and the human study. It encapsulates results from the previous chapter,

highlighting outcomes of the implemented methods, the effectiveness of the developed dataset, and the validation process involving participant interaction with the robotic hand.

Chapter 7:
This chapter critically examines the results obtained from the study, delving into their implications and broader significance. It also identifies potential limitations and areas for improvement, paving the way for future works and developments.

Chapter 8:
In the concluding chapter, the study's key findings are summarized, providing closure to the research endeavour.

Chapter 2

State of the Art

Delving deeper into the topic of slip detection necessitates an understanding of non-invasive sensory feedback methods. In this chapter, tactile and mechanotactile feedback will be explored as a starting point. Following this, an overview of the sensors required to enhance sensory experiences will be provided. Lastly, the focus will shift towards the core objectives of the thesis project: slip detection and shared control, along with an examination of current state-of-the-art methodologies.

2.1 Upper-limb Prostheses

Over the years, there has been notable evolution in upper limb prostheses, employing technological advancements to boost functionality and elevate the quality of life for those with upper-limb loss. This short chapter delves into the recent advancements and trends in upper limb prosthetics, examining different types of prosthetic devices, their hardware elements and control mechanisms.

2.1.1 Body Powered Prostheses

Conventional upper limb prostheses have long been the cornerstone of prosthetic rehabilitation, providing essential functionality for individuals with limb loss. Among these, body-powered prostheses remain a widely used option. Controlled through mechanical cables and harnesses, these prostheses rely on the movement of the residual limb to generate tension, enabling basic tasks such as grasping and lifting [10]. While cheaper, durable, reliable and requiring easy maintenance, body-powered prostheses are limited in their range of motion and dexterity so considered more suited to manual labour [11].

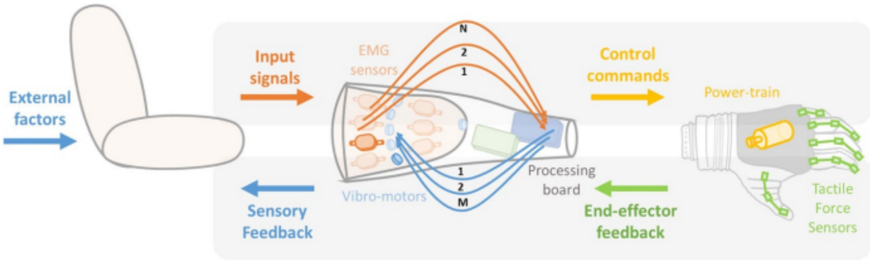
Within the realm of conventional prostheses, socket design is of paramount importance [12]. The socket serves as the interface between the residual limb and the prosthetic device, playing a pivotal role in ensuring comfort, stability, and proper transmission of control signals.

Figure 2.1: A body-powered prosthesis with highlighted principal components, from [13].

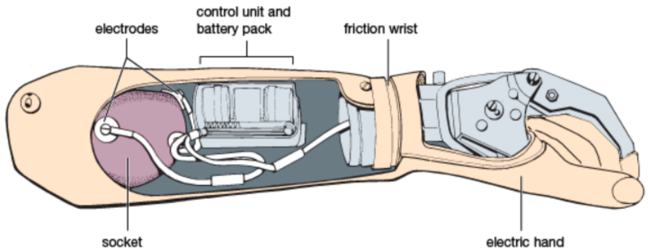
2.1.2 Modern Myoelectric Prostheses

In recent years, advances in technology have guided everyone into a new era of upper limb prosthetics, with myoelectric prostheses leading the way. Myoelectric prostheses utilize electromyographic signals generated by residual muscles to control a specific movement of the prosthetic limb. These algorithms, interpreting electromyographic signals to generate precise movements, offer users more intuitive and precise control, allowing for a wider range of movements and tasks without the need for mode switching [14]. Moreover, advancements in materials science coupled with the progress in the realm of 3D printing have resulted in the production of lightweight and long-lasting prosthetic components, which significantly improved comfort and usability for users [15].

Advanced prosthetic limbs, such as myoelectric prostheses, often incorporate sophisticated hardware components and control mechanisms. Socket design remains crucial, ensuring optimal fit and comfort for users [12]. Additionally, advanced prostheses may feature more complex sensor arrays, including inertial sensors and advanced feedback mechanisms to enhance functionality [16].



(a)



(b)

Figure 2.2: Figure a) illustrates the user-device interaction in an Upper Limb Prosthesis (ULP) system. The left panel showcases the user's perspective, encompassing input signals, sensory feedback, and external factors. The right panel depicts the device level, featuring control commands and feedback collected by the end-effector. The bidirectional exchange of information between the user and the device is highlighted. Image from [16]. Figure b) highlights components of a below-elbow myoelectric prosthesis, including the socket, electrodes, control unit with battery pack, friction wrist, and electric hand. Image from [17].

2.1.3 Open Challenges

Recent advancements in upper limb prosthetics have guided a new era of innovation, offering a range of promising design options and approaches. Decades of research on myoelectric prostheses have yielded a plethora of solutions, both invasive and non-invasive, for interfacing with body signals. These advances have significantly enhanced prosthetic capabilities, particularly in terms of control strategies and functional achievement.

Machine learning techniques hold promise for interpreting user intentions in prosthetic devices via non-invasive interfaces, augmenting control and usability. Short-term solutions, mainly surface electromyography (sEMG), offer advantages such as affordability and intuitive control. Nevertheless, often encounter issues with robustness due to susceptibility to various artifacts, thereby driving the need for continuous research and enhancement. Also worth mentioning, wearable technologies are promising, particularly for daily living activities. Conversely, long-term invasive solutions offer direct bidirectional interaction with the nervous system, thereby enhancing device functionality and usability.However, persistent challenges such as electrode invasiveness, signal quality, and stability continue to necessitate ongoing research endeavors aimed at optimizing physical interfaces and refining filtering processes. While brain-based approaches are still experimental and challenges remain, research in both academic and non-academic contexts (companies like Neuralink, Facebook Reality Labs, and Google DeepMind) is pushing the boundaries of neuroprosthetics, with the potential to revolutionize everyday life for amputees. [16] [18]

A further obstacle involves the biomechanical integration of artificial limbs with the body. Despite recent advancements in materials and high levels of customization in these technologies, the current options for sockets remain largely unsatisfactory for patients. Osseointegration emerges as a promising clinical alternative, directly attaching the prosthetic limb to residual skeletal structures. This approach alleviates discomfort and pain associated with pressure on soft tissues. [19]

Lastly, serving also as the foundation of this thesis project, there are feedback methods. These strategies play a crucial role in improving the acceptability and performance of robotic prosthetic hands. However, numerous commercial devices lack sensory feedback, causing users to heavily rely on visual inputs, leading to fatigue and potential errors. As a response, researchers are actively investigating tactile feedback mechanisms to offer users more intuitive and natural sensory experiences. [20] [21] [22]

2.2 Sensory feedback

2.2.1 Tactile Feedback

Tactile feedback enables us to experience textures and shapes, feeling the details of everything that surrounds us.

The replication of tactile sensations within prosthetic devices can be approached through two primary modes: vibrotactile feedback and electrotactile feedback.

- Vibrational feedback employs small, commercially available vibrators, typically compact and lightweight. These are applied to the skin surface, activating the Pacinian corpuscle mechanoreceptors, responsible for detecting touch, pressure or vibration changes. As users become familiar with the association between the vibration at that site and the sensory input from their prosthetic hand, a portable vibratory haptic feedback system integrated into the prosthesis has the potential to improve the grip force accuracy and gripping technique of upper-limb prosthetic users during daily life tasks [23].

Nevertheless, there are certain limitations associated with vibrational feedback. One noteworthy is the delay in stimulation, which can impact the sense of embodiment [24] [25].

Moreover, experiments showed that when grasping tasks are performed under visual control, the enhanced proprioception offered by a vibrotactile system is practically not exploited [26].

- In contrast to vibrational feedback, electrotactile feedback consists in the use of electrical stimulation applied directly to the skin. This method has found vast application in sensory restoration for prosthetic hands, thanks to its advantages, including non-invasive, decoupled parameters, compact electronics and a different number of electrode pads that can be strategically arranged.

  However, it is essential to acknowledge potential concerns about electrotactile feedback. Indeed, while effective, someone may find it uncomfortable, considering factors such as pain thresholds and placement positions. Moreover, it may require re-calibration in case of prolonged use, since it can lead to desensitisation of the person using it. Nonetheless, dual-parameter modulation can be used to substantially improve the performance in spatial localization of the stimulated tactile sensation [27].

  Another problem could be EMG interference. Several solutions were proposed to avoid that, for example, O time-division multiplexing [28] assures that myoelectric control and electrotactile stimulation are never occurring at the same time; or also, utilizing artifact blanking - or a minimal reduction in performance - it is possible to eliminate the negative influence of the stimulation artifact on EMG pattern classification in a broad range of conditions, thus allowing to close the loop in myoelectric prostheses using electrotactile feedback [29].

2.2.2 Mechanotactile Feedback

One of the methods of delivering sensory information is called “modality matching” [30], meaning the sensation’s production in the user is similar to the type of information to be transmitted. One way to go through this approach involves using mechanotactile feedback to provide tactile sensations or feedback to a user.

Preliminary tests conducted by Aziziaghdam and Samur in [31] showed that an object’s softness or hardness could be identified by analyzing the acceleration response obtained when tapping an object.

Moreover, it was verified in [32] that mechanotactile sensory feedback might not only be useful for improving the sense of ownership and location but also may have a modulating effect on the sense of agency when provided asynchronously during active motor control tasks.

However, it's important to note that this feedback was not statistically significant compared to visual feedback. Also, early implementations of mechanotactile feedback devices often were quite large and provided unnecessary bulk to prosthetic devices.

2.2.3 Others Indirect Feedback

Beyond these direct forms of sensory feedback, there are also other types of indirect feedback, each of which has a unique role to play in enhancing the haptic experience within prosthetic devices.

- **Thermal feedback**, introduces the perception of warmth and coldness, contributing to a more comprehensive sensory experience when interacting with the external environment. However, since it is not a priority to occur by itself, a potential focus of research would be to incorporate temperature feedback with another feedback method so that they occur simultaneously [7].
- **Audio feedback** complements tactile and pressure feedback, providing auditory cues to communicate robotic hand movements. For example in [33] the variance in volume represented the level of grasping force and the varying frequency corresponded with the location of two different regions of the hand, or in [34] a triad identified the movement of different fingers. However, each of these audio feedback experiments was conducted within the laboratory so it needs further testing to understand the usability given environmental noise.
- **Augmented reality**, with its capacity to overlay digital information onto the physical world, adds yet another layer to the haptic experience, but while effective, it also requires increased cognitive load from users and this high level of cognitive effort may affect the overall user experience [7].

Each of these feedback methods has its distinctive characteristics and applications, making them suitable for specific scenarios and user preferences. However, it is being studied the possibility to combine them, even if testing was only conducted on able-bodied subjects. [35] [36]

2.3 Tactile sensors

Sensors are what can be identified as the bridge between the physical world and users' sensory experiences. In prosthetic devices, indeed, sensors play an essential role, enabling users to regain perception and a deeper connection to their surroundings.

This section tries to shortly explain the role of sensors and how they work, in order to comprehend how they create a more adjusted sensory experience for those relying on prosthetic devices.

2.3.1 Resistive sensor

Resistive sensors operate on the principle of electrical resistance, which changes when strain is applied, enabling them to detect mechanical pressure and deformation. An example of a resistive sensor in prosthetic technology is found in [37], where a stretchable prosthetic skin is equipped with an ultra-thin single crystalline silicon nanoribbon (SiNR) strain, pressure and temperature array, with the integration of stretchable humidity and heater sensors (Figure 2.3).

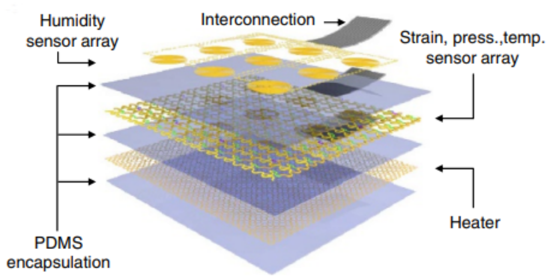


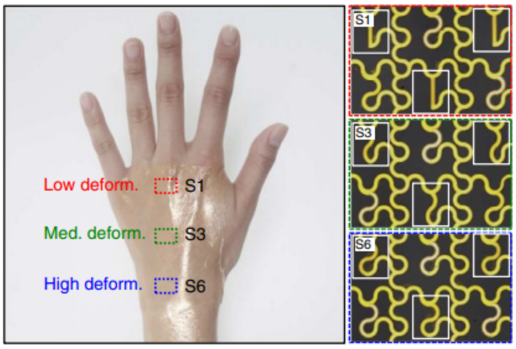
Figure 2.3: An exploded view of a resistive sensor. From [37].

This array of sensors exhibits a variety of geometric configurations, spanning from linear (S1) to progressively increasing curvature (S6), see Figure 2.4. By using this design strategy, the response to highly variable external environments is dramatically enhanced, providing the highest spatio-temporal sensitivity and mechanical reliability.

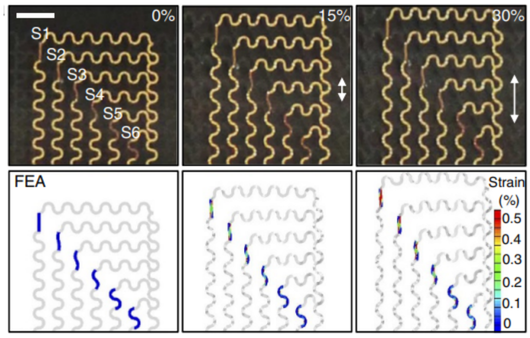
Utilizing a motion capture system, they could identify distinct hand zones undergoing various ranges of motion to strategically position sensors equipped with different SiRN strain gauges, as shown in Figure 2.4a.

To evaluate how strains impact various SiNR sensor designs, it was observed that as applied strains increased, SiNR strain gauges with minimal curvature underwent significantly higher strain levels than those with greater curvature. Yet, while the latter could endure more substantial applied strains, they exhibited reduced sensitivity. This phenomenon was examined by measuring relative resistance (ΔR/R) in relation to applied strain. Thus, it was determined that SiNR S1 is best suited to locations with limited stretching, while SiNR S6 is better suited to areas subjected to more significant stretching, as illustrated in Figure 2.4b and 2.4c.

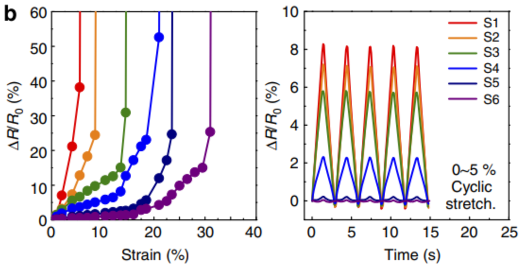
While the temperature sensor should ideally be unaffected by mechanical deformations, the divergence between I-V curves under different strains is notably reduced as sensor curvature increases. Figure 2.4d displays calibration curves for a specific current value. While S1 design shows significant shifts in response to strain, the S6 design remains stable. Deducing that S6-designed temperature sensors will be used to minimize the effects of mechanical deformations, ensuring reliable temperature monitoring under varying pressures.



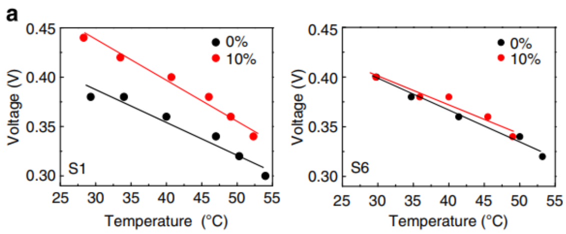
(a)



(b)



(c)



(d)

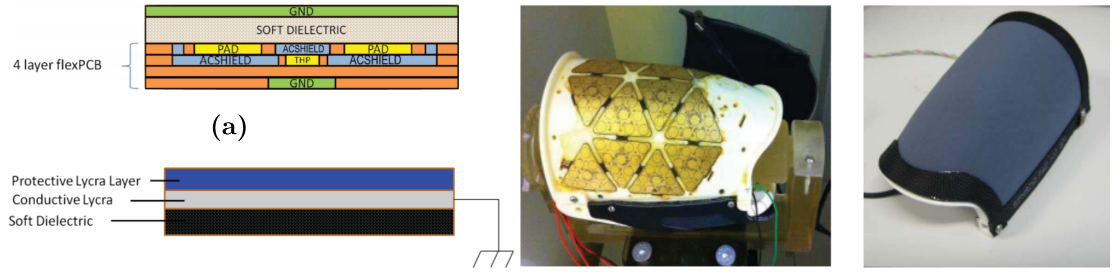
Figure 2.4: Image a) displays the fabricated site-specifically designed SiNR strain gauge arrays attached to the back of the hand. Magnified views of each design are shown on the right. Figure b) shows SiNR strain gauges (top frames) under different applied strains, along with corresponding FEA results (bottom frames). Figure c) shows on the left the resistance changes for different curvatures of SiNR, depending on the applied strain, and on the right, temporal resistance changes of different curvatures of SiNR under cyclical stretching. Finally, d) shows calibration curves of SiNR temperature sensors for representative designs (S1: graph on the left and S6: graph on the right) under stretched and unstretched conditions. All images are from [37]

In conclusion, the adaptability of this electronic skin was monitored in various real-life scenarios, including typing on a keyboard, catching a ball, handling hot/cold objects, touching diapers, and simulating body temperature. Each of these scenarios revealed improved functionality and high performance.

2.3.2 Capacitive sensor

Capacitive sensors measure changes in capacitance, a property that varies with the proximity of the two plates. These sensors play a significant role in detecting touch and interaction because they are small, compact and with high sensitivity.

Generally, their main problem is related to hysteresis and temperature sensitivity, but in the example proposed in [38] they present a novel solution using a thin layer of 3D fabric glued to a conductive and a protective layer (clothing industry techniques). Moreover, the capacitors used are insensitive to pressure so they can be used for temperature compensation (2 taxels inside FPCB).



(b) **(c)**

Figure 2.5: In a) a vertical section of the structure of the tactile module. Figure b) displays fabrics that will constitute the new dielectric for the sensor. Finally, c) shows the integration of a mesh of sensors on a prosthetic forearm. All images are from [38].

The sensor used in this study is built on a flexible PCB, with a conductive area forming the first capacitor plate. On top of the FPCB, there's a deformable dielectric (changing with applied pressure) and a conductive layer, which acts as the second plate and serves as a common ground plane to protect against electromagnetic interference.

The FPCB has a triangular shape and accommodates 12 sensors (2 taxels embedded in the FPCB + 10) and a Capacitance-to-Digital Converter (CDC, AD7147 from Analog Devices), which measures the capacitance of each sensor, performs analog-to-digital conversion and transmits values via a serial line.

Multiple triangles can be interconnected to create a mesh of sensors covering the desired area; moreover, being flexible, they can adapt to curved surfaces.

In conclusion, in [38] the objective was to assess the sensor's performance, specifically in terms of repeatability, sensitivity, hysteresis, and spatial resolution. They will demonstrate that the sensor exhibits satisfactory performances, with particular emphasis on its minimal hysteresis (an improvement from the previous sensor). Additionally, it will be shown the effective use of the introduced thermal sensors in the FPCB for compensating drift caused by temperature changes.

2.3.3 Inductive sensor

Inductive sensors operate based on electromagnetic induction to detect the presence of objects. These sensors have found applications in prosthetic devices as described in [39], with initial improvements detailed in [40] and subsequent enhancements in [41].

The sensor under consideration utilizes a single MLX90393 chip capable of providing 3-axis magnetic and temperature data. It is embedded within a soft material, specifically silicone rubber, with a small magnet placed approximately 5mm above it, as illustrated in Figure 2.6.

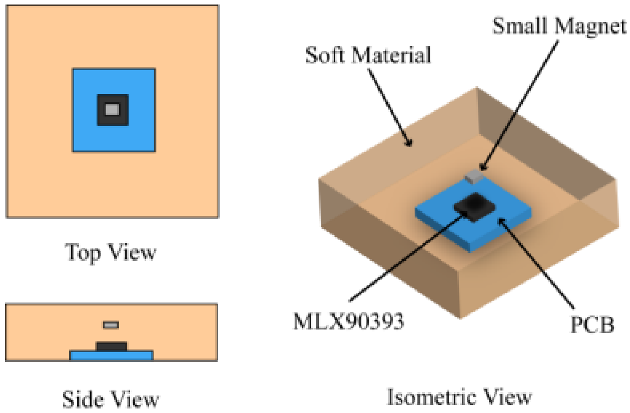


Figure 2.6: Design of an inductive sensor. From [39].

In the first study, two tests were conducted: one for measuring normal force and the other for assessing both normal and shear forces. Both tests yielded positive results, demonstrating the sensor's ability to detect normal and shear forces, particularly in the y and z axes. However, some unexpected readings were observed in the x-axis, which could be attributed to misalignment or crosstalk between axes.

While the initial work presented only preliminary results with the sensor, subsequent papers offered a more comprehensive characterization of the sensor.

These subsequent studies encompassed three tests: evaluating thermal drift, hysteresis, and load capacity. In the thermal drift assessment, it was found that the z-axis was the most affected by temperature variations. After implementing linear regression and temperature compensation, the results improved significantly. Further enhancements can be achieved with the use of a high-pass filter.

The second test focused on hysteresis, which is partly attributed to the silicone covering the sensor. However, the study did not primarily focus on the choice of optimal materials.

As for the last test, both normal and shear forces were found to have good correspondence, after calibration, of course. Additionally, the study assessed the sensor's capability to detect a minimal load of approximately 1 gf along the z-axis.

The integration of distributed sensors into the limited space of robot hands presents a significant challenge. To enhance the work described earlier regarding inductive sensors, a customized printed circuit board (PCB) equipped with 16 Hall-effect sensor chips has been developed (Figure 2.7) [41]. Each taxel is capable of measuring the applied 3D force vector using a Hall effect sensor and a magnet with an I2C digital output. Remarkably, each sensor module, consisting of 16 taxels, requires only seven wires.

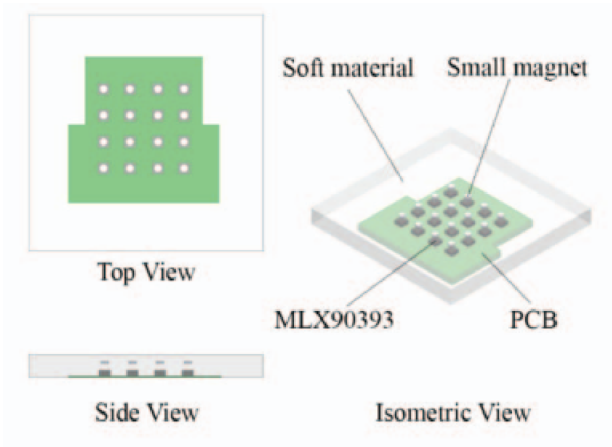


Figure 2.7: Design of a customized PCB equipped with 16 Hall-effect sensors. From [41].

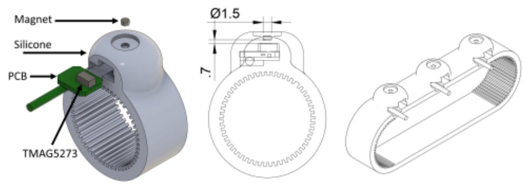
Forthcoming examinations will assess the measurement of normal and shear forces, examine potential crosstalk between the chips, and ensure sensor stability by applying repetitive force.

Similar to the previous tests, when only normal forces were applied, displacements were detected in the x-axis and y-axis, and vice versa when solely shear forces were applied displacements were detected also in the z-axis; probably due to a slight misalignment of the magnet. A crosstalk test was conducted to estimate any magnetic field interference between the sensors, and the results affirmed the sensors' robust functionality. Additionally, the final test confirmed the reliability of the sensors.

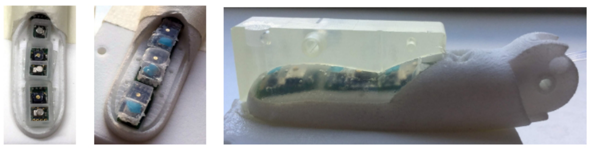
Integration

For what concerns the integration of the sensors, in this document will only be cited what regards the inductive sensor, being the one that will be successively used and implemented for experiments, see Chapter 5.1.2.

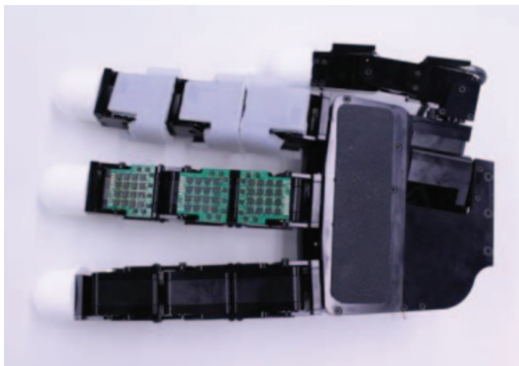
Several examples illustrate different approaches to sensor integration. One notable example employs customizable and scalable silicone bands [42], presented in Figure 2.8a, with two variations: a ring-shaped design for individual sensors and a band-shaped version to cover the palm. These silicone bands offer flexibility by fitting various finger and palm dimensions accurately. Additionally, a ribbed texture is incorporated to enhance friction between the silicone and the fingers.



(a)



(b)



(c)

Figure 2.8: Depicted in a) a design for customizable and scalable silicon bands containing one Hall-effect sensors, conceived to precisely fit a wider range of subjects. From [42]. b) Illustrate the casting process aimed at integrating various types of sensors on the fingertip. From [43]. And c) shows an Allegro Hand integrated with customized PCBs each equipped with 16 Hall-effect sensors, covered with skins (top finger). From [41].

Another approach demonstrated in [43], involves securing PCBs to each finger and encasing tactile sensors in silicone rubber. This process is depicted in Figures 2.8b left and center. Furthermore, an extra layer of silicone rubber, as shown in Figure 2.8b right, covers the entire finger pad to enhance stability. The design integrates holes and canals with undercuts into the fingertip to ensure stability; moreover, having a large part of the finger cast in silicone allows for a greater number of sensors and improved grasping capabilities.

For a hybrid solution, as seen in [41], a PCB with 16 Hall-effect sensors is utilized. After creating the silicone module, it is fitted onto the finger phalanges' motors, and a silicone band is employed to encircle the fingers (Figure 2.8c).

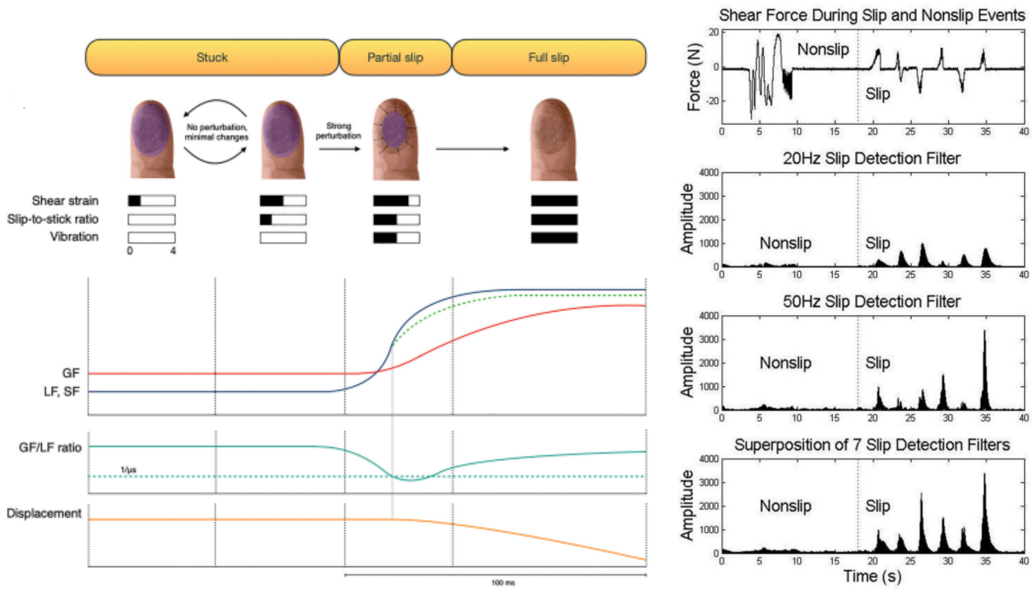
2.4 Slip Detection

Slip detection in the human hand relies on a sophisticated interplay of somatosensory feedback from mechanoreceptors to sense grip changes during object manipulation. This process encompasses distinct phases, including 'stuck,' 'partial slip,' and 'full slip' (Figure 2.9a). Various tactile units react to skin deformation, pressure, and vibration, aiding in slip detection, moreover, factors such as skin hydration and surface irregularities also affect slip dynamics. Collectively, these mechanisms enable humans to maintain a secure hold on objects and respond to slippage.[44]

Humans also possess a frictional memory system that adjusts the force applied to slippery objects based on past experiences with similar frictional characteristics [45]. However, replicating this mechanism in a prosthetic hand is challenging due to the complexity of the human body. The SensorHand Speed is the first commercially available prosthetic hand that attempts slip prevention by increasing grip force proportionally when tangential forces exceed predetermined values [46]. Nevertheless, this approach has limitations, as it does not adapt to varying friction conditions and can lead to object crushing or slippage.

In [47], three slip-prevention algorithms are introduced and experimentally evaluated. The first two are the sliding mode slip prevention (SMSP) controller and the integral sliding mode slip prevention (ISMSP) controller. They are compared to the proportional derivative (PD) shear force feedback slip prevention controller. Additionally, these controllers are compared to a sliding mode controller without provisions for preventing object slip.

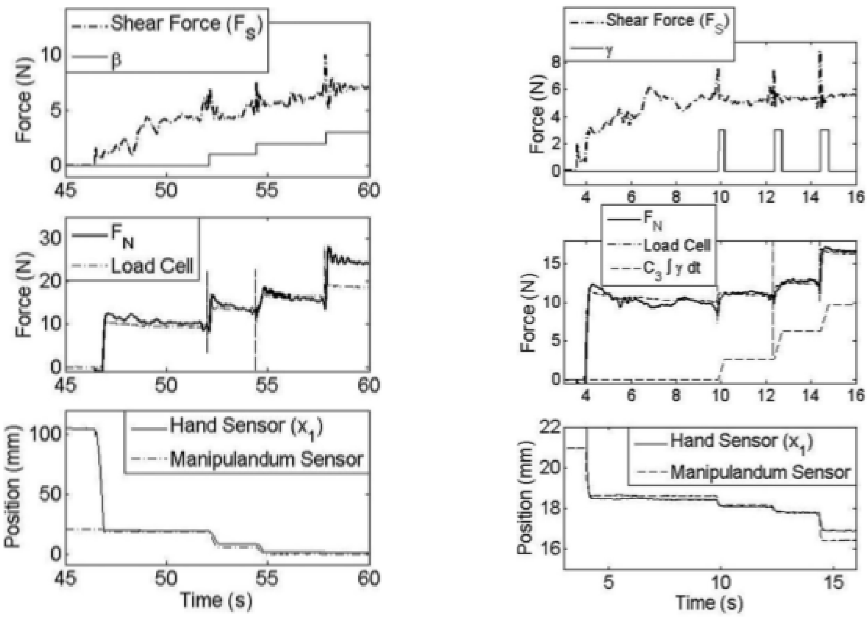
The adaptive SMSP control system is designed to address potential object slipping during grasping. It introduces a slip-dependent state (e_S) into the error equation, allowing slip events to influence the error state. Slip detection relies on strain gauges on the prosthesis, which detect vibrations generated during slip at the hand-object interface. Band-pass filters amplify these vibrations, with different frequencies indicating various slip events (Figure 2.9b). The controller also checks the amplitude of shear and normal forces to ensure reliable slip detection. When slip is detected, grip force is increased to prevent further slipping.



(a) (b)

Figure 2.9: In (a) a schematic quantification of shearing strain of the fingertip (shear strain), slip-to-stick ratio, and vibration, represented on a 5-point scale during each of the three slippage phases. The contact area between the fingertip and the object is highlighted in violet. The slip-to-stick ratio progressively declines from 0 (no slip) to 1 (full slip) as partial slip develops over time. Below is a relation between grip force (GF, in red), shear and load force (SF and LF, in green and blue) during each slippage phase. Image from [44]. Additionally, (b) depicts the effectiveness of band-pass filters resonating at 20 Hz and 50 Hz, with the superposition of seven slip-detection filters between 20 Hz and 50 Hz, proving to be highly effective in detecting slips. Image from [47]

Determining the appropriate increase in grip force when slip is detected is a real challenge. One approach involves defining the slip error state as $e_S = C\beta$. This way, the controller increases grip force each time it detects slip events. However, this approach can lead to rapid and excessive grip force increases, potentially crushing the object (Figure 2.10a). An alternative method, the Integral Sliding Mode Slip Prevention (ISMSP) controller, integrates the slip signal upon detection, resulting in a smoother increase in grip force to prevent object slip (Figure 2.10b). This approach minimizes deformation, maintains control over position and velocity, and avoids excessive force spikes.



(a) **(b)**

Figure 2.10: (a) Demonstration of the SMSP control algorithm. The SMSP controller increases the grip force in discrete, predetermined amounts during each of the three detected slip events. This often results in a larger grip force than is necessary. (b) Demonstration of the ISMSP control algorithm. The ISMSP controller integrates the slip signal to smoothly increase the grip force until the grasped object stops slipping. Thus, the minimal required grip force is applied to prevent more slip. Images from [47]

Both of these controllers were compared with a PD shear force feedback slip prevention controller. This controller operates by adjusting the applied grip force based on the measured shear forces, using positive feedback to achieve this. However, it may inadvertently crush objects, even when they don't slip, and might not effectively prevent slip in cases of low friction. This method resembles a commercially available scheme, OttoBock's SensorHand Speed.

As previously mentioned, the ISMSP controller significantly reduces manipulandum deformation as disturbances are applied, effectively preventing object dropping in all experiments. It is statistically different from the SMSP and PD controllers in terms of deformations (Figure 2.11). All three controllers allow minimal slip (less than 2 mm), with no significant difference in slip distance. In contrast, the sliding mode controller permits significant slip due to the absence of provisions for increasing grip force, resulting in object slippage in many cases (20 out of 25).

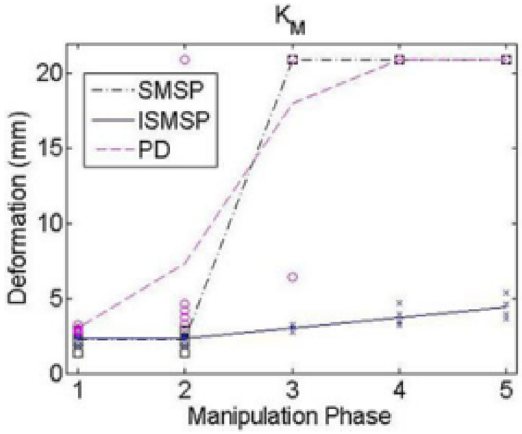


Figure 2.11

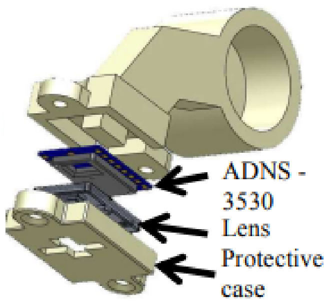
Figure 2.12: Deformation results for SMSP, ISMSP and PD controllers. [47]

Alternative approaches involve the use of force sensing resistors (FSR) sensors to identify slip occurrences through the analysis of the rate of force change, which is then compared to a predefined threshold. For instance, in [48] slip detection is accomplished by employing an FSR and the discrete wavelet transform (DWT) to monitor alterations in grasping force and identify slipping based on high-frequency signal changes. Another example can be found in [49]. This strategy combines proportional-integral (PI) and proportional-derivative (PD) control to manage finger positions, detect slip events, and uphold grip stability. FSR sensors play a key role in detecting slip by analysing the rate of force change and comparing it against a predefined threshold in this system.

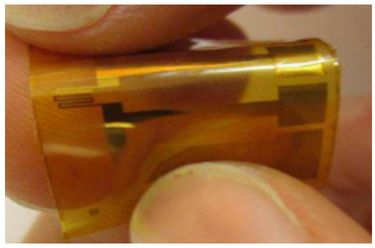
In [50] a method called friction cone analysis, which assesses the stability of a grasp by ensuring that contact forces remain within a designated cone, is presented. However, ensuring the reliability of slip detection requires setting a minimum threshold for the measured force, as noise could yield erroneous γ values. This threshold varies based on the material combination and the specific grasp executed.

A distinct methodology, highlighted in [51], emphasizes the potential of optical sensors in tackling grip control challenges in prosthetic devices. Here, the focus is on an optical-based sensor commonly utilized in optical computer mice, Fig.2.13a. Selected for its compact size, low power consumption, and reliability, this sensor exhibits promising abilities in detecting object displacement across diverse surface properties encountered in everyday life, including roughness, curvature, and reflectivity. However, challenges persist in reliably detecting slips on transparent surfaces, indicating a limitation that necessitates further investigation.

Another different strategy is outlined in [52], presenting a flexible slip microsensor relying on thermo-electrical phenomena, Fig.2.13b. Unlike conventional methods that hinge on mechanical vibrations or friction coefficient estimation, this novel sensor operates on thermo-electrical principles, eliminating the necessity for vibration detection. Its design permits integration onto curved or deformable surfaces, rendering it suited for applications in robotic fingertip prosthetics. By sidestepping the susceptibility to mechanical noise intrinsic in traditional slip detection methods, the thermo-electrical approach offers enhanced reliability. Experimental findings substantiate the sensor's precise discrimination of slip events, underscoring its potential to augment tactile feedback in hand robotic prostheses.



(a)



(b)

Figure 2.13: (a) Prototype of the solid model of the protective case enclosing the optical sensor and lens on the right. From [51]. (b) The flexible slip microsensor based on thermo-electrical phenomena. From [52].

2.5 Shared control

What is shared control? At its core, shared control involves congruent interaction between a human and intelligent agent(s) in a perception-action cycle, jointly executing dynamic tasks typically performed by humans [53] (Figure 2.14a). It's important to note that shared control doesn't imply that both entities focus on exactly the same aspects. Human beings bring inventiveness, adaptability, and problem-solving skills to the table, while the intelligent agent contributes precision, repeatability, and crucially, inexhaustibility, eliminating issues related to human fatigue and thereby enhancing safety.

Shared control entails keeping humans in the decision-making loop, with the provision for humans to override control when necessary. As outlined in [26], shared control delineates a collaboration between a high-level controller (HLC), responsible for interpreting user intentions, and a low-level controller (LLC), which executes the prosthetic hand's actions (Figure 2.14b). Meaning that transitions between different states, identified by the LLC, enable automatic control, whereas those identified by the HLC facilitate interactive control based on user intentions. Furthermore, this study explores three hierarchical control strategies (M1, M2, and M3) with varying degrees of shared control to strike a balance between the user's ability to achieve successful grasps and the cognitive effort required during operation. Ultimately, users favoured M2 (Figure 2.14c), which offers a valuable compromise between functionality and ease of use, resulting in improved grasp success in subsequent trials.

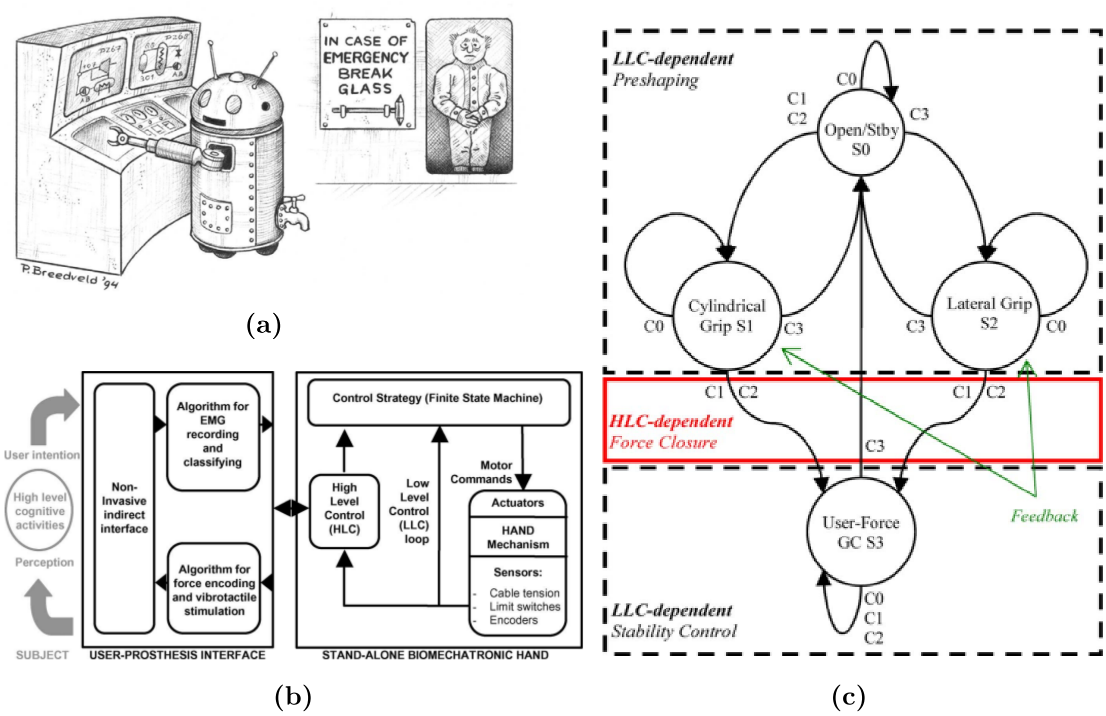
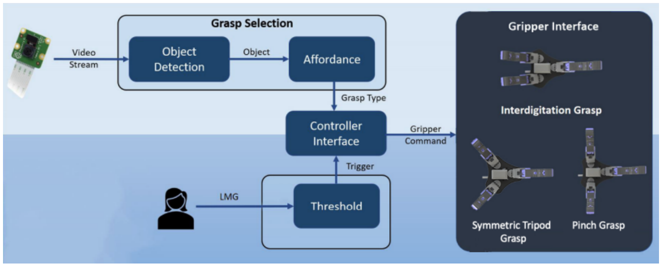


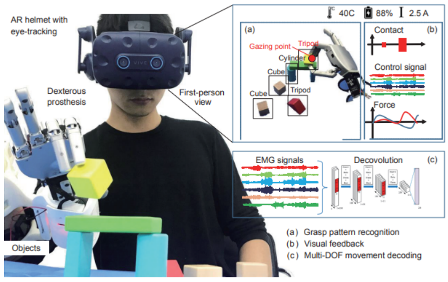
Figure 2.14: Figure (a) shows a light-hearted image on shared control. Figure (b) is a scheme of a prosthetic hand system. Here it is explained that the LLC loop is primarily responsible for grasp stability and the HLC system loop is responsible for selecting grasp configuration and force level requested by the user. From [26]. Finally, on the right, image (c) is an FSM diagram for control strategy M2. Circles represent states, with C0–C3 denoting EMG commands. User-selectable grasps include cylindrical grasp (S1) or lateral grip (S2) initiated by flexor or extensor contractions. Closure is arrested by a second flexor contraction, and the hand applies user-dependent force closure on the object (state S3). The hand reopens after an extensor contraction (S0), with stability and pre-shaping managed by LLC and force closure by HLC. From [26].

The evolution of shared control is also evident in [54], introducing an adaptive prosthetic training gripper with a variable stiffness differential mechanism and a vision-based shared control system (Lightmyography, LMG, for triggering grasps). This innovation empowers users to efficiently grasp a wide array of objects, enhancing accessibility and functionality in the realm of prosthetic devices.

For upper-limb amputees, shared control represents a transformative leap in prosthetic technology, as elaborated in [55]. This study highlights significant benefits, such as improved grip security, reduced cognitive effort, and a remarkable 49% decrease in muscle activity (and, consequently, physical effort), enhancing prosthetic control and the quality of life for those with upper-limb amputations.



(a) Framework integrating an RGB camera-based object detection scheme to select grasp types based on predetermined grasping affordances, along with an LMG-based grasp triggering scheme. From [54]



(b) Block diagram illustrating a human-machine shared control system. Components include (a) first-person view on grasp pattern recognition, (b) visual feedback, and (c) movement decoding. Form [56]

Figure 2.15: Two examples of shared-control systems for dexterous prostheses.

In summary, shared control is a groundbreaking concept in the field of prosthetic technology. Through a hybrid human-machine intelligence, the shared control methods could address the control problem of dexterous prostheses [56], as well as enhance functionality and user-friendliness. This collaborative approach has the potential to revolutionize the lives of individuals with limb loss, offering improved grip security, reduced cognitive and physical demands, and a user-centered design that optimizes the interaction between users and their prosthetic devices, but also with the environment.

Chapter 3

Proposed Shared-Autonomy Control Method

3.1 Slip Detection

In Chapter 2.4, diverse strategies aimed at detecting slip were examined and briefly described. Naturally, each of these strategies comes with its own set of advantages and limitations. Subsequently, the advantages and limitations of the two primary papers on which this thesis project is based will be dissected, to provide a comprehensive understanding of their applicability and efficacy.

The study referenced as [50] sheds some light on the friction cone concept, which emerges as particularly apt in addressing real-world scenarios. This method, thanks to its quick response time and its resilience against external disturbances, is highly desirable for practical applications. However, its effectiveness depends upon variables such as surface texture, as well as the weight and shape of the object being grasped. While constructing a model for the friction cone may be feasible within industrial settings characterized by repetitive tasks and controlled environments, its translation into everyday life encounters obstacles due to the need for extensive pre-analysis of each object and its surrounding environment before interaction.

In contrast, the bandpass method, also elucidated in [47], offers a straightforward implementation process, negating the necessity for prior knowledge regarding the object being grasped. Nevertheless, this approach is markedly more susceptible to problems arising from movement and environmental vibrations, posing challenges to its reliability in real-world applications.

Recognizing the inherent limitations of both methodologies, it becomes evident that a unified approach of these two approaches to slip detection holds great potential for enhancing grasping stability. By leveraging the strengths of each method while addressing their respective weaknesses, a more robust and reliable detection mechanism can be devised. One of the key aspects of this integrated strategy lies in the identification and utilization of critical points derived from the combination of these two methodologies. These critical points serve as pivotal indicators, encapsulating nuanced information about the interaction between the grasping surface and the object. By appropriately labeling these critical points, it becomes feasible to leverage them as input features for training a machine-learning model.

In this context, a random forest classifier emerges as a promising tool for machine-learning-based slip detection. By employing such a classifier, it becomes possible to harness the collective power of numerous decision trees, each trained on a subset of the dataset, to collectively reach a comprehensive understanding of the grasping dynamics and associated instability.

3.1.1 Friction Cone

Friction, in mechanics, plays an important role in understanding the behaviour of objects in contact. In particular, the friction cone stands out as a geometric representation essential for comprehending the dynamics of rigid bodies. However, before delving into the intricacies of the friction cone, it's crucial to distinguish between static and dynamic friction, or better static and dynamic friction coefficients.

Static friction arises when two surfaces are in contact but not moving relative to each other. It acts to prevent the initiation of motion, effectively keeping objects stationary or resisting the onset of motion. On the other hand, dynamic friction, also known as kinetic friction, comes into play when the surfaces are sliding past each other. It opposes the relative motion of the surfaces, acting tangentially to the contact area [57].

The coefficient of friction is a dimensionless scalar value. It is a ratio of the force of friction between two bodies (shear force) and the force pressing them together (normal force), in formulas 3.1 and 3.2. Both static and kinetic coefficients of friction depend on the pair of surfaces in contact. However, for a specific surface, the coefficient of static friction is always larger than the one of kinetic friction since it has to avoid motion.

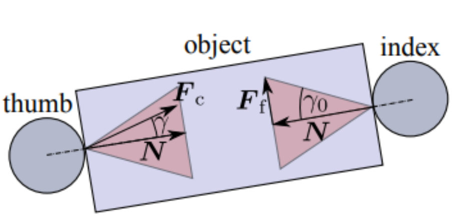
$$F_{shear} = \mu_s \cdot F_{normal} \tag{3.1}$$

$$\mu_s = \tan \gamma_0 \tag{3.2}$$

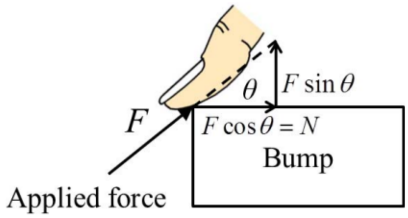
The friction cone, instead, is a geometric representation that delineates all the possible frictional forces at a contact point between two surfaces.

In two dimensions, the friction cone appears as a cone-shaped region around the normal force vector, see Fig.3.1, which is perpendicular to the contact surface.

This cone encompasses all possible directions in which frictional forces can act tangentially to the contact surface. And, if the contact force stays inside the friction cone, the grasp is known to be stable [50].



(a) Friction cone shown in the contact point plane, where N describes the normal force, F_f the friction force, and F_c the contact force [50].



(b) Friction cone shown in the contact point plane, where F is the applied force, $F \cdot \cos\theta$ the normal force and $F \cdot \sin\theta$ the friction (or shear) force [58].

Figure 3.1: Friction cone explanation in 2-dimensions.

From Fig. 3.1b, it is possible to obtain :

$$F \cdot \cos\theta = F_{normal} \tag{3.3}$$

And, using the formula 3.3 together with 3.1 it is possible to obtain formula 3.4:

$$F_{shear} = \mu_s \cdot F \cdot \cos\theta \tag{3.4}$$

Theoretically, from this, it is possible to deduce that when the pulling force has $\theta \neq 0°$, the friction exists, and it will decrease when θ increases. Therefore, a small θ is expected for sufficient friction, otherwise the object will slip [58].

In three dimensions, the friction cone becomes more complex, as frictional forces can act in various directions around the contact point. Nevertheless, the fundamental principle remains the same: the friction cone represents the range of possible frictional forces within a certain angular boundary relative to the normal force vector.

In the robotics field, friction cone is important for ensuring stability and control in robot motion. Robots often interact with their environment through physical contact, whether it's grasping objects or walking; friction cone provides valuable insights into the permissible range of forces and moments that the robot's actuators can exercise to maintain stability and achieve the desired motion.

In this study, the friction cone will help optimize the recognition of different actions that the Michelangelo hand will do. To achieve this, it was decided to apply the concept of the friction cone to the first principal component, leveraging a strategic approach that combines statistical analysis with practical application.

With six sensors capturing data from various points of contact, each corresponding to different fingers of the Michelangelo hand, the task of applying the friction cone becomes inherently more intricate. One of the primary challenges arises from an intrinsic variability in how each finger interacts with the object. Due to differences in finger morphology, mobility and contact dynamics, the distribution and magnitude of forces sensed by each sensor may vary significantly. Moreover, analyzing and interpreting data from six sensors simultaneously requires a robust analytical framework capable of processing and synthesizing information from multiple sources effectively.

To mitigate these challenges, focusing on the first principal component offers a pragmatic solution, converging the six sensors' readings into a holistic representation of the system dynamics.

Upon applying the first principal component, I computed the friction coefficient for both shear forces, F_{shear_x}/F_{normal} and F_{shear_y}/F_{normal}. Such analysis grants valuable insights into the nature and magnitude of frictional forces acting upon the robotic hand during different actions. Indeed, the outcome of this computation, in Fig. 3.2, displays two lines characterized by distinct peaks representing critical points of interest.

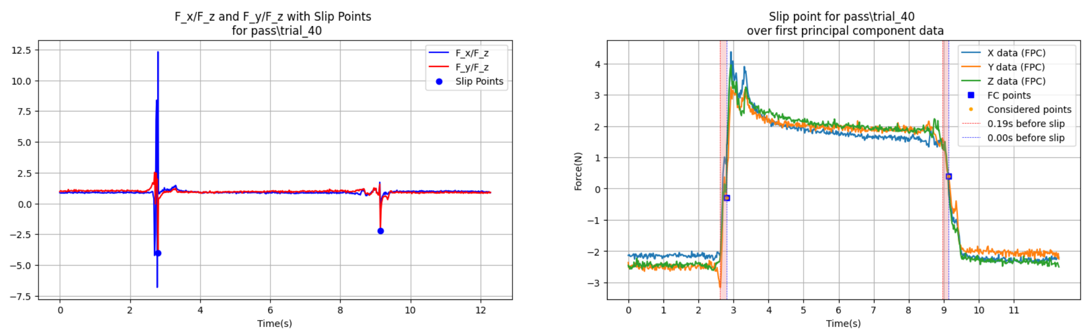


Figure 3.2: Results of friction cone analysis. On the left, the friction coefficients are plotted and the highlighted points indicate where the experimentally determined threshold is exceeded. On the right, the same results are visualized on the first component analysis x, y and z to visualize a global frame. The result shown is the ones obtained for the basket-ball passing action.

Selecting the point(s) exceeding an experimentally chosen threshold has been the criteria elected for identifying critical moments during the execution of actions by the Michelangelo hand. As an added measure of robustness, another criterion has been introduced to enhance the robustness of this analysis, considering the potential influence of sensor noise on results, see Fig.3.3. To mitigate the risk of unwanted peaks interfering with the training of our model, in addition to the friction coefficient surpassing the designated threshold, it was incorporated a requirement for the variation of the z-coordinate to exceed a predefined threshold. This additional criterion serves as a filter, ensuring that only those peaks indicative of significant frictional interactions are considered for further analysis.

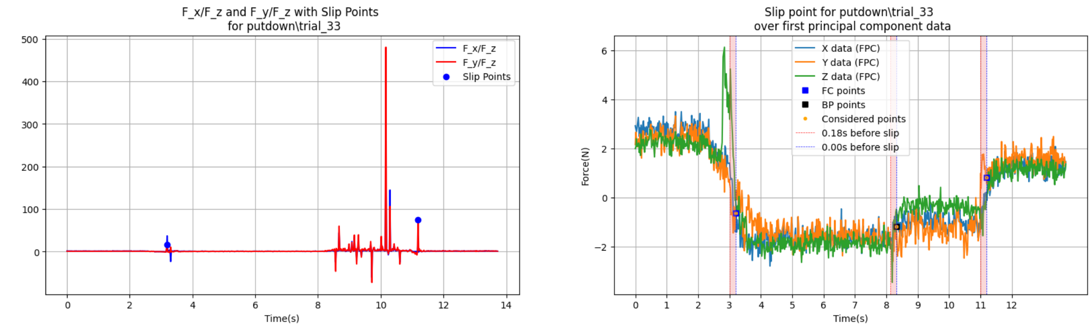


Figure 3.3: Results of friction cone analysis, as for 3.2. In this case, the result corresponds to the hard-ball putting-down action. Despite noise interference, due probably to the hardness of the object that causes some unwanted peaks, critical points were still accurately detected using the criteria outlined in this chapter. Note: for clarity also the critical points obtained with the Bandpass Filter are shown.

3.1.2 Bandpass Filtering

As already mentioned in Chapter 2.4, slip generates vibrations at the interface between the hand and the object, characterized by high-frequency oscillations. These high-frequency vibrations that occur during slip can be amplified by band-pass filtering the measured shear force derivative.

Indeed, in this study, a fifth-order digital filter will be employed. Initially, the signal undergoes differentiation via a high-pass filter, eliminating low-frequency components (steady state). Subsequently, the signal undergoes double filtration through two second-order bandpass filters, resonating near ω_n. This amplifies the vibrations, crucial for slip detection. Lastly, a low-pass filter with a cut-off frequency near 75 Hz attenuates high-frequency noise, ensuring accurate slip detection under varying conditions.

Following, all the formulas used are computed, starting from the transfer function in equation 3.5.

$$H(s) = s\left[\frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}\right]^2 \left[\frac{\omega_{LP}}{s + \omega_{LP}}\right] = s\left[\frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}\right]^2 \left[\frac{75 \cdot 2\pi}{s + 75 \cdot 2\pi}\right] \tag{3.5}$$

It was opted for the bilinear transform, formula 3.6, over the Euler method for mapping our continuous-time filter to the discrete-time domain. The bilinear transform offers great accuracy and stability preservation. Its accuracy ensures faithful preservation of frequency response characteristics, while stability preservation guarantees a stable filter response. Despite introducing frequency warping, the bilinear transform remains manageable and predictable, making it well-suited for this filter design.

$$s = \frac{1}{T}ln(z) = \frac{2}{T}\left[\frac{z-1}{z+1} + \frac{1}{3}\left(\frac{z-1}{z+1}\right)^3 + \frac{1}{5}\left(\frac{z-1}{z+1}\right)^5 + ...\right]$$
$$s \approx \frac{2}{T}\frac{z-1}{z+1} = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}} \tag{3.6}$$

Using 3.5, 3.6 and considering that $K = 2/T$, it is possible to obtain the difference equations respectively for the high-pass filter:

$$H_{hp}(z) = K\frac{1-z^{-1}}{1+z^{-1}} \longrightarrow y[n] = K(x[n] - x[n-1]) - y[n-1] \tag{3.7}$$

for the band-pass filter:

$$H_{bp}(z) = \frac{\omega_n^2 + 2\omega_n^2 z^{-1} + \omega_n^2 z^{-2}}{(K^2 + 2\zeta\omega_n K + \omega_n^2) + (2\omega_n^2 - 2K^2)z^{-1} + (K^2 + 2\zeta\omega_n K + \omega_n^2)z^{-2}}$$

$$\begin{aligned} y_{bp}[n] = &\frac{\omega_n^2}{K^2 + 2\zeta\omega_n K + \omega_n^2}x_{bp}[n] + \frac{2\omega_n^2}{K^2 + 2\zeta\omega_n K + \omega_n^2}x_{bp}[n-1] \\ &+ \frac{\omega_n^2}{K^2 + 2\zeta\omega_n K + \omega_n^2}x_{bp}[n-2] - \frac{2\omega_n^2 - 2K^2}{K^2 + 2\zeta\omega_n K + \omega_n^2}y_{bp}[n-1] \\ &- \frac{K^2 - 2\zeta\omega_n K + \omega_n^2}{K^2 + 2\zeta\omega_n K + \omega_n^2}y_{bp}[n-2] \end{aligned} \tag{3.8}$$

and for the low-pass filter:

$$H_{lp} = \frac{75 \cdot 2\pi(1 + z^{-1})}{(75 \cdot 2\pi - K)z^{-1} + K + 75 \cdot 2\pi}$$

$$y_{lp}[n] = \frac{75 \cdot 2\pi(x_{lp}[n] + x_{lp}[n-1]) - (75 \cdot 2\pi - K)y_{lp}[n-1]}{75 \cdot 2\pi + K} \tag{3.9}$$

The derived formulas were directly translated into code for implementation. The bandpass filter was applied in parallel at 10 different frequencies ranging from 10 Hz to 55 Hz with 5 Hz increments. It's worth noting that a cut-off frequency was strategically selected at 75 Hz, taking into account the potential influence of characteristic axis compression when employing the bilinear transform.

Subsequently, by employing a predetermined threshold, tailored to the specific object and action under consideration, the critical point detection phase could start. Since the parallel filtering of 12 input signals (x and y for 6 sensors) produced 120 elements, information from the filtered data at the 10 different passband filters was fused for each sensor. At this point, if at least one of these values surpassed the predetermined threshold, a critical point was detected. This concept is shown in Fig. 3.4.

Given the discernible presence of tails accompanying the peaks in the filtered data, only the first point of each peak surpassing the threshold was considered as a critical point. This meticulous methodology ensured a precise identification of critical events, optimizing subsequent analysis and decision-making processes.

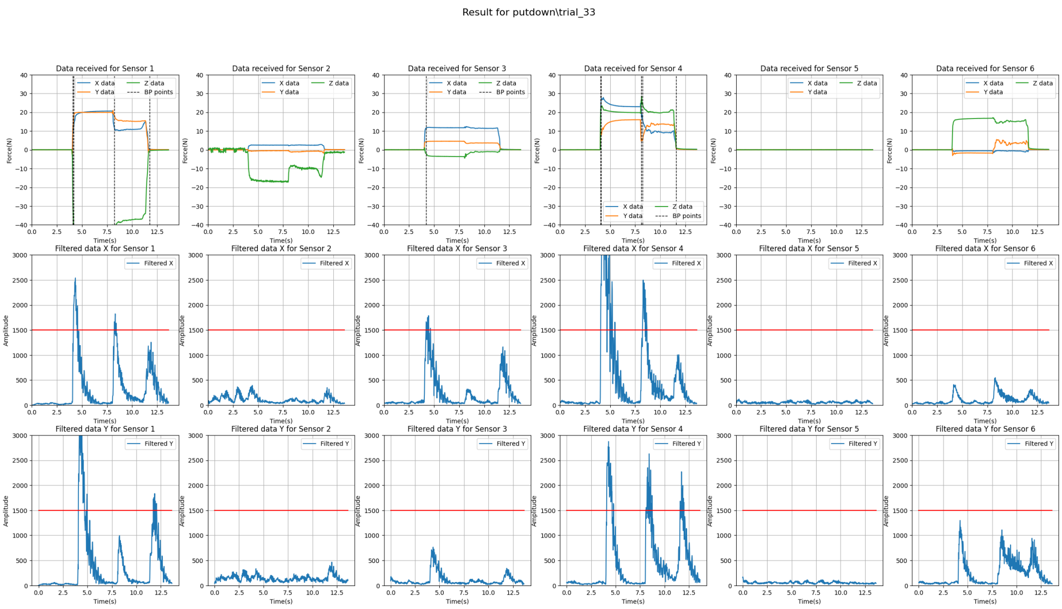


Figure 3.4: Results of bandpass filter analysis. The first row displays the values of x, y, z for each of the six sensors, along with vertical lines indicating critical points. These critical points are derived from the second and third rows of plots, where the filtered signal is plotted alongside the threshold. When the signal surpasses the threshold, a critical point is detected. The results shown are the ones obtained for the full plastic bottle putting-down action.

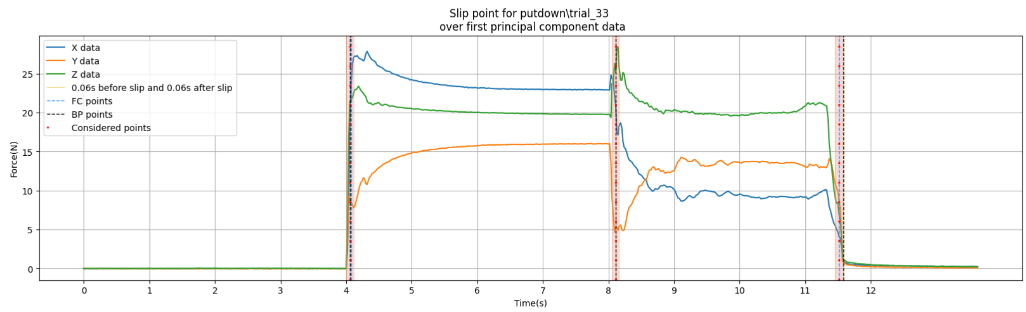


Figure 3.5: Plot showing the values of x, y, and z for one of the six sensors. The results from both the friction cone analysis and the bandpass filter analysis are represented, respectively as light blue vertical dashed lines and black vertical dashed lines. When both approaches detect a critical point, preference is given to the one occurring earlier. Critical points chosen for model training are highlighted with vertical red dotted lines. The red band, referred to as 'window', will be elaborated in Chapter 3.3.

3.2 Safe and Risky Slip

As discussed in Chapter 2.4, there are various methods and sensors available for recognizing slipping. However, this thesis project distinguishes itself through its entirely innovative approach. The primary objective is not only to recognize slips but to categorize them by risk. The aim is to differentiate between risky slips, such as an object falling or suddenly being pulled from someone's grasp, and safe slips, such as normal object handling or positioning.

This differentiation is crucial as it enables a possible introduction of shared control. In the event of a risky slip being anticipated with sufficient warning, the prosthesis would be capable of reacting autonomously, without requiring visual intervention from the user. This could not only enhance human-prosthesis integration but also increase grip safety. The ability to distinguish slips requiring action from those that do not, adds a level of complexity and utility to the system. It means not only predicting an imminent slip but also determining whether intervention is necessary to prevent or manage it appropriately.

The potential practical applications of this project are manifold. For instance, in a household setting, it could be used to prevent accidents in the kitchen or bathroom, where slippery objects are common. In an industrial setting, it could enhance workplace safety by reducing the risk of material damage or worker injuries caused by slips.

To realize this project, four main classes of actions have been considered:

- Nothing: This class includes moments when the robotic hand and sensors are not in contact with anything. The instants have been selected from the extremes of actions and have not been obtained through friction cone or band-pass filter analysis.

- Grasp: This class includes not only the exact moment when the object is grasped but also subsequent instants considered grip-keeping moments. These latter instants have been manually added as they were not recognized by the algorithms, while the former were correctly recognized by the algorithms.

- Safe Slip: This class includes those moments correctly recognized by the friction cone and/or the band-pass filter analysis, which are related to actions of passing, putting-down or releasing objects. These are considered safe slips because they are voluntary moves, and therefore do not require any action.

- Risky Slip: This class includes those moments correctly recognized by the friction cone and/or the band-pass filter analysis, which are related to actions such as falling or pulling objects. These are considered risky slips because they are involuntary moves, and therefore require action to avoid them.

To increase the accuracy and inclusivity of classification, a total of eight classes have been created, considering both light-weight (4 classes) and heavy-weight (4 classes) objects. Given the impossibility of considering all possible object characteristics, this was deemed the best choice. Five objects of varying sizes, weights, shapes, and friction surfaces have been selected for this project, as will be detailed in Chapter 5.2.

3.3 Shared Autonomy Control

Lastly, armed with all the necessary insights, it is possible now to delve into the potential of shared autonomy within the framework of our project.

As previously noted, shared autonomy seeks to empower individuals by augmenting their capabilities through intelligent automation. This collaboration is particularly pertinent in scenarios where human judgment and machine precision converge, as is the case with everyday task execution of our project.

As depicted in the scheme in Figure 3.6, the actions are done according to human intention and thanks to predictive insights coming from the machine learning model the prosthesis will know in real-time the situation, and just in case of risky slip autonomous execution of tasks to avoid it. Of course with the possibility for the user to overwrite it at every moment.

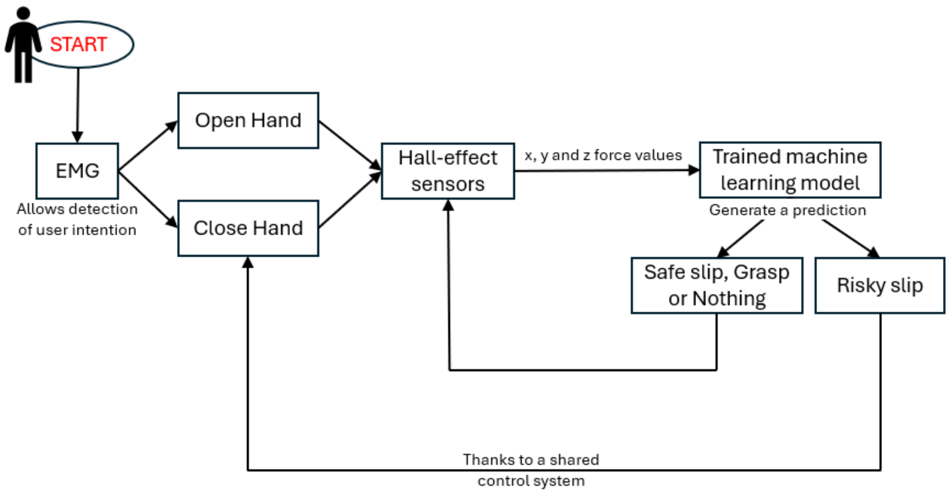


Figure 3.6: Representation of a proposed Shared-Autonomy Control framework for this project.

Illustrated in Figure 3.6, this scheme delineates how actions are executed based on human intention, supported by real-time predictive insights obtained from the machine learning model. This symbiotic relationship ensures that the prosthetic device remains aware of the current situation, executing tasks autonomously in case of potential risks, such as slips. Importantly, users keep the ability to override autonomous actions at any given moment, maintaining ultimate control and agency.

To realize the idea of shared autonomy control in this project, the reliance is on a trained machine learning model. However, before we delve into the specifics of model training and testing, it is crucial to introduce the concept of a 'window'.

In the context of training and testing machine learning models, it is essential to ensure that all data elements have uniform dimensions. To achieve this, a "training window" has been defined to ensure consistency in data dimensionality across different samples. As mentioned in the previous chapters, these windows can represent various actions and will be labelled accordingly. In the specific case of our model, these windows will cover a time interval ranging from 0.10 seconds before the critical point to the critical point itself (included), as illustrated in Figure 3.7. This approach allows us to ensure that this system can recognize different actions and, furthermore, may be capable of predicting them, even anticipating them by a few milliseconds.

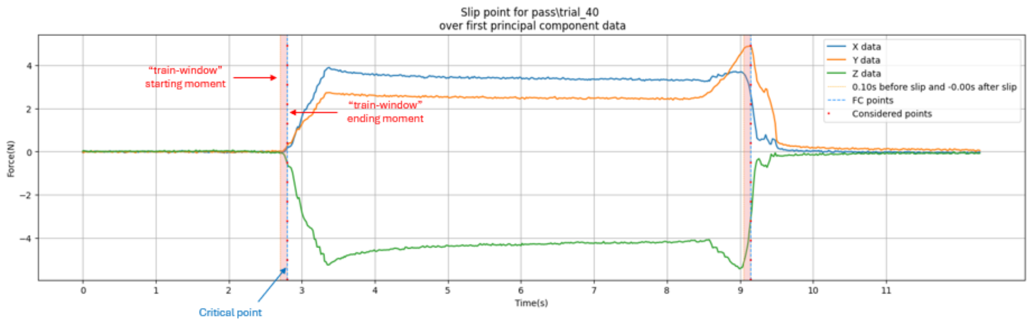


Figure 3.7: Illustration of the window concept in graphical form. Specifically, depicting the basketball passing action, the two critical points derived from the application of the friction cone theory are respectively the grasping action and the passing action. For each of these points, the window will consist of 5 sampling steps before the critical point till the critical point itself.

With the concept of the training window clarified, we are ready to send our data to the algorithm to train the model. Every single sample sent will be represented by matrices of dimensions 18x6, each corresponding to a training-window. However, before proceeding with training, there is still an important step to take.

Despite having collected a large dataset of 5194 elements, all of them cannot be used for training, as the dataset needs to be balanced. Balancing will be done using a random method, ensuring a representative and balanced dataset. This balancing process will ensure that our model is exposed to a variety of cases during training, thereby improving its generalization ability and predictive performance. With a few attempts, it was possible to obtain a trained model with satisfactory results that would be challenging to replicate given the randomness of the element chosen.

Upon successfully training our model, the testing phase will follow, The results of which will be elaborated upon in Chapter 6, providing insights into the practical applicability and effectiveness of this shared autonomy control framework.

Chapter 4

Preliminary Offline Analysis

In the upcoming chapters, it is delineated the step-by-step process used to enhance the offline analysis to its peak effectiveness. This chapter, in particular, will delve into the initial phase, focusing on analyzing pre-recorded data to gain a comprehensive understanding of the topic.

4.1 Data

The primary objective of this phase is to fully understand the nature of the available data in terms of type and structure and to determine the best strategy to maximize the effectiveness of the study.

The first step involved understanding the type of data required to train a machine learning algorithm. To do this, three distinct cases were considered, each characterized by specific peculiarities that allowed exploration of various aspects of the problem under study. However, before examining these cases, it is important to introduce the data used. This dataset was previously collected for an internship-project [59], and included the use of 20 Hall-effect sensors on a robotic hand, the Soft Hand, different from that used in the current work, see chapter 5.1.1. The available data is saved in .csv files, each representing a specific action, listed in Fig. 4.1, and each containing many rows as the sampling steps and 61 columns. The first column represents time, while the other 60 contain force values along the three axes for all 20 sensors.

Task Type	Task	Description	Manipulated Object
Pick and place	1	Picking up a bottle, moving it between two marked positions on a table and putting it down	Bottle
	2	Grasping a weighted ball, moving it between two marked positions on a table, putting it down	Ball
	3	Grasping a book, moving it between two marked positions on a table, putting it down	Book
	4	Grasping a pen, moving it between two marked positions on a table, putting it down	Pen
	5	Grasping a small glass cup, moving it between two marked positions on a table, putting it down	Shot Glass
Functional	6	Grabbing a key from the lock, extracting it, inserting it in the lock, twisting it fully	Key
	7	Grasping a pen from a table, drawing a house, releasing the pen back on the initial position	Pen
	8	Opening a jar fully, lifting the lid, closing it fully	Jar
	9	Picking up a screwdriver from a table, tightening a screw on a wooden block, putting down the screwdriver in the initial position	Screwdriver
	10	Grabbing a full cup from a table, pouring it's contents into another cup, putting both cups down in initial position	Plastic Cup
	11	Grabbing an empty cup from a table, holding it while the other hand pours from a filled cup, putting both cups down in initial position	Plastic Cup
	12	Grasping a bottle from a table, walking across the room and back, putting the bottle down in initial position	Bottle
Social	13	Handshaking	
	14	Grasping and passing a remote to another person	TV Remote
	15	Grasping and passing a weighted plastic cup to another person	Plastic Cup
Stability	16	Grasping a cylinder with unexpected weight from a table and lifting it up	3D Printed Cylinder
	17	Holding on to a remote being pulled by another person until failure	TV Remote

Figure 4.1: Explanation of the tasks executed and the elements manipulated within the existing dataset.

4.2 Application

The identification of various actions was considered most suitable through the utilization of a machine learning model integrated with classification algorithms. However, a meticulous and precise data analysis was necessary to determine the most fitting methodological approach among the initially proposed ones.

As previously mentioned, three distinct cases were investigated, each distinguished by specific characteristics simplifying the exploration of diverse characteristics of the underlying problem. Prior to delving into these cases, it is essential to re-propose the concept of 'window', already explained in Chapter 3.

As noted earlier, it is imperative that all samples, for both training and testing, adhere to the same dimensionality. To ensure this consistency, we employed what we refer to as the 'train-window.' These windows, as detailed in Chapter 3.3, denote various actions. However, in the initial phase of our study, the categorization is simply between 'safe' (0) and 'risky' (1) actions. These windows encompass the critical moments, which, at this juncture, are determined solely through the application of the Friction Cone Theory outlined in Chapter 3.1.1, without the inclusion of the bandpass filter method not yet introduced in our study. Furthermore, the presence of the critical point within the window denotes our focus on slip detection rather than prediction at this early stage. Specifically, an interval of approximately 0.05 milliseconds before and after the critical point will be considered.

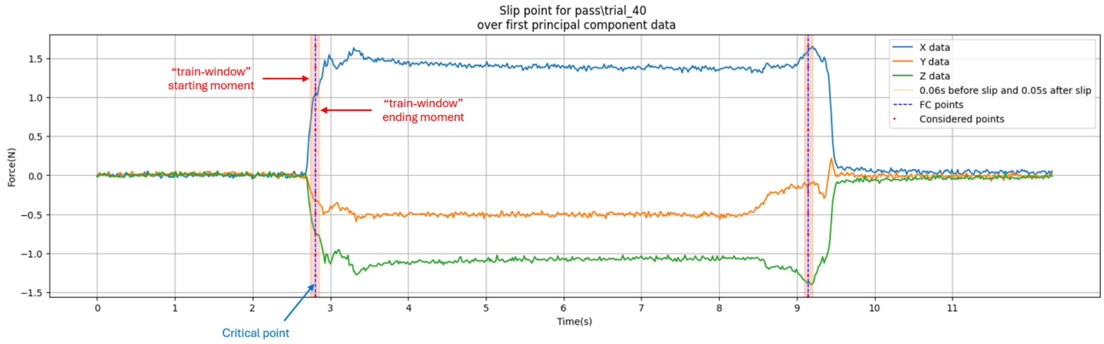


Figure 4.2: Illustration of the window concept in graphical form. Specifically, in depicting the basketball passing action, the two critical points derived from the application of the friction cone theory are respectively the grasping action and the passing action. For each of these points, the window will consist of three sampling steps prior to and three following the critical point.

With a clear understanding of this concept, it is possible to return to the three cases under consideration. The first case entailed the utilization of the x, y, z coordinates of all 20 sensors to train the algorithm, resulting in data of size (7, 60), where 7 represents the window size (including 3 sampling steps before and 3 after the critical point), and 60 comes from considering the 3 coordinates for all 20 sensors.

The second case involved the use of the friction cone not only for recognizing the critical point but also as data for model training, incorporating both f_x/f_z and f_y/f_z. This yielded a training sample of size (7, 40), where 40 represents the ratio between the shear force and the normal force for both shear forces x and y across the 20 sensors.

Lastly, the utilization of the first principal component was contemplated, incorporating the length and direction of the first principal component alongside the z coordinate, producing a training sample of size (7, 3).

The next step was to carefully examine the results obtained by applying various machine learning methods, known for their ability to handle complexity and extract information from large volumes of data. The results are presented in the following table.

	Random Forest Classifier	K-Nearest Neighbors	Naive Bayes	Gradient Boosting Machines	Neural Network
Case 1	86.25%	78.57%	64.64%	82.32%	84.57%
Case 2	48.39%	53.57%	49.11%	50.89%	49.29%
Case 3	45.58%	53.85%	43.27%	46.15%	42.12%

Table 4.1: Average accuracy, firsts considerations. The rows of the table represent the different cases considered (1,2, and 3, previously explained), while the columns display the various machine-learning models taken into consideration. The values within each cell denote the average accuracies obtained for that specific combination of case and machine learning model, calculated over a total of 5 subsequent iterations.

After an initial evaluation, k-nearest Neighbors and Naive Bayes were excluded as they did not demonstrate the ability to provide reliable results. Furthermore, the decision was made to focus attention on case 1, which seemed to be the most accurate, probably because it included the largest number of data fed to the model, which were also in their elementary form, thus offering a greater opportunity for the model to reprocess the data thoroughly and therefore obtain more precise predictions.

At this point, two additional cases emerged to expand the study. The first involved extending or shifting the considered time interval, considering a period of time preceding the critical instant of even just one millisecond. This further investigation aimed to verify if the adopted methodology was able to anticipate and therefore predict slipping, thus adding a temporal prediction element to the analysis process. The second action involved experimenting with a wider variety of classes. This would allow for the exploration and comparison of different contexts and conditions, increasing the richness of the obtained information and the robustness of the conclusions drawn from the study.

Therefore, to expand this preliminary study, a fourth and fifth case were introduced involving greater complexity. The fourth case was essentially identical to the first one, with the only difference being that the considered time window ranged from -7 to -1 sampling steps. The fifth case, on the other hand, in addition to also having a window from 7 to 1 sampling steps before the critical point, introduced a differentiation into four classes instead of two. These classes were identified as "safe slip", "grasp", "not slipping under disturbances", and "slipping". Please note, this more detailed subdivision of situations was intended to capture and distinguish various conditions and events more precisely, allowing for a more thorough and detailed analysis of the collected data; however, these classes are not those that will be used in this study, as it was already explained in Chapter 3.2.

4.3 Results

After further evaluation of the confusion matrices of these two new cases, within Tab. 4.2, the decision was made to adopt the Random Forest Classifier. This choice was motivated by the ability of this classification method to provide reliable and robust results, regardless of the quantity or diversity of the data, thus proving to be an optimal option for this study context.

	Random Forest Classifier	Gradient Boosting Machines	Neural Network
Case 4	84.42%	83.85%	80.58%
Case 5	73.49%	71.16%	67.91%

Table 4.2: Average accuracy, seconds considerations. The rows of the table represent the different cases considered (4 and 5, previously explained), while the columns display the various machine-learning models taken into consideration. The values within each cell denote the average accuracies obtained for that specific combination of case and machine learning model, calculated over a total of 5 iterations.

This phase played a fundamental role in defining the overall methodological approach of this study, providing a solid foundation on which to build and guiding the decisions made in the subsequent phases of the study. Thanks to the satisfactory results obtained, it was possible to proceed to the second phase of the work, explained in Chapter 5.

Chapter 5

Experimental Validation

This Chapter introduces the hand and sensors utilized in this project's development, followed by an overview of the final structure. Calibration of the sensors is discussed before detailing the study's execution, including objects, actions, and methodologies, setting the stage for subsequent phases. Finally, the reliability and precision of this study will be verified.

5.1 Experimental Setup

5.1.1 Michelangelo Hand

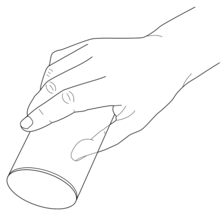
The Michelangelo Hand 8E500 (in Fig. 5.1), developed by Ottobock, is a cutting-edge prosthetic designed to restore numerous functions and aspects of the natural hand. It integrates the innovative Axon-Bus system, a self-contained data transmission system derived from safety-related systems in aviation and automobile industries. This system ensures communication between components, virtually eliminating losses in terms of data transmission, speed and functionality, offering users increased safety and reliability while reducing sensitivity to external interference — a crucial aspect for research involving reaction time analysis.

Figure 5.1: Michelangelo Hand 8E500, developed by Ottobock. [60]

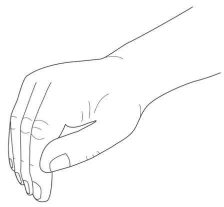
The Michelangelo Hand is a multi-articulated hand-wrist system that utilizes standard myoelectric control via two electrodes, capturing forearm muscle contractions for flexion and extension movements. Weighing approximately 150 g (excluding the Axon Rotation adapter and cosmetic glove), the hand offers a wide opening width of 120 mm and can withstand maximum loads of up to 10 kg over actively operated fingers (index and middle) in an open position, and 20 kg when closed.

It offers three main grip modes - opposition, lateral, and neutral - each with varying maximal grip strengths (70N, 60N, and 15N respectively), for a total of seven grip options.

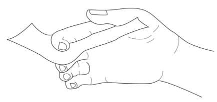
(a) Natural Mode



(b) Lateral Power Grip



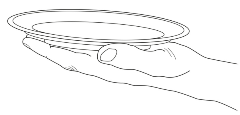
(c) Lateral Pinch



(d) Fingers Abduction/Adduction

(e) Tripod Pinch

(f) Opposition Power Grip



(g) Open Palm

Figure 5.2: The seven grip options the Michelangelo Hand can reproduce. [60]

Providing users with a natural appearance and ease of use, the hand automatically returns to a relaxed, neutral position 5.2a upon relaxation of the myosignal.

The hand's main drive is responsible for the gripping movements and force. Actively driven elements are the thumb, index finger and middle finger while the ring finger and little finger passively follow the other fingers. This design allows for lateral power grip 5.2b and lateral pinch 5.2c, with the thumb moving laterally toward the index finger to facilitate various holding positions for objects of different shapes and dimensions.

The thumb drive enables electronic positioning, allowing for diverse grip configurations from a wide open palm 5.2g, by rotating the thumb outward, to opposition power grip 5.2f and tripod pinch 5.2e, thereby enhancing the hand's adaptability in grasping objects. Additionally, finger abduction/adduction 5.2d allows for the secure clamping of flat, thin objects between fingertips by closing the hand.

In conclusion, the Michelangelo Hand stands as a remarkable achievement in prosthetic design, offering users advanced functionality and control while providing researchers with a reliable platform for studying human-machine interaction with an ergonomic design and versatile grip options.

5.1.2 Hall-effect sensors

If a current-carrying conductor crosses a magnetic field, a remarkable phenomenon known as the Hall-effect occurs.

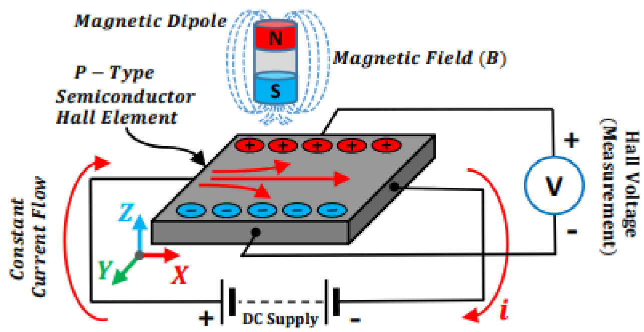
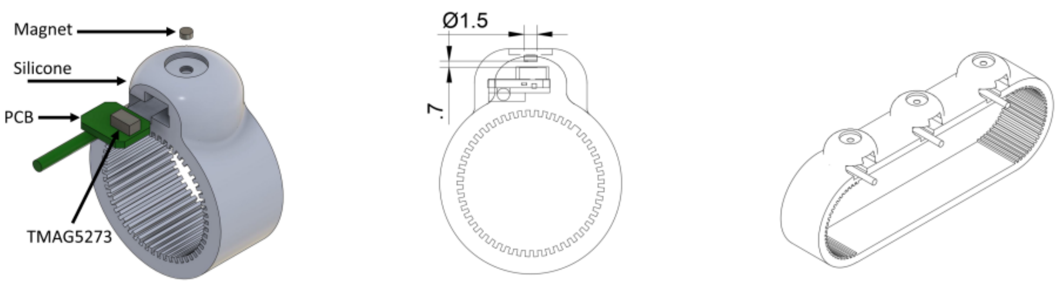


Figure 5.3: Single-axis Hall-effect sensor principle [61]. The output signal from a Hall-effect sensor is a function of the magnetic field density around the device.

When the conductor is subjected to a magnetic field perpendicular to the current flow, a specific voltage, termed the Hall voltage, emerges across the conductor. This voltage manifests due to the influence of the Lorentz force exerted on the current by the magnetic field, disrupting the uniform current distribution and generating a potential difference across the conductor. This effect, a fundamental principle underpinning the functionality of Hall effect sensors, arises from the interaction between the flowing current and the magnetic field. [62]



(a) Exploded view of the implemented ring sensor. This includes a magnet, silicone cover, PCB and Hall sensor (TMAG5273).

(b) Image showing the placement of the magnet, located 0.7 mm above the Hall sensor.

(c) Band design, used for the palm areas. It integrates the same components of the ring design.

Figure 5.4: Hall-effect sensors design. [42]

Each force sensor features a TMAG5273 magnetic sensor (Texas Instruments, USA), which includes three independent Hall-effect sensing elements, enabling the measurement of magnetic flux along three axes. The TMAG5273 sensor was configured with a range of ±40 mT for the x and y axes and a range of ±80 mT for the z axis. To ensure accurate readings, a small printed circuit board (PCB) was designed following the sensor's datasheet specifications. Temperature compensation was set at 0.12%/°C, aligning with the temperature coefficient of neodymium magnets. The update rate for the sensor was optimized to achieve the best signal-to-noise ratio (SNR), averaging every 32 samples. An experimental comparison led to the selection of silicone with a shore hardness of 35A and an N45 disc magnet (diameter 1.5 mm; height 0.5 mm) with axial magnetization. [42]

As explained before, by applying some force on the sensor the relative position between the magnet and the integrated circuit (IC) will change, resulting in a change in the magnetic field and thus a digital signal output.

Instead of conventional gloves, customizable and scalable silicone bands are used, to accommodate a wider range of subjects. These silicone bands were designed in two variations: a ring shape and a band shape; and, a ribbed texture was incorporated into the design to enhance friction between the silicone ring and the finger, ensuring stable sensor placement.

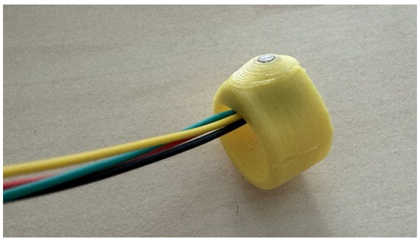


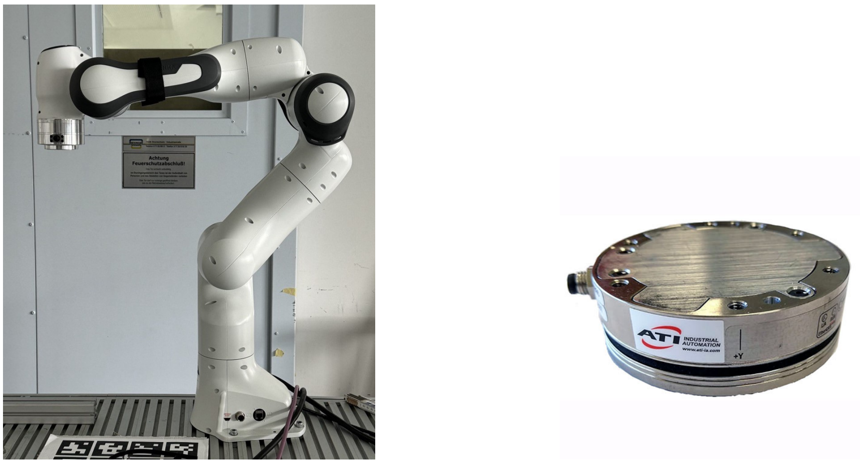
Figure 5.5: Updated version of the Hall-effect sensor utilized for this research investigation. This enhanced sensor model has been specifically designed and implemented to mimic the characteristics of a real finger, highlighting reduced protrusion for enhanced functionality and usability.

In this study, only the ring-shaped variant, designed to accommodate a single sensor and tailored to fit seamlessly on any finger, was employed. This design minimizes both translation and rotation of the force sensors, thereby reducing potential errors in shear force measurements. However, we opted for a slightly modified version of the sensor depicted in Figure 5.4a, as showcased in Figure 5.5. This variant features a shape more closely resembling a real finger and exhibits less protrusion, allowing for a more embedded integration into prosthetic devices.

5.1.3 Sensors' Calibration

To ensure accurate calibration accounting for manufacturing tolerances (manual fabrication) and potential cross-talk between axes, each sensor underwent individual calibration. as showed in figure 5.6, this calibration process was facilitated by a Panda robotic arm (Franka Emika, Germany) paired with an FT AXIA 80-M20 6 DoF sensor (ATI Industrial Automation, USA), providing ground truth data.

The calibration procedure involved three main phases. Initially, normal force increments were applied, ranging from 1 N to 14 N, with a 3-second holding time per step. Subsequently, forces were applied at 90° and 45° angles, using two different levels of normal force (5 N and 14 N) in the second and third phases.



(a) Franka robot.

(b) FT AXIA 80-M20 6 DoF sensor. [Photo from ATI Industrial Automation official page]

Figure 5.6: Setup for the calibration process.

5.1.4 Mechanical and Sensor Integration

Hand support

In this project, hand actions were chosen to be recognized while keeping the hand in a fixed position. This approach simplified recognition since all forces remained consistently directed. To facilitate this elaboration, a support structure was designed in SolidWorks, and then 3D-printed, to securely hold the hand in place (Fig. 5.7a). This support would then be attached to a profile on one side and to the table on the other.

(a) SolidWorks view of the support.

(b) Image of the 3D-printed support. ABS and QSR support materials were used.

(c) Final structure. The hand connected in this way was then used to record all trials.

Figure 5.7: Hand-support design.

The profile was 3D-printed, Fig. 5.7b, using a Stratasys F170 printer and then attached to both the hand and the profile. Everything was tested to ensure that the hand could be held securely in place during the experiments, as shown in Fig. 5.7c. Finally, the support structure proved to be effective and reliable, allowing the project to move forward.

Sensors' Placement

In the final design, each fingertip was equipped with a single sensor, except for the thumb, where two sensors were placed to provide more accurate force measurements, as the thumb is often the point of contact in most tasks. Sensors' Placement is shown in the following Fig. 5.8.

Figure 5.8: Sensors' placement in the final design of the robotic hand. Each fingertip is equipped with a single sensor, except for the thumb, where two sensors are positioned to enhance force measurement accuracy, considering the thumb's frequent contact in various tasks.

This decision was based on findings from previous studies, which presented heatmaps illustrating the average contribution of normal and shear forces across various trials and subjects (Fig. 5.9). Additionally, it was determined that using six sensors in total would be optimal to avoid too low sampling time.

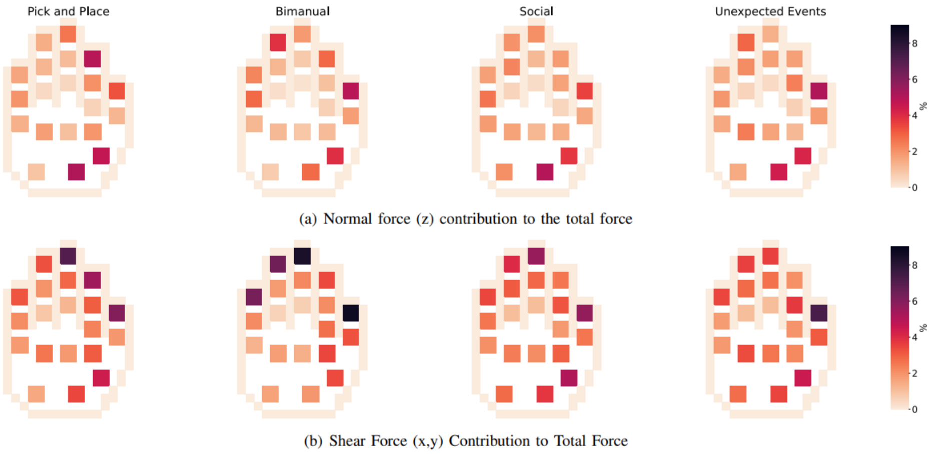


Figure 5.9: Heatmaps of the average force contribution of sensors placed on a hand. Panel (a) shows the normal force contribution of each sensor to the total force. Panel (b) shows the average contribution of shear forces (sum of x and y forces) of each sensor to the total force. [42]

Altogether, this placement was proven to be the most suitable to provide a comprehensive view of the finger's movement.

5.2 Preliminary Implementation of the Proposed Method

After confirming the feasibility of the study in Chapter 4, the focus moved to the implementation and refinement of the selected method, as well as the development of a new dataset that was consistent with the scope of the study both in terms of selected movements and type of robotic hand used, the Michelangelo Hand, previously discussed in Chapter 5.1.1.

Firstly, the objects to be used were chosen: three balls of different weights, an empty plastic bottle and a full one, see Fig. 5.10. A description, with also dimensions and weights of these objects, is reported in Table 5.1.

(a) Soft-ball **(b)** Empty plastic bottle **(c)** Basket-ball

(d) Hard-ball **(e)** Full plastic bottle

Figure 5.10: The 5 objects used for this study. (Not in scale)

	Description	Weight
Soft-ball	Soft and squeezable plastic spherical ball with 69 mm diameter	15.1 g
Empty bottle	Empty plastic bottle with 63 mm diameter (at the gripping point) and 230mm high	21.2 g
Basket-ball	Plastic spherical ball with 60 mm diameter	83.1 g
Hard-ball	Hard plastic spherical ball (3d printed) with 60 mm diameter	197.9 g
Full bottle	Full plastic bottle with 63 mm diameter (at the gripping point) and 230mm high	517.3 g

Table 5.1: This table provides a comprehensive overview of the objects utilized in the project, including their descriptions, dimensions, and weights. The dimensions are specified in millimeters (mm), while the weights are presented in grams (g). Note: Dimensions and weights are approximate and may vary slightly.

Secondly, some time was spent identifying key actions that would contribute to the understanding of the problem, and, then, the recording and analysis of the actions were carried out, the details of which will be illustrated in Table 5.2.

Object used	Task	Description	Quantity
Soft-ball	1	The Michelangelo hand is vertical on the table in a resting state. The user close the hand grasping the soft-ball handled by the collaborator, then the ladder slowly removes it from the front.	40
	2	The Michelangelo hand is vertical on the table in a resting state. The user close the hand grasping the soft-ball handled by the collaborator, then the ladder quickly removes it from the front.	40
	3	The Michelangelo hand is in a resting state, kept horizontally by the collaborator. The user close the hand grasping the soft-ball from the table and putting it in the air, then the ladder smashes the ball on the table and finally release it.	40
	4	The Michelangelo hand is vertical on the table in a resting state. The user close the hand grasping the empty plastic bottle, which is still held up by the collaborator, then the ladder slowly removes it from the front.	40
	5	The Michelangelo hand is vertical on the table in a resting state. The user close the hand grasping the empty plastic bottle, which is still held up by the collaborator, then the ladder quickly removes it from the front.	40

Table 5.2 continued from previous page

Empty plastic bottle	6	The Michelangelo hand is in a resting state, kept horizontally by the collaborator. The user close the hand grasping the empty plastic bottle from the table and putting it in the air, then the ladder smashes the bottle to the table and finally release it.	40
Basket-ball	7	The Michelangelo hand is vertical on the table in a resting state. The user close the hand grasping the basket-ball handled by the collaborator, then the ladder slowly removes it from the front.	40
	8	The Michelangelo hand is vertical on the table in a resting state. The user close the hand grasping the basket-ball handled by the collaborator, then the ladder quickly removes it from the front.	40
	9	The Michelangelo hand is in a resting state, kept horizontally by the collaborator. The user closes the hand grasping the basket-ball from the table and putting it in the air, then the ladder smashes the ball on the table and finally release it.	40
	10	The Michelangelo hand is vertical on the table in a resting state. The user close the hand grasping the hard-ball handled by the collaborator, then the ladder slowly removes it from the front.	40
	11	The Michelangelo hand is vertical on the table in a resting state. The user close the hand grasping the hard-ball handled by the collaborator and they continue holding it up, then the ladder quickly removes their hand and the ball should fall.	40

Table 5.2 continued from previous page

Hard-ball	12	The Michelangelo hand is in a resting state, kept horizontally by the collaborator. The user close the hand grasping the hard-ball from the table putting int in the air, then the ladder smashes the ball on the table and finally release it.	40
Full plastic bottle	13	The Michelangelo hand is vertical on the table in a resting state. The user close the hand grasping the full plastic bottle, which is still held up by the collaborator, then the ladder slowly removes it from the front.	40
	14	The Michelangelo hand is in a resting state, kept horizontally by the collaborator. The user close the hand grasping the full plastic bottle from the table and keeping it in the air, then the user opens slowly the hand to make the bottle slowly fall.	40
	15	The Michelangelo hand is in a resting state, kept horizontally by the collaborator. The user close the hand grasping the full plastic bottle from the table and putting it in the air, then the ladder smashes the bottle to the table and finally release it.	40

Table 5.2: This table provides a comprehensive overview of all actions recorded during the initial phase of this project realization. Each action is listed along with relevant details such as description, quantity and object used. In this table, even if they are the same person, the one using the EMG to control the prosthesis is referred to as user and the one interacting with the prosthesis is the collaborator.

At this point, a specific approach was outlined to identify critical points. In addition to the friction cone already mentioned, see Chapter 3.1.1, it was decided to use a combination of techniques, so the 'bandpass filter' was added, following what was explained in Chapter 3.1.2, in order to achieve greater precision and reliability in data analysis.

Once all these technical aspects were defined, the dilemma of determining the number and type of classes to consider in this model was addressed. This decision was influenced by a series of factors, including the complexity of the problem, the availability of data, and the computational resources at our disposal. A pragmatic approach was adopted, seeking a balance between the complexity of the model and its predictive capability. For this reason, various implementations and optimizations were experimented to improve the performance of the model.

Below are some ideas that were followed for various implementations, the one that will be chosen, as already explained in Chapter 3.2, is the last one.

```python
if prediction == 0:
    print("SAFE")
elif prediction == 1:
    print("RISKY")
elif prediction == 2:
    print("NOTHING")
else:
    print("No prediction")
```

```python
if prediction == 0:
    print("PASSING")
elif prediction == 1:
    print("GRASP")
elif prediction == 2:
    print("PULLING AWAY")
elif prediction == 3:
    print("FALLING")
elif prediction == 4:
    print("PUTTING DOWN")
elif prediction == 5:
    print("SAFE RELEASING")
elif prediction == 6:
    print("RISKY RELEASING")
else:
    print("No prediction")
```

```python
if prediction == 0:
    print("PASSING")
elif prediction == 1:
    print("GRASP")
elif prediction == 2:
    print("PULLING AWAY")
elif prediction == 3:
    print("FALLING")
elif prediction == 4:
    print("PUTTING DOWN")
elif prediction == 5:
    print("SAFE RELEASING")
elif prediction == 6:
    print("RISKY RELEASING")
elif prediction == 7:
    print("NOTHING")
else:
    print("No prediction")
```

```python
if prediction == 0:
    print("NOTHING Light")
elif prediction == 1:
    print("NOTHING Heavy")
elif prediction == 2:
    print("GRASP Light")
elif prediction == 3:
    print("GRASP Heavy")
elif prediction == 4:
    print("SAFE SLIP Light")
elif prediction == 5:
    print("SAFE SLIP Heavy")
elif prediction == 6:
    print("RISKY SLIP Light")
elif prediction == 7:
    print("RISKY SLIP Heavy")
else:
    print("No prediction")
```

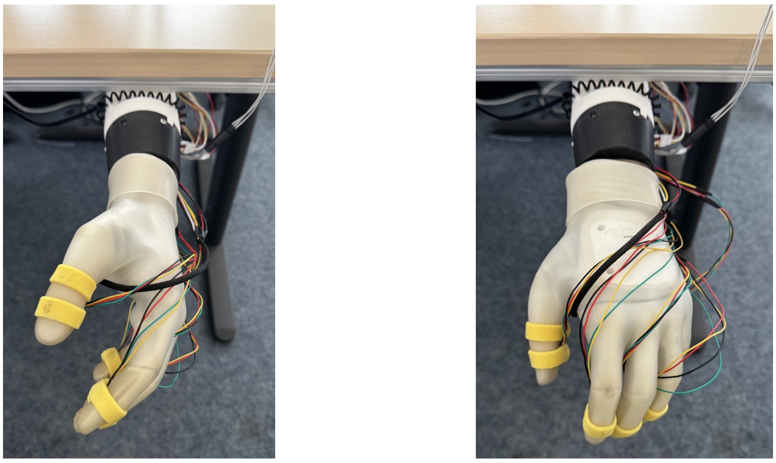
Figure 5.11: Various routes that were followed to understand which and how many classes suited better the study.

In conclusion, this preliminary implementation of the proposed method was characterized by intensive implementation and refinement work, during which a specific methodological approach was developed and optimized to address the challenges of this research.

5.3 Final Implementation of the Proposed Method

In the final phase of this study, the model was refined and optimized through a series of iterations and improvements.

A key objective of this final implementation was to ensure that the collected data were accurate and homogeneous. To this end, the action recording process ensured that all actions were performed in the same position and under the same conditions, ensuring consistency and comparability of the data without moving the hand during and/or between recordings, thereby avoiding interference or different values on the three axes of the collected force. To achieve this, a support for the hand was designed using SolidWorks and then 3D printed, more details can be found in Chapter 5.1.4. In Fig. 5.12a is displayed the primary hand position for various actions such as passing, pulling, and falling. However, it's notable that for the action of putting-down objects, specifically for the balls (smaller items), a different hand position is required, showed in Fig. 5.12b. This necessity arises due to a structural constraint: the smaller size of the ball compared to the hand's one prevents the ball from making contact with a surface when the hand is positioned on its side, avoiding so a good recording of the putting-down action. In this scenario, the lower part of the hand obstructs the ball by touching the surface, rendering the typical hand position ineffective. Interestingly, it was found that this adjustment did not adversely affect the performance of the algorithm.



(a) Side-position of the Michelangelo hand, used for almost all the actions.

(b) Down-position of the Michelangelo hand, specifically used for the action of putting-down.

Figure 5.12: Two distinct hand positions used for the recording of the actions.

Thereafter, a series of actions, detailed in Table 5.3, were recorded and their performances were carefully examined to identify possible areas of improvement. During this process, a thorough analysis of the results obtained was conducted, focusing particularly on evaluating the confusion matrix. After this, some additional actions were recorded to assess the impact that greater data could have on the final outcome.

Note: all actions will be recorded with the Michelangelo Hand in the position shown in Fig. 5.12a. However, regarding the put-down of the soft-ball, basket-ball, and hard-ball, there is an exception, and the Michelangelo Hand will be positioned as shown in Fig. 5.12b.

Object used	**Task**	**Description**	**Initial quantity**	**New quantity**
	1	The user closes the hand grasping the soft-ball handled by the collaborator, then the ladder slowly removes it.	40	60
	2	The user closes the hand grasping the soft-ball handled by the collaborator, then the ladder quickly removes it.	40	80
Soft-ball	3	The user closes the hand grasping the soft-ball, then the ladder smashes the ball on a moving box and then they release it.	40	60
	4	The user closes the hand grasping the empty plastic bottle handled by the collaborator, then the ladder slowly removes it.	40	60
	5	The user closes the hand grasping the empty plastic bottle handled by the collaborator, then the ladder quickly removes it.	40	80
Empty plastic bottle	6	The user closes the hand grasping the empty plastic bottle, then the ladder smashes the bottle on a moving box and then they release it.	40	60
	7	The user closes the hand grasping the basket-ball handled by the collaborator, then the ladder slowly removes it.	40	60
	8	The user closes the hand grasping the basket-ball handled by the collaborator, then the ladder quickly removes it.	40	80

Table 5.3 continued from previous page

Basket-ball	9	The user closes the hand grasping the basket-ball, then the ladder smashes the ball on a moving box and then they release it.	40	60
Hard-ball	10	The user closes the hand grasping the hard-ball handled by the collaborator, then the ladder slowly removes it.	40	60
	11	The user closes the hand grasping the hard-ball handled by the collaborator who continues keeping the ball, then the ladder quickly removes their hand and the ball should fall.	40	80
	12	The user closes the hand grasping the hard-ball, then the ladder smashes the ball on a moving box and then they release it.	40	60
Full plastic bottle	13	The user closes the hand grasping the empty plastic bottle handled by the collaborator, then the ladder slowly removes it.	40	60
	14	The user closes the hand grasping the full plastic bottle handled by the collaborator who continues keeping the bottle, then the ladder quickly removes their hand and the bottle should fall.	40	80
	15	The user closes the hand grasping the empty plastic bottle, then the ladder smashes the bottle on a moving box and then they release it.	40	60

Table 5.3: This table provides a comprehensive overview of all actions recorded during the final phase of this project realization. Each action is listed along with relevant details such as description, quantity and object used. In this table, even if they are the same person, the one using the EMG to control the prosthesis is referred to as user and the one interacting with the prosthesis is the collaborator.

In conclusion, this phase represented the culmination of the offline part of this study. During this period, considerable effort was dedicated to refining and optimizing the model through a series of iterations and little improvements. This phase enabled a comprehensive evaluation of the performance of this methodological approach, thus providing a solid foundation for addressing the more complex part of the study: the recognition and, potentially, real-time prediction of actions performed by prostheses' users. It is important to note that while slightly lower results are expected, compared to those obtained during the offline study, this is a natural consequence of the additional challenges present in everyday life.

5.4 Human Study

This chapter focuses on the crucial phase of ensuring the reliability and accuracy of this work. The validation of the model involves the direct interaction of two individuals with the robotic hand. Specifically, the author of this thesis project will manipulate the prosthetic hand using myoelectric sensors, mimicking the actions of a prosthesis user, while the participants engage with it.

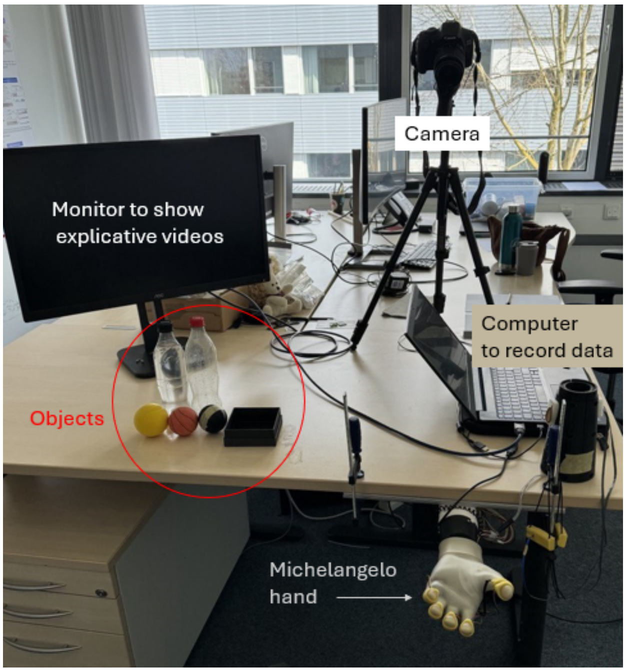


Figure 5.13: Setup for Human Study - Validation Experiment.

The experiment's procedure commenced with an explanation of the study's objectives, followed by the collection of personal information and obtaining signed consent forms, including consent for video recording. The experimental procedure remained consistent for each participant: they were presented with a 30-second video demonstrating the author's interaction with the prosthesis to familiarize them with the actions. Subsequently, they were tasked with replicating the actions demonstrated in the video three times each, aiming for as close a resemblance as possible.

The outcomes of their interactions will be subsequently analyzed in Chapters 6.3.1 and 6.3.2.

Participant 1 is a 25-year-old male with no prior experience in interacting with prostheses. Participant 2, aged 24, also lacks prior experience with prostheses. The inclusion of participants with no prior exposure to prosthetic devices aims to simulate a scenario where users approach the robotic hand with fresh perspectives, providing authentic feedback on its usability and effectiveness. As will be explained in Chapter 7, their participation offers an additional perspective on the ease of integration and functionality of this model in real-world scenarios.

Figure 5.14: Participant 1 and participant 2 interacting with the prosthesis.

The validation process adhered to a meticulously designed protocol outlined in the forthcoming table 5.4. This protocol ensures consistency and rigour in the assessment of the model's performance.

Before the study	- Prepare excel-file with a list of participants. In the same excel-file you will register participant demographics (name, gender, age, previous experience, etc) - Print informed consent - Prepare room and tripod			
Phase	**Time**	**Description**	**Material**	**ToDo**
0 – Preparation	2 min	Explain the objective of the study, get personal info, get consent form signed and ask consent for videos	Consent Form and pen	**Explain what the goal of the experiment is**
	1 min	Allow questions		
1 – Mounting	1 min	Put on EMG-sensors		
2 – Training with basketball 1	30 sec	Show video of passing action with basketball	Video Explanation	**Start recording video**
	30 sec	Allow questions		
3 - Testing with basketball 1	2 min	Record the action 3 times	Basketball	
4 – Training with basketball 2	30 sec	Show video of pulling action with basketball	Video Explanation	
	30 sec	Allow questions		
5 - Testing with basketball 2	2 min	Record the action 3 times	Basketball	
6 – Training with basketball 3	30 sec	Show video of putting-down action with basketball	Video Explanation	
	30 sec	Allow questions		
7 - Testing with basketball 3	2 min	Record the action 3 times	Basketball + box	**Stop recording video**

Table 5.4 continued from previous page

8 – Training with softball 1	30 sec	Show video of passing action with softball	Video Explanation	**Start recording video**
	30 sec	Allow questions		
9 - Testing with softball 1	2 min	Record the action 3 times	Softball	
10 – Training with softball 2	30 sec	Show video of pulling action with softball	Video Explanation	
	30 sec	Allow questions		
11 - Testing with softball 2	2 min	Record the action 3 times	Softball	
12 – Training with softball 3	30 sec	Show video of putting-down action with softball	Video Explanation	
	30 sec	Allow questions		
13 - Testing with softball 4	2 min	Record the action 3 times	Softball + box	**Stop recording video**
14 – Training with empty plastic bottle 1	30 sec	Show video of passing action with empty plastic bottle	Video Explanation	**Start recording video**
	30 sec	Allow questions		
15 - Testing with empty plastic bottle 1	2 min	Record the action 3 times	Empty plastic bottle	
16 – Training with empty plastic bottle 2	30 sec	Show video of pulling action with empty plastic bottle	Video Explanation	
	30 sec	Allow questions		
17 - Testing with empty plastic bottle 2	2 min	Record the action 3 times	Empty plastic bottle	

Table 5.4 continued from previous page

18 – Training with empty plastic bottle 3	30 sec	Show video of putting-down action with empty plastic bottle	Video Explanation	
	30 sec	Allow questions		
19 - Testing with empty plastic bottle 1	2 min	Record the action 3 times	Empty plastic bottle + box	**Stop recording video**
20 – Training with hardball 1	30 sec	Show video of passing action with hardball	Video Explanation	**Start recording video**
	30 sec	Allow questions		
21 - Testing with hardball 1	2 min	Record the action 3 times	Hardball	
22 – Training with hardball 2	30 sec	Show video of pulling action with hardball	Video Explanation	
	30 sec	Allow questions		
23 - Testing with hardball 2	2 min	Record the action 3 times	Hardball	
24 – Training with hardball 3	30 sec	Show video of putting-down action with hardball	Video Explanation	
	30 sec	Allow questions		
25 - Testing with hardball 3	2 min	Record the action 3 times	Hardball + box	**Stop recording video**
26 – Training with full plastic bottle 1	30 sec	Show video of passing action with full plastic bottle	Video Explanation	**Start recording video**
	30 sec	Allow questions		
27 - Testing with full plastic bottle 1	2 min	Record the action 3 times	Full plastic bottle	
28 – Training with full plastic bottle 2	30 sec	Show video of pulling action with full plastic bottle	Video Explanation	

Table 5.4 continued from previous page

	30 sec	Allow questions		
29 - Testing with full plastic bottle 2	2 min	Record the action 3 times	Full plastic bottle	
30 – Training with full plastic bottle 3	30 sec	Show video of putting-down action with full plastic bottle	Video Explanation	
	30 sec	Allow questions		
31 - Testing with full plastic bottle 3	2 min	Record the action 3 times	Full plastic bottle + box	**Stop recording video**

Total time	49 minutes

Table 5.4: Protocol for conducting Human study. This table outlines the protocol followed to conduct the study, each step describes the specific procedure or action taken during the experimental process with the timing and the objects needed.

Chapter 6

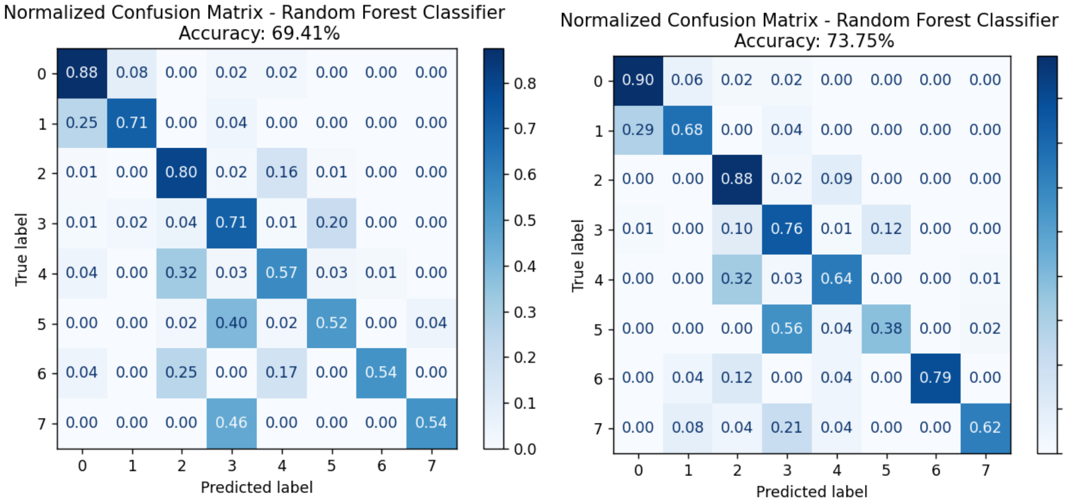
Results

This chapter presents the outcomes of the three main phases of the project. Initially, the offline analysis reveals findings from two distinct cases differing in the quantity of recorded data. Subsequently, the online analysis showcases results obtained from testing the trained machine learning model, considering new actions obtained by making interact with the prosthesis the same person who recorded the actions used to train the model. Finally, outcomes from the human study, involving two participants attempting to replicate the same action, are discussed.

6.1 Offline Analysis

An element of great relevance of this offline phase was the assessment of result variability, which was explored through a careful analysis of the confusion matrix. As highlighted in Figure 6.1, significant variations were observed in some classes considering some iterations of the model training.

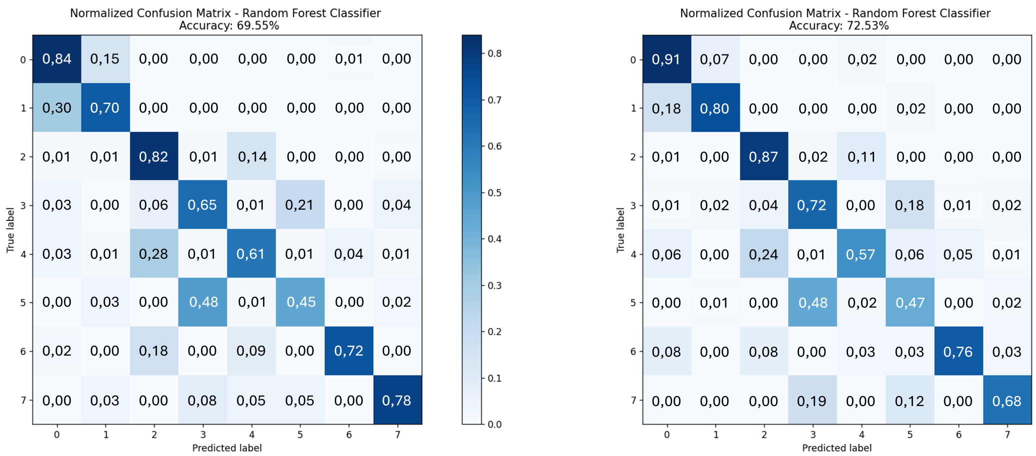
Notably, while the majority of the classes exhibited variability of not over 10%, which is still acceptable; Class 6 (Risky Slip Light) demonstrated a substantial change of approximately 25%, revealing significant instabilities. This phenomenon was attributed to the small size of the dataset produced, which was not sufficiently large to ensure stable results. It is important to emphasize that although machine learning techniques require extensive datasets, our data availability was time-limited.



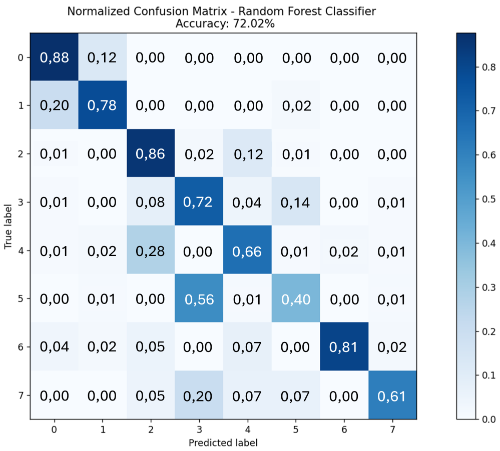
(a) **(b)**

Figure 6.1: Confusion matrices from the same reduced dataset.

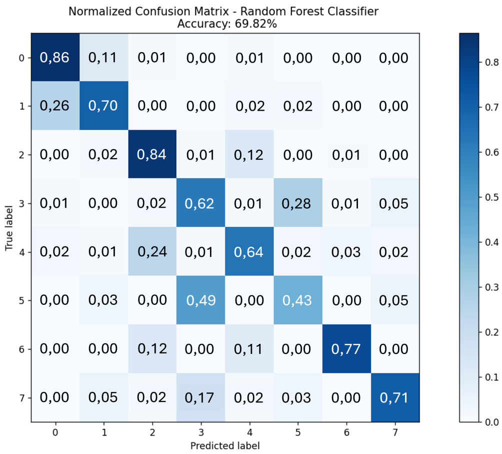
Having recognized that result variability is an intrinsic aspect of this small study, it was decided to expand the dataset by adding additional experiments of the already chosen actions. This approach allowed to assess the impact that greater data could have on the final outcome and explore possible strategies to reduce the observed variability. The resulting confusion matrices are shown in Fig. 6.3.



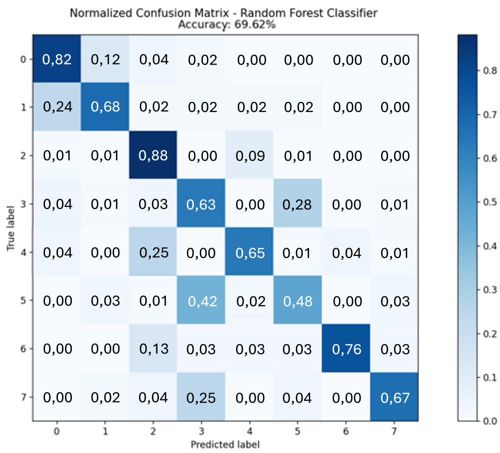
(a) **(b)**



(a)



(b)



(c)

Figure 6.3: Confusion matrices generated from the same full dataset and obtained by 5 subsequent iterations. Each matrix illustrates the performance of the model across different iterations, providing insights into its consistency over time, which got better after expanding the dataset.

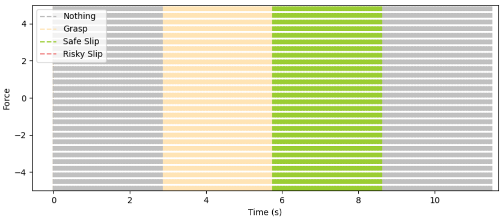
As evident from the matrices, the variability of almost all classes across 5 trials is now within 10%.

6.2 Online Analysis

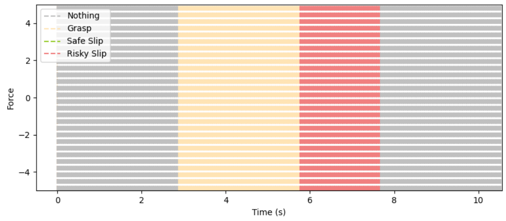
6.2.1 Ground Truth

Prior to showcasing the results derived from testing with the trained machine-learning model, as elaborated in Chapter 3, it is essential to outline the anticipated outcomes, serving as the ground truth for each action under consideration, i.e. pass, pull, fall, and putdown. These will be depicted in Figure 6.4.

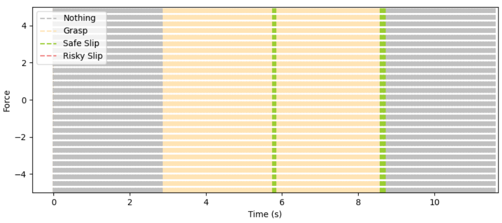
All recorded actions are detailed in Table 5.3. For the action termed ’pass’, the expected outcome includes nothing initially and at the end, common to every action, along with a yellow segment denoting grasp and a green segment indicating safe slip, specifically for the passing action. Regarding ’pull’ or ’fall’ actions, an initial yellow phase representing grasp is followed by a red phase corresponding to risky slip. As for the action of ’put-down’, as previously explained in Chapter 5.3, it entails 2 yellow intervals (representing grasp) and 2 green intervals (indicating safe slip), alternating between each other. This depicts the initial grasp followed by the putdown, referred to as safe slip, followed again by grasping due to the ongoing nature of the action. Finally, it concludes with the release, also considered a safe slip. Note: Of course, these images are figurative and the dimension of these intervals will depend on the actual actions.



(a) Pass



(b) Pull/Fall

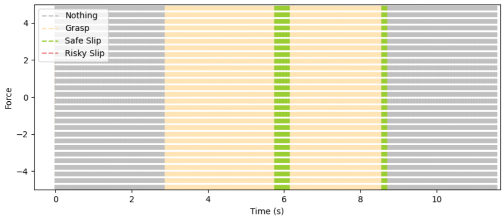


(c) Put-down

(d) Zoomed legend

Figure 6.4: Ground truths of the action recorded.

A singular case occurs in the ’pass’ action when dealing with a full plastic bottle. In this scenario, besides the difficulty of replicating the object passage while holding the hand slightly wider (as the heavy object tends to slip, simulating more of a ’fall’ action than a ’pass’), it was decided to replicate the actual passing of a heavy object in everyday life. This involves handing the object to another person and then releasing it when it is felt that the other person has actually taken it. It was attempted to replicate this dynamic by gently lifting the weight of the bottle during the intermediate phase before releasing it. In this way, the ground truth graph is very similar to that of 'put-down', and to be precise, it can be seen in Figure 6.5.



(a) Put-down

(b) Zoomed legend

Figure 6.5: Pass for plastic bottle.

Please be advised that this legend will remain applicable throughout the entire analysis of results, even if not explicitly displayed.

6.2.2 Object 1 - Soft-ball

Pass

In figures 6.6 and 6.7, it is possible to observe the results obtained from applying the trained machine learning model to the passing actions using the soft-ball. Image 6.6 shows an excellent result, nearly identical to the ground truth depicted in 6.4a. However, in image 6.7, a discrepancy is evident. Here, it can be observed that if a slight jerk occurs at the beginning or end of the passing action, the algorithm categorizes it wrongly as a risky situation.

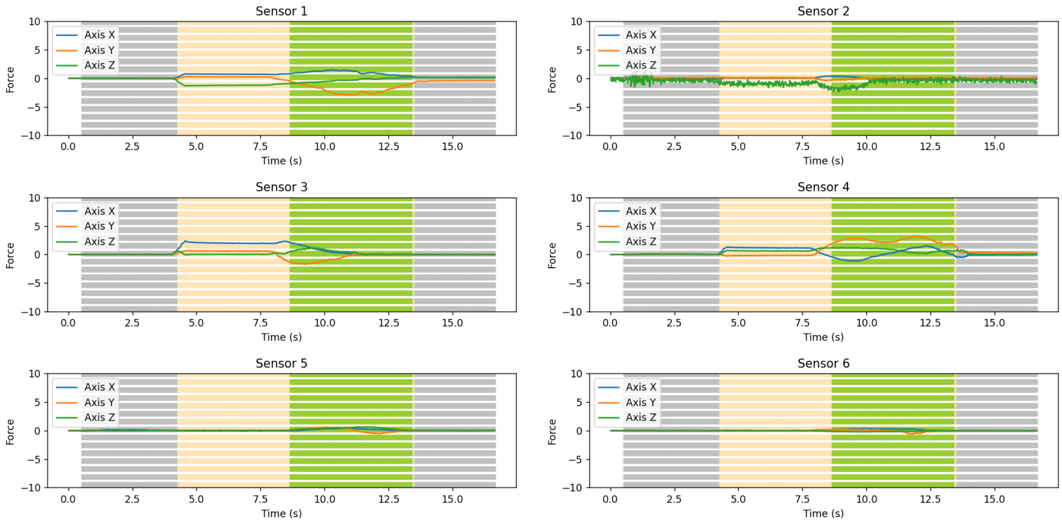


Figure 6.6: Test result after applying the trained machine-learning model to 'pass' action with the soft-ball.

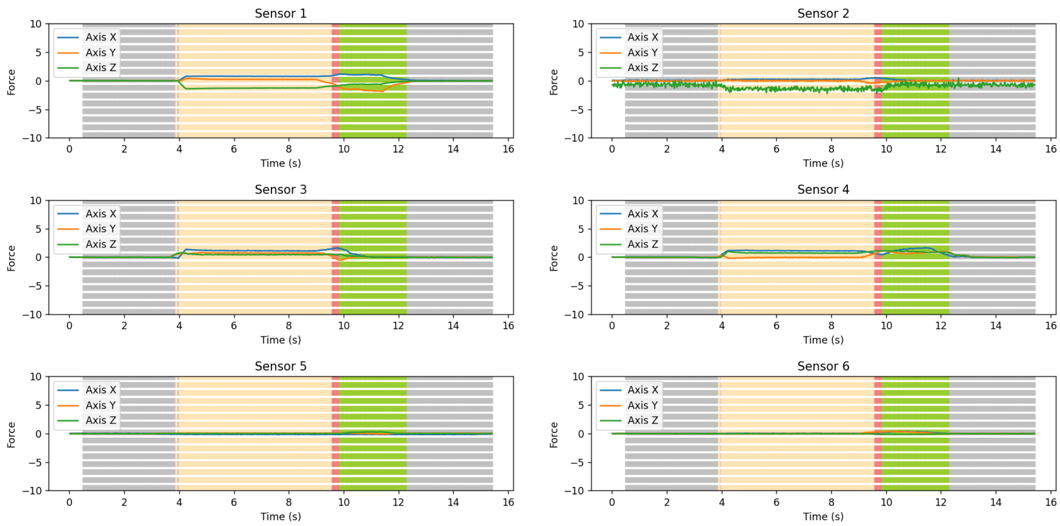


Figure 6.7: Test result after applying the trained machine-learning model to 'pass' action with the soft-ball.

Pull

In Figure 6.8, it is possible to see the results obtained from applying the trained machine learning model to the pulling actions using the soft-ball. In this case, as with all other analyses conducted for this action, the risky slip was correctly identified.

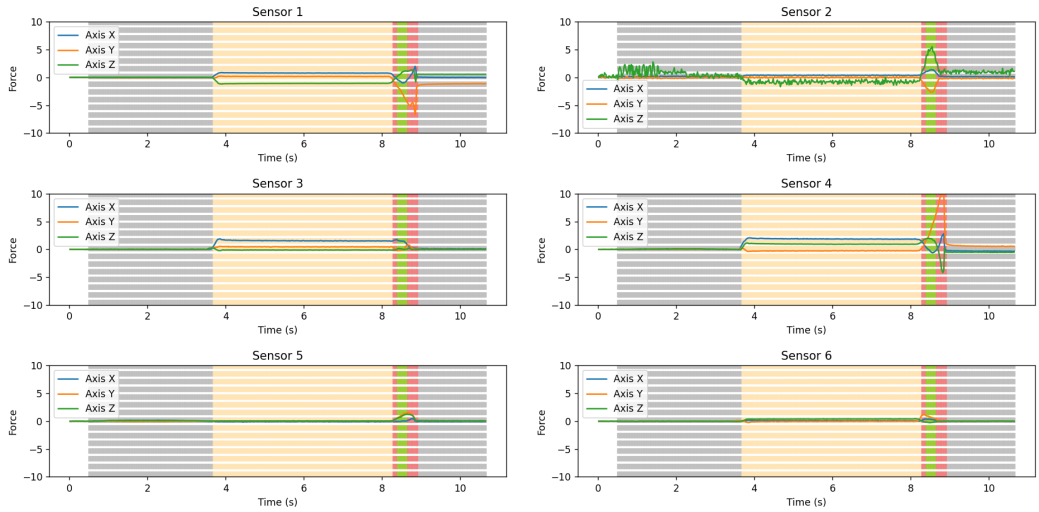


Figure 6.8: Test result after applying the trained machine-learning model to 'pull' action with the soft-ball.

Put-down

In Fig. 6.9, the results obtained from applying the trained machine learning model to the soft-ball putting-down action can be observed. In this case, as with all other analyses conducted for this action, the result is only partially correct. Indeed, the phases of ball putting-down and release are identified, but the interval between these two phases is not recognized as a grasp phase. See image 6.5a for reference.

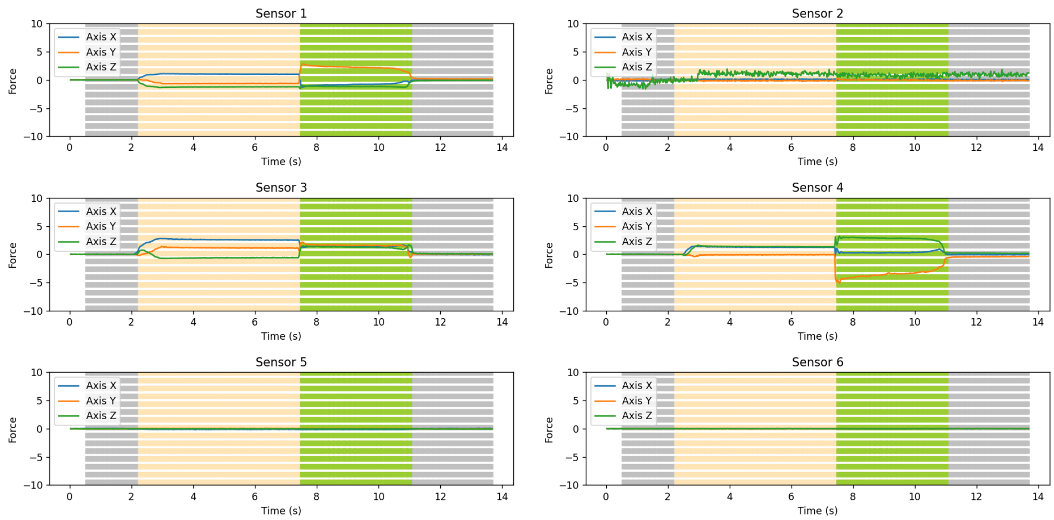


Figure 6.9: Test result after applying the trained machine-learning model to 'put-down' action with the soft-ball.

6.2.3 Object 2 - Empty plastic bottle

Pass

In Figure 6.10, the results obtained from applying the trained machine learning model to the passing actions for the empty plastic bottle can be observed. The result obtained in this image is excellent, identical to the ground truth depicted in Fig.6.4a. However, similar to what was found for the soft-ball, if a slight jerk occurs at the beginning or end of the passing action, the algorithm recognizes it as a risky situation.

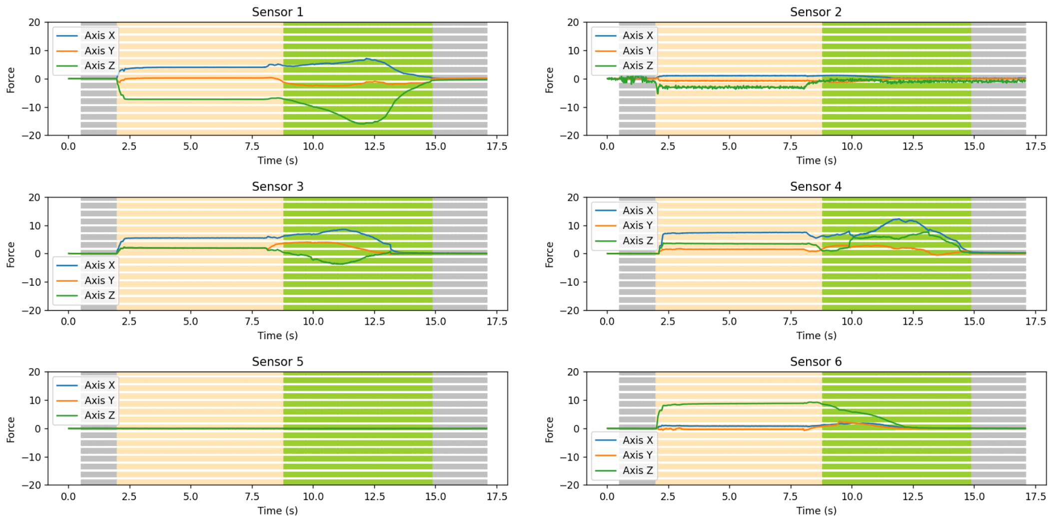


Figure 6.10: Test result after applying the trained machine-learning model to 'pass' action with the empty plastic bottle.

Pull

In Figure 6.11, it is possible to observe the results obtained from applying the trained machine learning model to the pulling actions using the empty plastic bottle. In this case, as with all other analyses conducted for this action, the risky slip was correctly identified as in 6.5b.

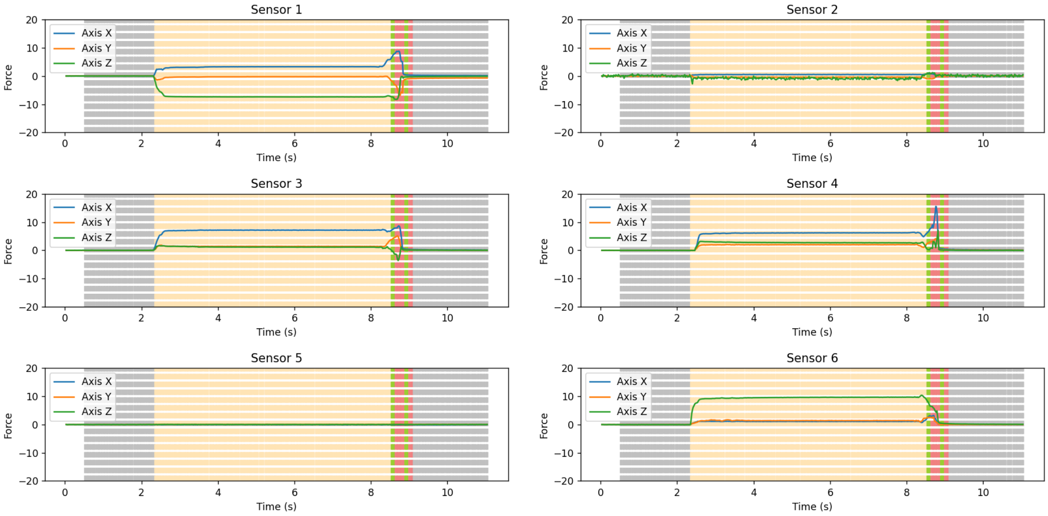


Figure 6.11: Test result after applying the trained machine-learning model to 'pull' action with the empty plastic bottle.

Put-down

In figures 6.12 and 6.13, the results obtained by applying the trained machine learning model to the putting-down actions for the empty plastic bottle can be observed.

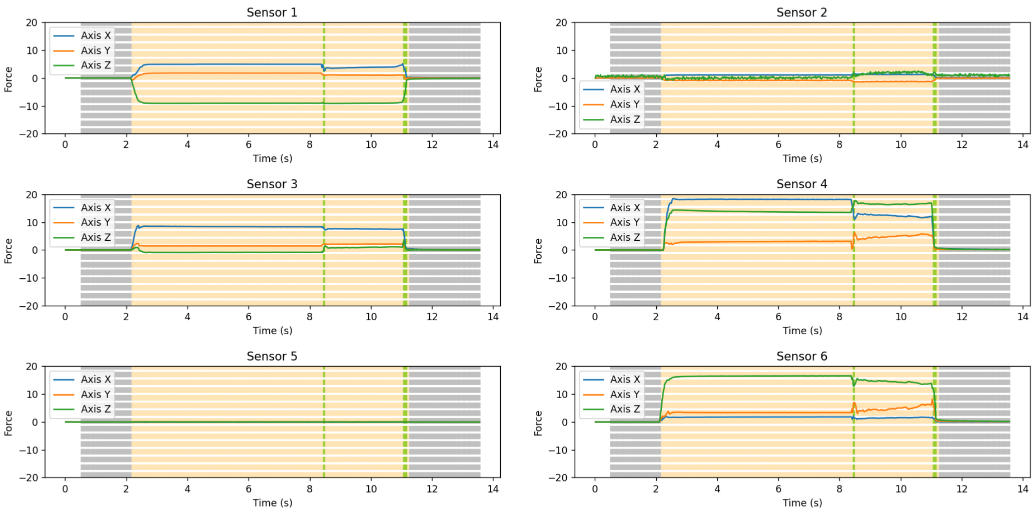


Figure 6.12: Test result after applying the trained machine-learning model to 'put-down' action with the empty plastic bottle.

The result obtained in image 6.12 is excellent, identical to the ground truth depicted in figure 6.5a. However, even the result obtained in figure 6.13, although not identical to the ground truth, can be considered correct since the phases of putting-down and release are correctly identified.

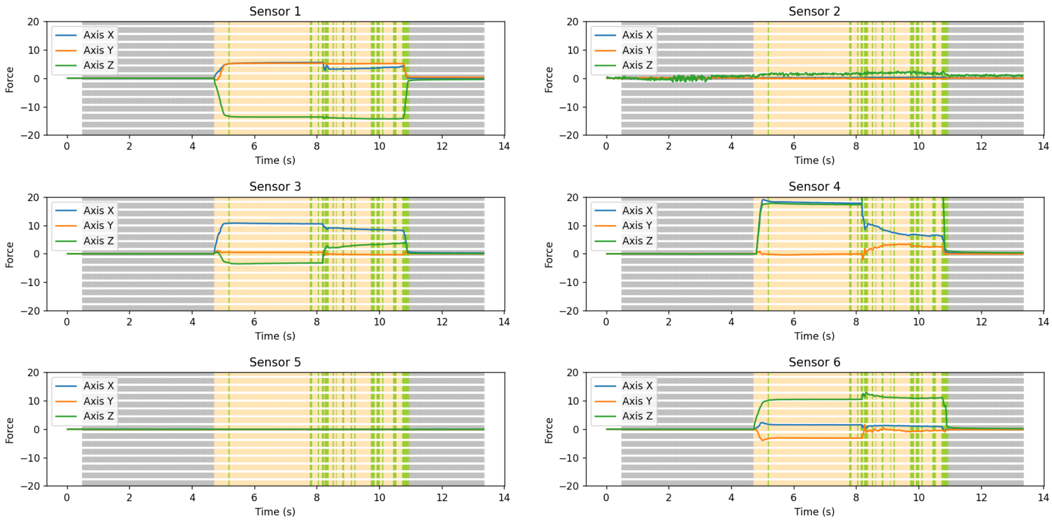


Figure 6.13: Test result after applying the trained machine-learning model to 'put-down' action with the empty plastic bottle.

6.2.4 Object 3 - Basket-ball

Pass

In figures 6.14 and 6.15, it is possible to observe the results derived from applying the trained machine learning model to the passing actions for the basket-ball. Image 6.14 exhibits an excellent result, matching the ground truth shown in figure 6.4a. However, as previously noted for the soft-ball and the empty plastic bottle, a discrepancy is evident in image 6.15. Here, it is observed that if a slight jerk occurs at the beginning or end of the passing action, the algorithm classifies it as a risky situation.

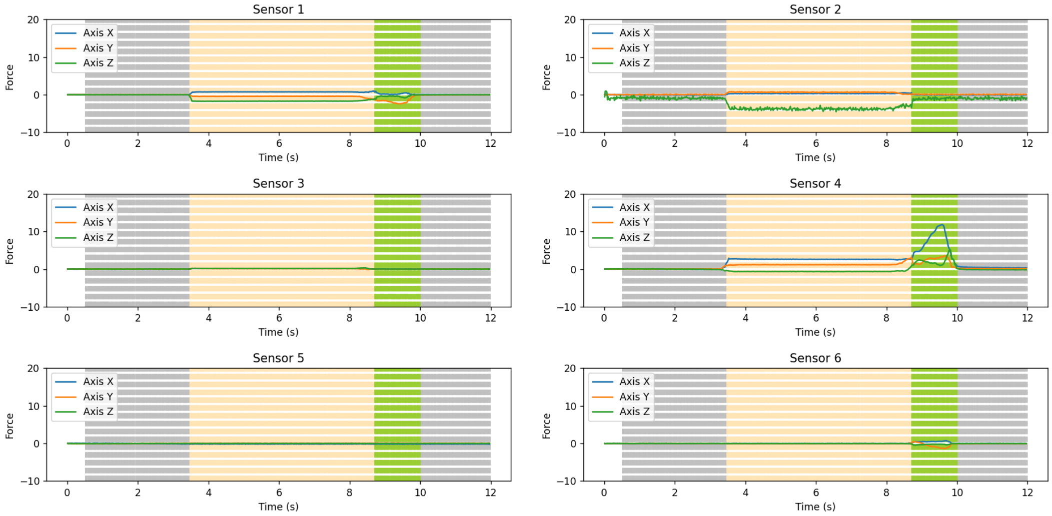


Figure 6.14: Test result after applying the trained machine-learning model to 'pass' action with the basket-ball.

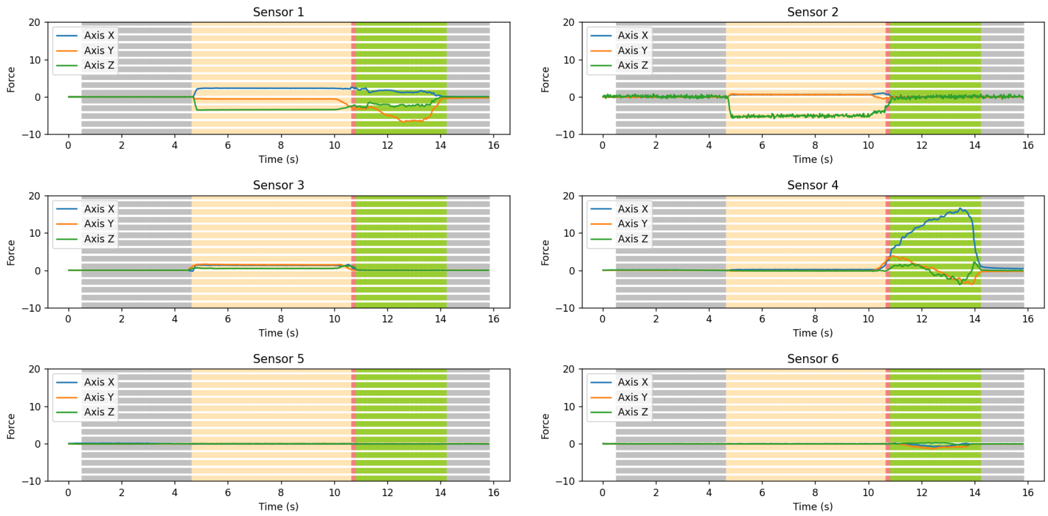


Figure 6.15: Test result after applying the trained machine-learning model to 'pass' action with the basket-ball.

Pull

In Figure 6.11, it is possible to observe the results obtained from applying the trained machine learning model to the pulling actions using the basket-ball. In this case, as with all other tests conducted for this action, the risky slip was correctly identified as in 6.5b.

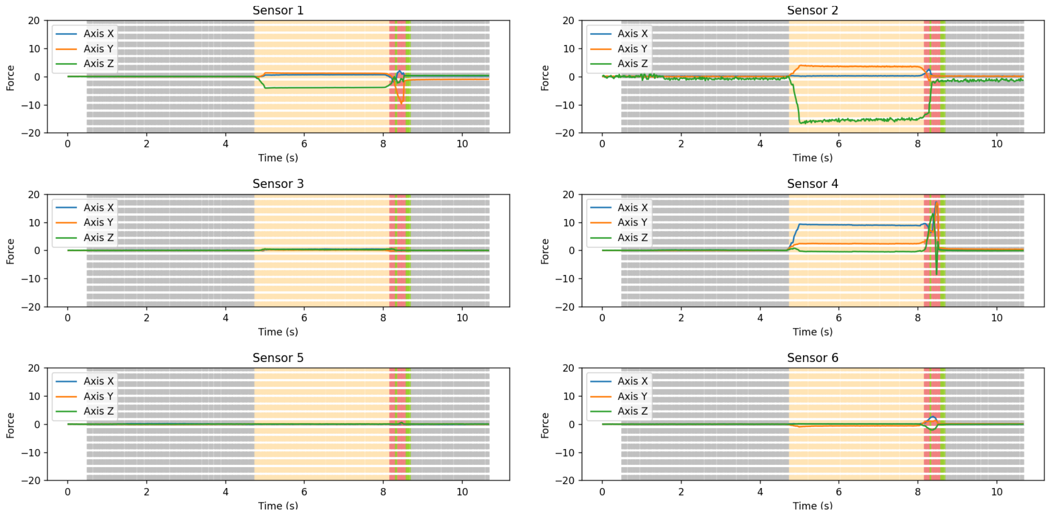


Figure 6.16: Test result after applying the trained machine-learning model to 'pull' action with the basket-ball.

Put-down

In figures 6.12 and 6.13, the results obtained by applying the trained machine learning model to the putting-down actions for the basket-ball can be observed. The result obtained in image 6.17 is excellent, identical to the ground truth depicted in Fig. 6.5a.

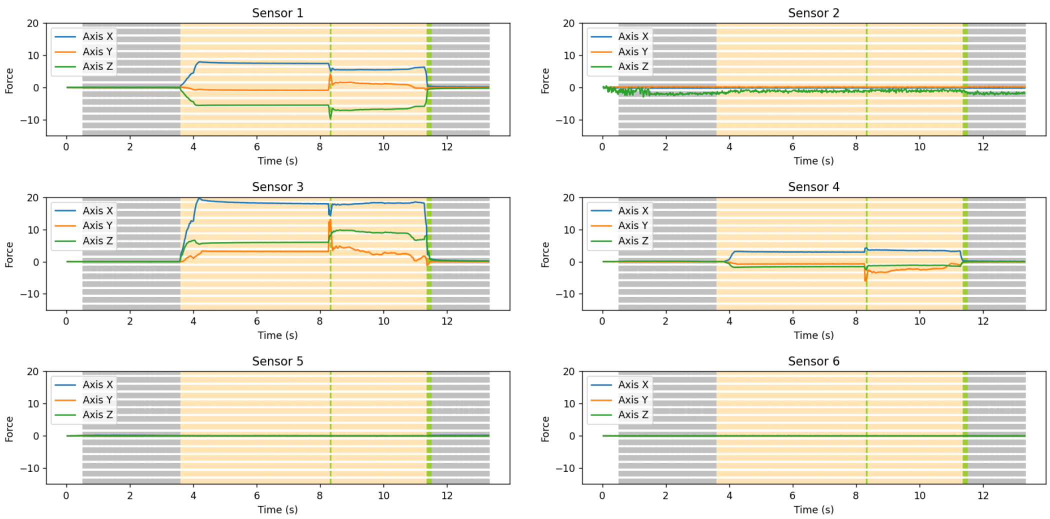


Figure 6.17: Test result after applying the trained machine-learning model to 'put-down' action with the basket-ball.

6.2.5 Object 4 - Hard-ball

Pass

In Figure 6.18, it is possible to observe the results obtained from applying the trained machine learning model to the passing actions using the hard-ball. Differently from all the previous object testing for the passing action, in this case, as with all other tests conducted for this action, the safe slip was correctly identified as in 6.5b.

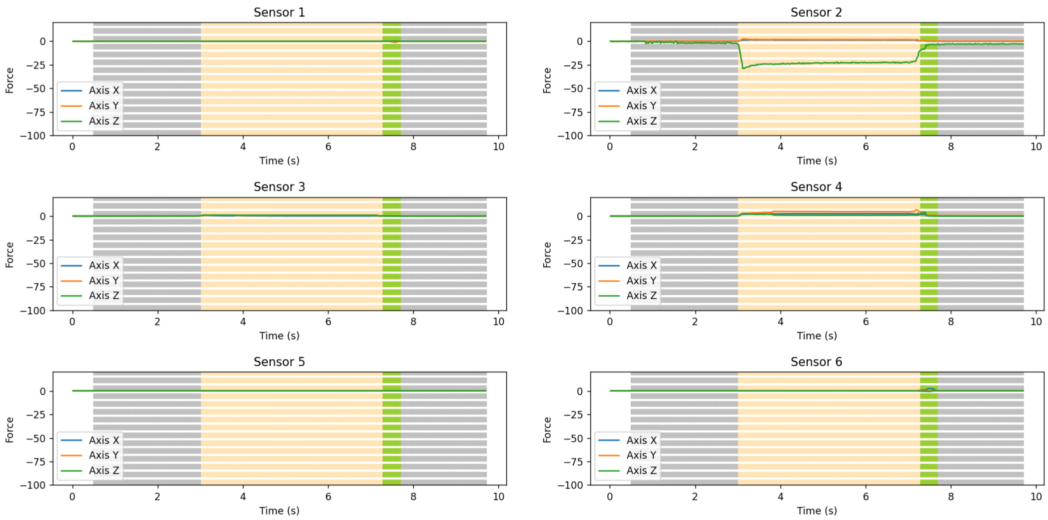


Figure 6.18: Test result after applying the trained machine-learning model to 'pass' action with the hard-ball.

Fall

In figures 6.19 and 6.20, the results obtained by applying the trained machine learning model to the fall actions for the hard-ball can be observed. The result obtained in figure 6.19 is excellent, identical to the ground truth depicted in the figure 6.5b. However, even the result obtained in figure 6.20, although not identical to the ground truth, can be considered correct since the phase of falling is correctly identified.

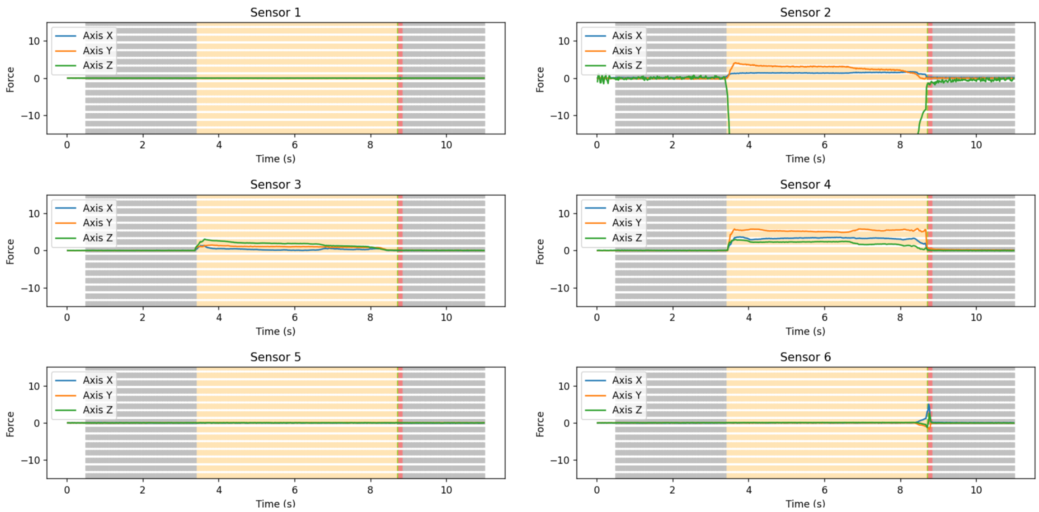


Figure 6.19: Test result after applying the trained machine-learning model to 'fall' action with the hard-ball.

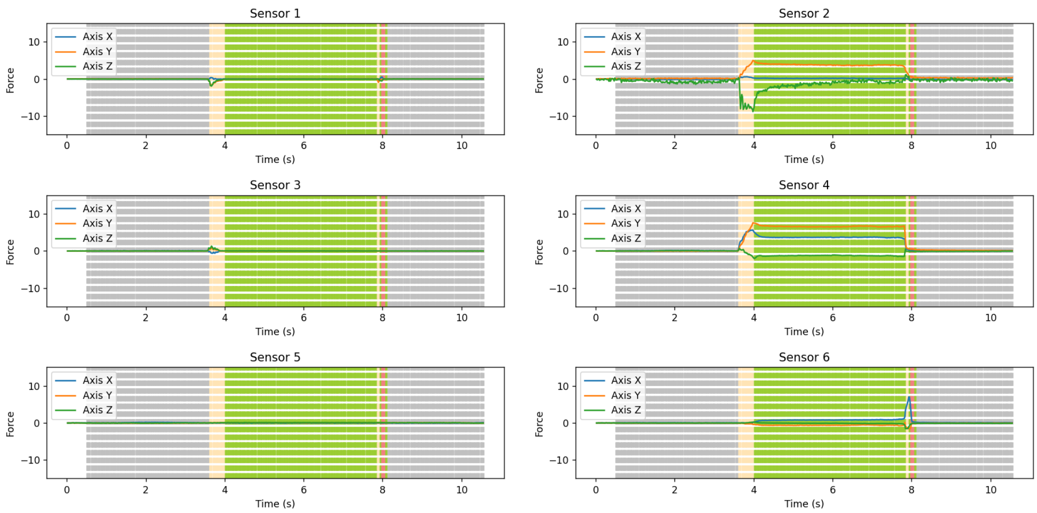


Figure 6.20: Test result after applying the trained machine-learning model to 'fall' action with the hard-ball.

Put-down

In Figure 6.21, the results obtained from applying the trained machine learning model to the hard-ball putting-down action can be observed. In this case, as with all other analyses conducted for this action, the result is only partially correct. Indeed, the phase of ball release is correctly identified together with the grasping phases, but put-down is not recognized. See image 6.5a for reference.

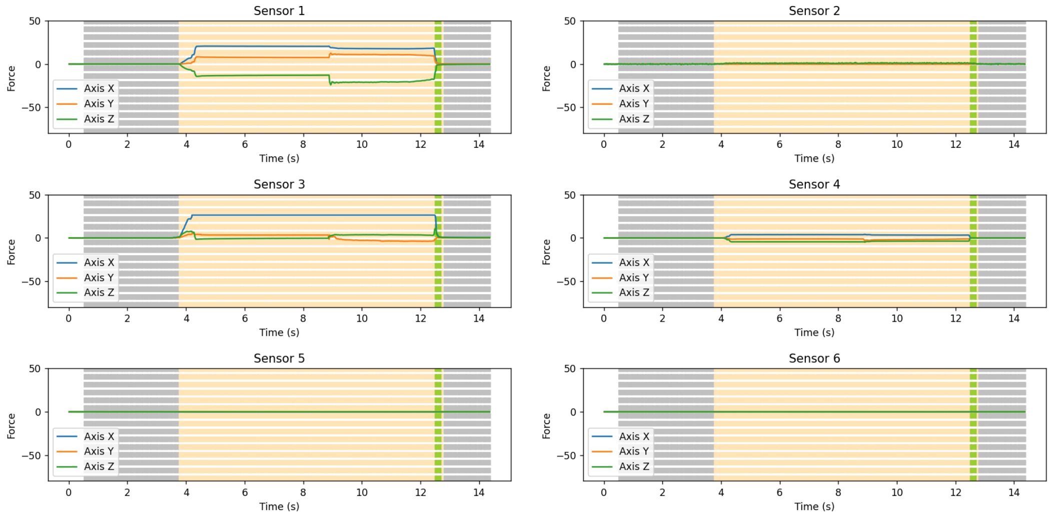


Figure 6.21: Test result after applying the trained machine-learning model to 'put-down' action with the hard-ball.

6.2.6 Object 5 - Full plastic bottle

Pass

In figures 6.22 and 6.23, the results obtained by applying the trained machine learning model to the passing actions for the full plastic bottle can be observed. Remember that this action was performed differently form the other pass actions with different objects. The result obtained in image 6.22 is excellent, almost identical to the ground truth depicted in the figure 6.5. However, even the result obtained in figure 6.20, although not identical to the ground truth, can be considered correct since the phase of passing is correctly identified.

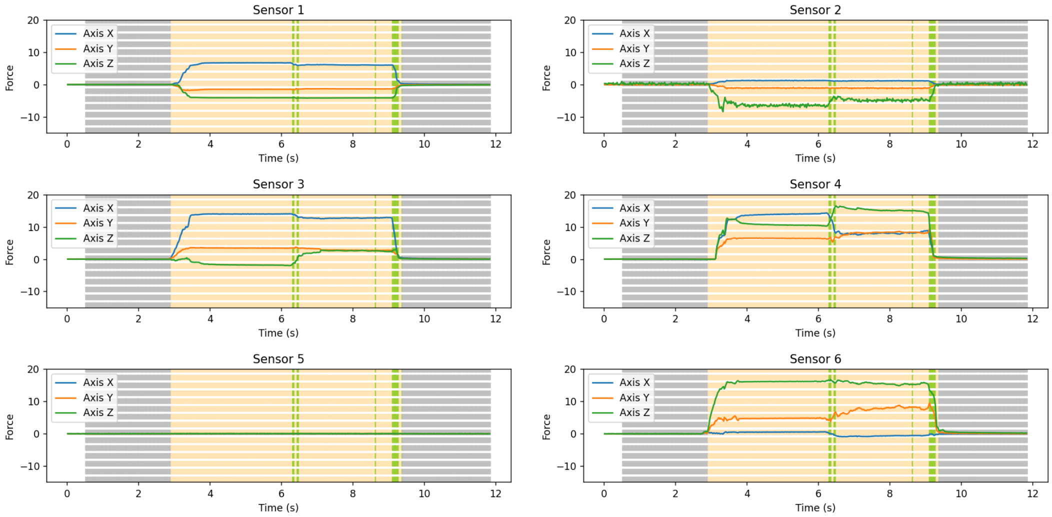


Figure 6.22: Test result after applying the trained machine-learning model to 'pass' action with the full plastic bottle.

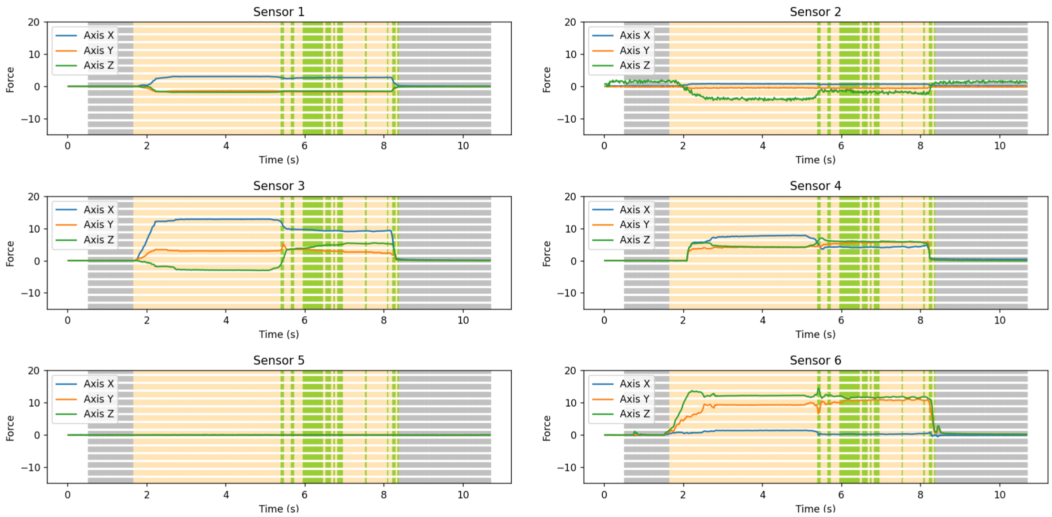


Figure 6.23: Test result after applying the trained machine-learning model to 'pass' action with the full plastic bottle.

Fall

In Figure 6.24, it is possible to observe the results obtained from applying the trained machine learning model to the falling actions using the full plastic bottle. In this case, as with all other tests conducted for this action, the risky slip was correctly identified as in 6.5b.

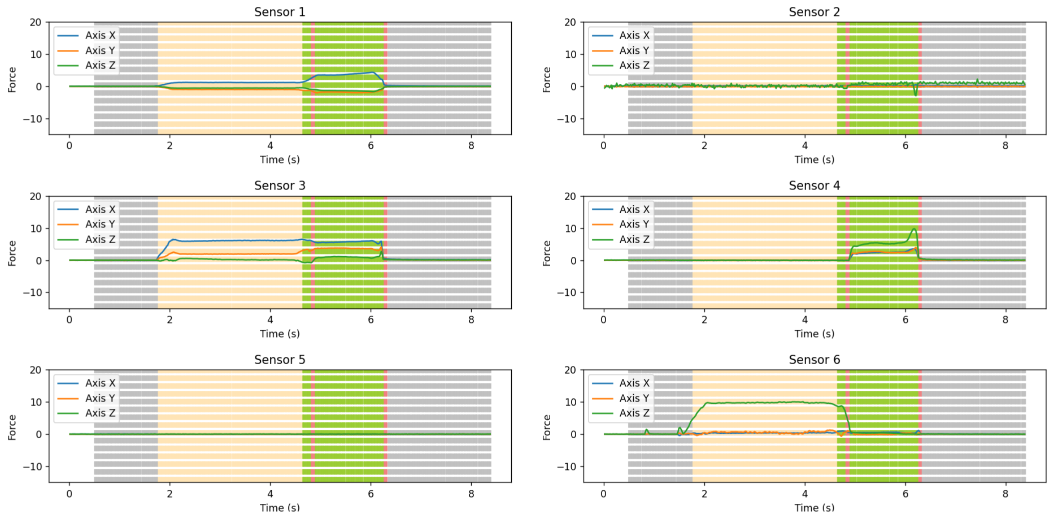


Figure 6.24: Test result after applying the trained machine-learning model to 'fall' action with the full plastic bottle.

Put-down

In figures 6.25 and 6.26, the results obtained by applying the trained machine learning model to the putting-down actions for the full plastic bottle can be observed.

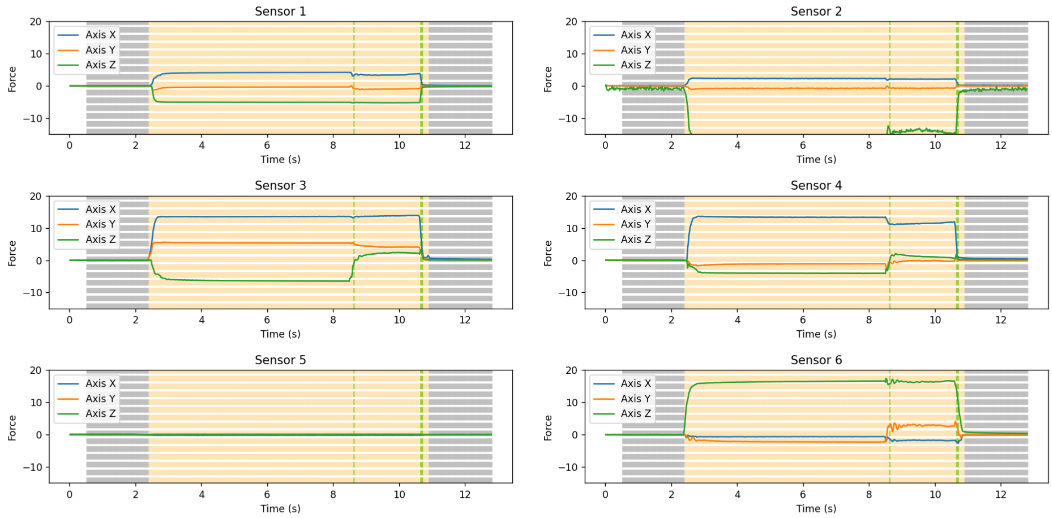


Figure 6.25: Test result after applying the trained machine-learning model to 'put-down' action with the full plastic bottle.

The result obtained in image 6.25 is excellent, identical to the ground truth depicted in the figure 6.5a. However, even the result obtained in figure 6.26, although not identical to the ground truth, can be considered correct since the phase of put-down and the release are correctly identified.

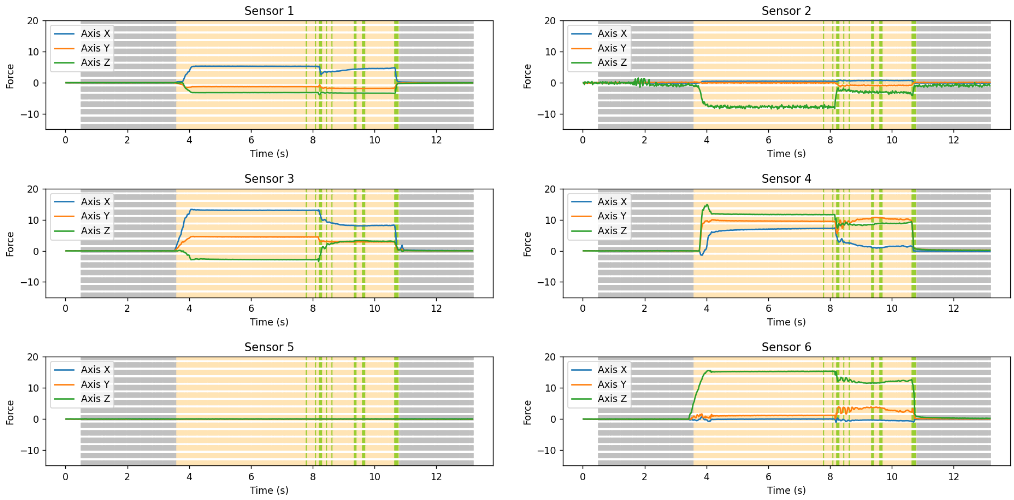


Figure 6.26: Test result after applying the trained machine-learning model to 'put-down' action with the full plastic bottle.

6.2.7 Unknown Object

To assess the validity of the model, not only was a human-study conducted, the findings of which will be detailed in Chapter 6.3, but trials were also recorded using an object unknown to the model. The object in question is a toy apple, visible in Figure 6.27, whose dimensions and weight are expressed in Table 6.1.

Figure 6.27: Apple used to test the model with an unknown object.

	Descriprion	Weight
Apple	Plastic toy 86.5 mm tall, neither hard nor squezable with a smooth surface. The upper diameter is 77 mm, while the lower one is 56 mm.	20.3 g

Table 6.1: This table provides a comprehensive overview of the unknown object, including its descriptions, dimensions, and weights. The dimensions are specified in millimeters (mm), while the weights are presented in grams (g). Note: Dimensions and weights are approximate and may vary slightly.

Pass

In Figure 6.28, it is possible to observe the results obtained from applying the trained machine learning model to the passing actions using the unknown object. In this case, as with all other tests conducted for this action, the safe slip was almost correctly identified.

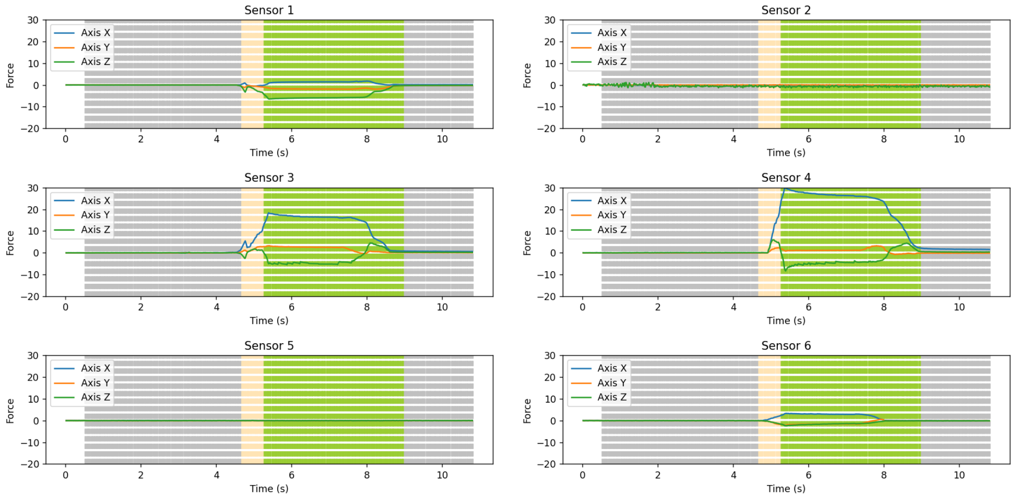


Figure 6.28: Test result after applying the trained machine-learning model to 'pass' action with the unknown object.

Pull

In Figure 6.29, it is possible to observe the results obtained from applying the trained machine learning model to the pulling actions using the unknown object. In this case, as with all other tests conducted for this action, the risky slip was almost correctly identified.

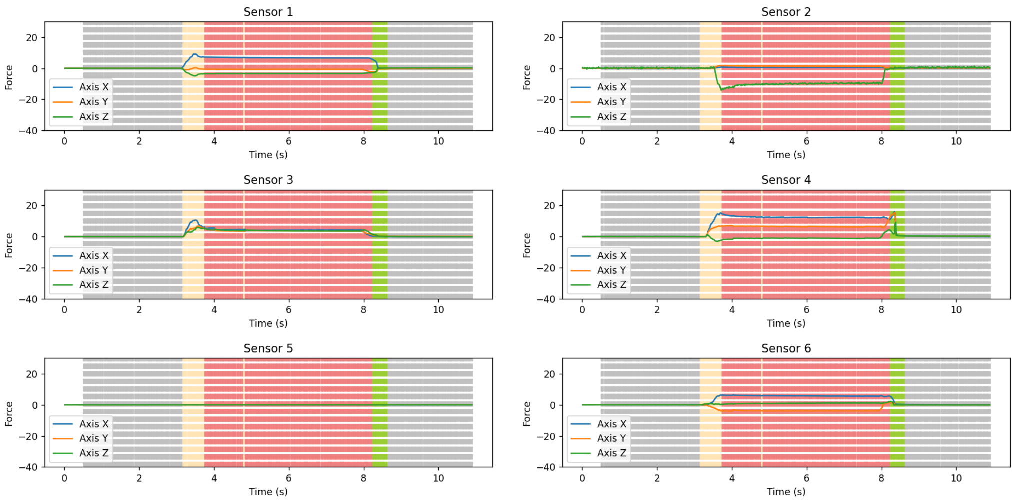


Figure 6.29: Test result after applying the trained machine-learning model to 'pull' action with the unknown object.

Put-down

In Figure 6.30, it is possible to observe the results obtained from applying the trained machine learning model to the putting-down actions using the unknown object. In this case, as with all other tests conducted for this action, the put-down and the release were correctly identified, but there were also a lot of additional safe slips.

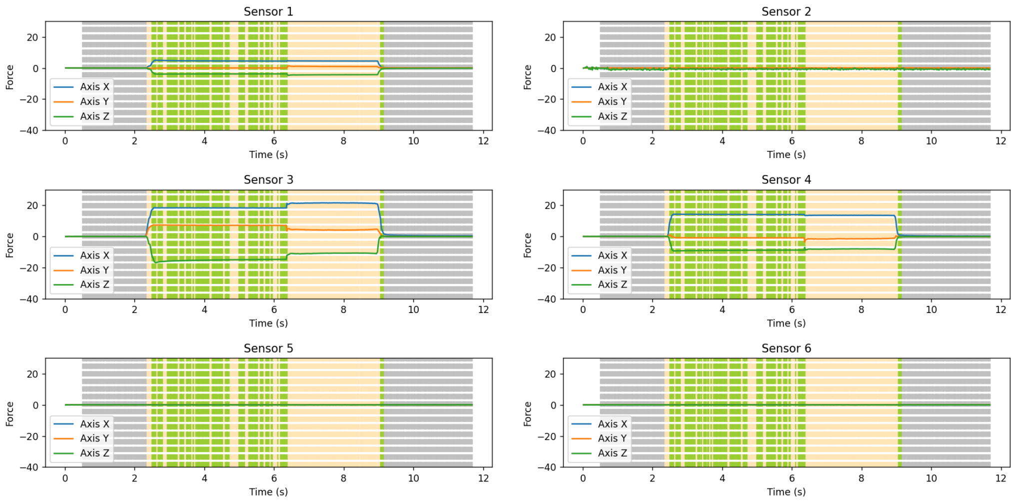


Figure 6.30: Test result after applying the trained machine-learning model to 'put-down' action with the unknown object.

6.3 Human study

6.3.1 Participant 1

Object 1 - Soft-ball

Pass

In Figure 6.31, it is possible to see the results obtained from applying the trained machine learning model to the passing actions using the soft-ball, accomplished by Participant 1. In this case, as with all other analyses conducted for this action, the safe slip was correctly identified.

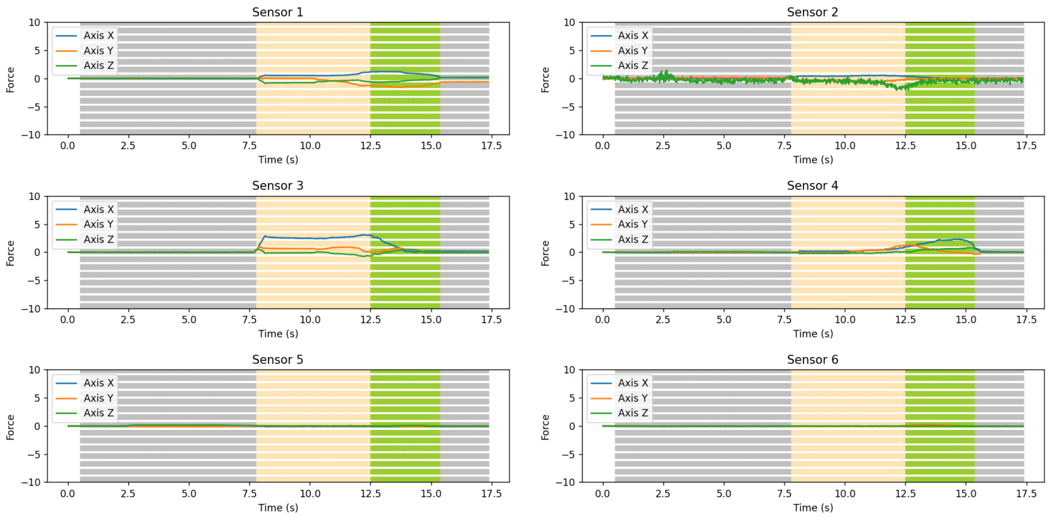


Figure 6.31: Test result for participant 1 after applying the trained machine-learning model to 'pass' action with the soft-ball.

Pull

In Figure 6.32, it is possible to see the results obtained from applying the trained machine learning model to the pulling actions using the soft-ball, accomplished by Participant 1. In this case, the risky slip was not correctly identified; the same happened for all other analyses conducted for this action.

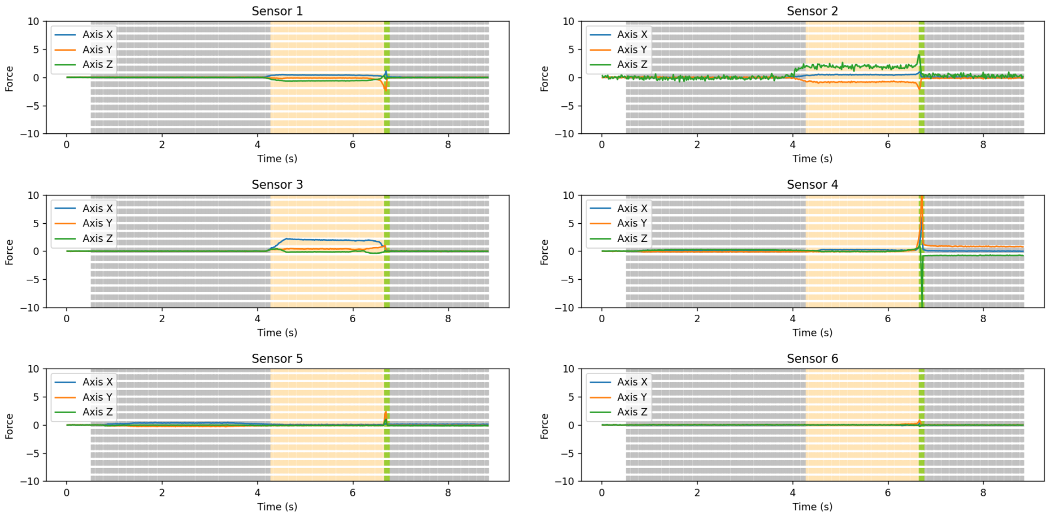


Figure 6.32: Test result for participant 1 after applying the trained machine-learning model to 'pull' action with the soft-ball.

Put-down

In Figure 6.33, it is possible to see the results obtained from applying the trained machine learning model to the putting-down actions using the soft-ball, accomplished by Participant 1. In this case, the put-down and the release were correctly identified, however, according to the ground truth in Fig. 6.5a, a grasp keeping is mistaken for a risky slip.

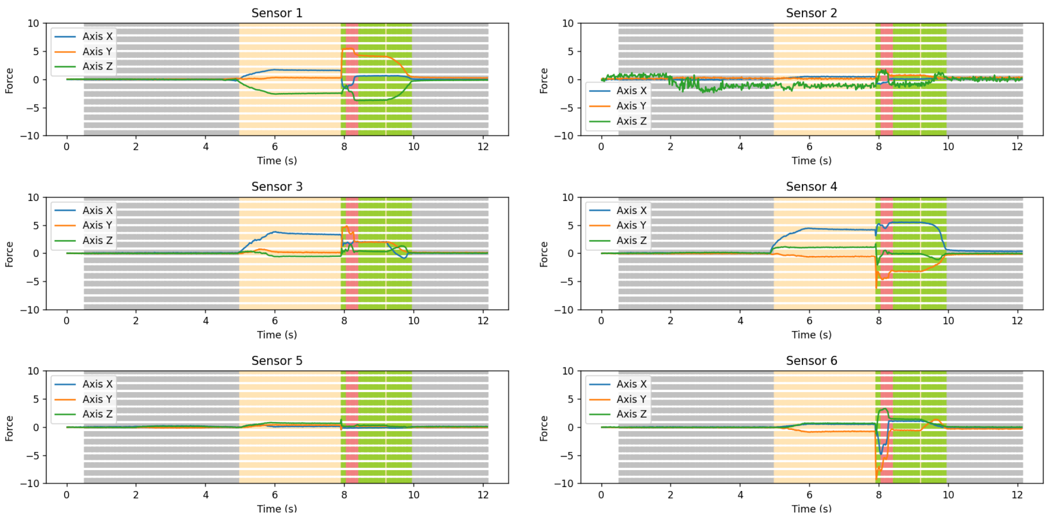


Figure 6.33: Test result for participant 1 after applying the trained machine-learning model to 'put-down' action with the soft-ball.

Object 2 - Empty plastic bottle

Pass

In Figure 6.34, it is possible to see the results obtained from applying the trained machine learning model to the passing actions using the empty plastic bottle, accomplished by Participant 1. In this case, the safe slip was not correctly identified; the same happened for all other analyses conducted for this action.

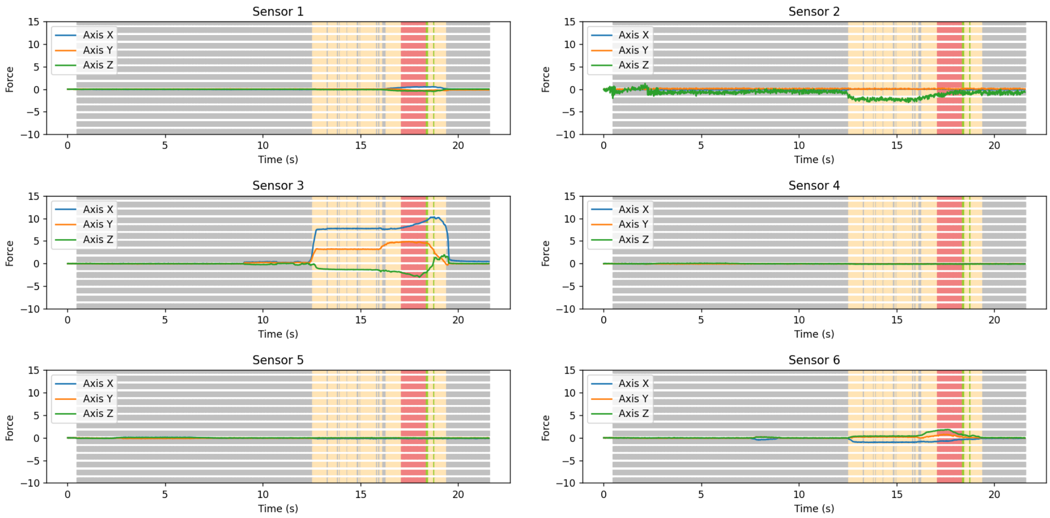


Figure 6.34: Test result for participant 1 after applying the trained machine-learning model to 'pass' action with the empty plastic bottle.

Pull

In Figure 6.35, it is possible to see the results obtained from applying the trained machine learning model to the pulling actions using the empty plastic bottle, accomplished by Participant 1. In this case, the risky slip was not correctly identified; the same happened for all other analyses conducted for this action.

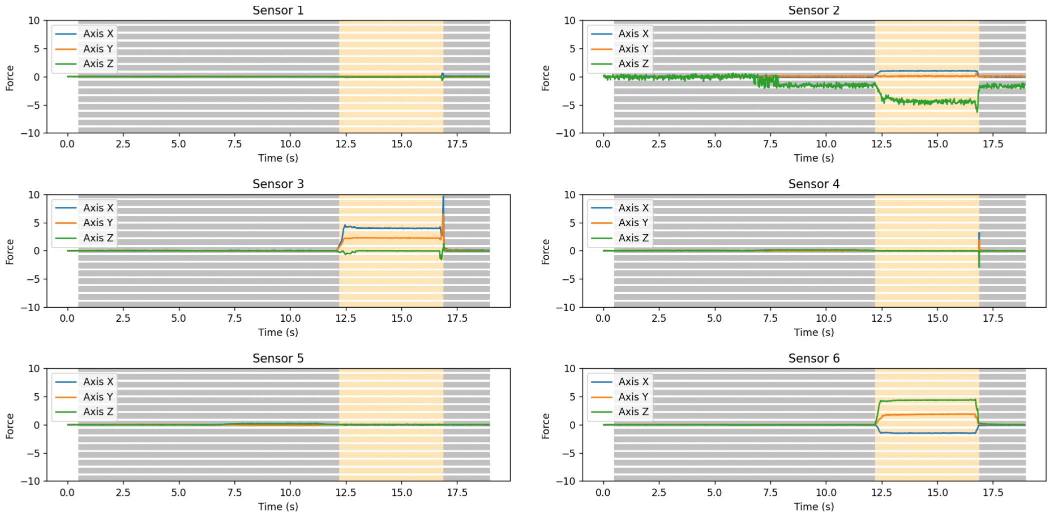


Figure 6.35: Test result for participant 1 after applying the trained machine-learning model to 'pull' action with the empty plastic bottle.

Put-down

In figures 6.36 and 6.37, it is possible to see the results obtained from applying the trained machine learning model to the putting-down actions using the empty plastic bottle, accomplished by Participant 1. For what it concerns Fig.6.36 the putting-down action was correctly recognised. However, the result also showed Fig. 6.37 which differently from the latter is not correctly identified.

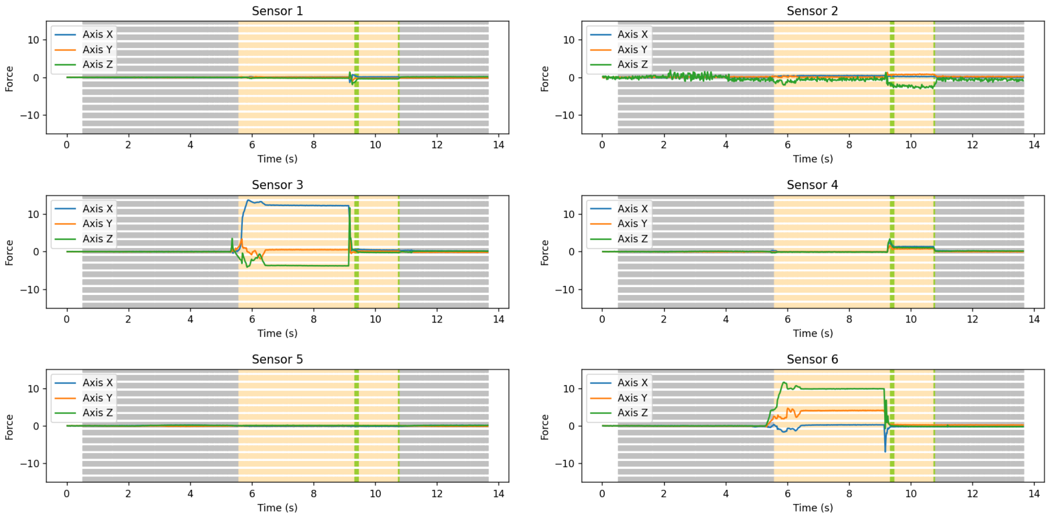


Figure 6.36: Test result for participant 1 after applying the trained machine-learning model to 'put-down' action with the empty plastic bottle.

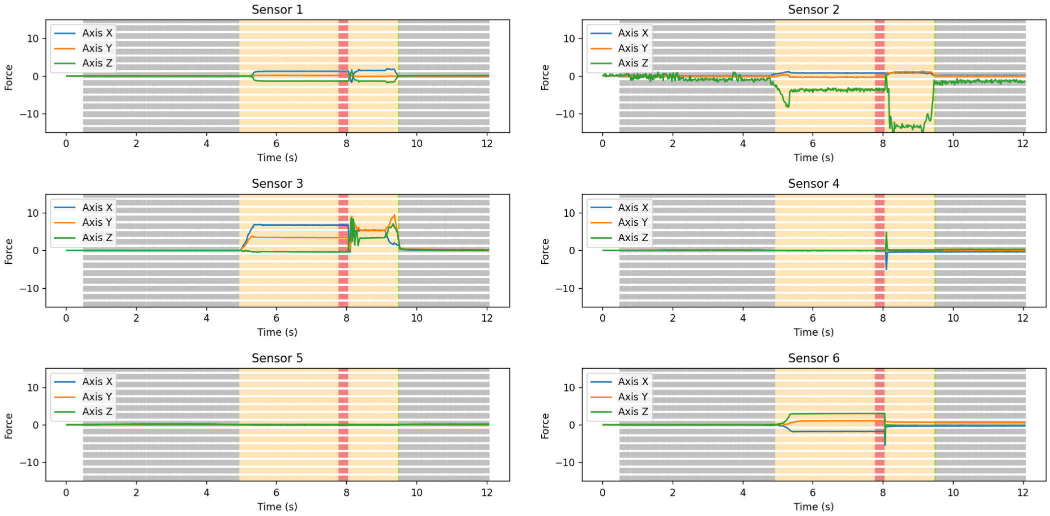


Figure 6.37: Test result for participant 1 after applying the trained machine-learning model to 'put-down' action with the empty plastic bottle.

Object 3 - Basket-ball

Pass

In Figure 6.38, it is possible to see the results obtained from applying the trained machine learning model to the passing actions using the basket-ball, accomplished by Participant 1. In this case, as with all other analyses conducted for this action, the safe slip was not correctly identified, as always mistaken for a risky slip.

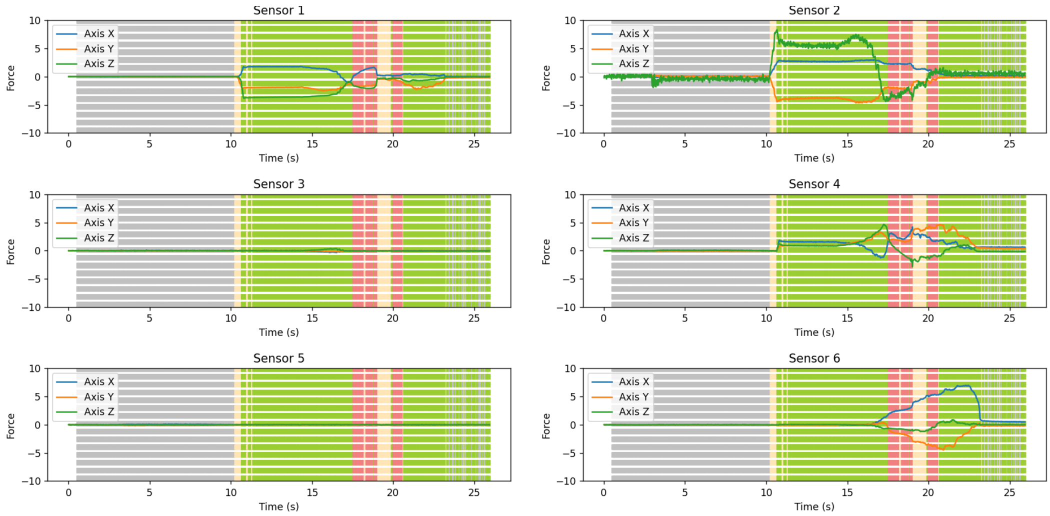


Figure 6.38: Test result for participant 1 after applying the trained machine-learning model to 'pass' action with the basket-ball.

Pull

In Figure 6.39, it is possible to see the results obtained from applying the trained machine learning model to the pulling actions using the basket-ball, accomplished by Participant 1. For what interests Fig.6.36 the pulling action was correctly recognised, however, other cases of the same action showed results similar to the ones of the pulling action for the soft-ball.

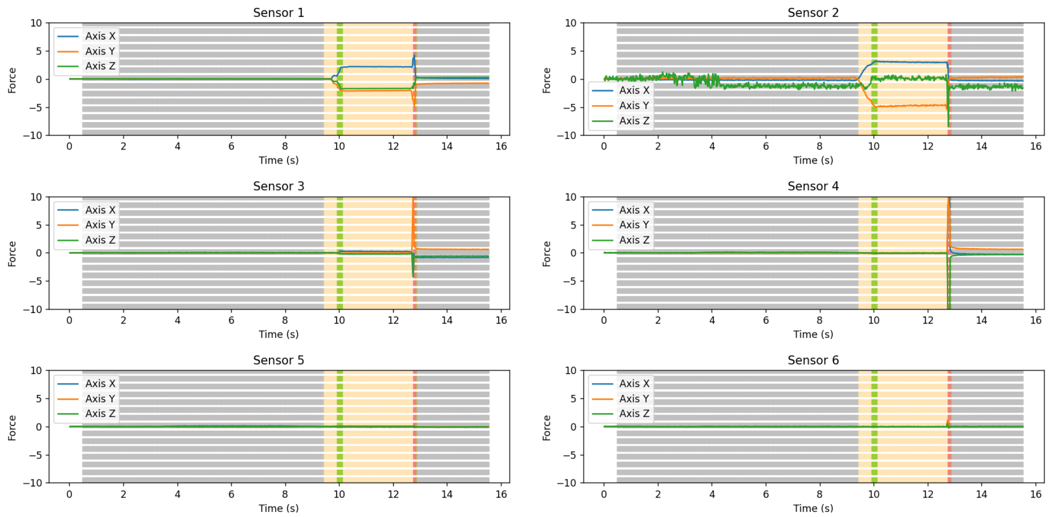


Figure 6.39: Test result for participant 1 after applying the trained machine-learning model to 'pull' action with the basket-ball.

Put-down

In Figure 6.40, the results obtained from applying the trained machine learning model to the pulling actions using the basket-ball, accomplished by Participant 1, can be observed. The outcome is really good, and additionally, other attempts at the same action, while not as successful, can still be deemed acceptable.

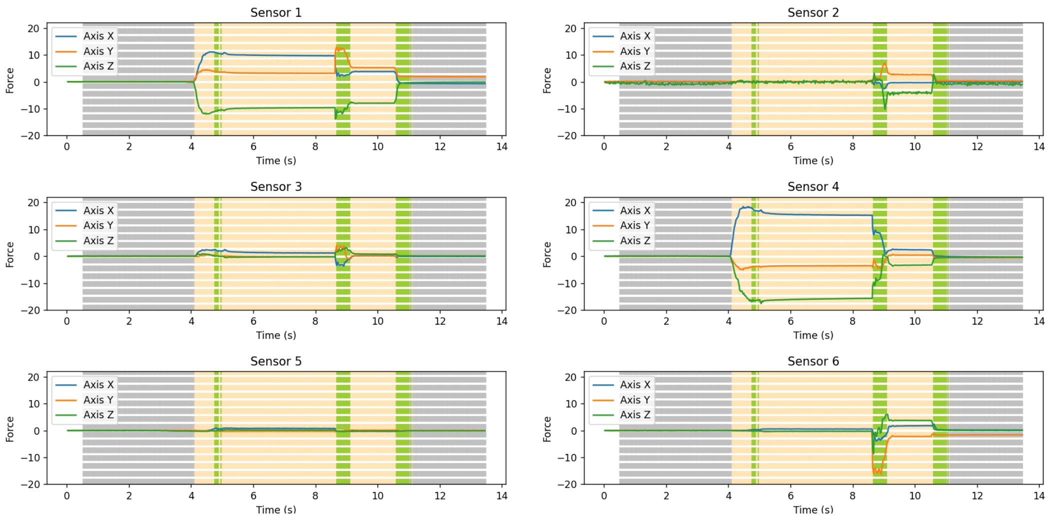


Figure 6.40: Test result for participant 1 after applying the trained machine-learning model to 'put-down' action with the basket-ball.

Object 4 - Hard-ball

Pass

In Figure 6.41, it is possible to see the results obtained from applying the trained machine learning model to the passing actions using the hard-ball, accomplished by Participant 1. In this case, as with all other analyses conducted for this action, the safe slip was not correctly identified, as always mistaken for a risky slip.

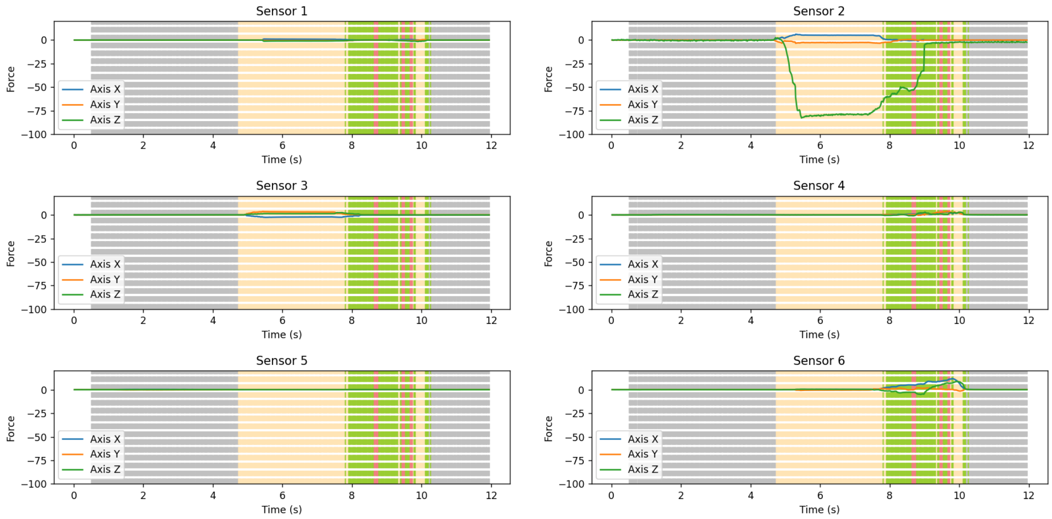


Figure 6.41: Test result for participant 1 after applying the trained machine-learning model to 'pass' action with the hard-ball.

Fall

In Figure 6.42, it is possible to see the results obtained from applying the trained machine learning model to the falling actions using the hard-ball, accomplished by Participant 1. In this case, as with all other analyses conducted for this action, the risky slip was not correctly identified, as the only prediction shown is the grasping one.

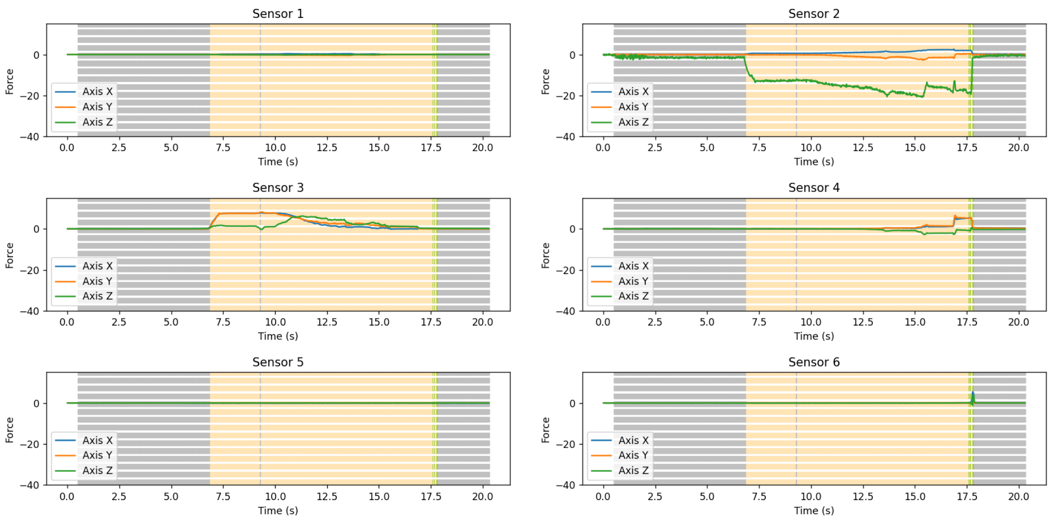


Figure 6.42: Test result for participant 1 after applying the trained machine-learning model to 'fall' action with the hard-ball.

Put-down

In Figure 6.43, the results obtained from applying the trained machine learning model to the putting-down actions using the hard-ball, accomplished by Participant 1, can be observed. The outcome is not perfect but it can still be considered correct, also other attempts at the same action can be considered acceptable.

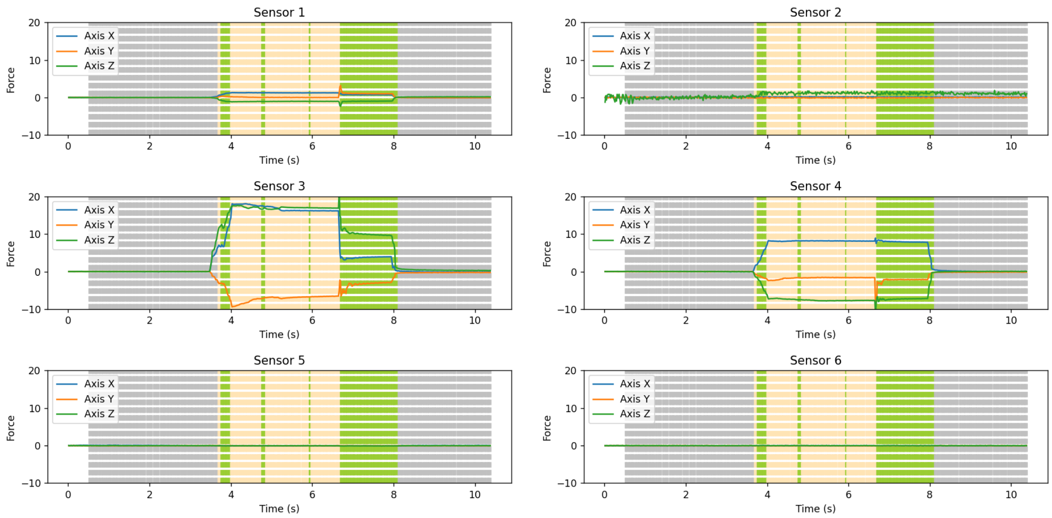


Figure 6.43: Test result for participant 1 after applying the trained machine-learning model to 'put-down' action with the hard-ball.

Object 5 - Full plastic bottle

Pass

In figure 6.44, it is possible to see the results obtained from applying the trained machine learning model to the passing actions using the full plastic bottle, accomplished by Participant 1. As for this image, the passing action was correctly recognised, recalling that the ground truth for this action is the one from Fig. 6.5. Nevertheless, other cases of the same action showed wrong results.

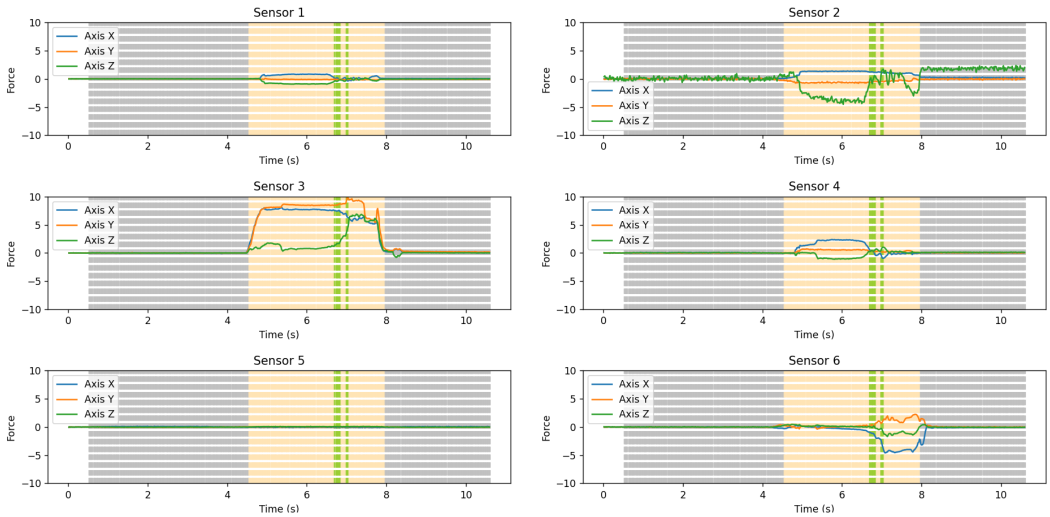


Figure 6.44: Test result for participant 1 after applying the trained machine-learning model to 'pass' action with the full plastic bottle.

Fall

In figure 6.45, it is possible to see the results obtained from applying the trained machine learning model to the falling action using the full plastic bottle, accomplished by Participant 1. In this case, as with all other analyses conducted for this action, the risky slip was correctly identified.

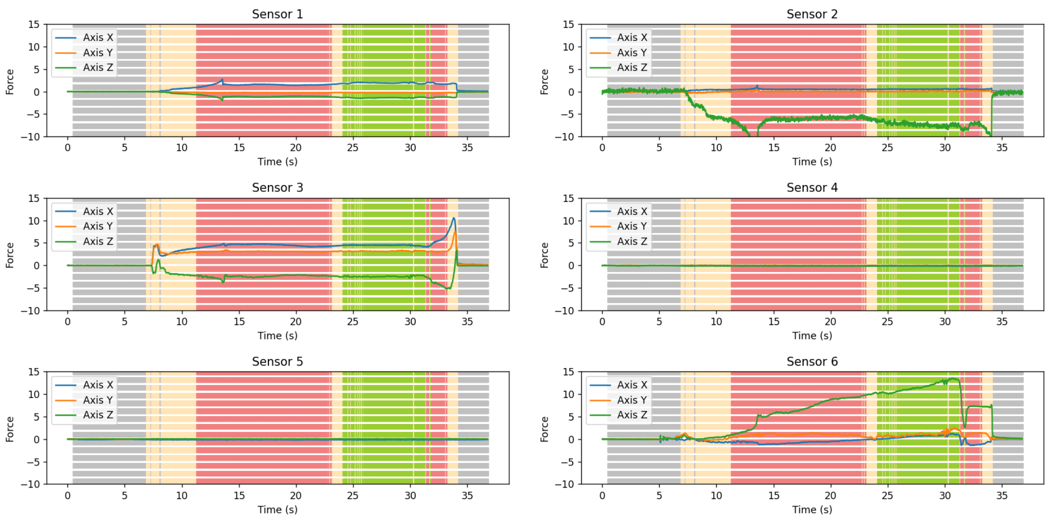


Figure 6.45: Test result for participant 1 after applying the trained machine-learning model to 'fall' action with the full plastic bottle.

Put-down

In figure 6.45, it is possible to see the results obtained from applying the trained machine learning model to the putting-down action using the full plastic bottle, accomplished by Participant 1. In this case, as with all other analyses conducted for this action, the put-down and the release were correctly identified, even if not perfectly as 6.5a.

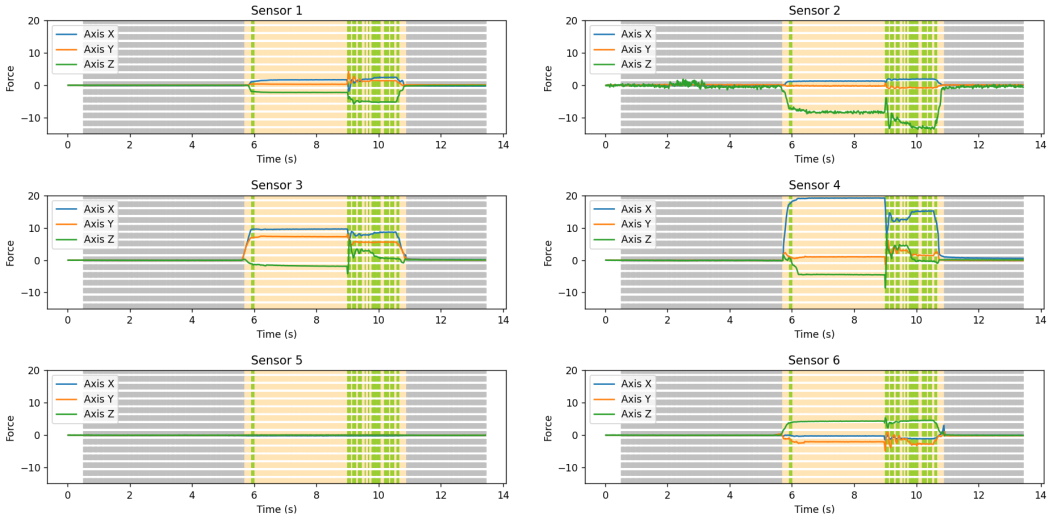


Figure 6.46: Test result for participant 1 after applying the trained machine-learning model to 'put-down' action with the full plastic bottle.

6.3.2 Participant 2

Object 1 - Soft-ball

Pass

In Figure 6.47, it is possible to see the results obtained from applying the trained machine learning model to the passing actions using the soft-ball, accomplished by Participant 2. As for this image, the passing action was correctly recognised. Nevertheless, other cases of the same action showed wrong results, mistaking the safe slip with a risky one.

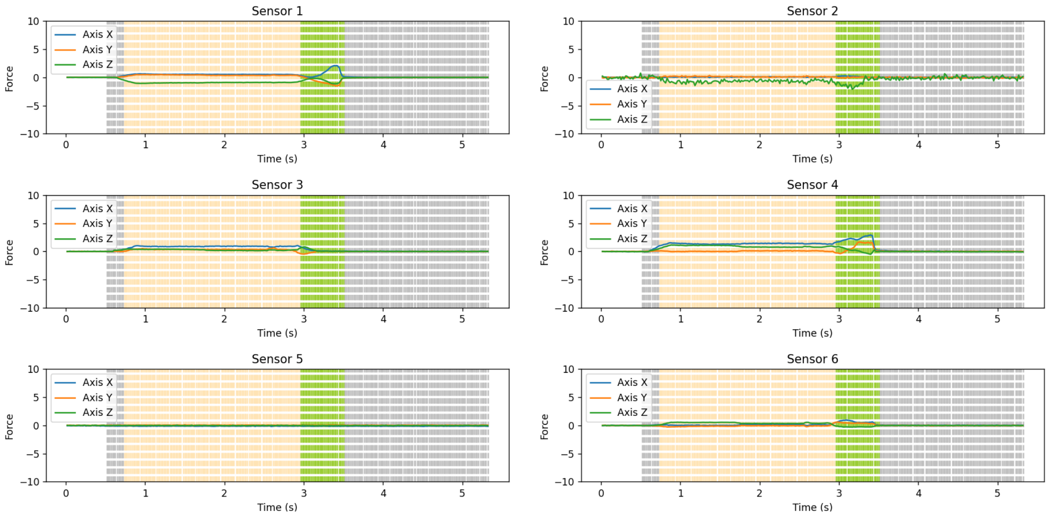


Figure 6.47: Test result for participant 2 after applying the trained machine-learning model to 'pass' action with the soft-ball.

Pull

In Figure 6.48, it is possible to see the results obtained from applying the trained machine learning model to the pulling action using the soft-ball, accomplished by Participant 2. In this case, as with all other analyses conducted for this action, the risky slip was correctly identified. However, Figure 6.49 reveals a misplaced safe slip. Despite this, the trial is not deemed incorrect, and the rationale behind this will be elaborated on in the discussion.

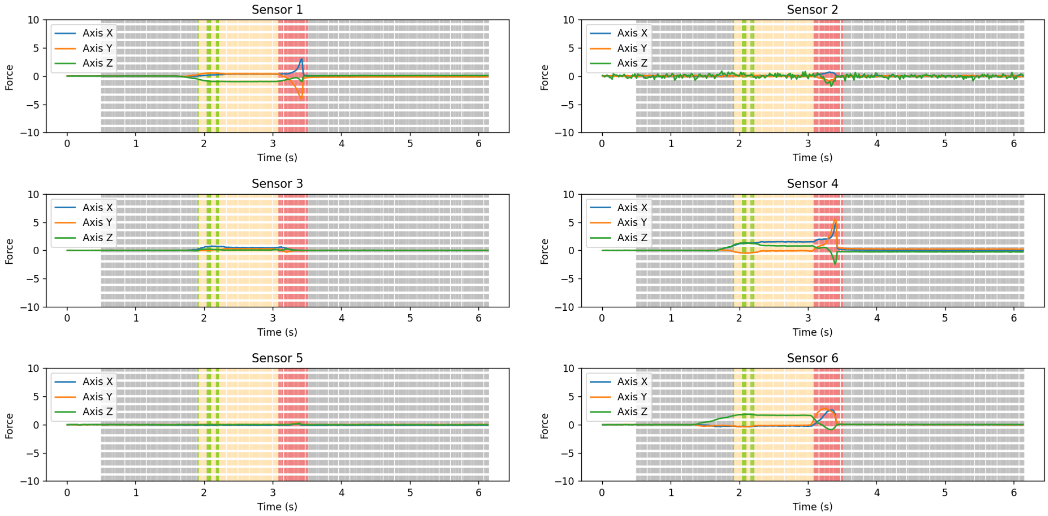


Figure 6.48: Test result for participant 2 after applying the trained machine-learning model to 'pull' action with the soft-ball.

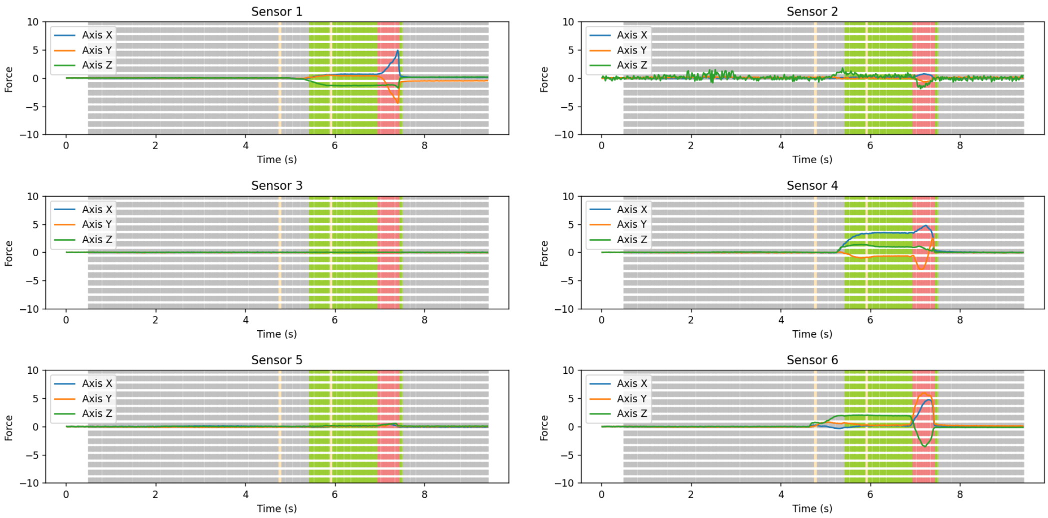


Figure 6.49: Test result for participant 2 after applying the trained machine-learning model to 'pull' action with the soft-ball.

Put-down

In Fig. 6.50, it is possible to see the results obtained from applying the trained machine learning model to the putting-down actions using the soft-ball, accomplished by Participant 2. In this case, as with all other analyses conducted for this action, the put-down and the release were correctly identified, even if not perfectly as 6.5a.

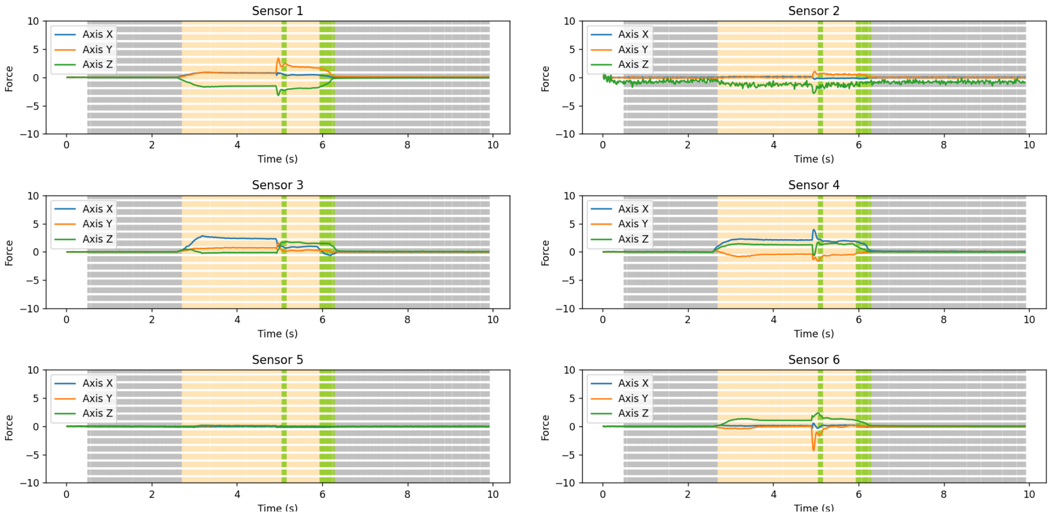


Figure 6.50: Test result for participant 2 after applying the trained machine-learning model to 'put-down' action with the soft-ball.

Object 2 - Empty plastic bottle

Pass

In Figure 6.51, it is possible to see the results obtained from applying the trained machine learning model to the passing action using the empty plastic bottle, accomplished by Participant 2. In this case, the passing was correctly identified, even if not perfectly as 6.5a. However, in one trial no action other than grasping was registered.

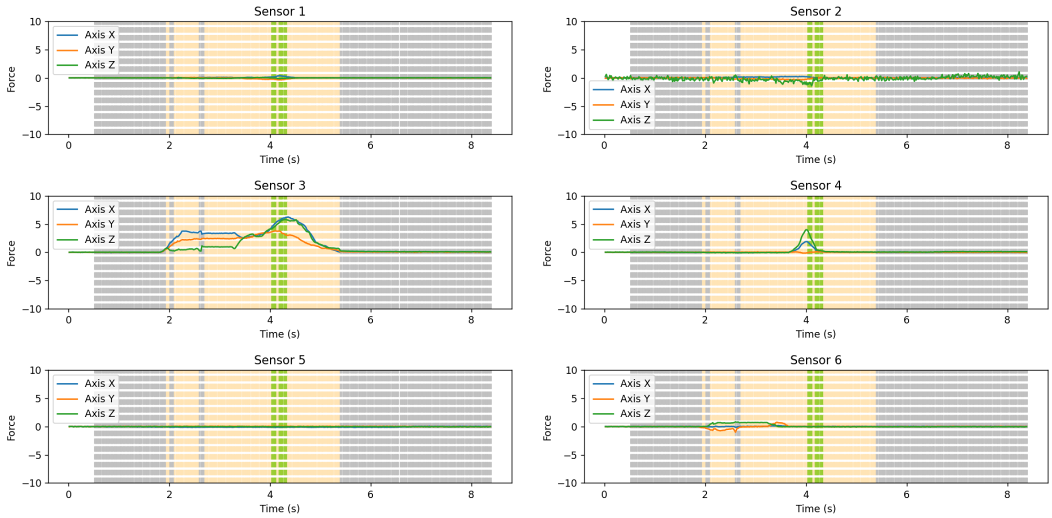


Figure 6.51: Test result for participant 2 after applying the trained machine-learning model to 'pass' action with the empty plastic bottle.

Pull

In Figure 6.52, it is possible to see the results obtained from applying the trained machine learning model to the pulling action using the empty plastic bottle, accomplished by Participant 2. In this case, the risky slip was correctly identified. However, in one trial no risky slip was registered.

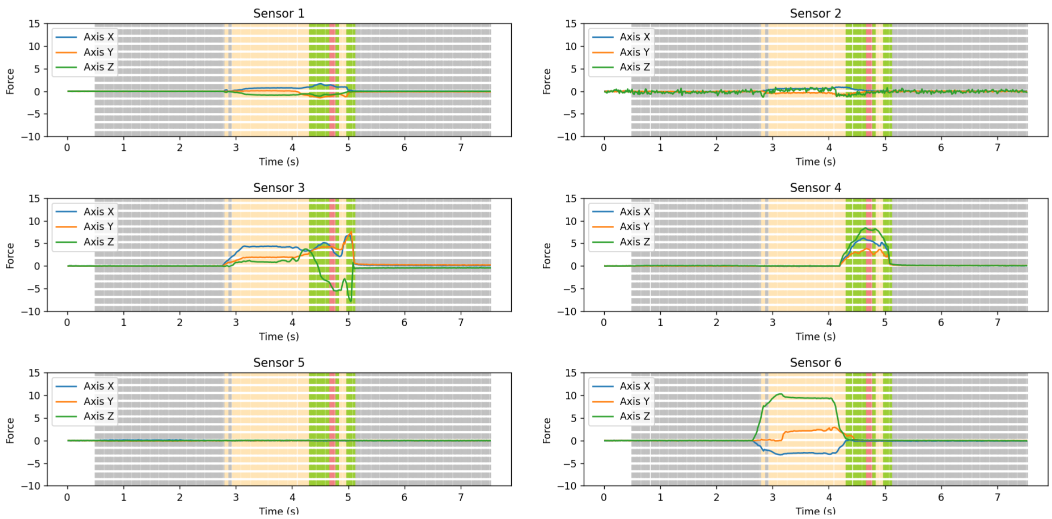


Figure 6.52: Test result for participant 2 after applying the trained machine-learning model to 'pull' action with the empty plastic bottle.

Put-down

In Figure 6.53, it is possible to see the results obtained from applying the trained machine learning model to the putting-down actions using the empty plastic bottle, accomplished by Participant 2. In this case, as with all other analyses conducted for this action, the put-down and the release were correctly identified, even if not perfectly as the ground truth in 6.5a.

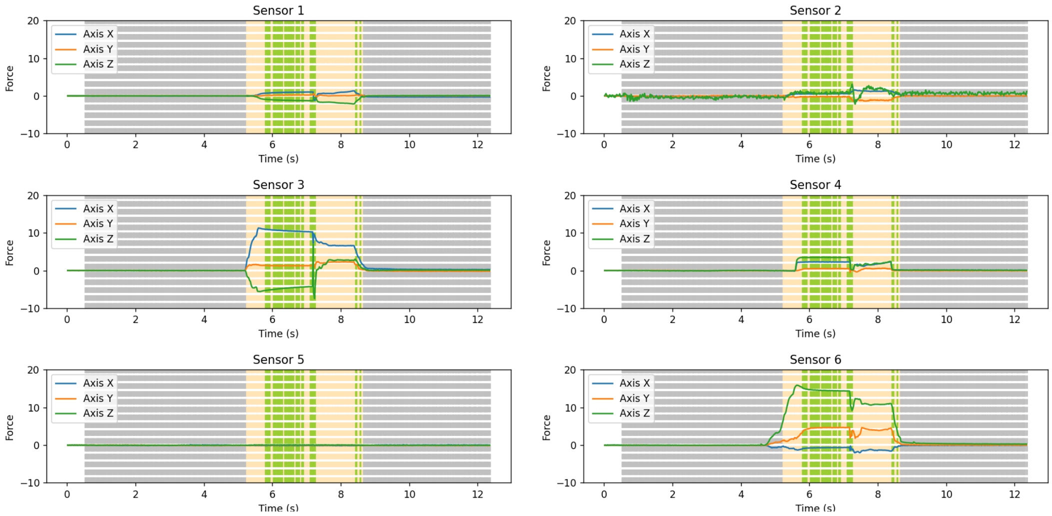


Figure 6.53: Test result for participant 2 after applying the trained machine-learning model to 'put-down' action with the empty plastic bottle.

Object 3 - Basket-ball

Pass

In Figure 6.54, it is possible to see the results obtained from applying the trained machine learning model to the passing action using the basket-ball, accomplished by Participant 2. In this case, the safe slip was correctly identified. However, another trial fro the same action in Figure 6.55 reveals a misplaced risky slip.

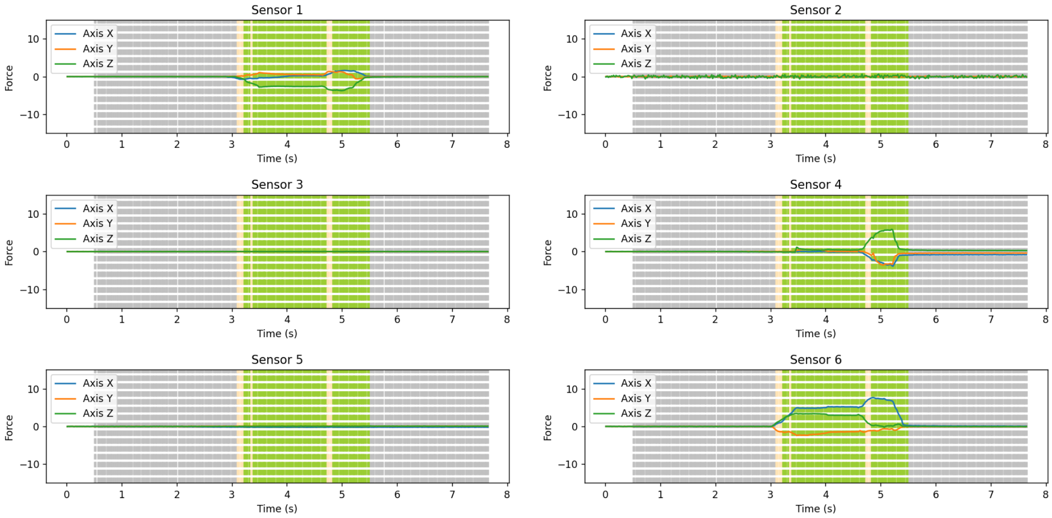


Figure 6.54: Test result for participant 2 after applying the trained machine-learning model to 'pass' action with the basket-ball.

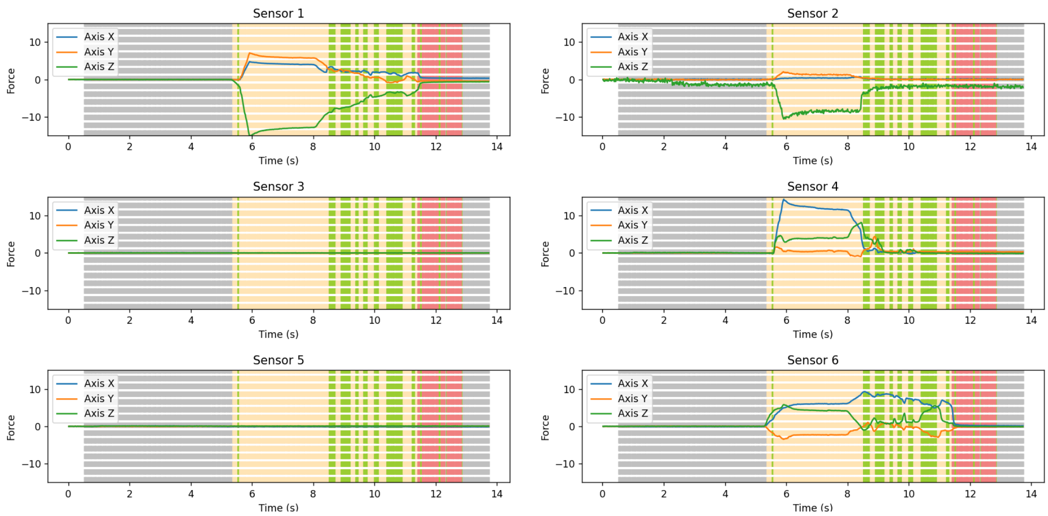


Figure 6.55: Test result for participant 2 after applying the trained machine-learning model to 'pass' action with the basket-ball.

Pull

In Figure 6.56, it is possible to see the results obtained from applying the trained machine learning model to the pulling action using the basket-ball, accomplished by Participant 2. In this case, as with all other analyses conducted for this action, the risky slip was correctly identified.

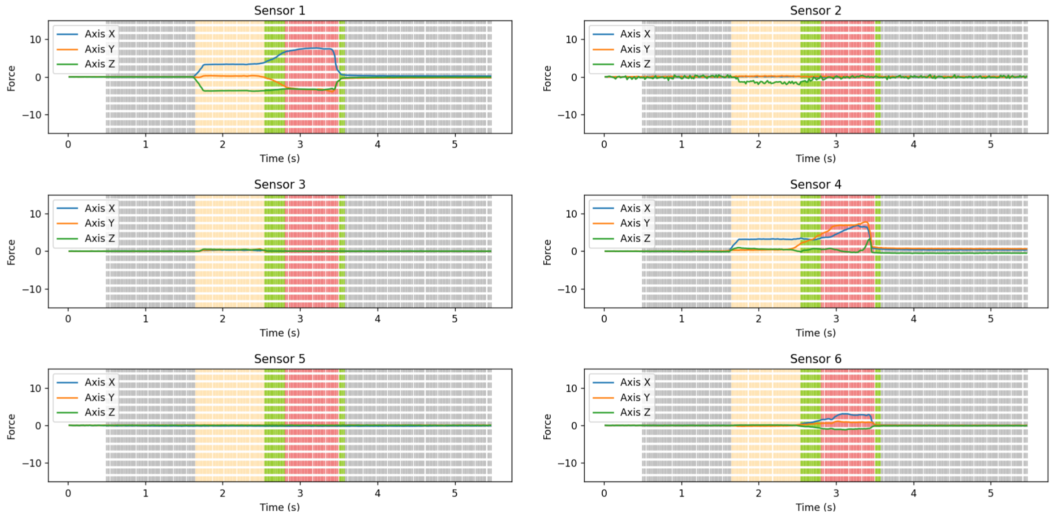


Figure 6.56: Test result for participant 2 after applying the trained machine-learning model to 'pull' action with the basket-ball.

Put-down

In Figure 6.57, it is possible to see the results obtained from applying the trained machine learning model to the putting-down actions using the basket-ball, accomplished by Participant 2. In this case, the put-down and the release were correctly identified. However, in other trials of the same action, as the example in Fig. 6.58, the model is not able to recognize the put-down.

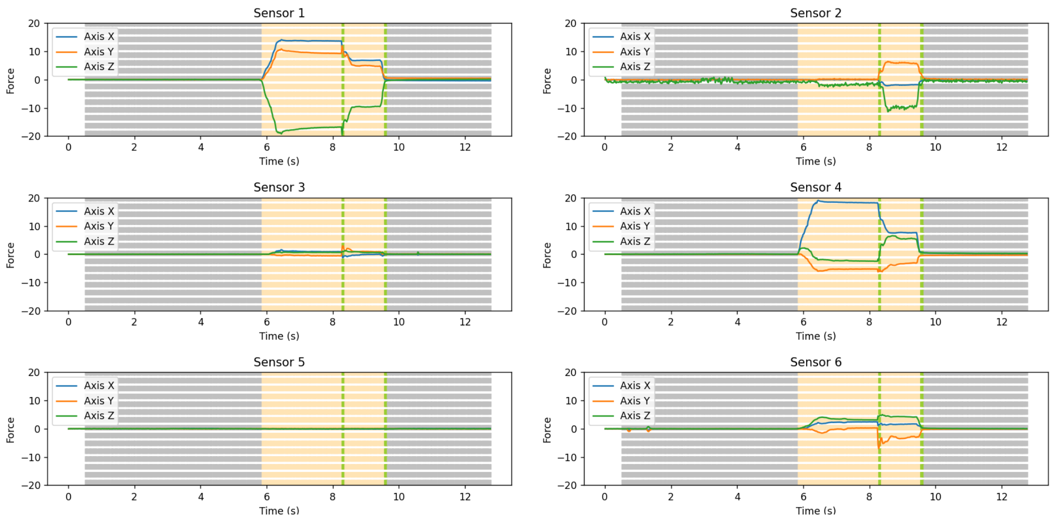


Figure 6.57: Test result for participant 2 after applying the trained machine-learning model to 'put-down' action with the basket-ball.

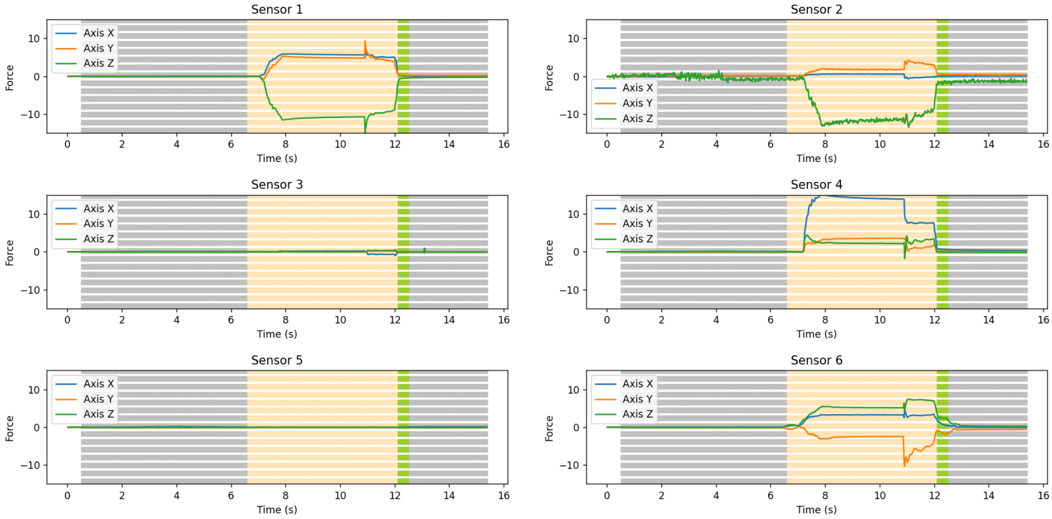


Figure 6.58: Test result for participant 2 after applying the trained machine-learning model to 'put-down' action with the basket-ball.

Object 4 - Hard-ball

Pass

In Figure 6.59, it is possible to see the results obtained from applying the trained machine learning model to the passing actions using the hard-ball, accomplished by Participant 2. In this case, the safe slip was almost correctly identified. However, in other trials of the same action, as the example in Fig. 6.60, the model is not able to recognize anything other than grasp.

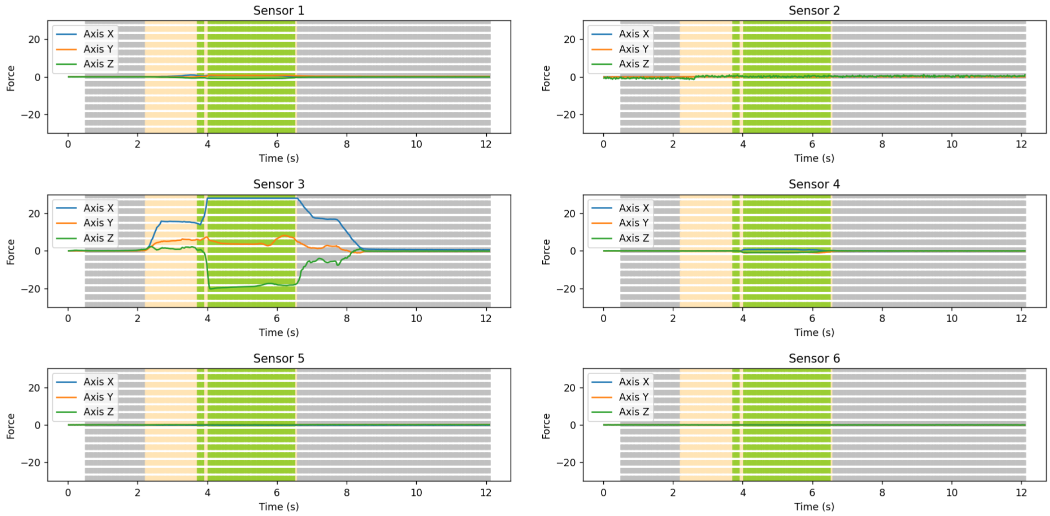


Figure 6.59: Test result for participant 2 after applying the trained machine-learning model to 'pass' action with the hard-ball.

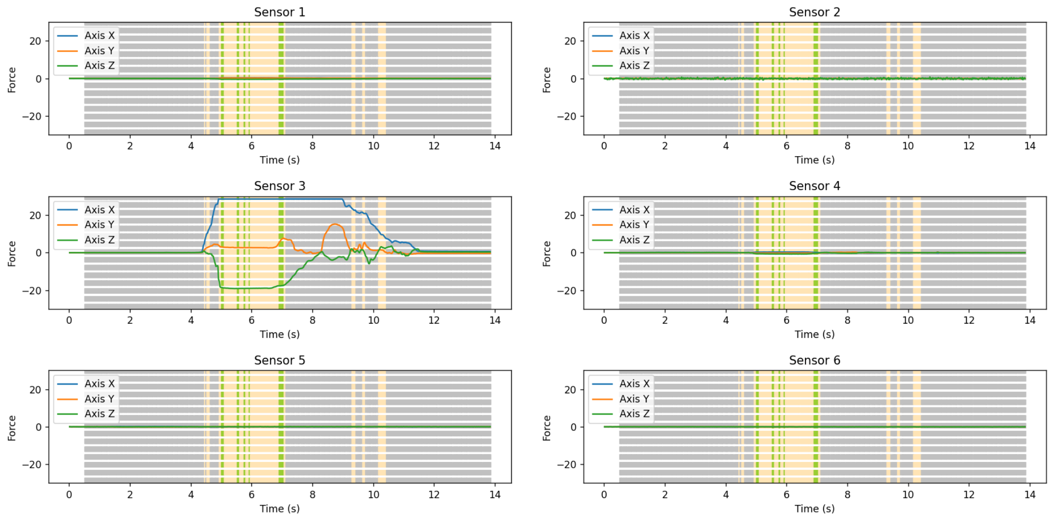


Figure 6.60: Test result for participant 2 after applying the trained machine-learning model to 'pass' action with the hard-ball.

Fall & Put-down

In figures 6.61 and 6.62, it is possible to see the results obtained from applying the trained machine learning model to the falling and putting-down actions using the hard-ball, accomplished by Participant 2. In these two cases, the only prediction shown is "nothing". An explanation for this will be given in the discussion.

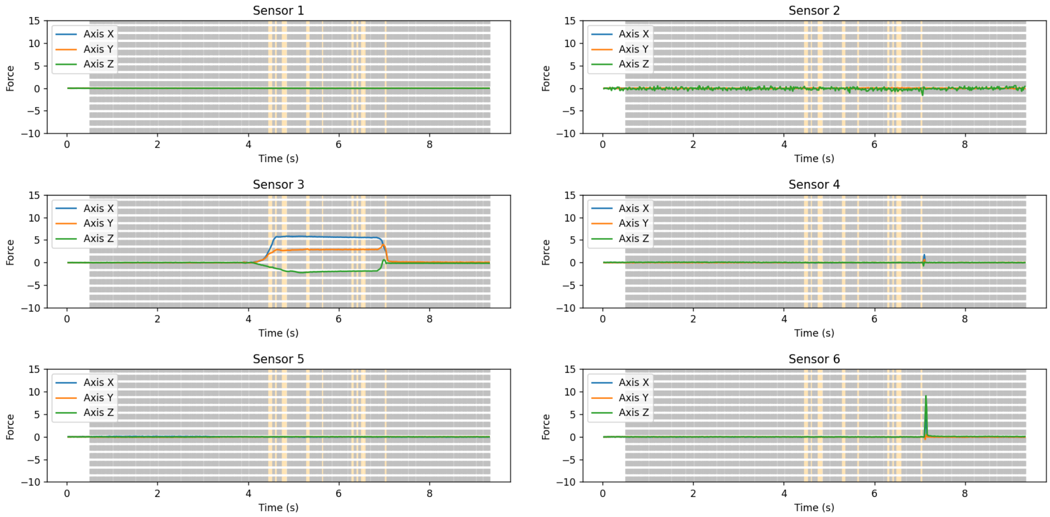


Figure 6.61: Test result for participant 2 after applying the trained machine-learning model to 'fall' action with the hard-ball.

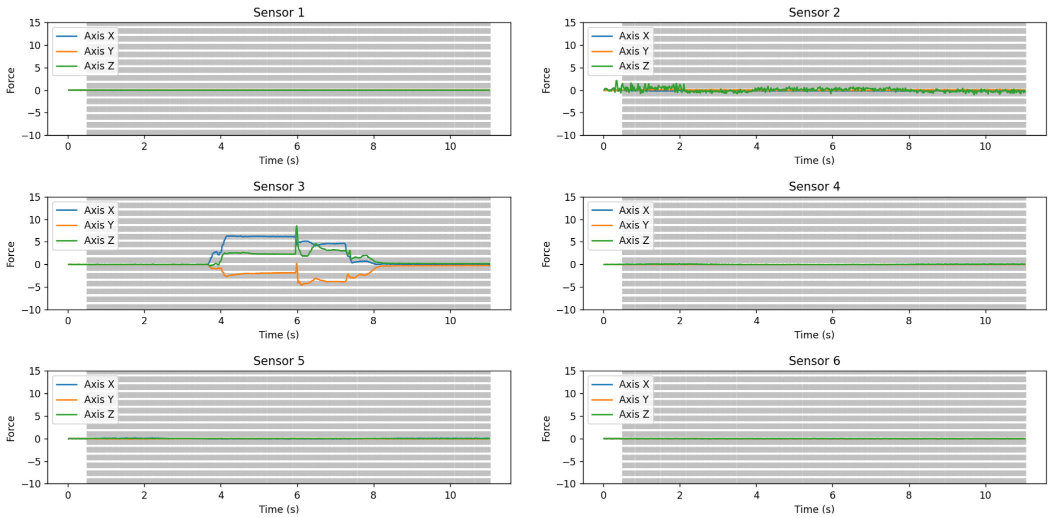


Figure 6.62: Test result for participant 2 after applying the trained machine-learning model to 'put-down' action with the hard-ball.

Object 5 - Full plastic bottle

Pass

In Figure 6.63, it is possible to see the results obtained from applying the trained machine learning model to the passing actions using the full plastic bottle, accomplished by Participant 2. In this scenario, the safe slip was not accurately identified, as evidenced by the ground truth depicted in Fig. 6.5. Nevertheless, no risky slip was detected, leading to the conclusion that the outcome is partially corrected. This outcome is consistent with all trials for the same action.

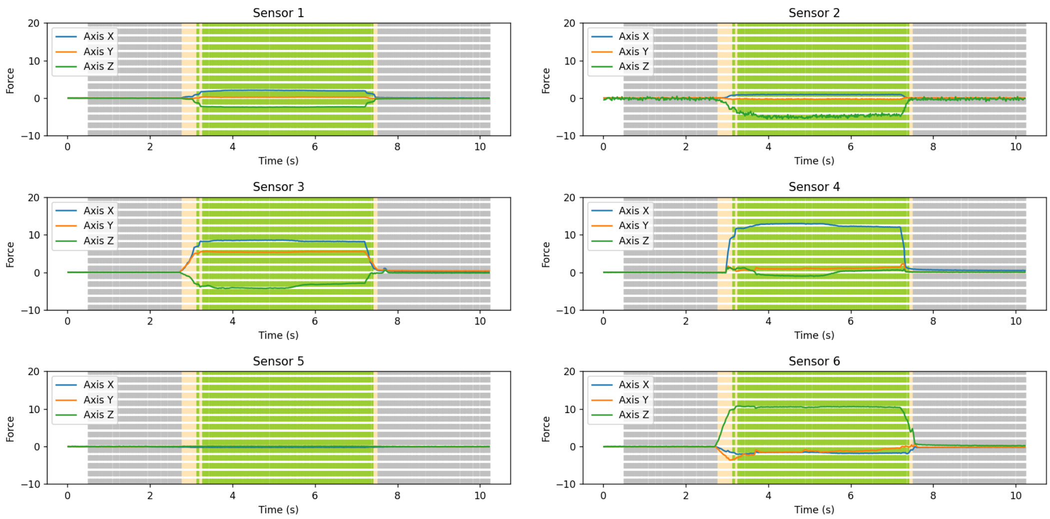


Figure 6.63: Test result for participant 2 after applying the trained machine-learning model to 'pass' action with the full plastic bottle.

Fall

In Figure 6.64, it is possible to see the results obtained from applying the trained machine learning model to the falling action using the full plastic bottle, accomplished by Participant 2. In this case, the risky slip was not correctly identified in any of the trials for this same action.

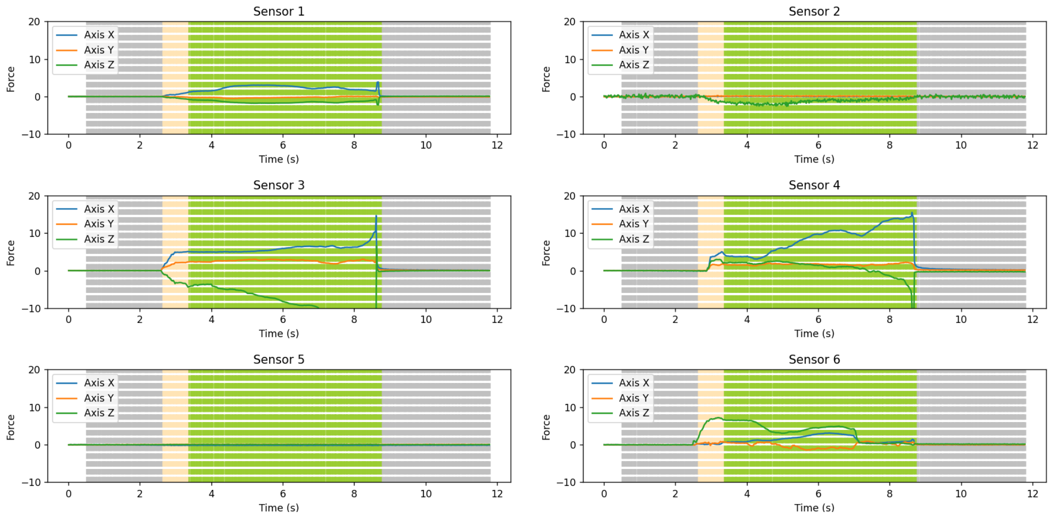


Figure 6.64: Test result for participant 2 after applying the trained machine-learning model to 'fall' action with the full plastic bottle.

Put-down

In Figure 6.57, it is possible to see the results obtained from applying the trained machine learning model to the putting-down actions using the full plastic bottle, accomplished by Participant 2. In this case, the put-down and the release were correctly identified in all the trials for this same action.

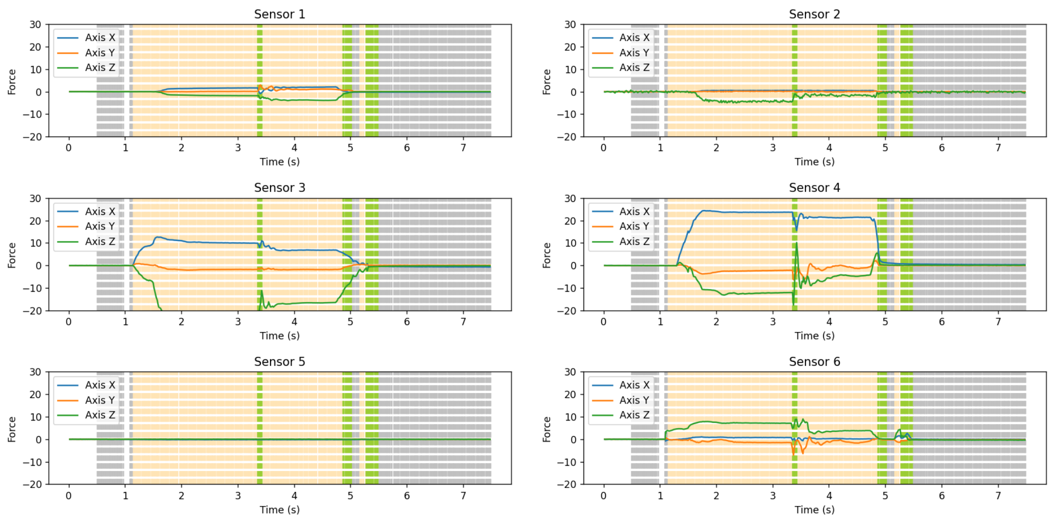


Figure 6.65: Test result for participant 2 after applying the trained machine-learning model to 'put-down' action with the full plastic bottle.

Chapter 7

Discussion & Future Works

After a careful analysis of the results, it emerged that the experiment produced promising outcomes, although they could be subject to improvements. Observing individual actions such as passing, it can be noted that for each object the action was correctly recognized at least once. However, a problem identified concerns the speed of passing: if executed slightly faster than expected, with a sudden movement at the beginning or end of the action, there is a chance of erroneously identifying a risky slip. This phenomenon appears to primarily affect lightweight objects such as the soft-ball, basket-ball, and empty plastic bottle, while no risky slips were observed in heavy objects like the hard-ball and full plastic bottle. This discrepancy could be attributed to how the model identifies risky slipping with pulling for lightweight objects and with falling for heavy ones, making pulling easily misunderstood with passing if too much force is exerted.

Regarding the pulling and falling actions, they were correctly recognized for all objects and recorded cases. However, the visual results concerning these risky actions may slightly differ from the ground truth in 6.5b. This discrepancy does not compromise the recognition of the risky action, as during slipping, there may be ambiguous moments that do not affect the correct identification of the risky action, because as soon as the risky action is recognized, the shared control will react accordingly, and therefore what comes after is not considered important.

Finally, concerning the put-down action, the situation is more complex. For objects like the basket-ball, empty plastic bottle, and full plastic bottle, the results are acceptable with correct recognition of both put-down and release, although, some instances might be erroneously identified as safe slips instead of grasps, probably due to insufficiently stable grip. However, since safe slip does not require intervention, this discrepancy does not pose a problem. As for the soft and hard balls, the situation is somewhat different. For the former, the model can recognize the disturbance during putting-down and release but fails to distinguish the variation after the object's put-down, preventing it from exiting the safe slip state. For the hard-ball, the lack of intensity in the put-down prevents the model from recognizing it. Nevertheless, as mentioned earlier, since no risky slips were observed, the put-down recognition is considered satisfactory in all cases.

After analyzing the collected data with the trained model, it was important to check for any biases in the machine-learning model. As a first step, an object unknown to the model was tested; the author of this project performed the same movements but this time with a toy apple. The results were quite good and in line with expectations. The pass was correctly recognized in all cases, as well as the pull. However, aware of the program's sensitivity, the pass was executed carefully to avoid sudden movements, which worked. However, in the development of a shared control, this aspect will certainly require further attention and improvement. Regarding the put-down, the results obtained were, in no case, identical to the ground truth due to the recognition of numerous small safe slips. Nevertheless, as anticipated, since no risky slips were observed, the results are not considered erroneous.

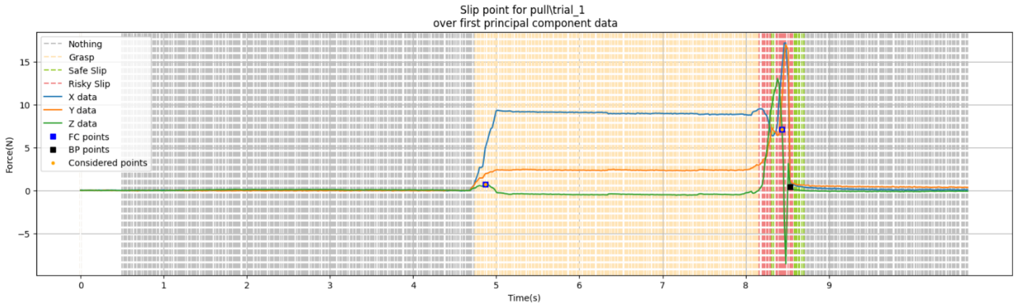
As for the human study, the situation differs significantly for the two participants. After viewing the explanatory videos, the first participant did not replicate exactly what was shown, while the second followed the instructions carefully. Let's now compare the results.

Regarding the passing action, the results for the second participant are very similar to those obtained by the author, with the only exception of passing for the full plastic bottle, which shows slightly different results from the others. This action, as explained in Chapter 6.2.1, has a slightly different execution and was not correctly identified because the weight of the bottle was not sufficiently attenuated, and consequently, only a minimal variation is visible in the plots of the various sensors, Fig. 6.63. Conversely, for the first participant, no cases of safe slip were correctly recognized for the basket-ball, the empty plastic bottle, and the heavy-ball. Regarding the soft-ball, the safe slip was correctly recognized without any risky slips observed in any case. Finally, for the full plastic bottle, the safe slip was correctly recognized in one case but erroneously labelled as risky in another. This happens due to a lack of correct execution of the action; indeed, while participant 1 did not exert enough force in lifting the bottle, participant 2 used an excessive amount, enough to move the bottle. At this point, it seems correct to say that the action was labelled as 'risky slip' because it was a reproduction of the pulling action.

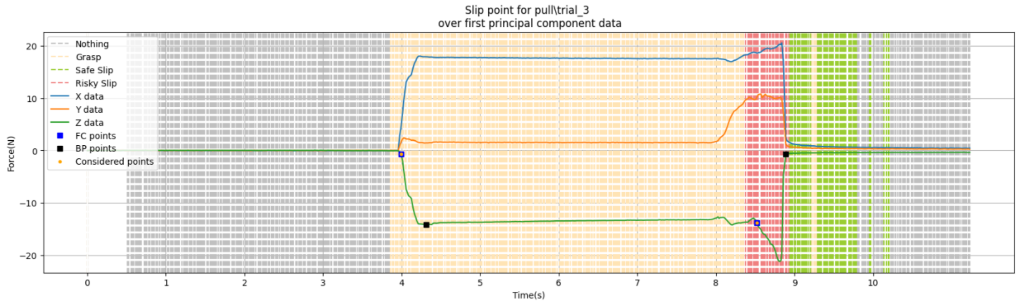
As for the pulling and falling actions, the results for the first participant were quite inaccurate for all objects except for the full bottle. This could be due to the fact that the first participant performed the pulling action so quickly that in almost all cases the model, trained on a more controlled pull, was unable to recognize it. As for the full plastic bottle, the fall was recognized in all cases. For the second participant, the results for lightweight objects are similar to those of the author, correctly recognizing the risky slip. For heavy objects, the situation is particular: for the hard-ball, it predicts "nothing", as will happen for the put-down action, probably due to its incorrect positioning during the pulling action. Indeed, the hard-ball, due to its shape, weight, and hardness, was the most difficult object to study. As for the full plastic bottle, from the figure 6.64, it is very evident that slipping is occurring, but it is probably labelled only as a safe slip because it is extremely slow and controlled.

Regarding the put-down, the results for the second participant, following the overall trend, were good for all objects except for the hard-ball, as anticipated. In general, the model was able to recognize the pulling and release actions at least once for each object. However, it is important to note that in some cases, the put-down action was not recognized. Regarding the first participant, the recognition of put-down and release was better than the previous actions. However, due to the excessive force with which the put-down was executed, as already explained for passing the full plastic bottle, many cases were erroneously recognized as risky. The only object for which the put-down was never recognized as risky is the full plastic bottle because, after viewing the video, it was emphasized not to apply excessive force. This was simply to avoid damaging the sensors, as they are very delicate and the object used was the heaviest in the project.

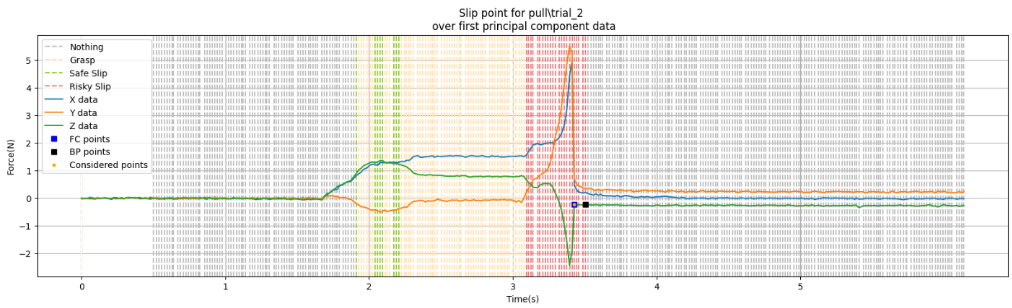
After analyzing all the results from the online test and the human study, an additional step was taken to further validate the efficacy of slip detection methods employed in this study. The data used for testing underwent the two traditional methods used for slip detection, namely friction cone and bandpass filtering, which until now were only employed to identify critical points for training the machine learning model. The results were remarkably satisfactory. For all objects in the online study except the soft-ball, in at least one trial, the machine learning model successfully predicted the risky slip, as evidenced by Figure 7.1a. Even for the soft-ball, although the results were not as strong, the model consistently identified the critical point alongside to the traditional methods. Moreover, this success extended to the unknown object, as depicted in Figure 7.1b. Finally, in the context of the human study, similar positive outcomes were observed, albeit with a focused analysis only on the objects previously identified as correctly recognized. For Participant 1, this encompassed the full plastic bottle, while for Participant 2, it included all lightweight objects.



(a) This figure showcases the outcomes regarding the basket-ball, utilized within the context of the online test. Demonstrating the capabilities of the machine learning model developed in this study, it exhibits the model can anticipate risky slips moments before the traditional method.



(b) This figure showcases the outcomes regarding the apple, an unknown object added to the online test. Demonstrating the capabilities of the machine learning model developed in this study, it exhibits the model can anticipate risky slips moments before the traditional method.



(c) This figure showcases the outcomes regarding the soft-ball, utilized within the context of the human study for Participant 1. Demonstrating the capabilities of the machine learning model developed in this study, it exhibits the model can anticipate risky slips moments before the traditional method.

Figure 7.1: Figures illustrating the comparison between traditional slip detection methods alongside the method developed in this project. Each figure focuses specifically on one of the six sensors, precisely sensor 4, positioned on the middle finger.

From the results obtained from the study, which highlight both strengths and weaknesses, several roads for future improvement of this work emerge. Initially, it is evident that the quantity of data used to train the model is insufficient. This is particularly critical in the passing action, which could benefit from considering a greater variety of cases. It is important to emphasize that the accuracy of predictions made by machine-learning algorithms is usually directly proportional to the size and diversity of the dataset used for training.

Another crucial aspect to consider concerns the sensors. During the recording of actions, there were repeated instances of magnets detaching. This not only caused practical inconveniences but also influenced the performance of the sensors, which could not be re-calibrated with every occurrence. Integrating the sensors directly into the prosthesis could significantly enhance the authenticity and reliability of the actions performed. Another solution, which maintains the external structure of the sensors, could be to encapsulate the chip and the magnet and introduce an air gap between them to mitigate the uncompressibility of silicone and glue.

Furthermore, for future developments, the inclusion of EMG recognition along with the study of Hall-effect sensor outputs could be evaluated for a more comprehensive study. Additionally, expanding the study to include various positions in which the hand interacts with the external environment could be beneficial. Indeed, this multi-directional approach could more accurately reflect the challenges and dynamics encountered in real life.

Finally, instead of implementing a shared control, an option to consider could be the use of vibrational feedback to assess the performance of the study. This could provide a more direct and immediate method for evaluating the response and interaction of the prosthesis with the user.

Chapter 8

Conclusions

During the course of this project, an innovative approach was developed to predict slipping using tactile sensors and a machine-learning model. The proposed method has proven effective in distinguishing slipping from grasping and recognizing various types of slipping, differentiating between safe and risky situations. However, while the preliminary results are promising, it has become clear that there is still room for improvement.

The human study and the study with an unknown object have certainly allowed us to understand that the model is capable of recognizing movements even when performed with unstudied objects, but it is very susceptible to changes in the way these actions are developed.

In the discussion phase of future work, several critical points requiring further development have been identified. For example, in the passing action, the analysis revealed a lack of sufficient data for model training, suggesting the need to acquire more cases to improve the accuracy and reliability of predictions. Furthermore, the stability of the sensors used is another area requiring improvement. The occasional detachment of magnets has influenced the sensor's performance, indicating the need for technical solutions to enhance their structural and functional integrity.

Despite these challenges, the proposed approach has proven capable of correctly predicting the situation in many cases, highlighting satisfactory results. Nonetheless, there remains an opportunity to enhance the system's performance. Consequently, potential changes and new directions for future developments are discussed in Chapter 7.

Appendix A

Code used to train the machine-learning model

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import os
import pickle
from scipy import signal, interpolate
from numpy.linalg import eigh
import sys
import math
import random
from scipy.signal import butter, lfilter, sosfilt, sosfreqz, filtfilt
import time
from pathlib import Path

from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler, PolynomialFeatures

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix,
    accuracy_score
from sklearn.model_selection import train_test_split

import joblib

# CLASS TO PROCESS DATA FROM EXCEL FILES

class DataProcessor_fromExcel:
    def __init__(self, filepath):
        self.filepath = filepath
        self.trial_data = None

    def load_data(self):
        self.trial_data = pd.read_csv(self.filepath)

    def correct_magnet_orientation(self):
        for column in self.trial_data.columns[1:]:
```

```
            if self.trial_data[column].iloc[0] < 0 and column[0] == 'Z':
                self.trial_data[column] = self.trial_data[column] * -1

    def correct_offset(self):
        for column in self.trial_data.columns[1:]:
            offset = np.mean(self.trial_data[column].iloc[:5])
            self.trial_data[column] = self.trial_data[column] - offset

    def apply_moving_average(self, window_size=5):
        for column in self.trial_data.columns[1:]:
            self.trial_data[column] = self.trial_data[column].rolling(window=window_size).mean()

    def clean_data(self):
        self.trial_data = self.trial_data.dropna().reset_index(drop=True)

    def normalize_time(self):
        self.trial_data['Time'] = (self.trial_data['Time'] - self.trial_data['Time'].iloc[0])

    def process_data(self):
        self.load_data()
        self.normalize_time()
        data=self.trial_data.iloc[:, 1:]
        time=self.trial_data.iloc[:, 0]

        return data, time

    def process_all_files(self, folder_path, num_files):
        all_data = []
        all_time_steps = []

        for file_number in range(1, num_files+1):
            filename = f'trial_{file_number}.csv'
            filepath = os.path.join(folder_path, filename)

            data_processor = DataProcessor_fromExcel(filepath)
            data, time_steps = data_processor.process_data()

            all_data.append(data)
            all_time_steps.append(time_steps)

        return all_data, all_time_steps


# CLASS CONTAINING BANDPASS FILTER FOR ALL SENSORS AND ALL FILES

class BandpassFilter:
    def __init__(self):

        #Resonance Frequencies for the bandpass
        self.w=[10, 15, 20, 25, 30, 35, 40, 45, 50, 55]

        #Filter Buffers
        self.in_b=[]
        self.hp_b=[]
        self.bp_b1=[]
        self.bp_b2=[]
        self.lp_b=[]

        #Sampling period
```

```
        self.fs=150
        self.T=1/self.fs
        self.K=2/self.T

        #Quality Factor
        self.g=0.1

        #LP frequency
        self.fc=75

        #initialize buffers
        for _ in range(3):
            a=[0]
            self.in_b.append(a)
            self.hp_b.append(np.zeros(np.shape(a)))
            temp=[]
            for _ in range(len(self.w)):
                temp.append(np.zeros(np.shape(a)))
            self.bp_b1.append(temp)
            self.bp_b2.append(temp)
        self.lp_b.append(temp)
        self.in_b.pop(0)
        self.bp_b2.pop(0)


    def highpass(self):
        input_minus_2 = np.array(self.in_b[-2])
        input_minus_1 = np.array(self.in_b[-1])
        hp_minus_1 = np.array(self.hp_b[-1])

        output = self.K * (input_minus_1 - input_minus_2) - hp_minus_1
        self.hp_b.append(output)
        self.hp_b.pop(0)


    def bandpass1(self):
        temp=[]
        for i, w in enumerate(self.w):
            w0=2*w*np.pi
            a=(self.K**2)+2*self.g*w0*self.K+(w0**2)
            b=w0**2
            value=(b*self.hp_b[-1] + 2*b*self.hp_b[-2] + b*self.hp_b[-3] - (2*b
    -2*(self.K**2))*self.bp_b1[-1][i] - (self.K**2-2*self.g*w0*self.K+b)*self.
    bp_b1[-2][i])/a
            temp.append(value)
        self.bp_b1.append(temp)
        self.bp_b1.pop(0)

    def bandpass2(self):
        temp=[]
        for i, w in enumerate(self.w):
            w0=2*w*np.pi
            a=(self.K**2)+2*self.g*w0*self.K+(w0**2)
            b=w0**2
            temp.append((b*self.bp_b1[-1][i]+2*b*self.bp_b1[-2][i]+b*self.bp_b1
    [-3][i]-(2*b-2*(self.K**2))*self.bp_b2[-1][i]-(self.K**2-2*self.g*w0*self.K+b)
    *self.bp_b2[-2][i])/a)
        self.bp_b2.append(temp)
        self.bp_b2.pop(0)

    def lowpass(self):
```

```
        fc=self.fc*2*np.pi
        a=self.K+fc
        temp=[]
        for i, w0 in enumerate(self.w):
            temp.append((fc*(self.bp_b2[-1][i]+self.bp_b2[-2][i])-(fc-self.K)*self.lp_b[-1][i])/a)
        self.lp_b.append([abs(value) for value in temp])
        self.lp_b.pop(0)


    def apply_bandpass_filter(self, data, time_steps, num_sensors):
            filtered_data = []

            for sensor_idx in range(num_sensors):
                sensor_data = data.iloc[:, sensor_idx * 3 : (sensor_idx + 1) * 3]
                filtered_sensor_data = []
                for i in range(len(sensor_data)):
                    signal = sensor_data.iloc[i].values
                    # Update filter states
                    self.in_b.append(signal)
                    self.in_b.pop(0)
                    self.highpass()
                    self.bandpass1()
                    self.bandpass2()
                    self.lowpass()

                    # Get the output from the lowpass filter for each frequency component
                    result = np.array(self.lp_b[-1])
                    result = result.flatten()
                    filtered_sensor_data.append(np.insert(result, 0, time_steps.iloc[i]))

                filtered_data.append(np.array(filtered_sensor_data))

            return np.array(filtered_data)

    def find_peaks(self, data, threshold):
        peaks = []
        peak_start = None

        for i in range(1, len(data)):
            if data[i] > threshold:
                if peak_start is None:
                    peak_start = i
            elif data[i] <= threshold and peak_start is not None:
                peak_end = i - 1
                peaks.append((peak_start, peak_end))
                peak_start = None

        if peak_start is not None:
            peaks.append((peak_start, len(data) - 1))

        return peaks

    def calculate_slip_points_bp_tresholds(self, data, filtered_data, time_steps, num_sensors, treshold):
        slip_bp=[]
        filtered_data_single_freq_x=[[] for _ in range(6)]
        for _ in range(6):
            for __ in range(len(filtered_data[0][:,0])):
```

```python
                filtered_data_single_freq_x[_].append([])

        filtered_data_single_freq_y=[[] for _ in range(6)]
        for _ in range(6):
            for __ in range(len(filtered_data[0][:,0])):
                filtered_data_single_freq_y[_].append([])


        for j in range(0, num_sensors):
            for i in range(len(filtered_data[j][:,0])):
                filtered_data_single_freq_x[j][i]=0
                filtered_data_single_freq_y[j][i]=0
                for k in range(0,10):
                    if filtered_data[j][i,k*3+1]>filtered_data_single_freq_x[j][i]:
                        filtered_data_single_freq_x[j][i]=filtered_data[j][i,k*3+1]
                    if filtered_data[j][i,k*3+2]>filtered_data_single_freq_y[j][i]:
                        filtered_data_single_freq_y[j][i]=filtered_data[j][i,k*3+2]

        for j in range(0, num_sensors):
            value_slip_bp = np.zeros((filtered_data.shape[1]))
            peaks_x0=0
            peaks_y0=0
            peaks_x=self.find_peaks(np.array(filtered_data_single_freq_x[j]), treshold)
            peaks_y=self.find_peaks(np.array(filtered_data_single_freq_y[j]), treshold)

            for n in range(len(peaks_x)):
                i=peaks_x[n][0]
                if i-peaks_x0>15:
                    if time_steps[i] > 0.5 and time_steps[len(time_steps)-1]-time_steps[i] > 0.5:
                        value_slip_bp[i] = 1
                peaks_x0=peaks_x[n][1]
            for n in range(len(peaks_y)):
                i=peaks_y[n][0]
                if i-peaks_y0>15:
                    if time_steps[i] > 0.5 and time_steps[len(time_steps)-1]-time_steps[i] > 0.5:
                        value_slip_bp[i] = 1
                peaks_y0=peaks_y[n][1]

            slip_bp.append(np.array(value_slip_bp))

        # putting together all the results for different frequencies and different sensors
        slip_bp=np.array(slip_bp)
        slip_bp_single=np.zeros(len(time_steps))
        for sensor_num in range(num_sensors):
            for i in range(len(time_steps)):
                if slip_bp[sensor_num][i]==1:
                    slip_bp_single[i]=1

        count_1 = 0
        count_t=0
        for i in range(len(slip_bp_single)):
            if slip_bp_single[i] == 1:
```

```
                # Reset counter after interval considered
                if i > count_t+30:
                    count_1 = 0

                if count_1 == 0:
                    count_1 += 1
                    count_t=i
                else:
                    # set to 0 following 1 in the interval
                    slip_bp_single[i] = 0

        return slip_bp, slip_bp_single


    def calculate_slip_points_bp(self, data, filtered_data, time_steps, num_sensors, directories):
        treshold=5000

        if 'basketball\pass/' in directories:
            treshold=450
        elif 'basketball\pull/' in directories:
            treshold=1450
        elif 'basketball\putdown/' in directories:
            treshold=1400

        elif 'softball\pass/' in directories:
            treshold=800
        elif 'softball\pull/' in directories:
            treshold=1500
        elif 'softball\putdown/' in directories:
            treshold=1000

        if 'emptybottle\pass/' in directories:
            treshold=500
        elif 'emptybottle\pull/' in directories:
            treshold=1500
        elif 'emptybottle\putdown/' in directories:
            treshold=1000

        elif 'hardball\pass/' in directories:
            treshold=500
        elif 'hardball/fall/' in directories:
            treshold=750
        elif 'hardball\putdown/' in directories:
            treshold=950

        elif 'colabottle\pass/' in directories:
            treshold=450
        elif 'colabottle/fall/' in directories:
            treshold=405
        elif 'colabottle\putdown/' in directories:
            treshold=1500

        # print(treshold)

        slip_bp, slip_bp_single= self.calculate_slip_points_bp_tresholds(data, filtered_data, time_steps, num_sensors, treshold)

        return slip_bp, slip_bp_single

```

```
# CLASS TO CALCULATE FIRST PRINCIPAL COMPONENT AND FRICTION CONE RATIO (for slip point) => CREATION OF WINDOWS
class PCABasedFeatureExtraction:
    def __init__(self, num_selected_sensors=None):
        self.num_selected_sensors = num_selected_sensors

    def apply_pca(self, data_x, data_y, data_z):
        sc = StandardScaler()
        sc.fit(data_x)
        scaled_data_x = sc.transform(data_x)

        sc.fit(data_y)
        scaled_data_y = sc.transform(data_y)

        sc.fit(data_z)
        scaled_data_z = sc.transform(data_z)

        cov_matrix = np.cov(scaled_data_y, rowvar=False)

        egnvalues, egnvectors = np.linalg.eigh(cov_matrix)

        total_egnvalues = sum(egnvalues)
        var_exp = [(i / total_egnvalues) for i in sorted(egnvalues, reverse=True)]

        cum_sum_exp = np.cumsum(var_exp)
        if self.num_selected_sensors is None:
            self.num_selected_sensors = np.argmax(cum_sum_exp >= 0.85) + 1

        pca = PCA(n_components=self.num_selected_sensors)
        data_pca_x = pd.DataFrame(pca.fit_transform(scaled_data_x), columns=range(1, self.num_selected_sensors + 1))
        data_pca_y = pd.DataFrame(pca.fit_transform(scaled_data_y), columns=range(1, self.num_selected_sensors + 1))
        data_pca_z = pd.DataFrame(pca.fit_transform(scaled_data_z), columns=range(1, self.num_selected_sensors + 1))

        return data_pca_x, data_pca_y, data_pca_z

    def calculate_slip_points(self, pca_data_x, pca_data_y, pca_data_z, time_steps):
        mi_x = np.divide(pca_data_x.iloc[:, 0], pca_data_z.iloc[:, 0])
        mi_y = np.divide(pca_data_y.iloc[:, 0], pca_data_z.iloc[:, 0])
        value_slip = np.zeros(len(time_steps))
        value = 0

        for i in range(1, len(time_steps)):
            if time_steps[i] > 1 and time_steps[len(time_steps)-1]-time_steps[i] > 1:
                if (abs(mi_y.iloc[i] - mi_y.iloc[i-1]) > 1 and abs(pca_data_y.iloc[i+15, 0]-pca_data_y.iloc[i-15, 0]) > 2) or \
                   (abs(mi_y.iloc[i] - mi_y.iloc[i-1]) > 1 and abs(pca_data_y.iloc[i, 0]-pca_data_y.iloc[i+15, 0]) > 2) or \
                    (abs(mi_y.iloc[i] - mi_y.iloc[i-1]) > 5 and abs(pca_data_z.iloc[i-5, 0]-pca_data_z.iloc[i+5, 0]) > 2):
                    if time_steps[i] - time_steps[value] <= 0.5:
                        # Check if the current value is higher than the previously recorded maximum
                        if abs(mi_y.iloc[i]) > abs(mi_y.iloc[value]):
                            value_slip[value] = 0
                            value = i
                            value_slip[i] = 1
```

```
                    else:
                        value = i
                        value_slip[i] = 1


            elif (abs(mi_x.iloc[i] - mi_x.iloc[i-1]) > 1 and abs(pca_data_x.iloc[i+15, 0]-pca_data_x.iloc[i-15, 0]) > 2) or \
                                (abs(mi_x.iloc[i] - mi_x.iloc[i-1]) > 1 and abs(pca_data_x.iloc[i, 0]-pca_data_x.iloc[i+15, 0]) > 2):
                if time_steps[i] - time_steps[value] <= 0.5:
                    # Check if the current value is higher than the previously recorded maximum
                    if abs(mi_x.iloc[i]) > abs(mi_x.iloc[value]):
                        value_slip[value] = 0
                        value = i
                        value_slip[i] = 1
                else:
                    value = i
                    value_slip[i] = 1


        return value_slip, mi_x, mi_y

    def create_interval_data(self, data, start_index, end_index, value_slip_softhard):
        # Sublist containing the relevant data
        interval_data=[]
        interval_data = pd.DataFrame(data.iloc[start_index:(end_index + 1), :].values.tolist())
        #print('interval_data',np.shape(interval_data))
        interval_data['Value'] = value_slip_softhard
        #print('interval_data',np.shape(interval_data))
        #print('slip_intervals', len(slip_intervals), slip_intervals[0].shape)
        return interval_data

    def create_slip_windows(self, data, time_steps, value_slip, slip_bp_single, directories, slip_intervals, grasp_intervals):

        zero, uno, two, three, four, five, six, seven = 0, 0, 0, 0, 0, 0, 0, 0
        basketball_grasp, basketball_pass, basketball_pull, basketball_fall, basketball_putdown, basketball_release = 0,0,0,0,0,0
        softball_grasp, softball_pass, softball_pull, softball_fall, softball_putdown, softball_release =0,0,0,0,0,0
        emptybottle_grasp, emptybottle_pass, emptybottle_pull, emptybottle_fall, emptybottle_putdown, emptybottle_release = 0,0,0,0,0,0
        heavyball_grasp, heavyball_pass, heavyball_pull, heavyball_fall, heavyball_putdown, heavyball_release = 0,0,0,0,0,0
        cola_grasp, cola_pass, cola_pull, cola_fall, cola_putdown, cola_release = 0,0,0,0,0,0
        basketball_nothing, softball_nothing, emptybottle_nothing, heavyball_nothing, cola_nothing = 0,0,0,0,0

        basketball_grasp_pass, softball_grasp_pass, emptybottle_grasp_pass, heavyball_grasp_pass, cola_grasp_pass = 0,0,0,0,0
        basketball_grasp_pass_keep, softball_grasp_pass_keep, emptybottle_grasp_pass_keep, heavyball_grasp_pass_keep, cola_grasp_pass_keep= 0,0,0,0,0
        basketball_grasp_pull,  softball_grasp_pull, emptybottle_grasp_pull, heavyball_grasp_pull, cola_grasp_pull= 0,0,0,0,0
```

```
    basketball_grasp_pull_keep, softball_grasp_pull_keep, emptybottle_grasp_pull_keep, heavyball_grasp_pull_keep, cola_grasp_pull_keep= 0,0,0,0,0
    basketball_grasp_fall, softball_grasp_fall, emptybottle_grasp_fall, heavyball_grasp_fall, cola_grasp_fall= 0,0,0,0,0
    basketball_grasp_fall_keep, softball_grasp_fall_keep, emptybottle_grasp_fall_keep, heavyball_grasp_fall_keep, cola_grasp_fall_keep= 0,0,0,0,0
    basketball_grasp_putdown, softball_grasp_putdown, emptybottle_grasp_putdown, heavyball_grasp_putdown, cola_grasp_putdown= 0,0,0,0,0
    basketball_grasp_putdown_keep, softball_grasp_putdown_keep, emptybottle_grasp_putdown_keep, heavyball_grasp_putdown_keep, cola_grasp_putdown_keep= 0,0,0,0,0
    basketball_grasp_release_keep, softball_grasp_release_keep, emptybottle_grasp_release_keep, heavyball_grasp_release_keep, cola_grasp_release_keep= 0,0,0,0,0

    slipperyslip=np.zeros(len(time_steps))
    for i in range(len(time_steps)):
        if slip_bp_single[i]==1 or value_slip[i]==1:
            slipperyslip[i]=1
    count_1 = 0
    count_t=0
    for i in range(len(slipperyslip)):
        if slipperyslip[i] == 1:
            if time_steps[i] > 1 and time_steps[len(time_steps)-1]-time_steps[i] > 1:
                if i > count_t+25:
                    count_1 = 0

                if count_1 == 0:
                    count_1 += 1
                    count_t=i
                else:
                    slipperyslip[i] = 0


    for i in range(len(value_slip)):

        if slipperyslip[i] == 1:
            # Extract data within the range
            start_index = max(0, i - 5)
            end_index = min(len(time_steps), i)

            if 'basketball\pass/' in directories:
                if time_steps[i] < 7:
                    value_slip_softhard = [2] * (end_index - start_index + 1) #grasp
                    two += 1
                    basketball_grasp_pass+=1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                    grasp_intervals[0][0].append(interval_data)
                elif time_steps[i] >= 7:
                    value_slip_softhard = [4] * (end_index - start_index + 1) # passing
                    four += 1
                    basketball_pass+=1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
```

```
                        slip_intervals[0][2].append(interval_data)
                elif 'basketball\pull/' in directories:
                    if  time_steps[i] < 7:
                        value_slip_softhard = [2] * (end_index - start_index + 1)
# grasp
                        two += 1
                        basketball_grasp_pull+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        grasp_intervals[0][2].append(interval_data)
                    elif time_steps[i] >= 7:
                        value_slip_softhard = [6] * (end_index - start_index + 1)
# taking away
                        six += 1
                        basketball_pull+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        slip_intervals[0][3].append(interval_data)
                elif 'basketball\putdown/' in directories:
                    if  time_steps[i] < 7:
                        value_slip_softhard = [2] * (end_index - start_index + 1)
# grasp
                        two += 1
                        basketball_grasp_putdown+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        grasp_intervals[0][6].append(interval_data)
                    elif time_steps[i] >= 7 and  time_steps[i] < 10:
                        value_slip_softhard = [4] * (end_index - start_index + 1)
# putting down
                        four += 1
                        basketball_putdown+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        slip_intervals[0][5].append(interval_data)
                    elif time_steps[i] >= 10:
                        value_slip_softhard = [4] * (end_index - start_index + 1)
# release
                        four += 1
                        basketball_release+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        slip_intervals[0][6].append(interval_data)


                elif 'softball\pass/' in directories:
                    if time_steps[i] < 7:
                        value_slip_softhard = [2] * (end_index - start_index + 1)
# grasp
                        two += 1
                        softball_grasp_pass+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        grasp_intervals[1][0].append(interval_data)
                    elif time_steps[i] >= 7:
                        value_slip_softhard = [4] * (end_index - start_index + 1)
# passing
                        four += 1
                        softball_pass+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
```

```
                        slip_intervals[1][2].append(interval_data)
                elif 'softball\pull/' in directories:
                    if time_steps[i] < 7:
                        value_slip_softhard = [2] * (end_index - start_index + 1)
  # grasp
                        two += 1
                        softball_grasp_pull+=1
                        interval_data=self.create_interval_data(data, start_index,
   end_index, value_slip_softhard)
                        grasp_intervals[1][2].append(interval_data)
                    elif time_steps[i] >= 7:
                        value_slip_softhard = [6] * (end_index - start_index + 1)
  # taking away
                        six += 1
                        softball_pull+=1
                        interval_data=self.create_interval_data(data, start_index,
   end_index, value_slip_softhard)
                        slip_intervals[1][3].append(interval_data)
                elif 'softball\putdown/' in directories:
                    if time_steps[i] < 7:
                        value_slip_softhard = [2] * (end_index - start_index + 1)
  # grasp
                        two += 1
                        softball_grasp_putdown+=1
                        interval_data=self.create_interval_data(data, start_index,
   end_index, value_slip_softhard)
                        grasp_intervals[1][6].append(interval_data)
                    elif time_steps[i] >= 7 and  time_steps[i] < 10:
                        value_slip_softhard = [4] * (end_index - start_index + 1)
  # putting down
                        four += 1
                        softball_putdown+=1
                        interval_data=self.create_interval_data(data, start_index,
   end_index, value_slip_softhard)
                        slip_intervals[1][5].append(interval_data)
                    elif time_steps[i] >= 10:
                        value_slip_softhard = [4] * (end_index - start_index + 1)
  # safe release
                        four += 1
                        softball_release+=1
                        interval_data=self.create_interval_data(data, start_index,
   end_index, value_slip_softhard)
                        slip_intervals[1][6].append(interval_data)


                elif 'emptybottle\pass/' in directories:
                    if time_steps[i] < 7:
                        value_slip_softhard = [2] * (end_index - start_index + 1)
  # grasp
                        two += 1
                        emptybottle_grasp_pass+=1
                        interval_data=self.create_interval_data(data, start_index,
   end_index, value_slip_softhard)
                        grasp_intervals[2][0].append(interval_data)
                    elif time_steps[i] >= 7:
                        value_slip_softhard = [4] * (end_index - start_index + 1)
  # passing
                        four += 1
                        emptybottle_pass+=1
                        interval_data=self.create_interval_data(data, start_index,
   end_index, value_slip_softhard)
```

```
                        slip_intervals[2][2].append(interval_data)
                elif 'emptybottle\pull/' in directories:
                    if  time_steps[i] < 7:
                        value_slip_softhard = [2] * (end_index - start_index + 1)
  # grasp
                        two += 1
                        emptybottle_grasp_pull+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        grasp_intervals[2][2].append(interval_data)
                    elif time_steps[i] >= 7:
                        value_slip_softhard = [6] * (end_index - start_index + 1)
  # taking away
                        six += 1
                        emptybottle_pull+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        slip_intervals[2][3].append(interval_data)
                elif 'emptybottle\putdown/' in directories:
                    if  time_steps[i] < 7:
                        value_slip_softhard = [2] * (end_index - start_index + 1)
  # grasp
                        two += 1
                        emptybottle_grasp_putdown+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        grasp_intervals[2][6].append(interval_data)
                    elif time_steps[i] >= 7 and  time_steps[i] < 10:
                        value_slip_softhard = [4] * (end_index - start_index + 1)
  # putting down
                        four += 1
                        emptybottle_putdown+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        slip_intervals[2][5].append(interval_data)
                    elif time_steps[i] >= 10:
                        value_slip_softhard = [4] * (end_index - start_index + 1)
  # release
                        four += 1
                        emptybottle_release+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        slip_intervals[2][6].append(interval_data)


                elif 'hardball\pass/' in directories:
                    if time_steps[i] < 7:
                        value_slip_softhard = [3] * (end_index - start_index + 1)
  # grasp
                        three += 1
                        heavyball_grasp_pass+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        grasp_intervals[3][0].append(interval_data)
                    elif time_steps[i] >= 7:
                        value_slip_softhard = [5] * (end_index - start_index + 1)
  # passing
                        five += 1
                        heavyball_pass+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
```

```
                        slip_intervals[3][2].append(interval_data)
                elif 'hardball/fall/' in directories:
                    if time_steps[i] < 7:
                        value_slip_softhard = [3] * (end_index - start_index + 1)
  # grasp
                        three += 1
                        heavyball_grasp_fall+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        grasp_intervals[3][4].append(interval_data)
                    elif time_steps[i] >= 7:
                        value_slip_softhard = [7] * (end_index - start_index + 1)
  # falling (risky)
                        seven += 1
                        heavyball_fall+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        slip_intervals[3][4].append(interval_data)
                elif 'hardball\putdown/' in directories:
                    if time_steps[i] < 7:
                        value_slip_softhard = [3] * (end_index - start_index + 1)
  # grasp
                        three += 1
                        heavyball_grasp_putdown+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        grasp_intervals[3][6].append(interval_data)
                    elif time_steps[i] >= 7 and  time_steps[i] < 10:
                        value_slip_softhard = [5] * (end_index - start_index + 1)
  # putting down
                        five += 1
                        heavyball_putdown+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        slip_intervals[3][5].append(interval_data)
                    elif time_steps[i] >= 10:
                        value_slip_softhard = [5] * (end_index - start_index + 1)
  # release
                        five += 1
                        heavyball_release+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        slip_intervals[3][6].append(interval_data)


                elif 'colabottle\pass/' in directories:
                    if time_steps[i] < 7:
                        value_slip_softhard = [3] * (end_index - start_index + 1)
  # grasp
                        three += 1
                        cola_grasp_pass+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        grasp_intervals[4][0].append(interval_data)
                    elif time_steps[i] >= 7 and  time_steps[i] < 10:
                        value_slip_softhard = [5] * (end_index - start_index + 1)
  # passing
                        five += 1
                        cola_pass+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
```

```
                        slip_intervals[4][2].append(interval_data)
                    elif time_steps[i] >= 10:
                        value_slip_softhard = [5] * (end_index - start_index + 1)
 # safe release
                        five += 1
                        cola_release+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        slip_intervals[4][6].append(interval_data)
                elif 'colabottle/fall/' in directories:
                    if time_steps[i] < 7:
                        value_slip_softhard = [3] * (end_index - start_index + 1)
 # grasp
                        three += 1
                        cola_grasp_fall+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        grasp_intervals[4][4].append(interval_data)
                    elif time_steps[i] >= 7:
                        value_slip_softhard = [7] * (end_index - start_index + 1)
 # falling
                        seven += 1
                        cola_fall+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        slip_intervals[4][4].append(interval_data)
                elif 'colabottle\putdown/' in directories:
                    if time_steps[i] < 7:
                        value_slip_softhard = [3] * (end_index - start_index + 1)
 # grasp
                        three += 1
                        cola_grasp_putdown+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        grasp_intervals[4][6].append(interval_data)
                    elif time_steps[i] >= 7 and time_steps[i] < 10:
                        value_slip_softhard = [5] * (end_index - start_index + 1)
 # putting down
                        five += 1
                        cola_putdown+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        slip_intervals[4][5].append(interval_data)
                    elif time_steps[i] >= 10:
                        value_slip_softhard = [5] * (end_index - start_index + 1)
 # safe release
                        five += 1
                        cola_release+=1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        slip_intervals[4][6].append(interval_data)

                elif 'nothing/' in directories:
                        value_slip_softhard = [0] * (end_index - start_index + 1)
 # nothing at the beginning
                        zero += 1
                        interval_data=self.create_interval_data(data, start_index,
 end_index, value_slip_softhard)
                        slip_intervals[0][1].append(interval_data)

                else:
```

```
                    break

            # sample window in the middle to represent steady grasp
            start_index = max(0, i - 5)
            end_index = min(len(time_steps), i)
            if 'basketball\pass/' in directories:
                if time_steps[i]-7>0 and time_steps[i]-7<0.015:
                    value_slip_softhard = [2] * (end_index - start_index + 1) # grasp
                    two += 1
                    basketball_grasp_pass_keep+=1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                    grasp_intervals[0][1].append(interval_data)
            elif 'basketball\pull/' in directories:
                if time_steps[i]-7>0 and time_steps[i]-7<0.015:
                    value_slip_softhard = [2] * (end_index - start_index + 1) # grasp
                    two += 1
                    basketball_grasp_pull_keep+=1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                    grasp_intervals[0][3].append(interval_data)
            elif 'basketball\putdown/' in directories:
                if time_steps[i]-7>0 and time_steps[i]-7<0.02:
                    value_slip_softhard = [2] * (end_index - start_index + 1) # grasp
                    two += 1
                    basketball_grasp_putdown_keep+=1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                    grasp_intervals[0][7].append(interval_data)
                elif 10-time_steps[i]>0 and 10-time_steps[i]<0.02:
                    value_slip_softhard = [2] * (end_index - start_index + 1) # grasp
                    two += 1
                    basketball_grasp_release_keep+=1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                    grasp_intervals[0][8].append(interval_data)

            elif 'softball\pass/' in directories:
                if time_steps[i]-7>0 and time_steps[i]-7<0.015:
                    value_slip_softhard = [2] * (end_index - start_index + 1) # grasp
                    two += 1
                    softball_grasp_pass_keep+=1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                    grasp_intervals[1][1].append(interval_data)
            elif 'softball\pull/' in directories:
                if time_steps[i]-7>0 and time_steps[i]-7<0.015:
                    value_slip_softhard = [2] * (end_index - start_index + 1) # grasp
                    two += 1
                    softball_grasp_pull_keep+=1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                    grasp_intervals[1][3].append(interval_data)
            elif 'softball\putdown/' in directories:
```

```
                if 7-time_steps[i]>0 and 7-time_steps[i]<0.02:
                    value_slip_softhard = [2] * (end_index - start_index + 1) # grasp
                    two += 1
                    softball_grasp_putdown_keep+=1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                    grasp_intervals[1][7].append(interval_data)
                elif 10-time_steps[i]>0 and 10-time_steps[i]<0.02:
                    value_slip_softhard = [7] * (end_index - start_index + 1) # grasp
                    two += 1
                    softball_grasp_release_keep+=1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                    grasp_intervals[1][8].append(interval_data)


            elif 'emptybottle\pass/' in directories:
                if time_steps[i]-7>0 and time_steps[i]-7<0.015:
                    value_slip_softhard = [2] * (end_index - start_index + 1) # grasp
                    two += 1
                    emptybottle_grasp_pass_keep+=1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                    grasp_intervals[2][1].append(interval_data)
            elif 'emptybottle\pull/' in directories:
                if time_steps[i]-7>0 and time_steps[i]-7<0.015:
                    value_slip_softhard = [2] * (end_index - start_index + 1) # grasp
                    two += 1
                    emptybottle_grasp_pull_keep+=1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                    grasp_intervals[2][3].append(interval_data)
            elif 'emptybottle\putdown/' in directories:
                if time_steps[i]-7>0 and time_steps[i]-7<0.02:
                    value_slip_softhard = [2] * (end_index - start_index + 1) # grasp
                    two += 1
                    emptybottle_grasp_putdown_keep+=1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                    grasp_intervals[2][7].append(interval_data)
                elif 10-time_steps[i]>0 and 10-time_steps[i]<0.02:
                    value_slip_softhard = [2] * (end_index - start_index + 1) # grasp
                    two += 1
                    emptybottle_grasp_release_keep+=1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                    grasp_intervals[2][8].append(interval_data)


            elif 'hardball\pass/' in directories:
                if 7-time_steps[i]>0 and 7-time_steps[i]<0.015:
                    value_slip_softhard = [3] * (end_index - start_index + 1) # grasp
                    three += 1
                    heavyball_grasp_pass_keep=1
```

```
                interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                grasp_intervals[3][1].append(interval_data)
        elif 'hardball/fall/' in directories:
            if time_steps[i]-7>0 and time_steps[i]-7<0.015:
                value_slip_softhard = [3] * (end_index - start_index + 1) # grasp
                three += 1
                heavyball_grasp_fall_keep+=1
                interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                grasp_intervals[3][5].append(interval_data)
        elif 'hardball\putdown/' in directories:
            if 7-time_steps[i]>0 and 7-time_steps[i]<0.02:
                value_slip_softhard = [3] * (end_index - start_index + 1) # grasp
                three += 1
                heavyball_grasp_putdown_keep+=1
                interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                grasp_intervals[3][7].append(interval_data)
            elif 10-time_steps[i]>0 and 10-time_steps[i]<0.02:
                value_slip_softhard = [3] * (end_index - start_index + 1) # grasp
                three += 1
                heavyball_grasp_release_keep+=1
                interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                grasp_intervals[3][8].append(interval_data)


        elif 'colabottle\pass/' in directories:
            if 7-time_steps[i]>0 and 7-time_steps[i]<0.02:
                value_slip_softhard = [3] * (end_index - start_index + 1) # grasp
                three += 1
                cola_grasp_pass_keep+=1
                interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                grasp_intervals[4][1].append(interval_data)
            elif 10-time_steps[i]>0 and 10-time_steps[i]<0.02:
                value_slip_softhard = [3] * (end_index - start_index + 1) # grasp
                three += 1
                cola_grasp_release_keep+=1
                interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                grasp_intervals[4][8].append(interval_data)
        elif 'colabottle/fall/' in directories:
            if time_steps[i]-7>0 and time_steps[i]-7<0.015:
                value_slip_softhard = [3] * (end_index - start_index + 1) # grasp
                three += 1
                cola_grasp_fall_keep+=1
                interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                grasp_intervals[4][5].append(interval_data)
        elif 'colabottle\putdown/' in directories:
            if 7-time_steps[i]>0 and 7-time_steps[i]<0.02:
                value_slip_softhard = [3] * (end_index - start_index + 1) # grasp
```

```
                    three += 1
                    cola_grasp_putdown_keep+=1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                    grasp_intervals[4][7].append(interval_data)
                elif 10-time_steps[i]>0 and 10-time_steps[i]<0.02:
                    value_slip_softhard = [3] * (end_index - start_index + 1) # grasp
                    three += 1
                    cola_grasp_release_keep+=1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                    grasp_intervals[4][8].append(interval_data)

            # sample window at the end to represent nothing
            if i == len(time_steps)-65: # or i == len(time_steps)-43:
                start_index = max(0, i - 5)
                end_index = min(len(time_steps), i)

                if 'basketball\pass/' in directories:
                    value_slip_softhard = [0] * (end_index - start_index + 1)
                    zero += 1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                    slip_intervals[0][0].append(interval_data)
                    basketball_nothing+=1
                elif 'basketball\pull/' in directories:
                    value_slip_softhard = [0] * (end_index - start_index + 1)
                    zero += 1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                    slip_intervals[0][0].append(interval_data)
                    basketball_nothing+=1
                elif 'basketball\putdown/' in directories:
                    value_slip_softhard = [0] * (end_index - start_index + 1)
                    zero += 1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                    slip_intervals[0][0].append(interval_data)
                    basketball_nothing+=1


                elif 'softball\pass/' in directories:
                    value_slip_softhard = [0] * (end_index - start_index + 1)
                    zero += 1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                    slip_intervals[1][0].append(interval_data)
                    softball_nothing+=1
                elif 'softball\pull/' in directories:
                    value_slip_softhard = [0] * (end_index - start_index + 1)
                    zero += 1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                    slip_intervals[1][0].append(interval_data)
                    softball_nothing+=1
                elif 'softball\putdown/' in directories:
                    value_slip_softhard = [0] * (end_index - start_index + 1)
                    zero += 1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
```

```
                slip_intervals[1][0].append(interval_data)
                softball_nothing+=1


            elif 'emptybottle\pass/' in directories:
                value_slip_softhard = [0] * (end_index - start_index + 1)
                zero += 1
                interval_data=self.create_interval_data(data, start_index,
    end_index, value_slip_softhard)
                slip_intervals[2][0].append(interval_data)
                emptybottle_nothing+=1
            elif 'emptybottle\pull/' in directories:
                value_slip_softhard = [0] * (end_index - start_index + 1)
                zero += 1
                interval_data=self.create_interval_data(data, start_index,
    end_index, value_slip_softhard)
                slip_intervals[2][0].append(interval_data)
                emptybottle_nothing+=1
            elif 'emptybottle\putdown/' in directories:
                value_slip_softhard = [0] * (end_index - start_index + 1)
                zero += 1
                interval_data=self.create_interval_data(data, start_index,
    end_index, value_slip_softhard)
                slip_intervals[2][0].append(interval_data)
                emptybottle_nothing+=1


            elif 'hardball\pass/' in directories:
                value_slip_softhard = [1] * (end_index - start_index + 1)
                uno += 1
                interval_data=self.create_interval_data(data, start_index,
    end_index, value_slip_softhard)
                slip_intervals[3][0].append(interval_data)
                heavyball_nothing+=1
            elif 'hardball/fall/' in directories:
                value_slip_softhard = [1] * (end_index - start_index + 1)
                uno += 1
                interval_data=self.create_interval_data(data, start_index,
    end_index, value_slip_softhard)
                slip_intervals[3][0].append(interval_data)
                heavyball_nothing+=1
            elif 'hardball\putdown/' in directories:
                value_slip_softhard = [1] * (end_index - start_index + 1)
                uno += 1
                interval_data=self.create_interval_data(data, start_index,
    end_index, value_slip_softhard)
                slip_intervals[3][0].append(interval_data)
                heavyball_nothing+=1


            elif 'colabottle\pass/' in directories:
                value_slip_softhard = [1] * (end_index - start_index + 1)
                uno += 1
                interval_data=self.create_interval_data(data, start_index,
    end_index, value_slip_softhard)
                slip_intervals[4][0].append(interval_data)
                cola_nothing+=1
            elif 'colabottle/fall/' in directories:
                value_slip_softhard = [1] * (end_index - start_index + 1)
                uno += 1
```

```
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                    slip_intervals[4][0].append(interval_data)
                    cola_nothing+=1
                elif 'colabottle\putdown/' in directories:
                    value_slip_softhard = [1] * (end_index - start_index + 1)
                    uno += 1
                    interval_data=self.create_interval_data(data, start_index, end_index, value_slip_softhard)
                    slip_intervals[4][0].append(interval_data)
                    cola_nothing+=1

            ml_classes= [zero, uno, two, three, four, five, six, seven]
            actions =   [[basketball_nothing, softball_nothing, emptybottle_nothing, heavyball_nothing, cola_nothing], \
                        [basketball_grasp, softball_grasp, emptybottle_grasp, heavyball_grasp, cola_grasp],\
                        [basketball_pass,  softball_pass, emptybottle_pass, heavyball_pass, cola_pass],\
                        [basketball_pull, softball_pull, emptybottle_pull, heavyball_pull, cola_pull],\
                        [basketball_fall, softball_fall, emptybottle_fall, heavyball_fall, cola_fall],\
                        [basketball_putdown, softball_putdown, emptybottle_putdown, heavyball_putdown, cola_putdown],\
                        [basketball_release, softball_release, emptybottle_release, heavyball_release, cola_release]]

            grasps =   [[basketball_grasp_pass, softball_grasp_pass, emptybottle_grasp_pass, heavyball_grasp_pass, cola_grasp_pass], \
                        [basketball_grasp_pass_keep, softball_grasp_pass_keep, emptybottle_grasp_pass_keep, heavyball_grasp_pass_keep, cola_grasp_pass_keep],\
                        [basketball_grasp_pull, softball_grasp_pull, emptybottle_grasp_pull, heavyball_grasp_pull, cola_grasp_pull],\
                        [basketball_grasp_pull_keep,  softball_grasp_pull_keep, emptybottle_grasp_pull_keep, heavyball_grasp_pull_keep, cola_grasp_pull_keep],\
                        [basketball_grasp_fall, softball_grasp_fall, emptybottle_grasp_fall, heavyball_grasp_fall, cola_grasp_fall],\
                        [basketball_grasp_fall_keep, softball_grasp_fall_keep, emptybottle_grasp_fall_keep, heavyball_grasp_fall_keep, cola_grasp_fall_keep],\
                        [basketball_grasp_putdown, softball_grasp_putdown, emptybottle_grasp_putdown, heavyball_grasp_putdown, cola_grasp_putdown],\
                        [basketball_grasp_putdown_keep, softball_grasp_putdown_keep, emptybottle_grasp_putdown_keep, heavyball_grasp_putdown_keep, cola_grasp_putdown_keep],\
                        [basketball_grasp_release_keep, softball_grasp_release_keep, emptybottle_grasp_release_keep, heavyball_grasp_release_keep, cola_grasp_release_keep]]

        return slipperyslip, slip_intervals, grasp_intervals, ml_classes, actions, grasps


# CLASS TO BALANCE THE DATA FOR THE MACHINE LEARNING ALGORITHM

class DatasetBalancer:
    def __init__(self, slip_intervals, grasp_intervals):
        self.slip_intervals = slip_intervals
```

```
        self.grasp_intervals = grasp_intervals

    def balance_dataset(self):
        all_slip_intervals=[]
        size=[[0,0,0,0,0],
              [0,0,0,0,0],
              [0,0,0,0,0],
              [0,0,0,0,0],
              [0,0,0,0,0],
              [0,0,0,0,0],
              [0,0,0,0,0]]

        size1=[[0,0,0,0,0],
              [0,0,0,0,0],
              [0,0,0,0,0],
              [0,0,0,0,0],
              [0,0,0,0,0],
              [0,0,0,0,0],
              [0,0,0,0,0],
              [0,0,0,0,0],
              [0,0,0,0,0]]

        for sublist1 in self.grasp_intervals:
            for sublist1a in sublist1:
                random.shuffle(sublist1a)

        for sublist2 in self.slip_intervals:
            for sublist2a in sublist2:
                random.shuffle(sublist2a)

        # to check if sizes are alright
        print('Check if this matrix is the same as before:')
        for i in range(len(self.slip_intervals)):
            for j in range(len(self.slip_intervals[i])):
                size[j][i] = len(self.slip_intervals[i][j])
        size = pd.DataFrame(size)
        column_names = ["basketball", "softball", "emptybottle", "hardball", "cola bottle"]
        row_names = ["nothing", "grasp", "pass", "pull", "fall", "putdown", "release"]
        size.columns = column_names
        size.index = row_names
        print(size)

        print('Check if this matrix is the same as before:')
        for i in range(len(self.grasp_intervals)):
            for j in range(len(self.grasp_intervals[i])):
                size1[j][i] = len(self.grasp_intervals[i][j])
        size1 = pd.DataFrame(size1)
        column_names = ["basketball", "softball", "emptybottle", "hardball", "cola bottle"]
        row_names = ["grasp_pass", "grasp_pass_keep", "grasp_pull", "grasp_pull_keep", "grasp_fall", "grasp_fall_keep", "grasp_putdown", "grasp_putdown_keep", "grasp_release_keep"]
        size1.columns = column_names
        size1.index = row_names
        print(size1)

        limits=[[125, 125, 125, 125, 125],
                [ 27,   0,   0,   0,   0],
                [ 88,  94,  92, 110,  62],
```

```
                 [ 86,  87, 109,   0,   0],
                 [  0,   0,   0, 115, 110],
                 [ 60,  60,  60,  59,  59],
                 [ 61,  59,  60,  63,  80]]


    for i in range(len(limits)):
                for j in range(len(limits[i])):
                    print('i, j, k = ', i,', ',j,', ',limits[i][j])
                    for k in range(limits[i][j]):
                        all_slip_intervals.append(np.array(self.slip_intervals[j][i][k]))


    limits_grasp=[[ 30,  30,  30,  30,  30],
                  [ 30,  50,  30,  30,  40],
                  [ 30,  30,  30,   0,   0],
                  [ 25,  25,  25,   0,   0],
                  [  0,   0,   0,  30,  30],
                  [  0,   0,   0,  25,  25],
                  [ 30,  30,  30,  30,  30],
                  [ 60,  60,  60,  60,  60],
                  [ 60,  66,  65,  60, 100]]

    for i in range(len(limits_grasp)):
        for j in range(len(limits_grasp[i])):
            print('i, j, k = ', i,', ',j,', ', limits_grasp[i][j])
            for k in range(limits_grasp[i][j]):
                all_slip_intervals.append(np.array(self.grasp_intervals[j][i][k]))

    zero_indices = [i for i, interval_data in enumerate(all_slip_intervals) if interval_data[0, -1] == 0]
    uno_indices = [i for i, interval_data in enumerate(all_slip_intervals) if interval_data[0, -1] == 1]
    two_indices = [i for i, interval_data in enumerate(all_slip_intervals) if interval_data[0, -1] == 2]
    three_indices = [i for i, interval_data in enumerate(all_slip_intervals) if interval_data[0, -1] == 3]
    four_indices = [i for i, interval_data in enumerate(all_slip_intervals) if interval_data[0, -1] == 4]
    five_indices = [i for i, interval_data in enumerate(all_slip_intervals) if interval_data[0, -1] == 5]
    six_indices = [i for i, interval_data in enumerate(all_slip_intervals) if interval_data[0, -1] == 6]
    seven_indices = [i for i, interval_data in enumerate(all_slip_intervals) if interval_data[0, -1] == 7]

    random.shuffle(zero_indices)
    random.shuffle(uno_indices)
    random.shuffle(two_indices)
    random.shuffle(three_indices)
    random.shuffle(four_indices)
    random.shuffle(five_indices)
    random.shuffle(six_indices)
    random.shuffle(seven_indices)

    train_zero = zero_indices
    train_uno = uno_indices
    train_two = two_indices
    train_three = three_indices
```

```
        train_four = four_indices
        train_five = five_indices
        train_six = six_indices
        train_seven = seven_indices

        balanced_train_intervals = ([all_slip_intervals[i] for i in train_zero] +
                                    [all_slip_intervals[i] for i in train_uno] +
                                    [all_slip_intervals[i] for i in train_two] +
                                    [all_slip_intervals[i] for i in train_three]+
                                    [all_slip_intervals[i] for i in train_four] +
                                    [all_slip_intervals[i] for i in train_five] +
                                    [all_slip_intervals[i] for i in train_six]+
                                    [all_slip_intervals[i] for i in train_seven])

        random.shuffle(balanced_train_intervals)

        balanced_train_intervals = np.array(balanced_train_intervals)

        train = balanced_train_intervals[:,:,:-1]

        train_target = balanced_train_intervals[:,:,-1].astype(int)

        return train, train_target


# MACHINE LEARNING ALGORITHM

class MLModelTrainer:
    def __init__(self):
        self.clf_rf = RandomForestClassifier(n_estimators=100, random_state=0)

    def reshape_data_for_ml(self, train, train_target):
        train = np.reshape(train, (train.shape[0], train.shape[1] * train.shape
    [2]))
        train_target = train_target[:,0].astype(int)
        # print(np.shape(train), np.shape(train_target))

        return train, train_target

    def train_and_save_random_forest(self, all_data, all_targets, path=r'D:\
    università\TUM\Python\definitivo_tesi\MLmodels/'):
        x_all = all_data
        y_all = all_targets

        # Train the model using all your data
        self.clf_rf.fit(x_all, y_all)

        # Save the trained model to a file
        joblib.dump(self.clf_rf, path+'model_x.pkl')

        print("Random Forest model trained and saved.")



# MAIN

if __name__ == "__main__":
    folder_path = r'D:\università\TUM\Python\definitivo_tesi/train/'
    directories = ['basketball\pass/', 'basketball\pull/', 'basketball/fall/', '
    basketball\putdown/',\
```

```
                    'softball\pass/', 'softball\pull/', 'softball/fall/', 'softball\putdown/',\
                    'emptybottle\pass/', 'emptybottle\pull/', 'emptybottle/fall/', 'emptybottle\putdown/',\
                    'hardball\pass/', 'hardball\pull/', 'hardball/fall/', 'hardball\putdown/',\
                    'colabottle\pass/', 'colabottle\pull/', 'colabottle/fall/', 'colabottle\putdown/',\
                     'nothing/']
    num_sensors = 6
    num_files_per_directory =  [60, 80,  0, 60,\
                                60, 80, 0, 60,\
                                60, 80, 0, 60,\
                                60, 0, 80, 60,\
                                60, 0, 80, 60,\
                                    1]
    all_filtered_data = []
    all_pca_data_x = []
    all_pca_data_y = []
    all_pca_data_z = []
    all_data = []
    all_time_steps = []
    all_slip_intervals = [[] for _ in range(5)]
    for _ in range(5):
        for __ in range(7):
            all_slip_intervals[_].append([])
    all_grasp_intervals = [[] for _ in range(5)]
    for _ in range(5):
        for __ in range(9):
            all_grasp_intervals[_].append([])
    total=0
    ml_classes=[0,0,0,0,0,0,0,0]
    actions = [[0,0,0,0,0],
               [0,0,0,0,0],
               [0,0,0,0,0],
               [0,0,0,0,0],
               [0,0,0,0,0],
               [0,0,0,0,0],
               [0,0,0,0,0]]
    grasps=[[0,0,0,0,0],
               [0,0,0,0,0],
               [0,0,0,0,0],
               [0,0,0,0,0],
               [0,0,0,0,0],
               [0,0,0,0,0],
               [0,0,0,0,0],
               [0,0,0,0,0],
               [0,0,0,0,0]]
    train = []
    train_target = []
    test = []
    test_target = []

    data_processor = DataProcessor_fromExcel(None)
    bpf = BandpassFilter()
    pca_fe = PCABasedFeatureExtraction()
    ml_trainer=MLModelTrainer()

    for directory, num_files in zip(directories, num_files_per_directory):
        print(directory, num_files)
```

```python
        all_data_prov, all_time_steps_prov = data_processor.process_all_files(folder_path + directory, num_files)

        for file_number in range(num_files):
            data = all_data_prov[file_number]
            all_data.append(data)
            time_steps = all_time_steps_prov[file_number]
            all_time_steps.append(time_steps)

            filtered_data = bpf.apply_bandpass_filter(data, time_steps, num_sensors)
            slip_bp, slip_bp_single = bpf.calculate_slip_points_bp(data, filtered_data, time_steps, num_sensors, directory)

            force_data_x = data.iloc[:, ::3]
            force_data_y = data.iloc[:, 1::3]
            force_data_z = data.iloc[:, 2::3]

            # Apply PCA
            pca_data_x, pca_data_y, pca_data_z = pca_fe.apply_pca(force_data_x, force_data_y, force_data_z)
            all_pca_data_x.append(force_data_x)
            all_pca_data_y.append(force_data_y)
            all_pca_data_z.append(force_data_z)

            # Calculate slip points
            slip_fc, mi_x, mi_y = pca_fe.calculate_slip_points(pca_data_x, pca_data_y, pca_data_z, time_steps)

            if any(value == 1 for value in slip_fc) or any(value == 1 for value in slip_bp_single):
                slipperyslip, all_slip_intervals, all_grasp_intervals, ml_classes1, actions1, grasps1 = pca_fe.create_slip_windows( data, time_steps, slip_fc, slip_bp_single, directory, all_slip_intervals, all_grasp_intervals)

                for i in range(len(ml_classes)):
                    ml_classes[i] += ml_classes1[i]

                actions1=np.array(actions1)
                for i in range(len(actions)):
                    for j in range(len(actions[0])):
                        actions[i][j] += actions1[i][j]

                grasps1=np.array(grasps1)
                for i in range(len(grasps)):
                    for j in range(len(grasps[0])):
                        grasps[i][j] += grasps1[i][j]

                print(f'File number {file_number+1} in directory {directory} - DONE!')

    actions = pd.DataFrame(actions)
    column_names = ["basketball", "softball", "emptybottle", "hardball", "cola bottle"]
    row_names = ["nothing", "grasp", "pass", "pull", "fall", "putdown", "release"]
    actions.columns = column_names
    actions.index = row_names
    print('Total number of actions:')
    print(actions)

    grasps = pd.DataFrame(grasps)
```

```
column_names = ["basketball", "softball", "emptybottle", "hardball", "cola bottle"]
row_names = ["grasp_pass", "grasp_pass_keep", "grasp_pull", "grasp_pull_keep", "grasp_fall", "grasp_fall_keep", "grasp_putdown", "grasp_putdown_keep", "grasp_release_keep"]
grasps.columns = column_names
grasps.index = row_names
print('Total number of grasps:')
print(grasps)

print(f"Shape of all_data: ({len(all_data)},{np.shape(all_data[0])})")
#print(f"Shape of slip_bp: {np.shape(slip_bp)}")
print(f"Shape of all_pca_data_y: ({len(all_pca_data_y)},{len(all_pca_data_y[0])})")
#print(f"Shape of slip_fc: {np.shape(slip_fc)}")

print('[zero, uno, two, three, four, five, six,seven]: ', ml_classes)

# Assuming you have all_slip_intervals and all_grasp_intervals as your input data
dataset_balancer = DatasetBalancer(all_slip_intervals, all_grasp_intervals)
train, train_target = dataset_balancer.balance_dataset()

print('Shape of train dataset:', np.shape(train), ' and shape of train targets: ', np.shape(train_target))

# Random Forest model
train, train_target = ml_trainer.reshape_data_for_ml(train, train_target)

print('Shape of reshaped train dataset:', np.shape(train), ' and shape of reshaped train targets: ', np.shape(train_target))

ml_trainer.train_and_save_random_forest(train, train_target)
```

content/machine_learning_training_def.py

Bibliography

[1] R. Chris Miall, Orna Rosenthal, Kristin Ørstavik, Jonathan D. Cole, and Fabrice R. Sarlegna. «Loss of haptic feedback impairs control of hand posture: a study in chronically deaferented individuals when grasping and lifting objects». In: (June 2019) (cit. on p. 1).

[2] Cambridge University Press. *Haptic in Cambridge Advanced Learner's Dictionary Thesaurus.* (n.d.) URL: `https://dictionary.cambridge.org/dictionary/english/haptic` (cit. on p. 1).

[3] James J. Gibson. *The Senses Considered as Perceptual Systems.* London: GEORGE ALLEN UNWIN LTD, 1966 (cit. on p. 1).

[4] Gabriel Robles-De-La-Torre. «The Importance of the Sense of Touch in Virtual and Real Environments». In: (July 2006), pp. 24–30 (cit. on p. 1).

[5] Jeremy D. Brown, Andrew Paek, Mashaal Syed, Marcia K. O'Malley, and Patricia A. Shewokis. «Understanding the Role of Haptic Feedback in a Teleoperated/Prosthetic Grasp and Lift Task». In: *IEEE World Haptics Conference 2013.* Daejeon, Korea, 414–18 2013, pp. 271–276 (cit. on p. 1).

[6] Alan Thurston. «Pare and prosthetics: The early history of artificial limbs». In: *ANZ Journal of Surgery* 77.12 (Dec. 2007), pp. 1114–1119 (cit. on p. 2).

[7] B. Stephens-Fripp, G. Alici, and R. Mutlu. «A Review of Non-Invasive Sensory Feedback Methods for Transradial Prosthetic Hands». In: *IEEE Access* 6 (2018) (cit. on pp. 2, 10).

[8] Katie Z. Zhuang et al. «Shared human–robot proportional control of a dexterous myoelectric prosthesis». In: *Nature Machine Intelligence* 1 (Sept. 2019), pp. 400–411 (cit. on p. 2).

[9] Todd R. Farrell and Richard F. Weir. «The Optimal Controller Delay for Myoelectric Prostheses». In: *IEEE Trans Neural Syst Rehabil Eng.* 15.1 (Mar. 2007), pp. 111–118 (cit. on p. 2).

[10] Maurice A. LeBlanc. «Innovation and Improvement of Body-Powered Arm Prostheses: A First Step». In: *Clinical Prosthetics Orthotics* 9 (1985), pp. 13–16 (cit. on p. 5).

[11] Stephanie L. Carey, Derek J. Lura, and M. Jason Highsmith. «Differences in myoelectric and body-powered upper-limb prostheses: Systematic literature review». In: *Journal of Rehabilitation Research Development (JRRD)* 52.3 (2015), pp. 247–262 (cit. on p. 5).

[12] Ivan Vujaklija, Dario Farina, and Oskar C Aszmann. «New developments in prosthetic arm systems». In: *Orthopedic Research and Reviews* 8 (2016), pp. 31–39 (cit. on pp. 5, 6).

[13] Samreen Hussain, Sarmad Shams, and Saad Jawaid Khan. «Impact of Medical Advancement: Prostheses». In: *Computer Architecture in Industrial, Biomechanical and Biomedical Engineering* (Nov. 2019) (cit. on p. 6).

[14] Taylor J. Bates, John R. Fergason, and Sarah N. Pierrie. «Technological Advances in Prosthesis Design and Rehabilitation Following Upper Extremity Limb Loss». In: *n The Use of Technology in Orthopaedic Surgery—Intraoperative and Post-Operative Managemen* (June 2020) (cit. on p. 6).

[15] Michael T. Leddy, Joseph T. Belter, Kevin D. Gemmell Jr., and Aaron M. Dollar. «Lightweight custom composite prosthetic components using an additive manufacturing-based molding technique». In: *Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (Aug. 2015) (cit. on p. 6).

[16] Andrea Marinelli et al. «Active upper limb prostheses: a review on current state and upcoming breakthroughs». In: *Progress in Biomedical Engineering* (Jan. 2023) (cit. on pp. 6–8).

[17] Sushmi Mazumdar, Angana Saikia, Nitin Sahai, and Sudip Paul. «Below Elbow Prosthetic: A Path to Independent Era». In: *International Journal of Advanced Information Science and Technology (IJAIST)* 4.11 (Nov. 2015) (cit. on p. 7).

[18] Smith LH and Hargrove LJ. «Comparison of surface and intramuscular EMG pattern recognition for simultaneous wrist/hand motion classification». In: *2013 35th annual international conference of the IEEE engineering in medicine and biology society (EMBC)* (2013), pp. 4223–4226 (cit. on p. 8).

[19] Patricia Capsi-Morales, Cristina Piazza, Lis Sjoberg, Manuel G. Catalano, Giorgio Grioli, Antonio Bicchi, and Liselotte M. Hermansson. «Functional assessment of current upper limb prostheses: An integrated clinical and technological perspective». In: *PLOS ONE* (Aug. 2023) (cit. on p. 8).

[20] Tasbolat Taunyazov, Luar Shui Song, Eugene Lim, Hian Hian See, David Lee, Benjamin C.K. Tee, and Harold Soh. «Extended Tactile Perception: Vibration Sensing through Tools and Grasped Objects». In: *2021 IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)* (Sept. 2021), pp. 1755–62 (cit. on p. 8).

[21] Chiara Lucarotti, Calogero Maria Oddo, Nicola Vitiello, and Maria Chiara Carrozza. «Synthetic and Bio-Artificial Tactile Sensing: A Review». In: *Sensors* 13 (Feb. 2013), pp. 1435–1466 (cit. on p. 8).

[22] H. Henrik Ehrsson, Birgitta Rosen, Anita Stockselius, Christina Ragno, Peter Kohler, and Goran Lundborg. «Upper limb amputees can be induced to experience a rubber hand as their own». In: *Brain* 131 (2008), pp. 3443–3452 (cit. on p. 8).

[23] Teri Rosenbaum-Chou, Wayne Daly, Ray Austin, Pravin Chaubey, and David A. Boone. «Development and Real World Use of a Vibratory Haptic Feedback System for Upper-Limb Prosthetic Users». In: *Journal of Prosthetics and Orthotics* 28.4 (2016), pp. 136–144 (cit. on p. 8).

[24] M. A. F. Ismail and S. Shimada. «Robot' hand illusion under delayed visual feedback: Relationship between the senses of ownership and agency». In: *PLoS ONE* 11 (July 2016) (cit. on p. 9).

[25] S. Shimada, K. Fukuda, and K. Hiraki. «Rubber hand illusion under delayed visual feedback». In: *PLoS ONE* 4 (July 2009) (cit. on p. 9).

[26] Christian Cipriani, Franco Zaccone, Silvestro Micera, and M. Chiara Carrozza. «On the Shared Control of an EMG-Controlled Prosthetic Hand: Analysis of User–Prosthesis Interaction». In: *IEEE TRANSACTIONS ON ROBOTICS* 24.1 (Feb. 2008), pp. 170–184 (cit. on pp. 9, 21, 22).

[27] Teri Rosenbaum-Chou, Wayne Daly, Ray Austin, Pravin Chaubey, and David A. Boone. «Dual-Parameter Modulation Improves Stimulus Localization in Multichannel Electrotactile Stimulation». In: *IEEE TRANSACTIONS ON HAPTICS* 13.2 (Apr. 2020), pp. 393–403 (cit. on p. 9).

[28] S. Dosen, M.-C. Schaeffer, and D. Farina. «Time-division multiplexing for myoelectric closed-loop control using electrotactile feedback». In: *Journal of NeuroEngineering and Rehabilitation* 11 (Sept. 2014) (cit. on p. 9).

[29] Cornelia Hartmann, Strahinja Došen, Sebastian Amsuess, and Dario Farina. «Closed-Loop Control of Myoelectric Prostheses With Electrotactile Feedback: Influence of Stimulation Artifact and Blanking». In: *IEEE TRANSACTIONS ON NEURAL SYSTEMS AND REHABILITATION ENGINEERING* 23.5 (Sept. 2015), pp. 807–816 (cit. on p. 9).

[30] Sasha B. Godfrey, Matteo Bianchi, Antonio Bicchi, and Marco Santello. «Influence of Force Feedback on Grasp Force Modulation in Prosthetic Applications: a Preliminary Study». In: *. Annual International Conference of the IEEE Engineering in Medicine and Biology Society.* Aug. 2016 (cit. on p. 9).

[31] M. Aziziaghdam and E. Samur. «Providing contact sensory feedback for upper limb robotic prosthesis». In: *IEEE Haptics Symposium* (Feb. 2014) (cit. on p. 9).

[32] Ahmed W. Shehata, Mayank Rehani, Zaheera E. Jassat, and Jacqueline S. Hebert. «Mechanotactile Sensory Feedback Improves Embodiment of a Prosthetic Hand During Active Use». In: *Frontiers in Neuroscience* 14 (Mar. 2020) (cit. on p. 9).

[33] Alison Gibson and Panagiotis Artemiadis. «Object Discrimination Using Optimized Multi-frequency Auditory Cross-Modal Haptic Feedback». In: *36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (Aug. 2014) (cit. on p. 10).

[34] Jose Gonzalez, Hiroyuki Suzuki, Nakayama Natsumi, Masashi Sekine, and Wenwei Yu. «Auditory Display as a Prosthetic Hand Sensory Feedback for Reaching and Grasping Tasks». In: *34th Annual International Conference of the IEEE EMBS*. San Diego, California USA, Aug. 2012 (cit. on p. 10).

[35] M. D'Alonzo, S. Dosen, C. Cipriani, and D. Farina. «HyVE: Hybrid vibro-electrotactile stimulation for sensory feedback and substitution in rehabilitation». In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 22.3 (Mar. 2014) (cit. on p. 10).

[36] M. D'Alonzo, S. Dosen, C. Cipriani, and D. Farina. «HyVE — Hybrid vibro-electrotactile stimulation—Is an efficient approach to multi-channel sensory feedback». In: *IEEE Transactions on Haptics* 7.2 (Apr. 2014) (cit. on p. 10).

[37] J. Kim et al. «Stretchable silicon nanoribbon electronics for skin prosthesis». In: *Nature Communication* 5.1 (Dec. 2014) (cit. on pp. 11, 12).

[38] P. Maiolino, M. Maggiali, G. Cannata, G. Metta, and L. Natale. «A Flexible and Robust Large Scale Capacitive Tactile System for Robots». In: *IEEE Sensors Journal* 13.10 (Oct. 2013) (cit. on p. 13).

[39] Tito Pradhono Tomo, Sophon Somlor, Alexander Schmitz, Shuji Hashimoto, Shigeki Sugano, and Lorenzo Jamone. «Development of a Hall-Effect Based Skin Sensor». In: *IEEE Sensors* (2015) (cit. on p. 14).

[40] T. Tomo, S. Somlor, A. Schmitz, L. Jamone, W. Huang, H. Kristanto, and S. Sugano. «Design and Characterization of a Three-Axis Hall Effect-Based Soft Skin Sensor». In: *Sensors* 16.4 (Apr. 2016) (cit. on p. 14).

[41] T. Tomo, W. K. Wong, A. Schmitz, H. Kristanto, A. Sarazin, L. Jamone, S. Somlor, and S. Sugano. «A Modular, Distributed, Soft, 3-Axis Sensor System for Robot Hands». In: *16th International Conference on Humanoid Robots (Humanoids)*. Cancun, Mexico, 1115–17 2016, pp. 454–460 (cit. on pp. 14–16).

[42] T. Spiegeler Castañeda, J. Matos, P. Capsi-Morales, and C. Piazza. «Spatial and Temporal Analysis of Normal and Shear Forces During Grasping, Manipulation and Social Activities». In: *IEEE International Conference on Rehabilitation Robotics (ICORR)*. 2023 (cit. on pp. 16, 44, 48).

[43] Pascal Weinerand Caterina Neef, Yoshihisa Shibataand Yoshihiko Nakamura, and Tamim Asfour. «An Embedded, Multi-Modal Sensor System for Scalable Robotic and Prosthetic Hand Fingers». In: *Sensors* 20 (2020) (cit. on p. 16).

[44] Andrea Zangrandi, Marco D'Alonzo, Christian Cipriani, and Giovanni Di Pino. «Neurophysiology of slip sensation and grip reaction: insights for hand prosthesis control of slippage». In: *Journal of Neurophysiology* (July 2021) (cit. on pp. 17, 18).

[45] I. Birznieks, M. Burstedt, B. Edin, and R. Johansson. «Mechanisms for force adjustments to unpredictable frictional changes at individual digits during two-fingered manipulation». In: *Journal of Neurophysiology* 80 (1998) (cit. on p. 17).

[46] G. Puchhammer. «The tactile slip sensor: Integration of a miniaturized sensory device on a myoelectric hand». In: *Orthopadie-Technik Q.* I/2000 () (cit. on p. 17).

[47] Erik D. Engeberg and Sanford G. Meek. «Adaptive Sliding Mode Control for Prosthetic Hands to Simultaneously Prevent Slip and Minimize Deformation of Grasped Objects». In: *IEEE/ASME Transactions on Mechatronics* 18.1 (Feb. 2013) (cit. on pp. 17–20, 24).

[48] Bei Yang, Xiaogang Duan, and Hua Deng. «A Simple Method for Slip Detection of Prosthetic Hand». In: *IEEE International Conference on Information and Automation*. Lijiang, China, Aug. 2015 (cit. on p. 20).

[49] F. Cordella, C. Gentile, L. Zollo, R. Barone, R. Sacchetti, A. Davalli, B. Siciliano, and E. Guglielmelli. «A Force-and-slippage control strategy for a poliarticulated prosthetic hand». In: *IEEE International Conference on Robotics and Automation (ICRA)*. 516–21 2015 (cit. on p. 20).

[50] Jens Reinecke, Alexander Dietrich, Florian Schmidt, and Maxime Chalon. «Experimental Comparison of Slip Detection Strategies by Tactile Sensing with the BioTac on the DLR Hand Arm System». In: *IEEE International Conference on Robotics Automation (ICRA)*. May 2014 (cit. on pp. 20, 24, 26).

[51] Hamidreza N. Sani and Sanford G. Meek. «Characterizing the performance of an optical slip sensor for grip control in a prosthesis». In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Sept. 2011) (cit. on pp. 20, 21).

[52] M. T. Francomano, D. Accoto, E. Morganti, L. Lorenzelli, and E. Guglielmelli. «A microfabricated flexible slip sensor». In: *The Fourth IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics* (June 2012) (cit. on p. 21).

[53] Tricia Gibo. «Honest Evaluations of Shared Human-Machine Control Systems». In: *IEEE SMC 2016 Tutorial* (2016) (cit. on p. 21).

[54] Mojtaba Shahmohammadi, Bonnie Guan, and Minas Liarokapis. «An Adaptive, Prosthetic Training Gripper with a Variable Stiffness, Compact Differential and a Vision Based Shared Control Scheme». In: () (cit. on p. 23).

[55] Marshall A. Trout, Taylor C. Hansen, Connor D. Olsen, David J. Warren, Jacob L. Segil, and Jacob A. George. «Shared control decreases the physical and cognitive demands of maintaining secure grip». In: *Conference: Myoelectric Control Symposium*. Fredericton, New Brunswick, Aug. 2022 (cit. on p. 23).

[56] Yang D.P. and Liu H. «Human-machine shared control: New avenue to dexterous prosthetic hand manipulation». In: *Science China* (Nov. 2020) (cit. on p. 23).

[57] A. Mingrino, A. Bucci, R. Magni, and P. Dario. «Slippage Control in Hand Prostheses by sensing Grasping Forces and Sliding Motion». In: *IEEE SENSORS JOURNAL* 3 (1994), pp. 1803–1809 (cit. on p. 25).

[58] S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok. «Friction-Assisted Pulling Force Detection Mechanism for Tactile Sensors». In: *JOURNAL OF MICROELECTROMECHANICAL SYSTEMS* 23.2 (Apr. 2014), pp. 471–481 (cit. on p. 26).

[59] Joana Matos. *Exploration of Shear Force in the Human Hand in Activities of Daily Living*. Internship Report. n.d (cit. on p. 36).

[60] *Michelangelo Hand 8E500*. Ottobock. Feb. 2022. URL: `https://www.ottobock.com/en-us/product/8E500` (cit. on p. 42).

[61] Edward Ramsden. *Hall-Effect Sensors: Theory and Application*. Burlington, MA: Elsevier, 2006 (cit. on p. 43).

[62] *Hall Effect Sensing and Application*. Freeport, Illinois: Honeywell (cit. on p. 44).