Universidade de Aveiro
2023

Pedro
Amaral

Reconhecimento de Configurações da Mão Humana para Previsão de Intenções em Tarefas Colaborativas

Recognition of Human Grasping Patterns for Intention Prediction in Collaborative Tasks

Pedro
Amaral

Reconhecimento de Configurações da Mão Humana para Previsão de Intenções em Tarefas Colaborativas

Recognition of Human Grasping Patterns for Intention Prediction in Collaborative Tasks

"*Predicting the future isn't magic, it's artificial intelligence.*"

— Dave Waters

Universidade de Aveiro
2023

Pedro
Amaral

Reconhecimento de Configurações da Mão Humana para Previsão de Intenções em Tarefas Colaborativas

Recognition of Human Grasping Patterns for Intention Prediction in Collaborative Tasks

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Robótica e Sistemas Inteligentes, realizada sob a orientação científica do Doutor Filipe Silva, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Vítor Santos, Professor Associado c/ Agregação do Departamento de Engenharia Mecânica da Universidade de Aveiro.

The present study was developed in the scope of the Project Augmented Humanity [POCI-01-0247-FEDER-046103], financed by Portugal 2020, under the Competitiveness and Internationalization Operational Program, the Lisbon Regional Operational Program, and by the European Regional Development Fund

o júri / the jury

agradecimentos / acknowledgements

Agradeço ao meu orientador Filipe Silva e co-orientador Vítor Santos por toda a dedicação e ajuda durante o desenvolvimento desta dissertação. Quero também agradecer aos meus amigos e família que me apoiaram durante o percurso.

Palavras Chave

Colaboração Humano-Robô, Aprendizagem Automática, Reconhecimento de Objetos, Antecipação de Ações, Modelo Preditivo, Controlador de Robô, Robô Colaborativo

Resumo

A Colaboração Homem-Robô surge como um conceito que complementa a flexibilidade e precisão de um trabalhador humano com a força e o menor custo de um trabalhador robótico no mesmo espaço de trabalho. Avanços recentes no campo da robótica colaborativa visam dotar os robôs industriais de capacidades de previsão e antecipação. Em muitas tarefas compartilhadas, a capacidade do robô de perceber e reconhecer com precisão os objetos que estão sendo manipulados pelo operador humano é crucial para fazer previsões sobre as intenções do operador. Neste contexto, esta dissertação tem dois objetivos principais. Em primeiro lugar, o objetivo é estabelecer a infraestrutura de hardware e software de um sistema colaborativo, que inclui um robô e duas câmeras utilizando o framework ROS. Esta infraestrutura inclui também um controlador de robô que antecipa o trabalhador humano, tomando decisões considerando as suas intenções derivadas dos dados dos sensores. Em segundo lugar, é desenvolvida uma framework baseada em aprendizagem para permitir que um robô auxiliar reconheça o objeto agarrado pelo operador humano com base no padrão das articulações da mão e dos dedos. A framework combina os pontos fortes do software MediaPipe na detecção de pontos de referência nas mãos a partir de uma imagem RGB com um classificador multiclasse profundo que prevê o objeto manipulado a partir dos pontos-chave extraídos. Este estudo concentra-se na comparação entre duas arquiteturas de aprendizagem profundas, uma Rede Neuronal Convolucional e um Transformer, em termos de exatidão de previsão, precisão, recall e F1-Score. O desempenho do sistema de reconhecimento é testado em diferentes conjuntos de dados com diferentes usuários e em diferentes sessões. Os resultados demonstram a eficácia dos métodos propostos, ao mesmo tempo que fornecem informações valiosas sobre os fatores que limitam a capacidade de generalização dos modelos.

Keywords Human-Robot Collaboration, Machine Learning, Object Recognition, Action Anticipation, Predictive Model, Robot Controller, Collaborative Robot

Abstract Human-Robot Collaboration emerges as a concept that complements the flexibility and precision of a human worker with the strength and lower cost of a robotic worker in the same workspace. Recent advances in the field of collaborative robotics aim to endow industrial robots with prediction and anticipation abilities. In many shared tasks, the robot's ability to accurately perceive and recognize the objects being manipulated by the human operator is crucial to make predictions about the operator's intentions. In this context, this dissertation has two main objectives. Firstly, the aim is to establish the hardware and software infrastructure of a collaborative system, which includes a robot and two cameras using the ROS framework. This infrastructure also includes a robot controller that anticipates the human worker, by making its decisions considering his intentions derived from sensor data. Secondly, a learning-based framework is developed to enable an assistive robot to recognize the object grasped by the human operator based on the pattern of the hand and finger joints. The framework combines the strengths of the commonly available software MediaPipe in detecting hand landmarks in an RGB image with a deep multi-class classifier that predicts the manipulated object from the extracted keypoints. This study focuses on the comparison between two deep architectures, a Convolutional Neural Network and a Transformer, in terms of prediction accuracy, precision, recall, and F1-score. The performance of the recognition system is tested on different datasets with different users and in different sessions. The results demonstrate the effectiveness of the proposed methods while providing valuable insights into the factors that limit the generalization ability of models.

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial Intelligence
CNN	Convolutional Neural Network
EMG	Electromyography
HCI	Human-Computer Interaction
HOI	Hand-Object Interaction
HRC	Human-Robot Collaboration
HRI	Human-Robot Interaction
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multi-Layer Perceptron
OMPL	Open Motion Planning Library
RNN	Recurrent Neural Network
ROS	Robot Operating System
SVM	Support Vector Machine

CHAPTER 1

Introduction

1.1 Background

The Third Industrial Revolution was characterized by a focus on automating repetitive and heavy tasks on the assembly lines. Still, this created a problem: whenever the manufacturers needed the robots to work in a different assembly process, they needed to be reprogrammed by an expert. The Fourth Industrial Revolution, also known as Industry 4.0, refers to the current trend of the manufacturing sector to become more intelligent and achieve greater automation. This trend takes advantage of the recent developments in artificial intelligence, the Internet of Things, and autonomous robots to pave the way for more efficient and flexible production processes. With Industry 4.0, robots are expected to be more adaptable and perform more actions without constant explicit programming.

The concept of Human-Robot Collaboration (HRC) emerges as part of Industry 4.0 and involves the research of mechanisms that allow humans and robots to work together to achieve a shared goal. Some of the most relevant topics in recent research include collision avoidance and human-aware planning of robot motions. However, to achieve true collaboration, it is not enough to react to the partner's movements and intentions, the robot must anticipate them.

Artificial Intelligence (AI) has significantly evolved in the last few years. With the increase of computational power, machine learning (ML), a subset of AI, has become an increasingly promising method to deal with complex data like images and text, heavily contributing to areas such as visual perception and speech recognition. The ability of ML models to learn from data with minimal human intervention and understand new data it has never seen before makes it a prime candidate to solve many problems in robotics and HRC in particular.

Anticipation is a research topic in many areas, such as biology, brain studies, psychology, social sciences, artificial intelligence, and engineering. One of the most cited definitions in the last decades and across the various fields is Rosen's [1]:

> An anticipatory system is a system containing a predictive model of itself and/or its environment, which allows it to change state at an instant in accord with the model's predictions pertaining to a later instant.

In the field of biology, Louie [2] claims that «Much, if not most, biological behavior is model-based ...» with the referred models being the «... internal predictive models of themselves and their environments ...». Poli [3] further claims that «... given that anticipatory behavior dramatically enhances the chances of survival, evolution itself may have found how to give anticipatory capacities to organisms, or to at least some of them.». For example, an animal predicts that it will be attacked by its predator and dodges said attack to survive.

In the case of humans, Louie [2] also stated, «We typically decide what to do now in terms of what we perceive will be the consequences of our action at some later time.» alluding to our anticipatory behavior. Therefore, human actions can result from reactive behavior when they are based on the past, from anticipatory behavior when they are based on predictions of the future, or from a mix of both.

1.2 Motivation

Human-Robot Collaboration is a research topic becoming increasingly important in modern industry, driven by the need to enhance productivity, efficiency, and safety in work environments [4]–[9]. The combination of human skills and robotic capabilities provides significant potential to improve the execution of complex and repetitive tasks. However, effective synchronization of actions and seamless communication between partners are open challenges that need to be further addressed [10]–[12]. In recent years, there has been a remarkable trend toward endowing collaborative robots with cognitive abilities, transforming them from simple automated machines into intelligent and adaptable collaborators. This shift is driven by the increasing demand for robots that can work alongside humans, understand their intentions, and actively contribute to complex tasks in dynamic environments. Collaborative cognition encompasses a range of essential abilities to enable robots to learn, predict, and anticipate human actions [9], [13], [14].

In collaborative scenarios, assistive robots are designed to work alongside humans in assembly processes or maintenance operations, providing timely support to enhance the overall efficiency of the task. Robots can assist the human worker by delivering a component, tool, or part, by holding a part while the operator works on it, or by performing autonomously a specific sub-task. In any case, the ability of an assistive robot to anticipate the upcoming needs of a human operator plays a pivotal role in supporting efficient teamwork. By anticipating human intentions, actions, and needs, robots can proactively assist or complement human tasks, providing timely support and improving overall efficiency.

The concept of intention anticipation is related to a double ability of robotic systems [15]–[18]. First, the ability to predict an action even before it occurs (or, before it is fully executed), by using the partial information provided up to a certain moment in time. The second ability consists of using this information to proactively plan their actions and adjust their behavior in real-time, providing smoother collaboration and minimizing potential conflicts or delays. However, achieving cognitive capabilities poses significant challenges, such as the need for robust real-time perception systems, efficient learning algorithms, and context understanding.

This anticipation ability can be achieved through distinct approaches, such as recognizing subtle cues, observing the progress of the task, or predicting human-object interactions [16], [19]–[21]. Different cues can contribute to the legibility of human actions according to the activity being performed. For example, eye gaze, head orientation, changes in body language, facial expressions, and even voice tone may be used to discriminate the operator's emotional state, level of engagement, and potential intentions. The source of anticipatory information may also result from monitoring the progress of the ongoing task and assessing the completion status of various sub-tasks. Therefore, the robot can anticipate the future by observing the sub-tasks performed in the past and reasoning about the future based on the structure of the complete task. Along the same line, the robot can predict the likely sequence of actions the operator is about to take by analyzing patterns of behavior and decision-making, enabling it to adjust its own actions.

This dissertation aims at the development of an anticipatory system to enhance human-robot collaboration in industrial settings under the AUGMANITY mobilizing project[1]. It is inspired by a collaborative scenario in which the robot observes the actions of the human operator, makes predictions about the human's intention, and reacts accordingly by either waiting for more observations or executing a physical action. This work focuses on the robot's ability to accurately perceive and recognize the object being manipulated by the human operator as a key element in making predictions about its needs. Knowing the object in the user's hand provides valuable contextual information revealing both current activity and future intention. However, this method comes with a limitation, while being handled, occlusions and partial views may compromise the actual perception. Another major challenge is the lack of labeled training data, as well as the time and costs of generating them.

The proposed concept is generic and it can be used in many human-robot interaction (HRI) scenarios. The robot can use this information to adjust its posture accordingly and/or to provide relevant assistance. For example, suppose a robot is collaborating with a worker on a factory assembly line. The robot observes the worker's action and realizes he/she just picked up a specific object. Based on object recognition, the robot can adjust its own position or prepare the required tool to assist the worker in the assembly process. In this way, the robot anticipates the human's action and streamlines the workflow. This example illustrates how anticipation can be applied in HRC scenarios to enhance interaction and overall efficiency: the robot understands the user's needs by recognizing the object being grasped or manipulated (Figure 1.1).

[1]AUGMANITY website: `https://www.augmanity.pt`

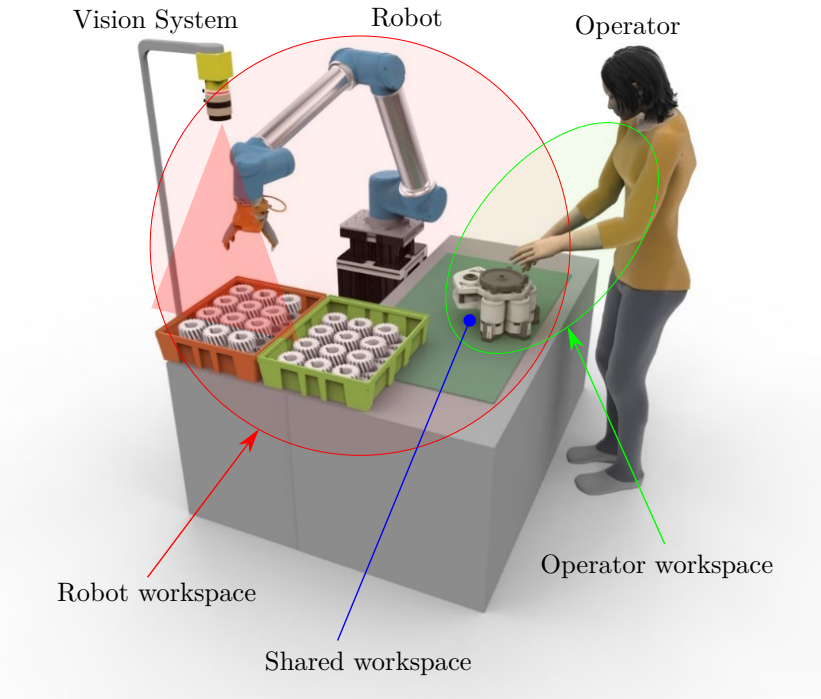


Figure 1.1: Illustration of a robot-assisted assembly system in a collaborative cell (adapted from [22]).

1.3 Objectives

This dissertation addresses the importance of anticipation in collaborative robotic tasks, exploring its challenges and practical applications in industry settings. The focus is on a framework that is able to derive the user's intention from sensor data by recognizing the grasped object. In summary, the main objectives are the following:

1. Review important concepts in HRC and, in this context, study the current research direction of action anticipation including common machine learning methods and how they impact the behavior of the robot. Additionally, look into the state of the art methods related to object recognition.
2. Establish the hardware and software tools used in this dissertation. Develop an infrastructure in ROS to support a practical implementation of action anticipation in the context of HRC with a robot controller that considers the human partner's intentions to make appropriate decisions during the execution of a sequential assembly task.
3. Explore and apply the potentialities of the MediaPipe framework. Produce deep learning models capable of perceiving and recognizing the objects being grasped by the user by using the right-hand keypoints. Perform extensive experiments, comparing the different models developed, to demonstrate the potential and limitations of the proposed approach analyzing, in particular, the generalization performance and/or model failure across different trials for the same user and across multiple users.

1.4 Document Structure

The remainder of the document is organized into four chapters. Chapter 2 provides an overview of the existing literature and research about collaborative robotics, action anticipation, and object recognition, contextualizing the work within the current state of the art. Chapter 3 reviews tools used for this study and describes the initial more simple implementation of an anticipatory robot controller. Chapter 4 presents the core components of the proposed approach, including the data acquisition process, the preprocessing steps applied to the collected data, and details of the neural network architectures implemented as multi-class classifiers and then discusses the results obtained from the experiments carried out, shedding light on the potential and limitations of the proposed approach for object recognition. Chapter 5 concludes the document by outlining the main conclusions of the study and it highlights potential avenues for future research.

CHAPTER 2

State of the Art

This chapter looks into previous work on collaborative robotics, action anticipation, and object recognition. Section 2.1 covers the background concepts associated with collaborative robotics including communication methods and safety. Section 2.2 explains the necessary fundamentals about action anticipation and provides some approaches from the current research direction in this theme. Section 2.3 explores the methods commonly used in object recognition.

2.1 Collaborative Robotics

Human-Robot Collaboration (HRC) consists of robots and humans working in the same workspace towards a common goal. Classical industrial robots are usually automated to perform repetitive tasks that require high physical strength. On the other hand, tasks that require cognitive knowledge, flexibility, and precision are better suited for humans, even if they are physically weaker. HRC aims to take advantage of both of their strengths and complement each others' weaknesses to increase manufacturing efficiency.

In a HRC scenario, robots need to be different from the traditional ones, given that they will work in the same workspace as humans. According to Castro *et al.* [9], «Collaborative robots need to be endowed with a set of abilities that enable them to act in close contact with humans, such as sensing, reasoning, and learning. In turn, the human must be placed at the center of a careful design where safety aspects and intuitive physical interaction need to be addressed as well.». In [23], it is stated that nowadays, collaborative robots are developed to be compact, easy to install and program, flexible, mobile, consistent, and precise. Additionally, they positively impact employees since they are responsible for monotonous and dangerous actions and reduce the production cost for the company.

2.1.1 Human-Robot Communication

Humans and robots can communicate through several methods, which can be direct such as using a console or a remote, or indirect, resulting from data captured from sensors. Based

on [9], [24], [25], the main methods for indirect communication can be seen in the diagram in Figure 2.1 and can be described as follows:

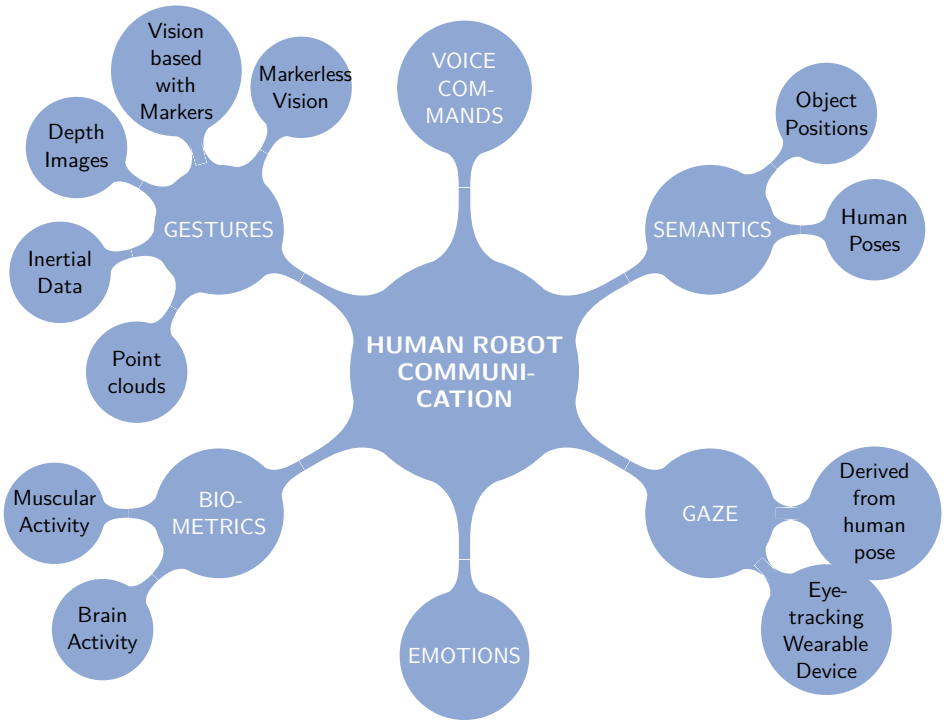


Figure 2.1: Common data sources in Human-Robot Collaboration.

- **Gestures**: these are one of the main ways humans communicate, whether through simple movements or formal sign language. In the literature about HRC, gestures can also commonly be found since they have the advantage of resisting ambient noise. Usually, gestures are captured with vision-based methods with either an RGB or RGB-D camera, so there is no need for unnatural movements. With vision, it is possible to include markers, but these may lead to occlusions and hinder the worker's movements. Consequently, there is also work in the literature that uses markerless vision to allow more unrestricted movements. Another way to capture the movements of the human worker would be to use wearable inertial sensors, which contain accelerometers and gyroscopes, but, once again, wearables can hinder the worker's movements. Finally, capturing point clouds using a LIDAR presents another possibility of capturing gestures without restricting the worker's motion.
- **Voice Commands**: talking is the most intuitive way for humans to communicate with each other. The advances in voice recognition and natural language processing make this a possible communication solution with robots. However, despite being intuitive, simple, effective, and even robust against lighting variations, when it comes to an industrial setting that contains significant sound noise, it becomes less valuable than the alternatives.
- **Semantics**: semantic information about the objects can also help the global workflow. For example, suppose the robot is trained to recognize certain features in objects related

to how it can pick them up. In this case, the robot can pick up a new object it has never seen before if it has a similar structure. Human actions can also be represented semantically by obtaining the poses of the human as a specific set of limbs, even if only partially. During action recognition, this can be used to know which objects the worker can interact with. Having semantic information about the pose of the human body also helps in the path-planning phase of the robot since it can use this information to avoid the worker and prevent collisions.

- **Gaze**: this can be used to determine where the user's attention resides, giving a considerable amount of information that can trigger some action. There are two options to obtain the user's gaze. Wearable sensors can provide better results but are expensive and intrusive. On the other hand, algorithms that detect head pose and assume the gaze from it can also be used, which is a cheaper and non-intrusive solution.
- **Emotions**: although this is a relatively new idea, some applications analyze the user's emotions from his facial expressions to have even more information in the algorithms.
- **Biometrics**: electromyography (EMG) sensors can measure electrical signals generated by muscle contractions, while electroencephalography (EEG) signals are commonly used in brain-computer interfaces (BCIs).

2.1.2 Safety

Safety is one of the most critical topics in collaborative robotics and the first step toward establishing a collaborative environment. According to [23], collaborative robots are able to safely work with people because they have sensitive sensors that can detect the human interrupting them, causing them to stop their actions, while traditional robots would potentially injure the worker. However, given that there are tasks that require the robot to move very close to the worker, some norms were implemented: ISO 10218-1 and 10218-2. From these two standards, Castro *et al.* [9] and Villani *et al.* [5] describe the four criteria (Figure 2.2) from which at least one must be met:

1. **Safety-rated monitored stop**: when a human enters the cobot's workspace, it completely stops.
2. **Hand guiding**: when an operator manually moves the cobot, it is compliant.
3. **Speed and separation monitoring**: as the human moves closer to the cobot, it becomes gradually slower.
4. **Power and force limiting**: the cobot has its operation restricted in terms of force and torque.

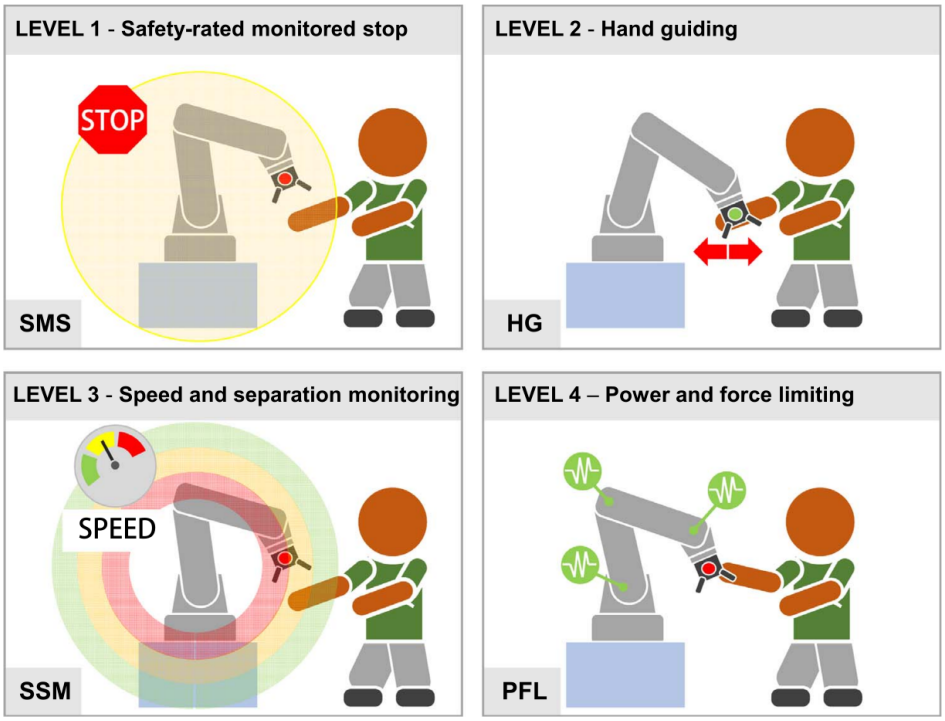


Figure 2.2: The four collaborative operative modes identified by robot safety standards ISO modes 10218-1/2 [5].

2.2 Anticipation System

2.2.1 Concepts

The concept of anticipation has been studied in several research fields and, in general, anticipation is viewed as the impact of predictions on the current behavior of a system, be it natural or artificial. A prediction model provides information about the possible future state of the environment and/or system. This perspective of looking to the future is related to the purpose of incorporating that information into a decision-making or planning process. Accordingly, the system becomes anticipatory when it incorporates such a model and, simultaneously, when it uses the model to change its current behavior.

Over the last few decades, experimental evidence of the existence of anticipatory biological processes at different levels of organization has been reported [3], [26]. The ability to modify behavior in anticipation of future events offers an adaptive advantage to living organisms with an impact on behavioral execution and learning. Anticipation is also considered one of the required abilities of cognitive robots operating in dynamically changing environments. The role of anticipation is to connect the robot's action in the present to its final goal, helping the design of robots with an increased level of autonomy and robustness.

The fundamental aspects of anticipation lie at the intersection of concepts such as time and information, involving abilities such as perception and prediction. The above definition of anticipation contains a temporal element that provides a key division between anticipatory and non-anticipatory robots. Anticipatory robots make decisions based on current states and predicted future states using predictive models of the environment. At the other extreme of

the spectrum are the robots that live in the present based on the current state of the observed environment, which are usually called reactive robots (e.g., the Braintenberg's vehicles [27]). However, the behavior of a purely reactive robot is limited by its temporal horizon since they have no memory of the past to build a model of the world. Most of the current robots present a behavior influenced either by the current perception as well as by the memory of past perceptions but still lacking a perspective of the future.

Information provides another defining aspect of anticipation since the prediction of a future state depends on sensory data. The challenge arises from the moment that an anticipatory system operates based on a potential future state (even before it occurs) that can only be inferred from past and current information. The inherent uncertainty associated with the prediction process can be reduced through the acquisition of information, namely by using different sensory modalities. In this context, sensory fusion is a process often adopted to merge data from multiple sensors such that to reduce the amount of uncertainty that may be involved to produce more reliable knowledge about the future.

The nature of anticipation and the mechanisms that support it are considered open questions in AI and robotics. Current research addresses fundamental questions such as: in which situations is anticipation useful? How can anticipatory processes be modeled and implemented in robotic systems? What are the impacts that may result from an anticipatory behavior? In the context of this dissertation proposal, anticipation is considered a combination of prediction and decision-making, as illustrated by the blocks diagram in Figure 2.3. The prediction model offers the possibility of incorporating action selection in their planning through a decision-making block, while the planning module relates to the robot's actions. These modules can be developed separately, or an end-to-end learning technique could be used where the model learns the different parts from the perception to the feedback control.

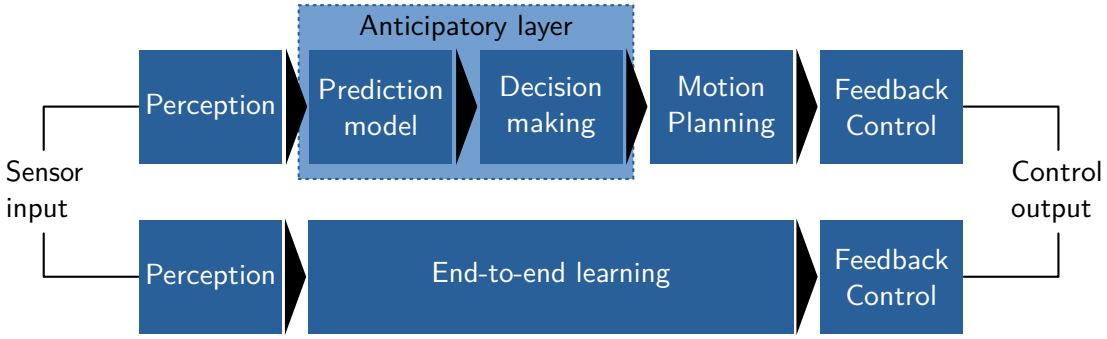


Figure 2.3: Functional blocks of an anticipatory robotic system considering two alternative approaches: modules developed separately vs end-to-end learning.

There are different situations in which an anticipatory response seems to be an essential ability for effective robot behavior. In an attempt to distinguish different types of anticipatory behaviors, three contexts in which a robot can operate are categorized below and the respective task requirements are presented as follows:

- **Time synchronization**: the interception of moving objects is central to several benchmark robotic tasks such as ball-catching and playing table tennis [28], [29]. These

tasks are challenging due to the demanding spatial–temporal constraints, which require continuous coordination between visual, planning, and control systems. On the one hand, frequent repredictions of the target location are required as new observations become available. On the other hand, this progressive refinement imposes an online re-planning of robot motion such that the goal is achieved in time.

- **Preventive safety**: systems that manage risk require some form of anticipatory mechanism such that the robot can adapt its behavior when an undesired situation occurs. Autonomous driving is an example of how predicting future events and reacting properly are important abilities to mitigate risk. Modeling behavior and predicting the future intentions of pedestrians are core elements to ensure that the driver stops the car safely or avoids the pedestrian in time.
- **Coordinate joint activities in human-robot interaction (HRI)**: humans have the ability to coordinate their actions when carrying out joint tasks with other partners [15], [30]. In the same line of thought, anticipation can enhance the ability of a robot to interact with a human partner by predicting their actions (or intentions) before selecting its own action plan. In collaborative contexts such as those that occur during manufacturing or assembly tasks, the main challenge is combining anticipation and planning in a context of high uncertainty due to the variability of human behavior in complex industrial environments. Anticipation seems to have a significant potential for more fluid and natural interaction with an impact on safety and cycle time.

2.2.2 Approaches

The ability of robots to accurately predict and anticipate human actions and intentions can greatly improve their ability to work safely and efficiently with humans in a shared workspace. Human intention recognition involves using sensors and machine learning algorithms to predict a human's intended action or task based on their movements, posture, and other contextual cues.

Regarding the sensors used to capture the raw data, most literature suggests using a RGB camera. However, the captured images may be used in the following different ways:

- directly used as input to models which can extract features from the images;
- used as input to frameworks that receive an image, process it, and return the keypoints, such as the skeleton joints of the person in the image; these keypoints can also then be used to assume the gaze of the human in the image such as in Canuto *et al.* [31] where the authors used OpenPose (explored in detail in subsection 3.2.1) to obtain not only the skeleton joints but also the worker's gaze;
- used to process the optical flow[32]–[35];
- if the human was wearing markers, the image can be used to obtain the positions of the markers obtaining gestures from the sequence of those positions [36];

Besides RGB cameras, some works, such as the one described in Moutinho *et al.* [37], indicate the use of an RGB-D camera to capture both the color and the depth images, which contain the gestures and pose of the worker. Other than cameras, in Tortora *et al.* [38] IMU and EMG data was used as input to capture the gestures and anticipate the worker's action.

When it comes to obtaining the worker's gaze, it is possible to do so from the RGB images as mentioned above, but it is also possible to use wearable sensors to capture it, such as in Schydlo *et al.* [39].

After knowing which data is usually captured and provided to an algorithm, the remainder of this subsection explores possible algorithmic solutions present in previous work starting with those that are only about predicting the action of the human worker and then those that go a step further and reference how to go from a prediction to the action that the robot must execute as a response.

Predictive Modeling Techniques

In the state-of-the-art, anticipation is generally represented as a classification problem about predicting the next action of the worker, frequently done by using a sequence of images that must be classified as a particular future action class. Using Figure 2.4 as an example, the high-five action should be predicted before the frames that contain it are captured. The previous work with this kind of algorithm mainly includes Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), with the latter being the most common.

Figure 2.4: Action Anticipation using Supervised Learning diagram [32].

In Furnari and Farinella [35], the authors aimed to predict the subsequent actions that someone wearing a camera would perform and the objects he would interact with. They used three datasets containing RBG frames from which they derived the optical flow and the objects in the environment. This data is then passed on to a Rolling-Unrolling Long Short-Term Memory (LSTM). The Rolling LSTM (R-LSTM) is a network that continuously encodes the received observations and keeps an updated summary of the past. When it is time to make predictions about future actions, the Unrolling LSTM (U-LSTM) is used with its hidden and cell states equal to the current ones of the R-LSTM.

In Schydlo *et al.* [39], the authors used an encoder-decoder recurrent neural network topology to predict human actions and intent where the encoder and the decoder are both LSTM networks. At each step, the decoder returns a discrete distribution of the possible actions making this algorithm able to consider multiple action sequences, which are then subject to a pruning method that reduces them to obtain the right action finally. In their work, these algorithms were tested in two different datasets, one containing RGB images with optical markers and gaze information from wearable sensors and another with RGB-D images.

In Moutinho *et al.* [37], the authors aimed to increase the natural collaboration between the robot and the human in an assembly station by interpreting implicit communication cues. The data related to the environment was captured using an RGB-D camera. This data was then passed on to a ResNet-34, a pre-trained neural network that extracted the features from the images. These features are used as the input to a LSTM to perform human action anticipation.

In Gammulle *et al.* [32], the authors aimed to predict future frames while at the same time predicting the following action. In their implementation, they used public datasets with videos from which they obtained RGB images and optical flow streams. To consider both data sources, they also used two ResNet-50's, which are pre-trained networks, one to get the input features from the image and another from the optical flow, and 2 LSTMs to take into account both sequences of inputs. Then the two results are merged into a final classification. They also used two Generative Adversarial Networks (GAN) to generate the subsequent frames, but this is different from the focus of the analysis.

In Wang *et al.* [40], the authors used video datasets to train a model that would predict a future action from the observed frames. They used three pre-trained neural networks in their work: VGG-16, TS, and ConvNet, to extract features from the images. Then these features were aggregated using a Temporal Transformer Module (TTM), and finally, a Progressive Prediction Module (PPM) would anticipate the worker's future action. This article also addresses the issue of specifying what the algorithm should consider as an action. Although most of the literature often implies that the last frames captured by the camera are considered an action, given that those are the frames that contain the last action made by the user, the authors of this article go into greater detail. They tested and evaluated how many frames should be considered as the last action to obtain the best results using a metric from Geest *et al.* [41] named per-frame calibrated average precision (cAP) calculated with (2.1). In [40] it is defined with

$$cAP = \frac{\sum_k cPrec(k) \times I(k)}{P}, \tag{2.1}$$

«... where calibrated precision $cPrec = \frac{TP}{TP+FP/w}$, $I(k)$ is an indicator function that is equal to 1 if the cut-off frame k is a true positive, P denotes the total number of true positives, and w is the ratio between negative and positive frames. The mean cAP over all classes is reported for final performance.».

In Rodriguez *et al.* [34], the authors aimed to predict the next action by first predicting the following motion images. They used datasets containing videos and then processed them to obtain motion images. These motion images become the input of a convolutional autoencoder network that generates the following motion images. These images are then passed to a CNN that processes them and makes action predictions for the future. The final action prediction is obtained from the results of the previous network and those of a second CNN, which analyzes the original RGB images.

In Wu *et al.* [33], the author's goal was to predict the following action someone wearing a camera would perform after some time. Initially, the optical flow was obtained from the captured images, and both were used as input to the model. The model is comprised of a

Temporal Segment Network (TSN), a CNN, and a LSTM to predict the future frame features and then use them to perform the required classification.

From Prediction to Planning

After predicting the next action of the worker, the robot must execute some action as a response to complete the anticipation process. This subsubsection contains articles that go beyond the predictive model and have relevant details for the integration of the model in a controller.

In Canuto *et al.* [31], the authors aimed to predict the following action using a LSTM. In their work, they used a dataset captured with an RGB camera. From these images, they obtained the objects in the environment, the human skeleton joints extracted over time using OpenPose, and the gaze derived from the joints. Then the three data sources were given to the LSTM as input to perform the desired classification. In this process, the authors use an adaptive threshold on the uncertainty of the recurrent neural network, which makes the model need a certain level of certainty to classify the action as a particular class. This creates a more robust solution since a standard supervised learning algorithm would predict the class with the highest probability even if the model has low certainty about every category.

In Maeda *et al.* [36], the authors aimed to reduce the delay in the robot's response by anticipating the human worker and providing a screw or a plate accordingly. They captured the environment using an RGB camera and tracked the hand using optical markers. Then they predicted the following human action using a look-up table containing different orders for assembly actions. With the nearest neighbor algorithm, the actions of the human would be matched with a particular order. The limitation of this method is that all possible sequences need to be on the table because if they are not there, then the robot will match with a different order which may be undesirable. If the robot eventually notices that it did the wrong action, it would then follow a hard-coded contingency trajectory to return to the pre-grasping position. When performing a handover action, the previously captured data is used to generate possible trajectories and this is given to the feedback controller as a reference.

In Zhang *et al.* [42], the authors aimed to predict the intention of the human worker to provide him with the required piece. To achieve this, they used an RGB camera to capture the data from the environment. Then the images are given to a convLSTM framework where the CNN part is in charge of extracting features from the input images, and these features are then passed on to the LSTM to predict the intention. This article also tackles the issue of having several possible assembly orders. It solves it by creating a phase at the beginning of the collaboration in which the robot learns the assembly actions and their order from a demonstration. After the prediction of the intention of the worker, the robot proceeds to fetch the required piece. It uses a CNN to recognize said piece and ROS Open Motion Planning Library (OMPL) to handle the trajectory planning jobs. In terms of safety, the authors defined speed limits for the robot and ensured that the robot would avoid the workspace of the human. Then when it needs to move closer to the user, its speed is reduced to guarantee the user's safety.

In Huang and Mutlu [17], the authors' goal is to make the robot use the anticipated actions of the worker to decide its tasks. It monitors the worker's gaze using a wearable device and uses it to predict his intent using a Support Vector Machine (SVM). After predicting it, the robot uses an anticipatory motion planner named «MoveIt!» to plan its motion according to a certain confidence threshold. This means that while it is unsure of what the human wants, the robot starts to move toward the item it thinks the user wants but only really moves completely when it surpasses the threshold.

2.3 Object Recognition

The idea underlying this study is to explore the possibility of establishing a relationship between the object grasped by the human operator and his current needs in collaborative scenarios. The immediate conceptual roots can be traced to the Activity Theory [43] whose main concept is "activity" defined as a purposeful and developing interaction between actors ("subjects") and the world ("objects"). This framework has established itself as a key concept for research in human-computer interaction (HCI) and interaction design.

This section provides an overview of existing literature and research relevant to this study. In particular, the recognition of the object grasped by the user is the central block when it comes to achieving the main purpose. It is organized into two subsections – "object sensing" and "hand sensing" – to help categorize and differentiate methods that primarily focus on capturing and analyzing data related to the object itself from those that explicitly use data from the human hand to infer the object being grasped.

2.3.1 Object Sensing

Approaches within the "object sensing" category leverage visual information extracted from images or videos of the objects the user is interacting with, by using techniques from computer vision and machine learning to discern object identities based on their visual attributes. A common approach involves the extraction of visual features that can encompass color histograms, texture descriptors, contour shapes, and local key-points. Early works in this domain [44]–[46] applied traditional image processing techniques to extract features such as shape moments and color histograms, leading to initial success in recognizing simple objects. The surge of progress seen in recent years is largely due to the latest developments in deep learning [47], particularly CNNs, and geometric reasoning [48].

Deep learning has had an enormous impact on perception tasks with the design of effective architectures for real-time object recognition, providing significant advancements in accuracy and robustness. CNNs have demonstrated remarkable performance in extracting hierarchical features from images [49]. Transfer learning, where pre-trained models are fine-tuned for specific tasks, has enabled efficient object recognition even with limited training data [50]. A relevant vision-based approach is the one in which the process of recognizing the human-grasped object, across consecutive frames, comprises two sub-processes: hand tracking and object recognition. The hand detection and tracking system is commonly used for defining a bounding box around the grasped object that describes its spatial location. This initial step

can, in turn, simplify the object recognition algorithm as it can focus attention solely on the region where the object is likely to be present. This reduces the search space and the required computational resources. Object detection frameworks like YOLO (You Only Look Once) and Faster R-CNN fall under this category. They divide the RGB image into a grid and predict bounding boxes and class probabilities directly from the grid.

In parallel to deep learning, the recent availability of inexpensive RGB-D sensors has enabled significant improvements in scene modeling and human pose estimation. Some studies explore the fusion of multiple modalities to enhance object recognition. These approaches combine visual information with other sensory data, such as depth information from 3D sensors [51], [52]. This integration of modalities has shown promise in improving recognition accuracy, especially in scenarios with varying lighting conditions or occlusions. Researchers have also studied how to leverage information from multiple viewpoints (i.e., multi-view 3D object recognition) to enhance recognition accuracy [53]. This approach is particularly relevant for 3D objects, where recognizing an object's 3D structure from different viewpoints can aid in robust recognition. Techniques like using 3D point clouds, multi-view CNNs, or methods that combine RGB images and depth information fall under this category.

Despite their successes, methods within the "Object Sensing" category are often constrained by the variability of object appearances, limited viewpoint coverage, and sensitivity to illumination changes. As a result, the focus on object characteristics alone may not provide a complete solution, particularly in situations where the human hand's interaction with the object plays a crucial role.

2.3.2 Hand Sensing

Recognizing objects based on the interactions of the human hand is a complex problem due to the intricate nature of hand-object interactions (HOIs) and the variability in grasp patterns and gestures [21], [54]–[56]. Achieving accurate and real-time recognition involves understanding the relationships and dynamics between a human hand and the objects it interacts with (e.g., the interaction context, the person's actions, and the patterns that emerge over time), as well as the tactile and kinesthetic feedback generated during manipulation. Additionally, variations in grasp styles, object sizes, and orientation further worsen the complexity of the task. Several works propose interaction reasoning networks for modeling spatio-temporal relationships between hands and objects in egocentric videos during activities of daily life, such as playing an instrument, kicking a ball, opening a drawer (one-handed interaction), opening a bottle (two-handed interaction, cutting a vegetable with a knife). Main advances are due to the development of several human-centric datasets (e.g., V-COCO [57], HICO-DET [54] and HCVRD [58]) that annotate the bounding boxes of each human actor, the object with which he/she is interacting and the corresponding interaction. However, the creation of large-scale, diverse, and annotated datasets was still an ongoing effort.

Some works consider the HOI as a manifestation of human intention or purpose of action [59]–[63]. Despite the growing need for detection and inference of HOIs in practical applications, such as collaborative robotics, the problem of recognizing objects based on

hand-object interactions is inherently complex. Instead of addressing the full complexity of HOI recognition, several works have adopted targeted approaches that address specific aspects of the problem without necessarily delving into the entire spectrum of interactions. A recent work investigated the influence of physical properties of objects such as shape, size, and weight on forearm electromyography (EMG) signals and the opportunities that this sensing technology brings in hand-object interaction recognition and/or for object-based activity tracking [64]. Despite the relevance of the work, it is difficult to apply in collaborative assembly scenarios given the complexity of the required setup that requires sensor attachment, calibration, and training. Some other limitations may include user-dependent variability, muscle fatigue and discomfort, and/or interference from other electrical devices.

Another line of research, closest to this work, focuses on tracking the positions of hand and finger landmarks during interactions. By monitoring the spatial relationships of these landmarks, these methods aim to deduce the object's identity based on the specific manipulations applied. This approach captures critical information about the hand's interaction without necessarily modeling the full complexity of interactions. A glove-based interaction approach has been proposed by Paulson et al. [65] in the HCI domain to investigate a grasp-based selection of objects in office settings. Authors showed that hand posture information solely can be used to recognize various activities in an office, such as dialing a number, holding a mug, typing at the keyboard, or handling the mouse. The classification of hand posture is performed using the nearest-neighbor algorithm. In a similar work based on a data glove, Vatavu et al. [66] proposed the automatic recognition of the size and shape of objects using the posture of the hand during prehension. The objects used in the experiments consisted of six basic shapes (cube, parallelepiped, cylinder, sphere, pyramid, and a thin plate) and, for each shape, three different sizes (small, medium, and large). Twelve right-handed participants took part in the experiments using a 5DT Data Glove Ultra equipped with 14 optical sensors. These sensors are distributed as follows: 10 sensors measure finger flexion (two sensors per finger) and four sensors measure abduction between fingers.

The study compared several classifiers derived from the nearest-neighbor approach with a Multi-Layer Perceptron (MLP) and a multi-class SVM. The best results were achieved with the K-nearest-neighbor classification approach when combining the results of individual postures across an entire time window of half a second. The experiments carried out included the capture of hand postures when grasping and maintaining a stable grip for reliable translation of the objects. The results showed that object size and shape can be recognized with up to 98 % accuracy when using user-specific metrics. The authors also pointed out the lower accuracy for user-independent training and the variability of individual grasping postures during object exploration. Although in general, the proposed approach recognizes the physical properties of the grasped objects with high accuracy, wearing a glove directly on the hand is intrusive and troublesome, interfering with the natural movement of the fingers.

When attempting to model human grasping, researchers have focused their attention on defining a comprehensive taxonomy of human grasp types [67] and the multifaceted factors that influence the choice of grasping, including user intentions [68], object properties [69], and

environmental constraints [70]. MacKenzie and Iberall [68] theorize the existence of a cognitive model that converts the object's geometry properties and user's intent into a motor program driving the hand and finger motions. From this seminal work, several studies on human reach-to-grasp actions have consistently shown that the natural kinematics of prehension allows for predicting the object he/she is going to grasp, as well as the subsequent actions that will be carried out with that object. Feix *et al.* [69] provided an analysis of human grasping behaviors showing the correlation between the properties of the objects and the grasp choice. More recently, the works of Betti *et al.* [71] and Egmose and Køppe [72] focus on the reach-to-grasp phase. Their finding shows that grasp formation is highly correlated with the size and shape of the object to be grasped, as well as strongly related to the intended action. These insights promise improved interaction by exploring the ability in which the robot can predict the object the user intends to grasp or to recognize the one he/she is already holding, provided that the hand kinematics information is extracted and processed in real-time.

In line with this, Valkov *et al.* [73] investigated the feasibility and accuracy of recognizing objects based on hand kinematics and LSTM networks. The data is extracted from a Polhemus Viper16 electromagnetic tracking system with 12 sensors attached to the hand and fingers. On the one hand, the study focuses on the size discrimination of 9 synthetic objects: three regular solids (sphere, box, and cylinder) in three different sizes (small – 2 cm, medium – 4 cm and large – 6 cm). On the other hand, a different set of seven objects (pen, glue, bottle, Rubik's cube, volcano-egg, toy, and scissors) was used for object discrimination. The data recorded during the experiments includes a phase in which participants were asked to reach and grasp the object starting from a fixed initial position. The results demonstrated that LSTM networks can predict the time point at which the user grasps an object with 23 ms precision and the current distance to it with a precision better than 1 cm. Furthermore, the size and the object discrimination during the reach-to-grasp actions were achieved successfully with accuracy above 90 % using K-fold cross-validation. Although the results are still preliminary, the leave-one-out cross-validation showed a significant degradation in the performance of the models compared to the K-fold validation. While the tracking system offers many advantages, there are also practical limitations such as sensor attachment and comfort, line-of-sight requirements, interference and noise, and calibration and drift.

CHAPTER 3

Human-Robot Collaboration System

This chapter describes the hardware and software tools used during development and the ROS-based architecture to support a practical implementation of action anticipation in human-robot collaboration. In particular, Section 3.1 details the system architecture including the available hardware and the software tools chosen to establish communication between all the parts. Section 3.2 describes the research methodologies employed in this dissertation. Section Section 3.3 delves into the implementation of initial experiments which integrate all the parts to simulate anticipatory behavior. Section 3.4 provides final remarks about the content of this chapter and its connection to the next.

3.1 System Architecture

The work described in this dissertation is part of the AUGMANITY project[1] that aims to develop technologies to support people operating in industrial environments. The experimental setup comprises the integration of both hardware and software components in a prototype collaborative cell (LARCC) at the Laboratory for Automation and Robotics (LAR) located in the Department of Mechanical Engineering at the University of Aveiro, as illustrated in Figure 3.1. The LARCC is equipped with a UR10e collaborative robot and multimodal sensor devices, including three LiDAR Velodyne sensors and four Orbbec 3D cameras distributed throughout the work volume. The software architecture is built upon the Robot Operating System (ROS) middleware [74], providing a robust framework for communication and coordination among the various components. In this context, this section provides a description of the materials used during this study and the methodological approaches followed to face the key challenges.

[1] AUGMANITY website: `https://www.augmanity.pt`

Figure 3.1: Prototype collaborative cell LARCC.

3.1.1 Robot Operating System (ROS)

ROS[75][2,3] is an open-source collection of tools and software libraries used to develop a robotics application and, in this work, it is used to establish communication throughout all of the infrastructure. ROS was chosen due to the hardware abstraction it offers given that it contains driver packages to deal with some hardware devices, allowing for easier communication with the robot and the cameras. Other relevant features include:

- **message broker**: every process in the project is a node in the ROS network and communicates with the other nodes mainly through topics (asynchronous publish/subscribe streaming of data) or services (synchronous RPC-style communication).
- **code reuse**: executables and packages are written to be as independent as possible, making the developer able to reuse them in another project.
- **rich ecosystem**: there are several open-source packages available to the developer that can be easily integrated.
- **scalability**: given that the nodes are so loosely coupled, it allows for node distribution.
- **language independence**: nodes can be written in any language since communication is established through well-defined objects.
- **data visualization**: there are tools to visualize the data and the functioning of the system in real-time, such as Rviz.
- **simulator support**: ROS has support for simulators with Gazebo being the most common.

For this system, the ROS 1 Noetic distribution was chosen over the more recent ROS 2 distributions so as to take advantage of work already done by other members of the AUGMANITY project at the University of Aveiro.

[2]ROS 1 documentation: `https://wiki.ros.org`
[3]ROS 2 documentation: `https://docs.ros.org/en/humble`

3.1.2 Perception System

In order to capture the necessary information from the environment, two Orbbec Astra Pro RGBD cameras were used (Figure 3.2). This camera model was developed by Orbbec Technologies and it is frequently used in computer vision and robotics [76]. Among the available cameras, it was chosen since it allowed to capture both color and depth images.

Figure 3.2: Orbbec Astra Pro [76].

In the experimental setup, one of the cameras is placed above the workspace facing downwards allowing the perception of the objects in the table through the color image and the position of the user through the depth image. The second camera is above and slightly behind the robot to capture the user in front of the robot with the images from this camera being the ones used for detecting the hands keypoints. The communication with the cameras is established through ROS with the *usb_cam* package being used for the color image and the *ros_astra_camera* package being used for the depth image. In order to have better control over the lightness in the environment, only artificial lighting was used and the back-light compensation was raised to the maximum in the *usb_cam* configurations. Then the frame rate was also set to 10 frames per second in both cameras given that none of the following image processing tasks require a high frame rate.

As said before, the color image of the first camera is used to detect objects in the environment. These objects are blocks of specific colors and therefore they can be identified with color segmentation, which is done with the help of OpenCV[4], a popular computer vision library. With this library, the image from the camera is initially cropped to a specific region of interest consisting of the table area, which is then converted from BGR to HSV given that the intervals of a color are more reliable in this representation. From the HSV image and the color intervals, a mask is obtained for each color. These masks are then subject to a closing morphological operation to reduce any possible noise. Finally, the resulting areas are considered an object if they surpass a certain area threshold further reducing noise.

On the other hand, the depth image is used to obtain the position of the human in the workspace by cropping it to a certain region of interest corresponding to where the user would normally be and then detecting the highest point.

3.1.3 Manipulator Arm Control

The collaborative robot available for this work is a UR10e model which was developed by Universal Robots (left side of Figure 3.3). This model has six degrees of freedom with six

[4]OpenCV documentation: `https://docs.opencv.org/4.x/`

rotational joints, allows for payloads up to 12.5 kg, and has a reach of 1300 mm being suitable for tasks such as machine tending, palletizing, and packaging[77]. In this work, the robot is equipped with a 2F-140 gripper developed by Robotiq (right side of Figure 3.3), commonly used together with robot models from Universal Robots[78].

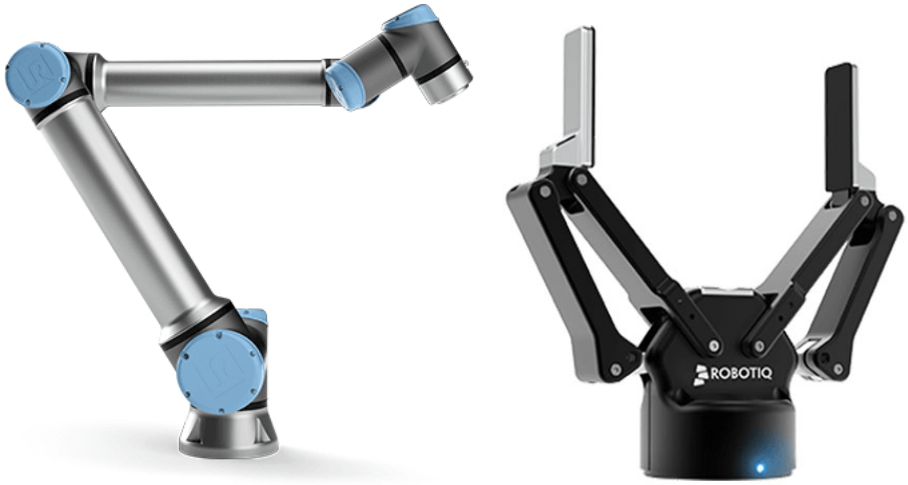


Figure 3.3: UR10e Collaborative Robot [79] and Robotiq 2F-140 Gripper [78].

Both the robot and the gripper have ROS packages containing their drivers making their integration easier. The planning and execution of the arm movements are done through the MoveIt[5] framework, which is a widely-used open-source framework for robotics applications involving motion planning, manipulation, 3D perception, kinematics, control, navigation, and collision checking, with OMPL being chosen to handle the motion planning tasks.

The configurations of the drivers and MoveIt was already done by other members of the AUGMANITY project and can be found on Github[6] along with a higher-level API that encloses that logic. During this work, an additional feature to stop movements was added.

3.1.4 Computational Systems

The tasks involved in this work, such as training deep-learning models and analyzing images in real-time require high computational resources. To handle the real-time processing of images and robot control, the central computer present in the setup was used. To handle the deep-learning model training, the deep-learning research server from LAR was used, codenamed Deeplar:

- AMD RyzenTM Threadripper 2950X;
- Four NVIDIA GEFORCE® RTX 2080 Ti;
- 128GB DDR4 RAM.

The model training in Deeplar is executed using docker images, which allows multiple people to use the computer with each having their own isolated training environment with their own dependencies. The images used to design and train machine learning models in this work are based on the latest TensorFlow official image for GPUs which uses Python.

[5]MoveIt documentation: `https://ros-planning.github.io/moveit_tutorials`
[6]Github LarCC Repository: `https://github.com/afonsocastro/larcc_interface`

TensorFlow is one of the most popular machine learning frameworks along with Pytorch. In this work, the former was chosen over the latter since the higher-level API allowed for faster development. The main features of Tensorflow[7] are:

- **prepare data**: load data, data pre-processing and data augmentation;
- **build models**: design and train custom models with little code or use pre-trained ones (transfer learning);
- **deploy models**: helps using models in different platforms such as locally, in the cloud, in a browser, or in mobile;
- **implement MLOps**: run models in production, tracking their performance and identifying issues.

As seen above, this work employs several tools, with Python being the common programming language between them and, therefore, the chosen language to ensure compatibility.

3.2 Software Tools and Deep Architectures

This section covers the software tools and deep architectures employed in this dissertation to research anticipatory systems.

3.2.1 Keypoints Detection Frameworks

This subsection reviews OpenPose, OpenPifPaf, and MediaPipe which are three projects containing models to detect keypoints in images, such as the human skeleton joints.

OpenPose

OpenPose[55], [80]–[82][8] is an open-source project that aims to detect keypoints in the human body, face, hands, and feet from images (Figure 3.4). Its main features are:

- 2D real-time keypoint detection based on the body/foot, the hand, or the face of multiple people;
- 3D real-time keypoint detection based on images from multiple cameras of one person;
- estimation of camera calibration parameters;
- single-person tracking;
- can be used through the command line or using an API for Python or C++.

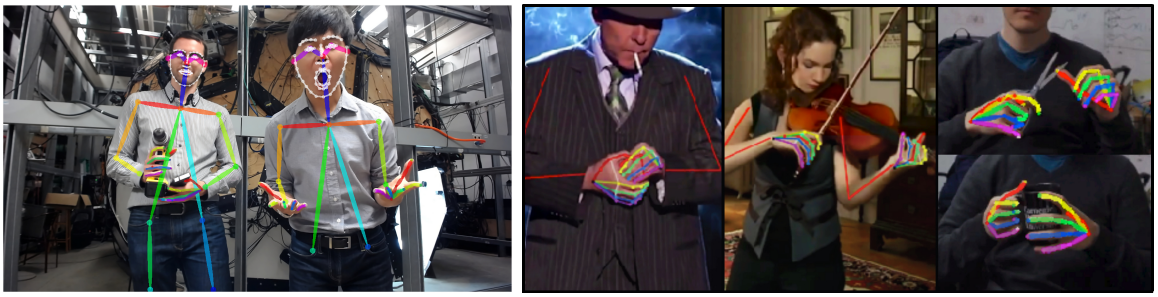


Figure 3.4: OpenPose Examples [55], [80].

[7] Tensorflow documentation: `https://www.tensorflow.org/api_docs`
[8] OpenPose documentation: `https://cmu-perceptual-computing-lab.github.io/openpose`

OpenPifPaf

OpenPifPaf[83], [84][9] is an open-source project that aims to detect, associate, and track semantic keypoints (Figure 3.5). Detecting human joints is an example of its usage but it is also able to generalize this detection to other classes such as cars and animals. It can be installed as a Python package which can then be imported.

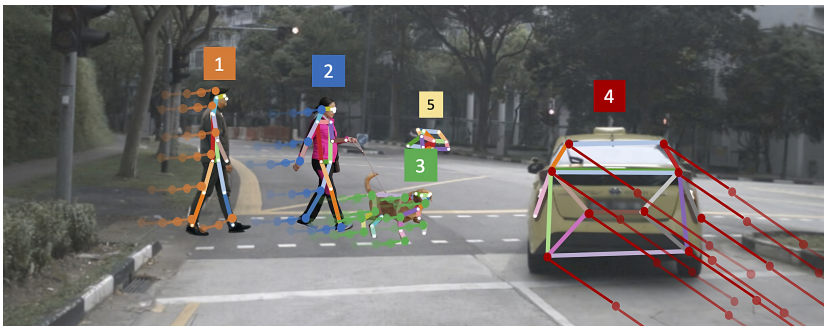


Figure 3.5: OpenPifPaf Example [83].

MediaPipe

MediaPipe[10] consists of a set of libraries and tools to apply AI and ML techniques in other applications, particularly in pipelines for advanced real-time vision-based applications [85]. Although it contains many features, in this work the focus is on the Hand and the Pose Landmark Detection Models. The Hand Landmark Detection model [86] uses two sub-modules: a hand palm detection model and a hand landmark model. Each frame of an RGB input is fed into the palm detection model, which produces a bounding box based on the palm. The hand landmark model uses this bounding box and returns the keypoint localization of 21 landmarks, including the fingertips, the finger joints (knuckles), and the base of the palm (Figure 3.6a). The Pose Landmark Detection model also uses two sub-modules working in a similar way to return 33 landmarks over the entire body (Figure 3.6b).

(a) **(b)**

Figure 3.6: Mediapipe landmarker models [87]: (a) Hand Landmarker and (b) Pose Landmarker.

[9]OpenPifPaf documentation: `https://openpifpaf.github.io`
[10]MediaPipe documentation: `https://developers.google.com/mediapipe`

From the three keypoint detection frameworks reviewed, MediaPipe was chosen for this work due to being easier to install compared to OpenPose, and having no dependency conflicts with other tools, which OpenPifPaf had. Additionally, MediaPipe was already being used by other members of the AUGMANITY project.

3.2.2 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are currently one of the most popular deep learning architectures and they are typically used to take advantage of the matrix structure of input data such as images and video where the input is too massive for other methods [88]. This architecture was originally introduced in a 1988 article [89] and has since evolved, giving rise to new variants. According to Alom *et al.* [90], the main building blocks are:

- **Convolutional Layer**: in this layer, feature maps are extracted from receptive fields, which are parts of the input such as a region of pixels in an image or lower-level feature maps. This is done using a learnable kernel, consisting of a sliding window that goes through the entire input, computing the feature maps.
- **Sub-Sampling Layer**: this layer is responsible for downsampling the input feature maps effectively reducing the dimension of each feature map. It can also be named a pooling layer with the most common strategy being Max Pooling which simply returns the maximum value of the input.
- **Classification Layer**: this layer is composed of a variable number of fully connected layers that use the higher-level features computed in earlier layers to obtain the final scores of each class.

The global architecture is then described as the junction of two main parts: Feature Extractors and a Classifier (Figure 3.7). The first part is composed of one or more pairs of convolutional and pooling layers. The repetition of these layers results in the progressive reduction of feature dimensions, allowing higher-level features to be derived from the previous layers. Finally, the classifier is made up of fully-connected feed-forward layers and a final classifier layer and it is responsible for converting the higher-level feature maps into an array of scores for each class.

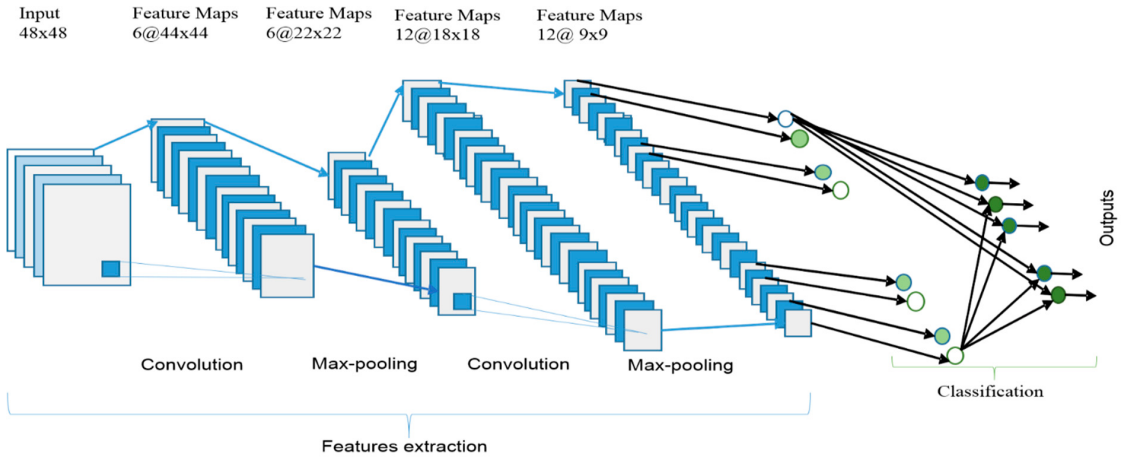


Figure 3.7: CNN Architecture [90].

3.2.3 Transformer Neural Networks

Transformer Neural Networks consist of a new kind of neural network architecture that aims to solve tasks involving sequence data such as those in natural language processing. They were initially proposed in the "Attention Is All You Need" paper published by Google Research in 2017 [91] and proceeded to surpass several established architectures in numerous tasks using attention mechanisms that identify complex relationships between elements in the input sequence. According to Kapoor *et al.* [92], the transformer's architectures are built upon 4 core concepts:

- **Positional Encoding** - RNNs are able to work with sequences given that the tokens are processed sequentially. However, this approach comes with the disadvantage of the model having greater difficulty in analyzing long sequences, since important data might be forgotten. Transformers address this issue by using positional encoding, assigning a unique number to each token representing its position in the input sequence. This enables the transformer to learn the significance of each token's position.
- **Attention** - Attention is a concept that consists of measuring the relative importance of the input tokens to the output tokens. It was initially designed to facilitate language translation given that, when translating text from one language to the other, each word in the output can be influenced by multiple words from the input, and it became a key idea behind the transformer's architecture.
- **Self-attention** - Self-attention derives from attention but consists of measuring the relative importance of the input tokens to the other tokens of the input sequence instead of the output tokens. This concept allows the model to learn the relationships between tokens and their relevance in the input sequence even if they are far away.
- **Multi-head (self-)attention** - Multi-head (self-)attention refers to the fact that a transformer can have multiple attention heads. Each attention head performs a parallel process of (self-)attention allowing for multiple resulting weight matrices and, therefore, multiple definitions of relevance between the tokens.

Figure 3.8 shows the first published Transformer architecture. Initially, positional encodings are added to the input and output embeddings providing information about the relative positions. Then, in the encoder (left side of the figure), the input goes through a multi-head (self-)attention layer and a position-wise feedforward layer. In the decoder (right side of the figure), a different multi-head attention is applied on the output embeddings, and its output is used alongside the output of the encoder in another multi-head self-attention layer ending in a feedforward layer. Finally, the output of the last Transformer block is passed on to a dense layer converting it into a sequence of probabilities.

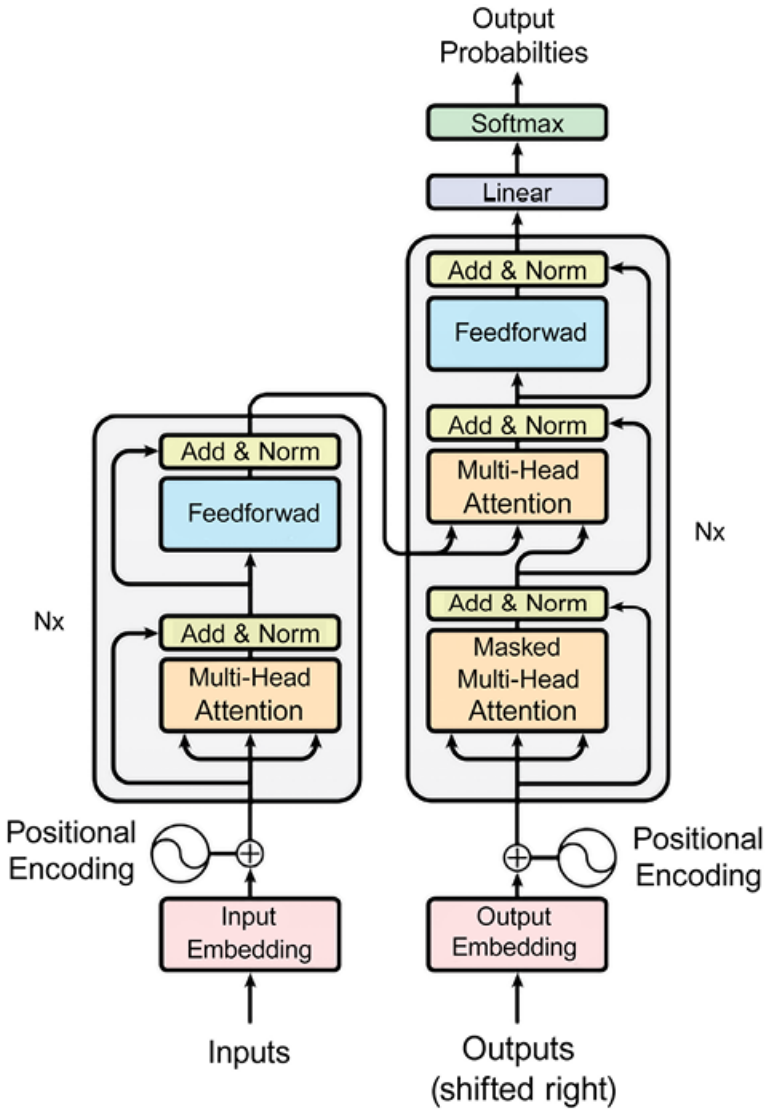


Figure 3.8: Transformer Architecture [91].

3.3 Integration of First Anticipation Experiments

This section covers the integration of previously described tools and then the implementation of the first experiments to simulate anticipatory behavior in HRC.

3.3.1 Assembly Task: Build a Striped Flag

To establish the required infrastructure, it was essential to define a concrete problem for evaluation, preferably involving some form of anticipation. For this work, it was chosen to put together sequences of big Lego blocks. The goal of the interactions would be to put three blocks together whose colors would be comparable to the stripes of a flag (Figure 3.9). In total, ten big cubic Lego blocks were used for this task: two red, two dark blue, two light blue, one white, one yellow, one green, and one orange. Additionally, there are also eight small blocks, one of every color mentioned in addition to violet, that are used for the interactions with the user.

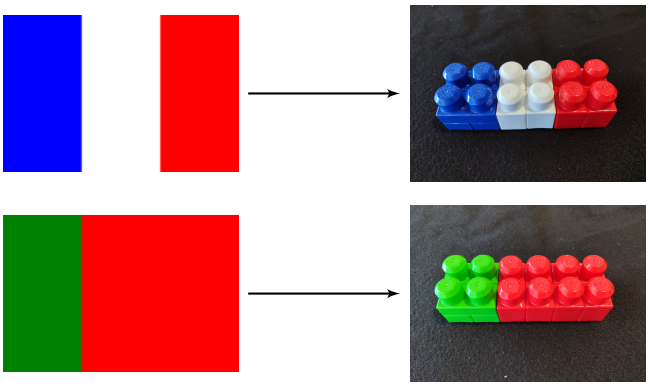


Figure 3.9: Flag examples.

Figure 3.10 shows the work environment with the table and the robot. The big blocks are initially placed behind the robot, with the robot being responsible for picking them up and placing them next to the user so that the user can build a flag. Then, there are also eight small blocks placed in front of the user that can be used for the interactions. The small violet block is used to tell the robot that it is fetching the wrong block by hovering it. The remaining small blocks are used to request that the next block is of a certain color by hovering them.

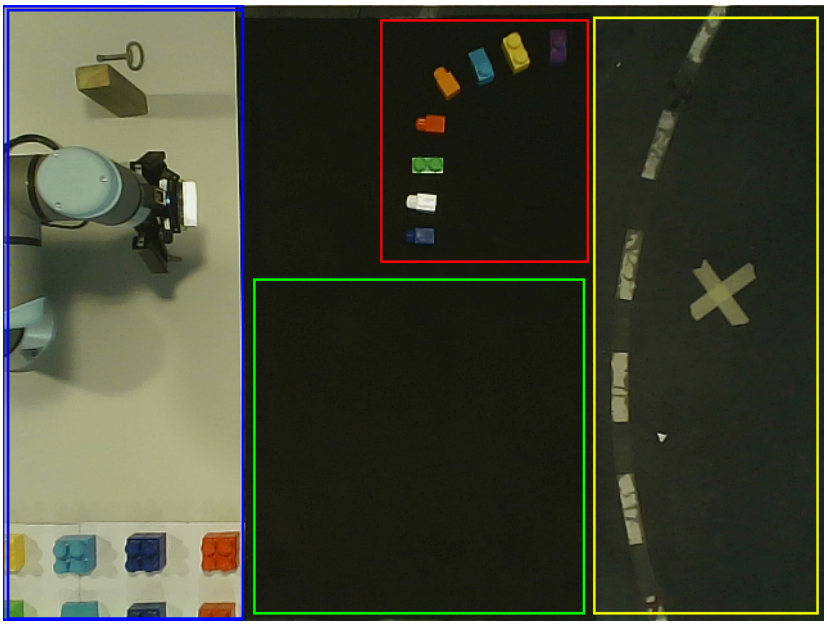


Figure 3.10: Work environment: red - small block hovering area, blue - robot workspace with the remaining blocks, green - flag assembling area, yellow - user area.

Before the anticipation experiments, a base infrastructure was developed to build upon, integrating the perception system described in subsection 3.1.2 and the manipulator arm control system described in subsection 3.1.3. Both of them are connected to the ROS network and therefore an additional node was added to it to manage the communication between them and the decision-making in the system (Figure 3.11). For this purpose, this node contains

a state machine where each state corresponds to a method of a class. As such, to integrate additional features or logic in the system, one only needs to create a subclass and redefine the relevant methods.

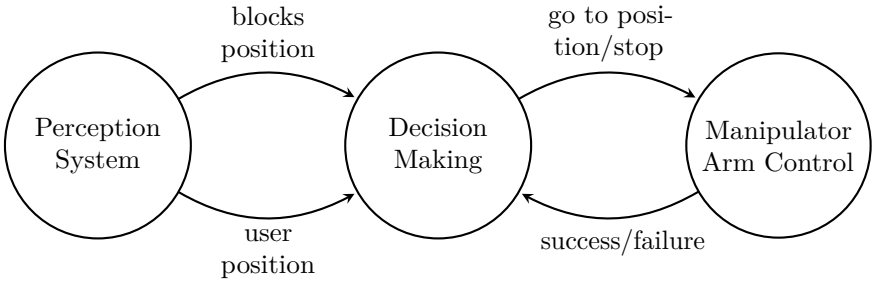


Figure 3.11: General ROS architecture with decision-making node.

The initial decision-making logic developed relied on interactions between the robot and the user. When the user puts his hand above a small block, if the robot is idle, then it proceeds to fetch a block of the same color. Additionally, when the user hovers his hand over the violet block while the robot is fetching a block, it means that the robot is fetching the wrong block, so it must stop and put the block back where it was before if it was already picked up. These interactions are also the fallback behavior if the methods in the following experiments fail to anticipate the block that the user desires.

State Machine Logic

As said before, the decision-making node contains an internal state machine. Although the logic of some states changes depending on the solution, all implementations follow the same general design represented in the state machine diagram shown in Figure 3.12.

- **idle**: state corresponding to when the robot does not know which is the next block. If the sequence has not started yet then it waits for the user to choose the first block and then the state changes to *picking up*. If the sequence has already started then the database is fetched for the possible next blocks and then if at least one block is returned the state changes to *picking up* or if none is returned the system waits for the user to choose the next block.
- **picking up**: state corresponding to while the robot is picking up a certain block. After it picks it up the state changes to *moving closer* unless it was picking the wrong object in which case it changes to *stop wrong guess*.
- **moving closer**: state corresponding to the movement until the put-down position opposite to the user so that the robot does not constrain him. After the robot reaches the put-down position the state changes to *putting down*. If the robot is holding the wrong object then the state changes to *stop wrong guess* instead and if the user changes side the state changes to *stop side switch*.
- **putting down**: state corresponding to the movement necessary to put down the block in the table and the retreat of the robot outside the user's workspace. After that, the state changes to *idle*.

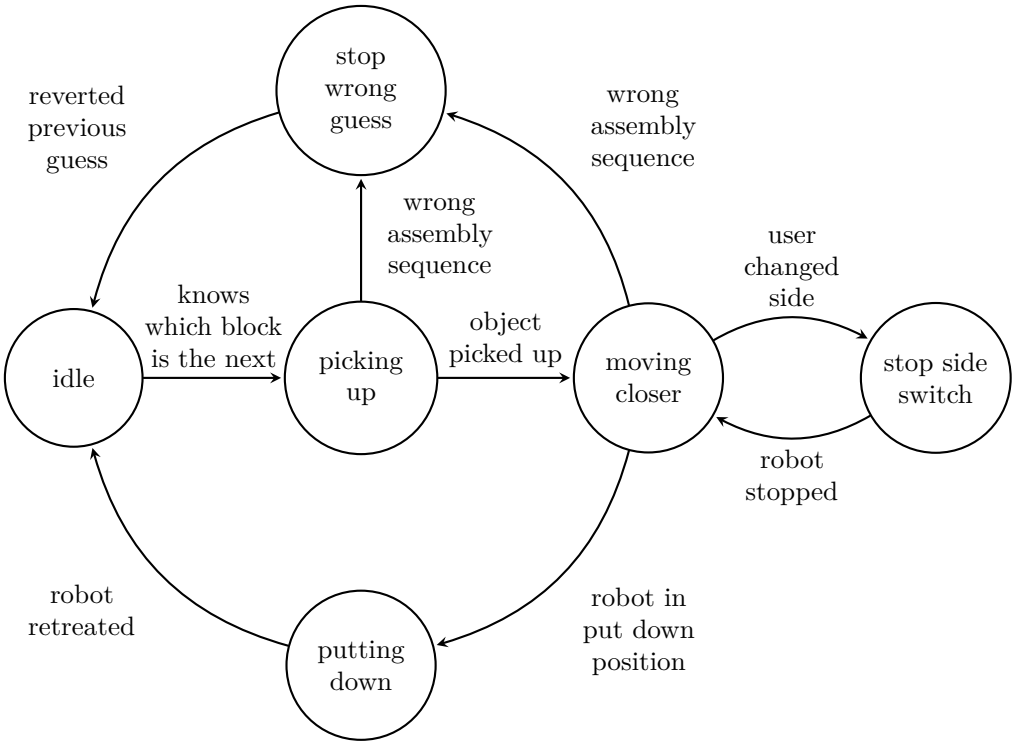


Figure 3.12: State machine diagram.

- **stop side switch**: State corresponding to the action of stopping the robot because the user changed sides. After the robot is stopped the state changes back to *moving closer*.
- **stop wrong guess**: state corresponding to the action of stopping the robot because the user indicated that it was the wrong block. If the robot was already holding a block then that block is put back where it was while in this state. After the robot is stopped and it is not holding a block the state changes to *picking up* if there are still more possible blocks in the database. Otherwise, the state changes to *idle* so that it waits for a user request.

3.3.2 First Anticipation Experiments

This subsection describes the implementation and the workflow of the different experiments performed.

Probability Based

The first experiment consisted of having a database of probabilities that the robot would check to anticipate the block that the user would need next. In this experiment, when the decision-making node starts, it reads a configuration file with all the possible flags and then calculates and saves in a database the conditional probabilities of a certain color considering the previous ones.

When assembling a flag, the first block is still requested by the user but the following blocks are automatically given to the user according to the probabilities. However, if the robot chooses the wrong block the user can still communicate that fact by interacting with

the small violet block and, if the robot exhausts all possibilities that it knows of, the user can request the following block using the interactions with the other small blocks.

Rule Based

The second experiment consisted of having a set of rules that the robot follows to anticipate the next block. These rules are read from a configuration file when the decision-making node starts and then managed by Experta[11]. Experta is a Python library to create and deploy rule-based systems that make informed decisions using a knowledge base of rules. In this library, rules are defined as combinations of conditions and actions allowing for an intuitive syntax. Then, the engine uses these rules to make decisions using the declared information. For this experiment, some rules were defined about what should be the next block considering the blocks previously provided and the blocks rejected since the last accepted one. For example, if the last accepted block was red and the user has since then rejected a dark blue block then the next block should be a yellow one.

Similarly to the previous experiment, the first block is requested by the user but for the following blocks, the robot checks if a rule is present for the current situation. If there is one, the robot follows that rule and provides a block and if there is not, it waits for the user to request the next block. The interactions for requesting and refusing a block are maintained.

Rule Based + MediaPipe

The third experiment is very similar to the previous one. The rule-based decision-making is maintained with the same rules but the first block is no longer requested by the user. Instead, the user picks up the first block and the images from the camera facing the user are used to detect the color of the object. For this purpose, the pose model from MediaPipe is employed to detect the hand's position, which is subsequently used for cropping the received image to a region of interest. The resulting image is segmented similarly to the images of the other camera and if a color is detected, then that color is considered the color of the first block.

3.4 Final Remarks

This chapter has reviewed the necessary components to build a Human-Robot Collaboration (HRC), beginning by the collaborative cell and how the different components are configured and capable of establishing communication through ROS. Then, the different tools that constitute this research are further described, including keypoint detection frameworks such as MediaPipe, Convolutional Neural Networks (CNNs), and Transformer Neural Networks. Finally, all components are connected with a a node that determines the robot's actions. The first implementation included a database of probabilities that guided which action the robot should take but this solution proved to have reduced flexibility. Then, the second implementation used Experta to set up a list of rules to decide the action the robot should take next with a configuration file, a characteristic that made the robot's behavior more

[11] Experta documentation: `https://github.com/nilp0inter/experta`

flexible since the rules could be easily changed. Finally, the last implementation integrated MediaPipe not only to reduce the direct communication between the robot and the user but also in preparation for its use in the next chapter.

CHAPTER 4

Learning-Based Recognition of Human-Grasped Objects

This chapter covers the development of machine learning models to recognize human-grasped objects. In particular, Section 4.1 formulates the problem with detail and describes the evaluation metrics. Section 4.2 details the data collecting and the dataset preprocessing and splitting. Section 4.4 and Section 4.5 describe the implementation of the CNN and Transformer models respectively and their results. Section 4.6 compares the results of both models in different cases. Section 4.7 delves into a possible implementation of a real-time implementation of anticipation along with possible limitations and solutions. Section 4.8 provides the final remarks about the models' performance.

4.1 Proposal Framework

4.1.1 Approach

This dissertation proposes a learning-based framework to enable an assistive robot to recognize the object grasped by the human operator. As illustrated in Figure 4.1, the proposed framework combines the strengths of MediaPipe in detecting hand landmarks in a RGB image with a deep multi-class classifier that predicts the object based on the configuration of the user's hand after grasping it. Accordingly, the developed object recognition system operates based on different principles, including the sensing device, the tracking method, and the machine learning approaches. From the point of view of the application in industrial settings, the proposed system has two strengths when compared to the use of data-gloves or electromagnetic motion capture systems. First, the simplicity of installation is associated with a much less complex and costly setup. Second, the non-intrusiveness of the required setup is a valuable factor in accelerating the acceptance of these technologies by humans in carrying out collaborative tasks (a process also referred to as "user adoption"). In contrast, vision-based hand tracking is affected by occlusions, changes in light conditions, and cluttered backgrounds.

Furthermore, these problems are difficult to overcome with deep-learning techniques given the data dependency and generalization problems against hands, objects, and lighting conditions outside the training sets. Overall, this paper contributes to advances in understanding the opportunities and limitations of using this novel approach for the recognition of human-grasped objects.

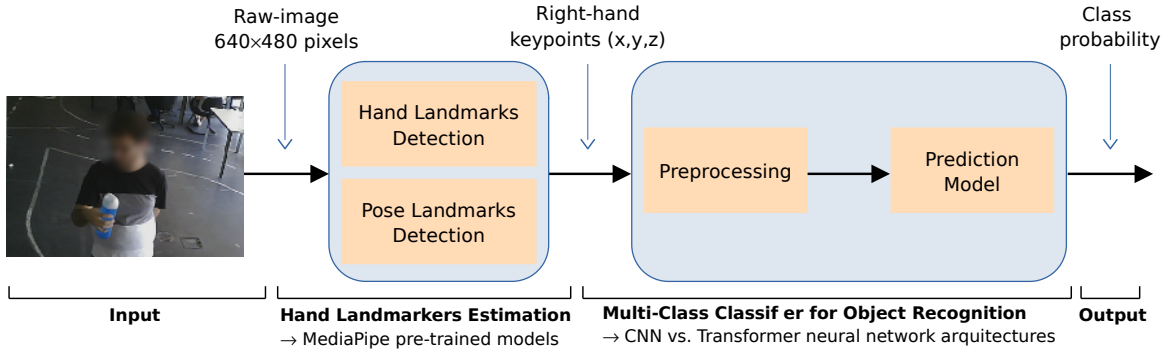


Figure 4.1: The proposed learning-based framework for object recognition based on the hand keypoints.

The output of the pre-trained model provides the (x, y, z) coordinates of landmarks for each detected hand. The (x, y) coordinates represent the horizontal and vertical positions of the landmark on the image plane, while the z-coordinate represents an estimate of the relative depth with respect to the wrist reference [93]. This work focuses on tracking the right hand by combining the Hand Landmark detection and the Pose Landmark Detection pre-trained models. This strategy proved to be useful to enhance the reliability of the process of extracting the coordinates of the right-hand keypoints from each frame.

The multi-class classifier for object recognition faces several challenges. First, there is limited information about the three-dimensional configuration of the hand, namely if the hand configurations involve overlapping fingers or positions close to each other in the image plane. Consequently, the z-coordinate (relative depth) revealed to be a critical element for discriminating complex hand configurations. Second, the coordinates provided by MediaPipe can vary in scale and rotation depending on the hand's distance from the camera and the hand's orientation in the image, adding complexity to the task. For these reasons, a deep learning model able to learn complex features directly from the MediaPipe coordinates will be explored and evaluated with a view to its generalization ability in different scenarios and for various users.

The learning problem involves a mapping between two spaces $f(X, \theta) : X \rightarrow Y$, where $X \in \mathbb{R}^{3\times21}$ is the set of possible spatial coordinates of the hand keypoints, $Y \in \mathbb{R}^{M}$ is the set of possible M output classes, and θ the model parameters. Let $D = \{(x_1, y_1), \cdots, (x_n, y_n)\}$ be a training dataset of n examples where $x_i \in X$ is an input and $y_i \in Y$ is the corresponding ground truth class label. Given a new instance x_{new}, the task is to predict its corresponding object class y_{pred}, such that:

$$y_{pred} = f(x_{new}). \tag{4.1}$$

The model aims to minimize a chosen loss function L that quantifies the dissimilarity between the predicted class y_{pred} and the ground truth class y_i. Formally, the training process seeks to find the optimal parameters $\hat{\theta}$ of the mapping function f by solving the following optimization problem:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L\left(f_\theta(x_i), y_i\right), \tag{4.2}$$

where $f_\theta(x_i)$ is the predicted class label for sample x_i using the model with parameters θ, and L is a suitable loss function. Upon successful training, the model f can be used for predicting the object class y_{pred} for new instances of hand keypoints.

4.1.2 Evaluation Metrics

To properly ascertain the performance and generalization capability of deep learning models, metrics must be employed. Accuracy is one of the most widely used metrics in the realm of deep learning, representing the proportion of correctly classified examples among all samples in the dataset, calculated as follows:

$$ACC = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}\ . \tag{4.3}$$

Although accuracy clearly assesses the model's performance, it may not be recommended in some situations, like when working with an imbalanced dataset. Therefore other metrics should be used, such as:

- True Positive (TP) - instances correctly classified as positive;
- True Negative (TN) - instances correctly classified as negative;
- False Positive (FP) - instances incorrectly classified as positive;
- False Negative (FN) - instances incorrectly classified as negative;
- Positive Predictive Value (PPV) or Precision, which is the percentage of instances correctly classified as positive relative to all instances classified as positive:

$$PPV = \frac{TP}{TP + FP}\ ; \tag{4.4}$$

- True Positive Rate (TPR) or Recall, which is the percentage of positive instances that are classified as positive:

$$TPR = \frac{TP}{TP + FN}\ ; \tag{4.5}$$

- F1-Score, which is the harmonic mean between the precision and the recall:

$$TPR = 2\frac{PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}\ . \tag{4.6}$$

Precision is a relevant metric to use when the aim is to minimize false positives, while Recall is relevant to minimize false negatives. F1-Score includes the two of them, with both false positives and false negatives influencing the result.

In a multi-class classification problem, such as the one in this work, these metrics are obtained separately for each class and then averaged across all classes to obtain a final metric.

In addition to the previous metrics, a Confusion Matrix can be used to visually represent the model's performance in a tabular way. Each entry i, j contains the number of instances from the class i that are classified as belonging to the class j (e.g., Figure 4.2a). A confusion matrix can also undergo normalization by dividing each entry by the sum of its row so that each entry provides ratios instead (e.g., Figure 4.2b).

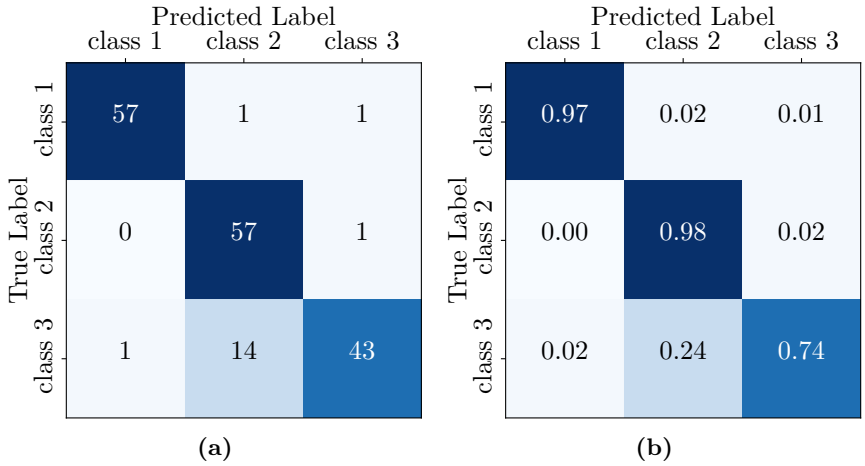


Figure 4.2: Confusion matrices examples: (a) non-normalized and (b) normalized.

4.2 Data Representation

The four objects selected for this study are all "graspable", i.e., more or less rigid. They include a cylindrical water bottle, a Rubik's cube, a flat and thick smartphone, and a small and sharp screwdriver (Figure 4.3). Given the differences in shape, size, and/or weight, the goal is to discriminate these four objects based on the configuration adopted by the hand while interacting with them. This section will start by validating the MediaPipe framework given the fact that it is an external tool and then it will detail the dataset acquisition, the data pre-processing, and the data splitting.

Figure 4.3: The objects used in the study include a water bottle, a Rubik's cube, a smartphone, and a screwdriver.

4.2.1 MediaPipe Suitability Validation

The MediaPipe software was used to extract the required landmarks automatically from the video input. A recent study by Amprimo *et al.* [93] evaluated the performance of the basic MediaPipe Hand [86] and an enhanced solution using depth data [94] against a motion capture system using an Optitrack solution comprising six Prime13 cameras. The focus was on the usage of such hand-tracking frameworks in clinical applications, such as automatic motion quality assessment, as well as the influence on the tracking quality of factors such as distance from the camera, camera viewing angle, and velocity of the motion. The results show that the use of the hands model based on an RGB input provides a good level of trust in terms of tracking accuracy.

In the same line of thought, the question arose of evaluating the reliability of the hand tracking software in situations where the hand grasps an object, taking into account the model's level of confidence [86] in the localization of the hand landmarks. The MediaPipe Hands Model can detect a variable number of hands in an image, and it was set up so that it would only detect up to two hands, meaning that its output can consist of no hands detected, of a set of 21 points hand detected, which can be either the left or the right hand, or two sets of 21 points if both hands were detected. Thus, before acquiring the dataset, the success of the MediaPipe framework in identifying the hand keypoints was evaluated in two cases.

Firstly, it is important to understand if it can consistently detect the right hand in different situations. To achieve this, four scenarios were considered: hand open, hand closed, hand holding bottle, and hand holding phone with all of them being recorded both still and making similar movements. First, the hand remained stationary during the acquisition of 250 frames, considering both the hand without interacting with an object and the hand grasping each of the objects selected in the study. The results in Table 4.1 show that MediaPipe achieves a success rate of nearly 100 % when the hand is not interacting with an object and fluctuates between 93.2 % and 99.2 % when it grasps an object. Second, the hand performed random movements, resulting in lower success rates: approximately 98.4 % without an object and values between 83.2 % and 84.8 %, depending on the object. These success rates can be considered acceptable since once in operation the classifier will tend to consider several frames, and not just one, to make a more reliable decision. Despite this, in Table 4.1 which shows the longest sequence of consecutive frames without generating keypoints, more complex occlusion situations can be observed in which MediaPipe did not return valid coordinates for a 4-second interval. This may lead to rethinking the best camera location (e.g., environment- versus robot-mounted camera) and, eventually, the number of cameras to use.

Table 4.1: Percentage of frames with detected right-hand keypoints

	Hand Open	Hand Closed	Holding Bottle	Holding Phone
No Movement	100 %	99.6 %	99.2 %	93.2 %
In Movement	93.2 %	98.4 %	83.2 %	84.8 %

Table 4.2: Longest sequence of empty frames

	Hand Open	Hand Closed	Holding Bottle	Holding Phone
No Movement	-	1	2	17
In Movement	17	4	42	38

Given that the scope of this work is restricted to scenarios where the user grasps the objects with his right hand, the detected hands in each instance need to be filtered according to their handedness to avoid wrong predictions. This information can be given directly by the hands landmarker and by using the pose landmarker. The latter option is more reliable but it also increases the amount of processing. To decide wether the pose landmarker should be incorporated into the data processing or the hand landmarker is reliable enough, the results in Table 4.3, consisting of the percentage of frames both models agree on, were obtained with videos of a user in different scenarios. These results show that for most frames both models agree about the handedness of the detected hands, but when the right hand is grasping objects that require less common hand geometries, especially the screwdriver, the hands model is only able to correctly classify the handedness 55.1 % of the frames. To avoid this source of error, the images are also processed by the more reliable pose landmarker.

Table 4.3: MediaPipe hand and pose model concordance percentage in different scenarios

	Hand Visible	Holding Bottle	Holding Cube	Holding Phone	Holding Screwdriver	Hand Obstructed
Left Hand	92.7 %	91.9 %	71.5 %	97.3 %	57.1 %	88.7 %
Right Hand	95.5 %	88.7 %	99.2 %	92.0 %	96.4 %	98.7 %
All Hands	97.2 %	81.5 %	71.1 %	89.3 %	55.1 %	87.4 %

4.2.2 Dataset Acquisition

The first step to train a supervised machine learning model is to find a dataset. However, given that this problem is particular, the dataset had to be manually collected. For this purpose, a dataset was collected consisting of videos where one person would move and rotate a particular object (example frames in Figure 4.4). This acquisition involved the participation of three right-handed (male) volunteers aged between 23 and 26 years old. Participants were asked to naturally grab and hold an object placed on a table, followed by executing small movements of the hand in free space. These movements were performed while introducing random variations in the hand's orientation relative to the RGB camera to ensure diversity in the points of view from which the hand-object interaction is observed.

Naturally, the successive frames could lead to similar grasping patterns from different views. To investigate intra-user variability and to ensure robust model training, users are instructed to perform multiple grasping trials of the selected object across four distinct acquisition sessions. Bearing this in mind, the data acquisition system was designed to facilitate the fast generation of training datasets, accommodating the inclusion of new users and/or additional acquisition sessions. On the one hand, the system is integrated into the workflow of the

Figure 4.4: Dataset examples holding a bottle (left) and a phone (right).

proposed object recognition framework. On the other hand, it is particularly well-suited for implementation in industrial settings where end-users may not possess extensive expertise in machine learning or computer vision. The instructions provided to users during the data acquisition sessions were intentionally straightforward, ensuring that non-experts could readily participate in the process.

Videos over four sessions per user were recorded at 10 frames per second. For each object and each user, four data acquisition sessions were carried out, which gave rise to the dataset used in the study. Therefore, the dataset consists of a total of 11 054 samples, distributed practically equally across the three participants (around 3600 samples per participant) and the four objects (between 2618 and 2849 samples per object). The exact number of samples of the entire dataset per class and per user is shown in Table 4.4.

Table 4.4: Number of samples in the dataset per class and user

Dataset	Bottle	Cube	Phone	Screwdriver	Total
User1	828	928	950	957	3663
User2	886	926	939	946	3697
User3	904	907	937	946	3694
Total	2618	2761	2826	2849	11 054

4.2.3 Preprocessing

After having a dataset, the data had to be processed to have a fitting structure to be used in the model training. The images from the videos were processed using the Mediapipe hands model resulting in 21 points for each hand detected (Figure 4.5).

The points corresponding to the right hand are then subject to further transformations and normalization. First, the original coordinates of the keypoints (raw data), which are already normalized within the range of 0 to 1 are converted into coordinates relative to a reference. Specifically, for each keypoint $P = (x, y, z)$, the coordinates of the reference point are subtracted $P_{ref} = (x_{ref}, y_{ref}, z_{ref})$ from them to obtain relative coordinates $P_{rel} = (x_{rel}, y_{rel}, z_{rel})$. In this study, the reference is defined as the centroid C of the set of hand

Figure 4.5: Points detected on the pictures in Figure 4.4 by Mediapipe Hands Model.

keypoints This transformation into relative coordinates is particularly useful because the absolute position of the hands in the image may vary from frame to frame due to different distances from the camera or hand orientations. Instead, relative coordinates are translation invariant and they reduce the influence of any rotations that might be present in the raw data. Therefore, the network will focus on the spatial relationships between keypoints, rather than their absolute positions, making it less sensitive to hand orientations and scale variations.

After obtaining the relative coordinates with respect to the reference point, scaling is applied to each dimension independently by dividing by an appropriate constant to ensure that the hand's representation spans the entire range, as follows:

$$scaleFactor = \frac{0.5}{\max(\{|x_i|, |y_i|, |z_i|\} : i = 1, \cdots, n)} \ , \tag{4.7}$$

where $\{x_i, y_i, z_i\}$ denote relative coordinates. This feature scaling revealed to be a valuable pre-processing step to help make the data more consistent, helping the model to learn the relevant patterns without being influenced by variations in hand position, hand size, or scale. Further, it helps to maximize the separation among keypoints, helping the model to discriminate the output class. Finally, a uniform adjustment is made by adding 0.5 to each coordinate, centering the points between 0 and 1 on the scale. It is important to note that throughout the point processing, the order of the points is never changed, and therefore the models can take advantage of this structure. Figure 4.6 shows examples of the normalized keypoints representation expressed according to the previous steps, that is:

$$P_{norm} = (P - C) \times scaleFactor + 0.5 \ . \tag{4.8}$$

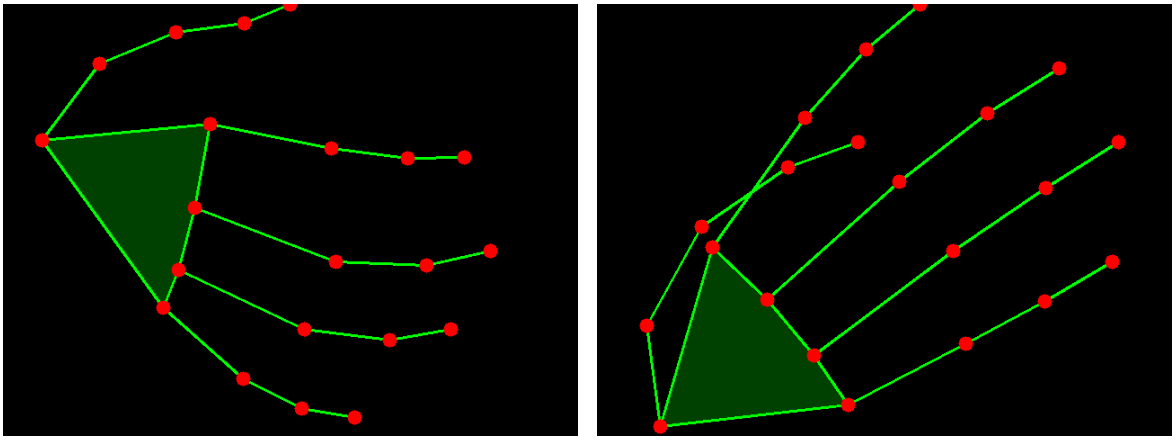


Figure 4.6: Points from the pictures in Figure 4.5 after normalization.

4.2.4 Train-Validation-Test Split

In order to train the model, the entire dataset is initially split into training and test sets using an indicative 80-20 ratio (the exact ratio can vary slightly depending on the experiment). Subsequently, the training set is further divided into a training subset and a validation subset using the same ratio. Additionally, early stopping was set up so that the model would stop training after 200 epochs without a better validation loss and restore the best weights. During hyperparameter optimization, the different combinations are tested using 4-fold cross-validation, which means that the model is trained four times, and therefore every sample of data in the initial training and validation set was used both for training and validation.

4.3 K-Means Clustering

Before describing the methodology adopted for recognizing grasping patterns, a statistical analysis was conducted on the entire dataset obtained from MediaPipe using K-means clustering. Initially, this algorithm was applied to learn the cluster structure within the training data. Subsequently, the model's performance was assessed using a separate and previously unseen test dataset.

The analysis results are visualized in a 4×4 matrix, denoted as the "True Labels" vs. "Assigned Cluster" matrix (see Figure 4.7). This matrix provides a comprehensive view of the K-means clustering method's performance in object recognition and it enables the computation of the proportion of objects assigned to each cluster. This cluster analysis provides valuable insights into the relationships between hand poses and class labels. First, the analysis of the relationship between clusters and class labels in hand pose data reveals that certain clusters, like clusters 0 and 3, exhibit a diverse mix of all classes. Second, in the test data, cluster 0 encompasses a substantial 39.8 % of the samples. Third, and of utmost importance, the K-means clustering results indicate that the clusters do not closely align with the class labels, which may indeed present challenges for the chosen classifier.

Figure 4.7: The distribution of test dataset samples from each class within each cluster.

The complexity of the problem is increased by user dependency and the variability observed within the same subject during different trials of grasping the same object. Given these challenges, a supervised learning approach, such as a CNN, emerges as a suitable choice, as it can autonomously learn hierarchical features and patterns directly from the data, irrespective of the initial cluster structure. Deep learning models, in particular, excel in addressing the intricacies of hand posture recognition and demonstrating robust generalization across different users, effectively capturing intra-user variations, provided that the training dataset exhibits diversity and represents the target scenarios.

4.4 Convolutional Neural Network Classifier

Convolutional Neural Networks (CNNs) excel at detecting local relations, which makes them an advantageous solution for this problem given that each sample provided to the model is made of the 21 3D points that always follow the same structure representing the right hand. Although the input data is bi-dimensional (21×3) resulting in a 2D kernel, a one-dimensional convolutional neural network was used so that the kernel only moves in one direction including all the point coordinates.

4.4.1 Model Selection

The backbone of the developed CNN comprises a total of three convolutional layers each with 64 feature maps and ReLU activation functions. The first layer uses a kernel size of 3×3 pixels performing a 1D convolution on the 3×21 data with a stride of 1 pixel. The flattened output from the final layer is connected to a dense layer with 128 neurons, followed by another dense layer with the number of neurons equal to the number of classes. The output layer consists of the final connect layer with softmax activation. The softmax function takes a vector of real-valued scores (often called logits) and transforms them into a probability distribution

over multiple classes. For the classification task with 4 classes, the output layer has 4 neurons, each representing the probability of the input belonging to a particular class. To prevent overfitting, dropout layers are incorporated after each fully connected layer.

To optimize the described architecture, three common hyperparameters in CNNs were optimized to obtain even better results. These were the initial learning rate for the model training, the kernel size used by the convolutional layers, and the dropout rate. The number of convolutional layers was also tested alongside them, given that, according to manual testing, it also affected the model results without changing the number of trainable parameters. Table 4.5 shows the values tested for each hyperparameter.

Table 4.5: Tested hyperparameter values (CNN model)

Initial Learning Rate	0.01, 0.001, 0.0001
Kernel Size	2, 3
Dropout Rate	0.0, 0.1, 0.2, 0.3, 0.4, 0.5
Number of Convolutional Layers	1, 2, 3

In order to choose the best combination of these hyperparameters, all combinations were tested using 4-fold cross-validation, and the average result of each combination was obtained. The combination in Table 4.6 resulted in the lowest average loss and an average accuracy of 94.20 %.

Table 4.6: Best hyperparameters (CNN model)

Initial Learning Rate	0.001
Kernel Size	3
Dropout Rate	0.5
Number of Convolutional Layers	2

The final model can be seen in Figure 4.8 and it has 156 644 trainable parameters. It is made of two convolutional layers followed by three dense layers, with the third being the output layer. Between the convolutional and the dense layers and between both dense layers, there is also a dropout layer to help with overfitting.

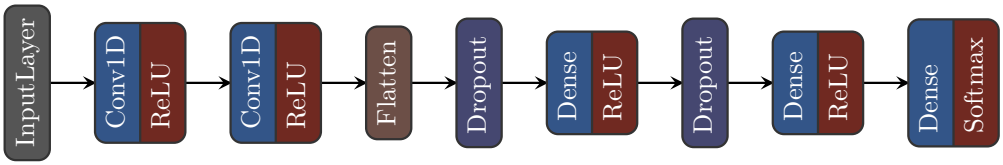


Figure 4.8: CNN model architecture.

4.4.2 Performance Evaluation

The final CNN architecture was trained resulting in the learning curves in Figure 4.9 and Figure 4.10. According to the figures, the best validation loss occurred around the 375th epoch, with the training stopping 200 epochs later.

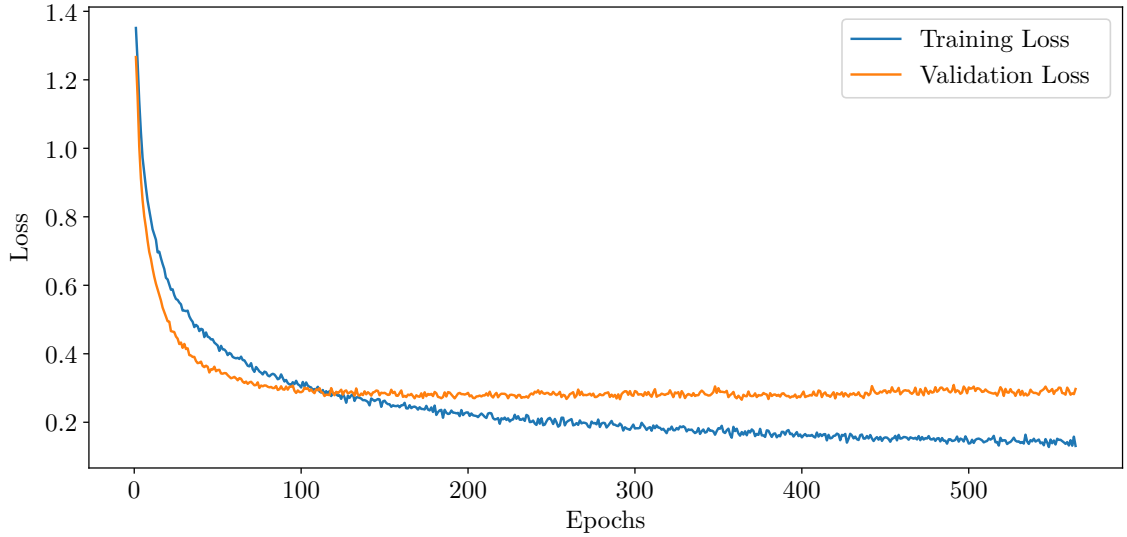


Figure 4.9: Training and validation loss evolution during the CNN's training.

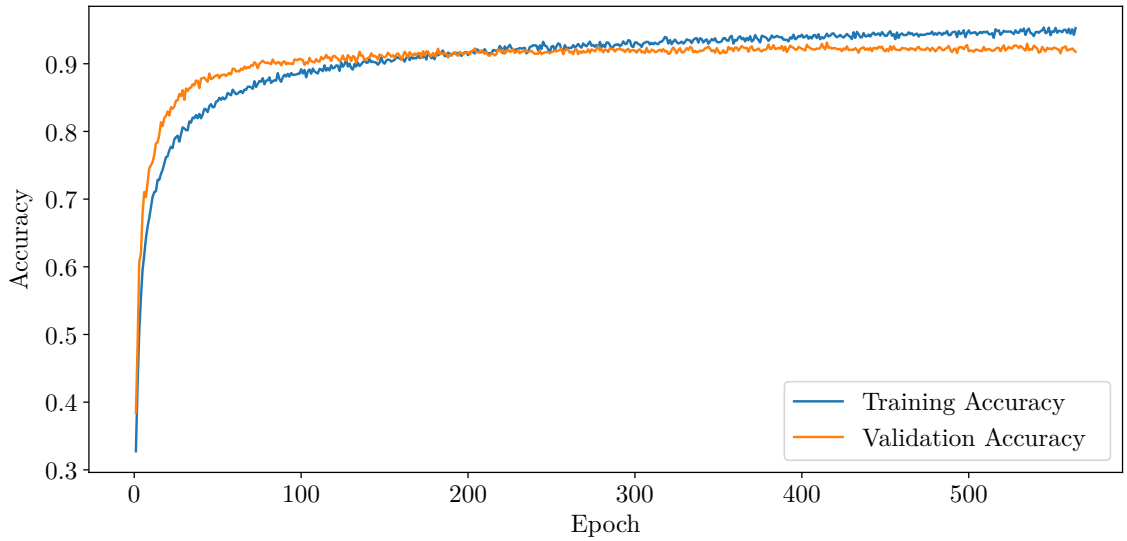


Figure 4.10: Training and validation accuracy evolution during the CNN's training.

After the model finished training, the metrics in the Table 4.7 were obtained. Given that the test accuracy was over 92 %, it can be concluded that the model managed to generalize its knowledge from the training data to classify data it has never seen before. Additionally, the confusion matrix in Figure 4.11 shows that the screwdriver was the object the model managed to predict more accurately. This can be because the hand geometry that allows a person to intuitively grasp a screwdriver is more restricted.

Table 4.7: CNN metrics

Accuracy	Precision	Recall	F1-Score
0.9240	0.9242	0.9240	0.9240

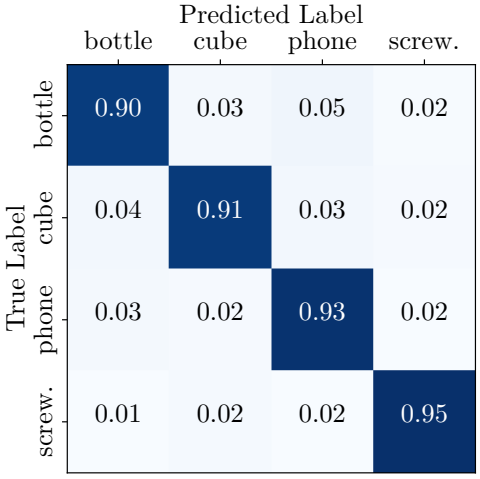


Figure 4.11: CNN confusion matrix.

4.5 Transformer Neural Network Classifier

As said in subsection 3.2.3, Transformer Neural Networks shine at capturing long-range dependencies and relationships. Therefore, because the points obtained from Mediapipe have a specific order, this ability can be used to process structured data and effectively capture dependencies and patterns.

4.5.1 Model Selection

For this work, a Transformer architecture adapted from an example in the Keras documentation[1] was tested and manually optimized. The resulting architecture can be seen in Figure 4.13 and it has 16 384 trainable parameters. It is made of two Transformer encoder stacks (Figure 4.12) comprised of the following layers: multi-head self-attention, layer normalization, and feedforward neural networks. Within each encoder, multi-head self-attention is applied to capture dependencies among the keypoints, where four attention heads are used for enhanced feature extraction. Following self-attention, two position-wise feedforward neural networks are employed to process the attended features and capture complex patterns. Layer normalization is applied after each sub-layer to stabilize the activations and facilitate training convergence.

To further optimize the previous architecture, three hyperparameters were optimized to improve the performance of the model. These were the initial learning rate for the model training, the dropout rate inside each transformer block, and the dropout rate of the MLP at the end of the model. Table 4.8 shows the values tested for each hyperparameter.

Table 4.8: Tested hyperparameter values (Transformer model)

Initial Learning Rate	0.01, 0.001, 0.0001
Dropout Rate	0.0, 0.1, 0.2, 0.3, 0.4, 0.5
MLP Dropout Rate	0.0, 0.1, 0.2, 0.3, 0.4, 0.5

[1]Transformer Keras Example: `https://keras.io/examples/timeseries/timeseries_transformer_classification`

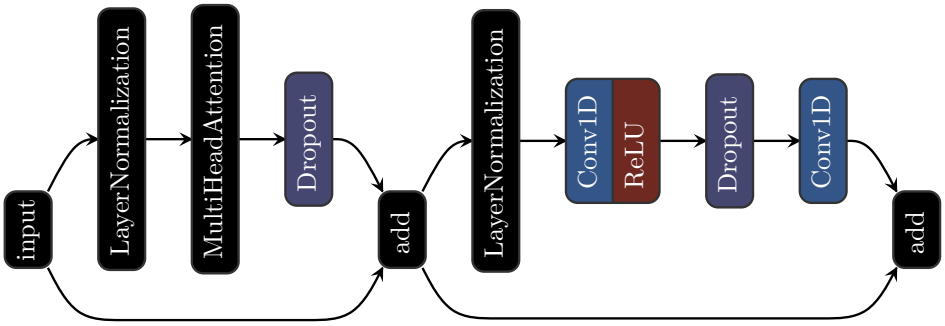


Figure 4.12: Transformer encoder block.

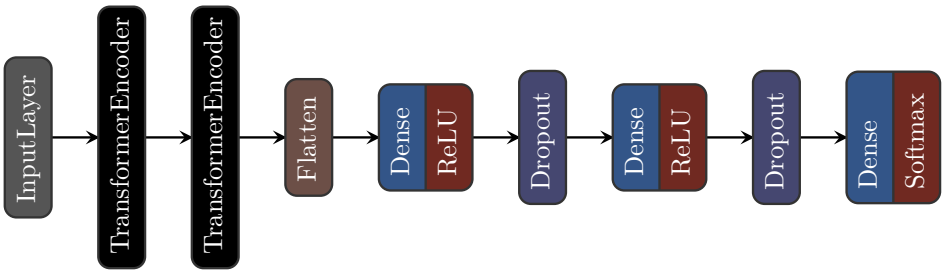


Figure 4.13: Transformer model architecture.

In order to choose the best combination of these hyperparameters, all combinations were tested using 4-fold cross-validation, and the combination with the smallest average validation loss was chosen. This combination can be seen in Table 4.9 having an average accuracy of 92.24 %.

Table 4.9: Best hyperparameters (Transformer model)

Learning Rate	0.0001
Dropout Rate	0.5
MLP Dropout Rate	0.1

4.5.2 Performance Evaluation

The final architecture was trained resulting in the learning curves in Figure 4.14 and Figure 4.15. According to the figures, the best validation loss occurred around the 3500th epoch, with the training stopping 200 epochs later.

With the model training phase finished, the metrics on Table 4.10 were obtained using the test set. Considering that the accuracy surpassed 91 %, one can conclude that the model successfully generalized the knowledge from the training data to classify previously unseen data. Additionally, the results from the obtained confusion matrix (Figure 4.16) were similar to the CNN with the screwdriver being the object predicted more accurately.

Table 4.10: Transformer metrics

Accuracy	Precision	Recall	F1-Score
0.9109	0.9115	0.9109	0.9109

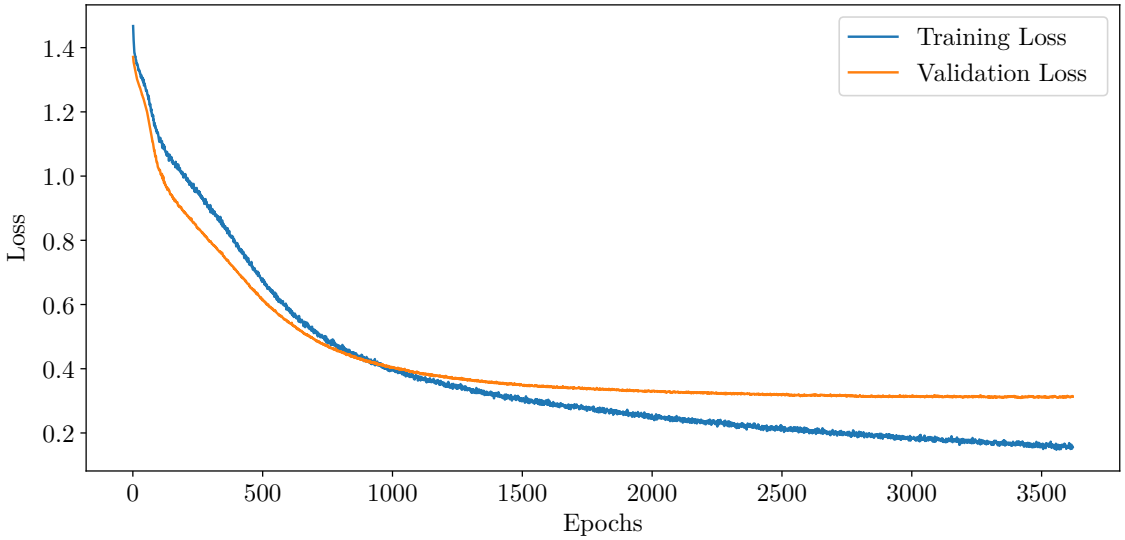


Figure 4.14: Training and validation loss evolution during the Transformer's training.

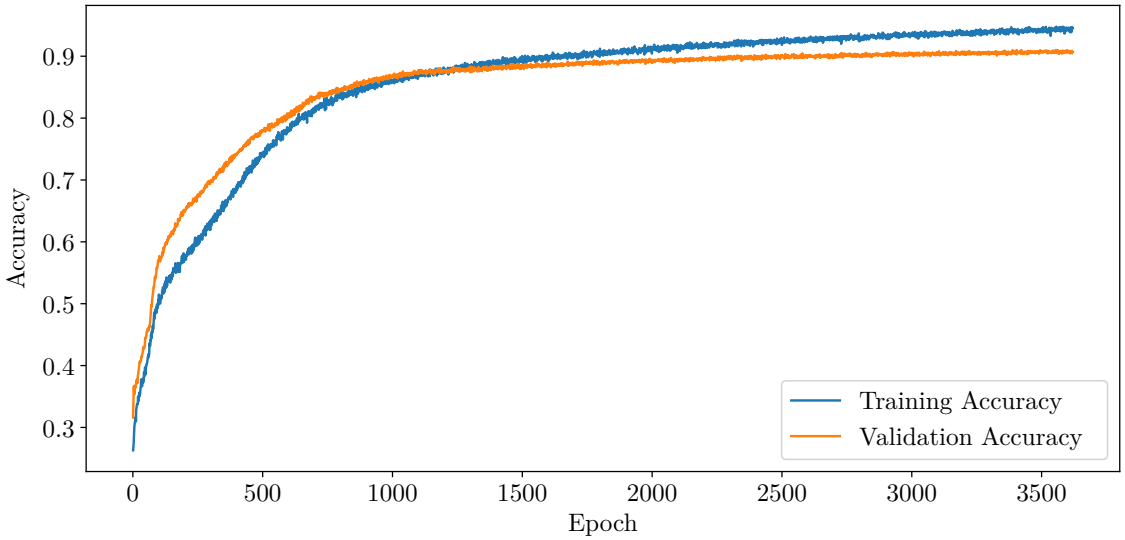


Figure 4.15: Training and validation accuracy evolution during the Transformer's training.

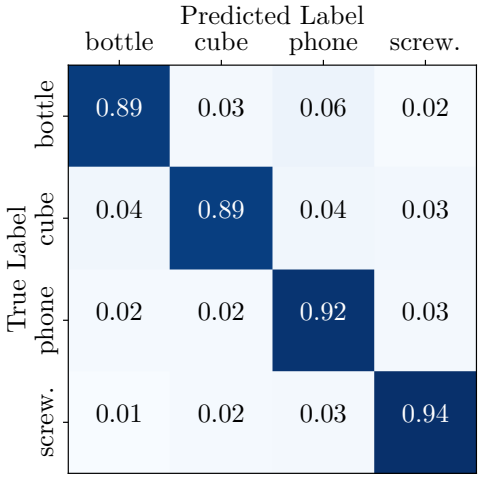


Figure 4.16: Transformer confusion matrix.

4.6 Comparative Analysis of Deep Models Generalization Ability

This section provides a comparative analysis of the performance of a CNN model against a transformer model for the recognition of the human-grasped object from the hand keypoints. Three main experiments were designed to evaluate the classification performance of the two architectures under comparison when they were fed with the same data. With the three experiments, the intent was to study the generalization capacity of the deep model in unseen data, taking into account the different users and the different data acquisition sessions.

4.6.1 Experiments and Metrics

In the context of this research on developing a hand-object recognition classifier utilizing keypoints provided by the MediaPipe Hands Model, a series of experiments was conducted to evaluate the classifier's performance comprehensively, as follows:

- **Experiment 1: Session-Based Test**: the first experiment aims to assess the impact of session-based testing on the classifier's performance. For that purpose, the classifier will be trained on data from all users and all acquisition sessions except one that will be used for testing.
- **Experiment 2: User-Specific Test**: in the pursuit of refining the hand-object recognition classifier, a second experiment with a focus on individual user data was conducted. This experiment aims to provide insights into how the classifier performs when trained and tested on data collected from a single user, with the process repeated separately for each of the selected users.
- **Experiment 3: Leave-One-User-Out Test:**: the third experiment follows a distinctive approach termed the "Leave-One-User-Out Test". This experiment is designed to evaluate the classifier's performance when trained on data from two users and tested on data from the third user.

All experiments carried out take into account the intrinsic variability of performance estimation by conducting 50 Monte-Carlo simulations. Metrics such as accuracy, precision, recall, and F1-score were used to quantify the model's effectiveness in recognizing the grasped object. Also, confusion matrices help identify specific areas where the model may excel or struggle in the classification task.

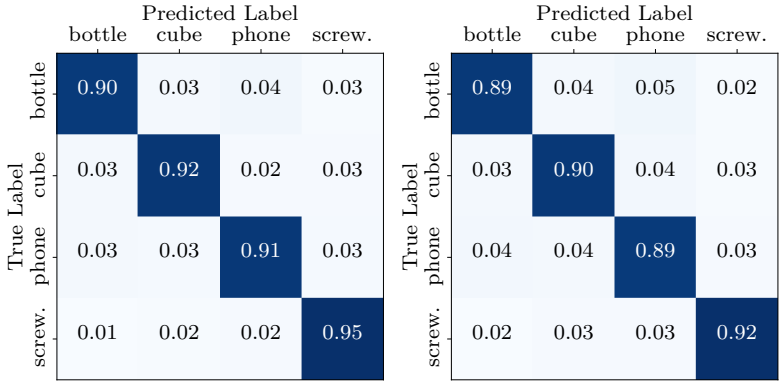
4.6.2 Session-Based Testing

In order to investigate the influence of session-based testing, the dataset was divided into two configurations. In the first configuration, referred to as the "Full Dataset" scenario, the entire dataset was employed consisting of 11 054 samples for both training and testing. First, the dataset was randomly split into training (6632 samples), validation (2211 samples), and testing (2211 samples) subsets. This setup aims to evaluate the classifier's performance when trained on a diverse set of hand-object interactions. Table 4.11 summarizes the results of evaluating the performance of the two models in the "Full Dataset" scenario in terms of accuracy, precision, recall, and F1-score. The CNN and Transformer show similar results with robust scores around 92 % and 90 %, respectively. The confusion matrices in Figure 4.17

show that the classifiers excel in distinguishing between the different classes, maintaining high accuracy. However, this outcome underscores the classifier's ability to generalize across a diverse range of hand-object interactions, as it was trained on a dataset encompassing multiple users and multiple data acquisition sessions.

Table 4.11: "Full Dataset" performance metrics

Model	Accuracy	Precision	Recall	F1-Score
CNN	**0.9210**	**0.9214**	**0.9211**	**0.9211**
Transformer	0.9017	0.9020	0.9016	0.9017



(a) **(b)**

Figure 4.17: "Full Dataset" confusion matrices: (a) CNN model, (b) Transformer model.

In the second configuration, one session was isolated from each user for testing (2731 samples), while the remaining sessions from all users were used for training (8289 samples). The aim was to determine how the inclusion of session-specific data impacts the classifier performance. The results depicted in Table 4.12 show the discrepancy in classifier accuracy between the "Full Dataset" and the "Session-Based Testing" scenarios. Further, the confusion matrix in Figure 4.18 compares the actual target with those predicted by the CNN model. The classifier is making accurate and correct predictions for the majority of classes, while struggling to recognize accurately examples belonging to specific classes, such as the "phone" in the third data acquisition session (the same with the Transformer).

Table 4.12: "Session-Based Testing" performance metrics where data from each session only appears in one set. For example, the "Session 1" column means that data from that session of all users is used in testing, while the remaining sessions are used for training.

Metric	Model	Session 1	Session 2	Session 3	Session 4
Accuracy	CNN	**0.8493**	**0.8138**	0.7844	**0.7718**
	Transformer	0.8458	0.8027	**0.7902**	0.7613
Precision	CNN	**0.8515**	**0.8160**	0.8024	**0.7723**
	Transformer	0.8469	0.8028	**0.8045**	0.7623
Recall	CNN	**0.8493**	**0.8138**	0.7844	**0.7718**
	Transformer	0.8458	0.8027	**0.7902**	0.7613
F1-Score	CNN	**0.8499**	**0.8136**	0.7878	**0.7717**
	Transformer	0.8461	0.8019	**0.7932**	0.7614

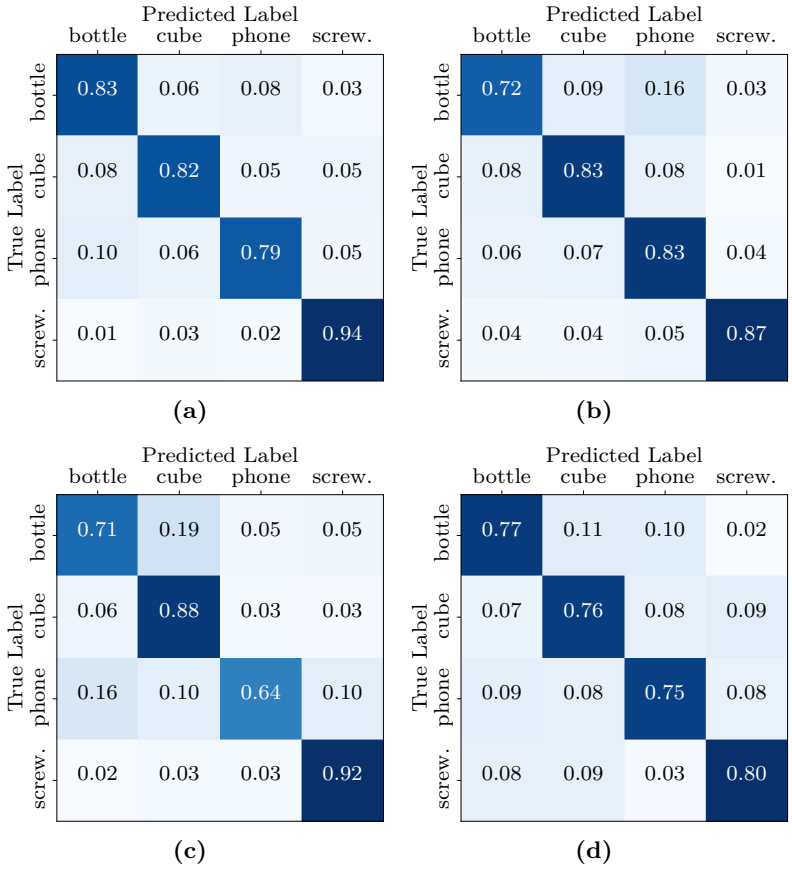


Figure 4.18: "Session-Based Testing" confusion matrices (CNN model): session 1 (a) to session 4 (d).

4.6.3 User-Specific Test

The experiment described in this subsection focuses on the variability in hand configurations and keypoint patterns for the same user over time (i.e., under different conditions imposed by the four data acquisition sessions carried out). Once again, two distinct dataset configurations were constructed for each user. In the first configuration ("Full User Dataset"), the entire

dataset for a single user was used, with the process repeated separately for all others. This simulates a scenario where the classifier is trained and tested on all available data for a single user based on a random split using an 80-20 ratio. The performance of the CNN and transformer models in the "Full User Dataset" scenario is summarized in Table 4.13. The CNN presents slightly better results than the transformer in different metrics, while User1's performance stands out. The confusion matrices show that the classifiers excel in distinguishing between the different classes with high accuracy (see Figure 4.19 relating to the CNN model).

Table 4.13: "Full User Dataset" performance metrics.

Metric	Model	User1	User2	User3
Accuracy	CNN	**0.9674**	**0.9100**	**0.9163**
	Transformer	0.9423	0.8929	0.8730
Precision	CNN	**0.9678**	**0.9109**	**0.9171**
	Transformer	0.9435	0.8943	0.8745
Recall	CNN	**0.9674**	**0.9100**	**0.9163**
	Transformer	0.9423	0.8929	0.8730
F1-Score	CNN	**0.9675**	**0.9101**	**0.9164**
	Transformer	0.9423	0.8930	0.8729

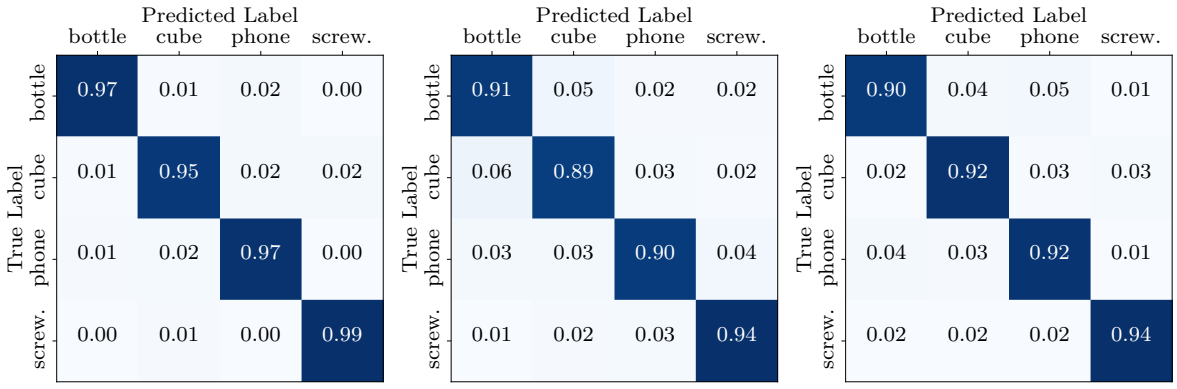


Figure 4.19: "Full User Dataset" confusion matrices (CNN model): (a) User1, (b) User2, and (c) User3.

In the second configuration, the data of one acquisition session for a specific user was isolated, dedicating it exclusively to the testing phase. Meanwhile, the remaining sessions' data from the same user were employed for training. This "Session-Based User Testing" setup mirrors scenarios where a classifier must adapt to recognize objects manipulated by a user based on a limited history of interactions. The results in Table 4.14, relating to User1, reveal interesting insights into the classifier's adaptability within the context of different user behaviors across multiple sessions. First, varying levels in all metrics across sessions with a range between 78.8 % and 92.6 % can be observed. The decrease in the evaluation metrics between the "Full User Dataset" and "Session-Based User Testing" scenarios highlights the

importance of user-specific adaptation. The confusion matrices (see Figure 4.20) also reveal lower performances in certain classes, but these vary from session to session. These results emphasize the importance of considering session-specific variations and user behaviors when training and evaluating the classifier.

Table 4.14: "Session-Based User1 Testing" performance metrics (each column indicates the specific session used in testing the model).

Metric	Model	Session 1	Session 2	Session 3	Session 4
Accuracy	CNN	**0.9257**	**0.8364**	**0.9053**	**0.7883**
	Transformer	0.9078	0.8171	0.8742	0.7636
Precision	CNN	**0.9288**	**0.8543**	**0.9135**	**0.7971**
	Transformer	0.9112	0.8401	0.8806	0.7846
Recall	CNN	**0.9257**	**0.8364**	**0.9053**	**0.7884**
	Transformer	0.9078	0.8172	0.8742	0.7636
F1-Score	CNN	**0.9261**	**0.8382**	**0.9073**	**0.7892**
	Transformer	0.9083	0.8196	0.8759	0.7671

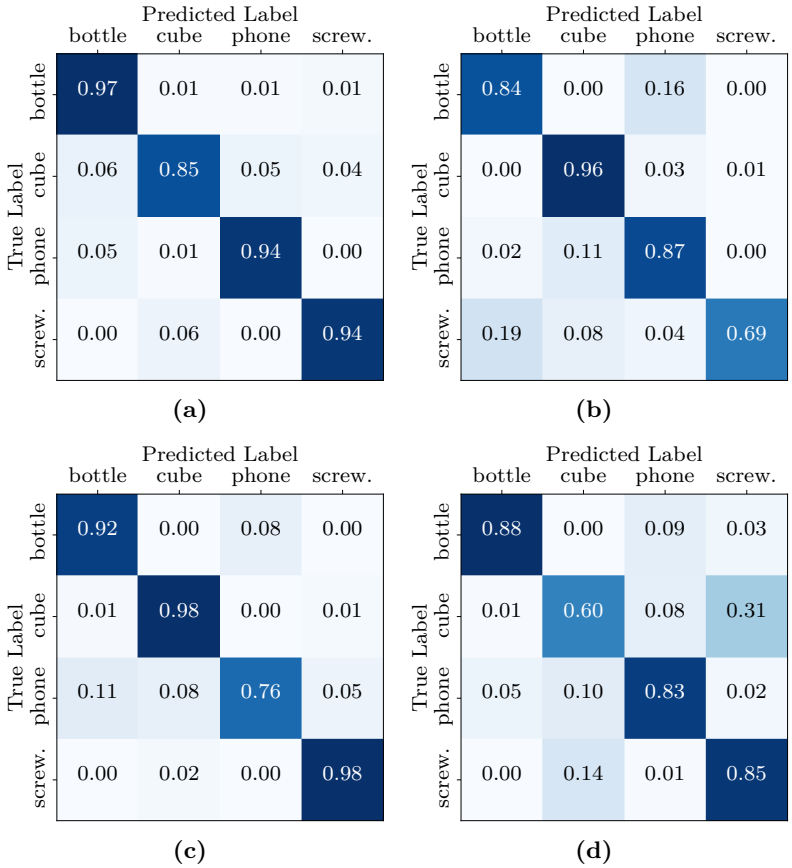


Figure 4.20: "Session-Based User1 Testing" confusion matrices (CNN model).

4.6.4 Leave-One-User-Out Test

The third experiment follows a distinctive approach termed the "Leave-One-User-Out Test". This is particularly relevant in scenarios where the data is collected from multiple users (i.e., dealing with user-dependent data), and the goal is to evaluate the model's generalization ability to new individuals who were not part of the training data. In this case, the process involves training the model on data from all users except one (the user to be left out) and then evaluating the model's performance on the data from the left-out user. This experiment allows the observation of a trend and it showcases the challenges of generalization to unseen users. In order to identify the trend, the process is repeated for each user, training the model on all users except the one being evaluated. This approach ensures that each user's data is used once as a test set (around 3660 samples), while the remaining data is employed for training (around 7350 samples).

Table 4.15 and Figure 4.21 shows the results obtained considering that only one user's data (all sessions) is used in testing the model. Although the classifier recognizes to some extent objects grasped by "User1" when it has not been explicitly trained on, the lower performance for "User2" and "User3" indicates limitations in generalization to users with distinct grasping patterns. The results indicate the difficulties the deep model faces when adapting to a new user in the absence of a dedicated training period. This emphasizes the need for personalized models or strategies that can adapt to individual user behaviors. In real-world scenarios, users may exhibit diverse hand-object interaction patterns and models should be capable of accommodating these variations. Whatever strategy is adopted, ensuring diverse and representative data (e.g., a bigger dataset) will be crucial for improving the results, either using a convolutional network as a transformer.

Table 4.15: "Leave-One-User-Out Test" performance metrics where data from each user only appears in the test set. For example, the "User1" column means that data from that user is used in testing, while the data from the remaining users is used for training."

Metric	Model	User1	User2	User3
Accuracy	CNN	0.7969	**0.5827**	**0.5488**
	Transformer	**0.8006**	0.5730	0.5350
Precision	CNN	**0.8123**	**0.5889**	**0.5675**
	Transformer	0.8094	0.5794	0.5492
Recall	CNN	0.7969	**0.5827**	**0.5488**
	Transformer	**0.8006**	0.5730	0.5350
F1-Score	CNN	0.8008	**0.5791**	**0.5483**
	Transformer	**0.8028**	0.5702	0.5321

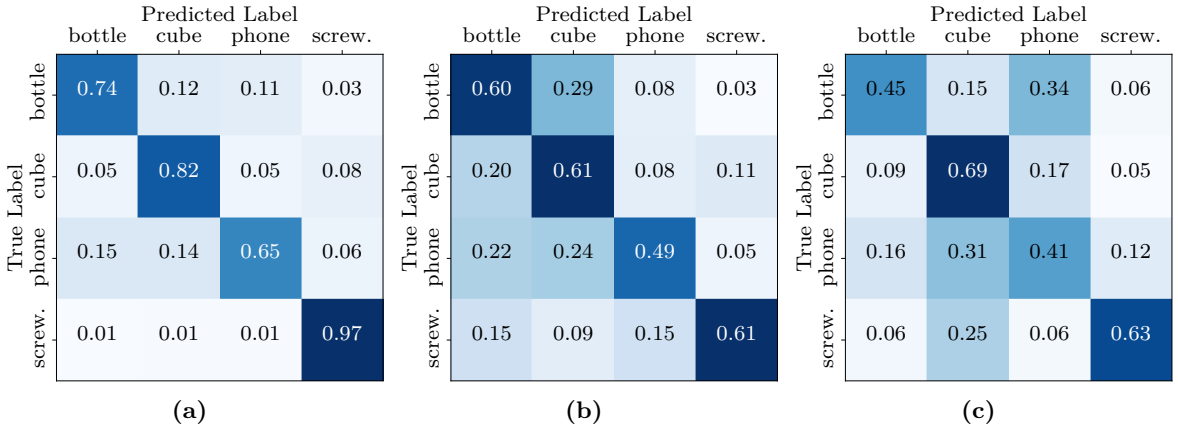


Figure 4.21: "Leave-One-User-Out Test" confusion matrices (CNN model).

4.7 Human Intention Prediction in Shared Tasks

After detailing all the processes related to the models' creation and validation, this section will cover how they can be integrated into a real-time application. In the proposed implementation, the MediaPipe Hands and Pose models have a corresponding ROS node that communicates with the others using actions. This communication method ensures that the models can run in parallel with each other and that the node that makes the requests does not need to wait for the results of the first model to make a request to the second as would be the case if services were used. Although communicating using publish/subscribe nodes would also respect the previous requirements, using actions also has the advantage of guaranteeing that the images analyzed by both models are the same. This process is managed by the Right Hand Keypoints Detection node which sends the images to the MediaPipe nodes, associates their outputs, and publishes only the right hand keypoints. The Points Normalization node is responsible for applying the remaining preprocessing steps and then publishing the normalized right-hand keypoints. Finally, the Model Prediction node contains a model trained in this work and predicts the grasped object. Figure 4.22 shows the developed implementation in ROS of data preprocessing and model prediction.

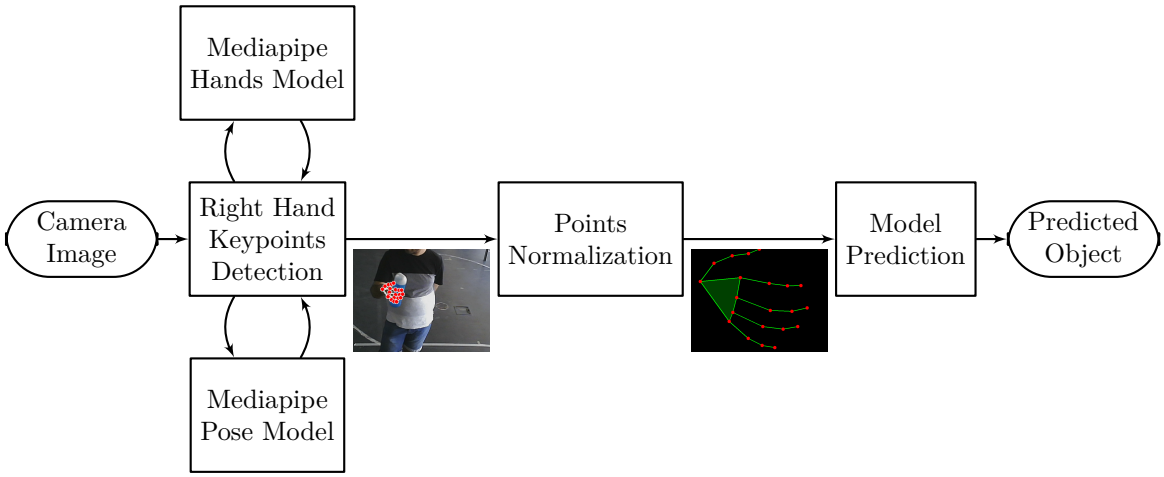


Figure 4.22: ROS nodes in the object recognition pipeline.

In a real-time implementation, additional processing could be made. For example, given that the system processes a continuous video stream, it can generate multiple predictions per second, which can be combined to make a final decision. This approach could give priority to the most frequent label to prevent a single incorrect prediction from negatively affecting the robot's actions. However, certain limitations would also need to be addressed. One example is how to deal with frames where the user is not holding an object. To solve this, the system could, for instance, have a threshold on the model's output probabilities to know if the hand is in fact holding an object. Another example is how to deal with occlusions given that the object can occlude the hand enough that MediaPipe stops detecting it. This could be solved using object recognition models that directly classify the object from the images or by adding more cameras. The information about which object is being grasped can then be used to make a decision. Although it was not implemented, a possible solution would be to add a new rule to the rule-based controller described in subsection 3.3.2.

4.8 Final Remarks

The experiments carried out provide valuable insights into the classifier's performance in different testing scenarios, especially for real-world applications. The results of the "User-Specific Test" show that the classifier's performance is influenced by the variability in interaction patterns across different sessions. This emphasizes the importance of considering session-specific variations when training and evaluating the classifier in user-centric applications (e.g., single-operator workstations). The lower accuracy in certain sessions and classes suggests the need for model refinements to deal with changes in data distribution between different work sessions. Pre-training a model on a large and diverse dataset and then fine-tuning it using data from a new session can be effective. Likewise, monitoring the model's performance in real-time and periodically retraining it with new and diverse data can help the model adapt to changing conditions and variations in different work sessions.

In dynamic settings where multiple users may interact with objects differently across multiple sessions, it is crucial for the classifier to adapt and maintain performance. The declining trend observed in the "Session-Based Testing" scenario suggests that the classifier may have difficulties in recognizing grasping patterns effectively when faced with new sessions that deviate from those in training. Once again, these results highlight the importance of considering session-specific variations when acquiring the dataset. From the perspective of model development in practical applications, the findings of this experiment emphasize the need for ongoing model refinements and adaptation strategies. Techniques such as session-specific fine-tuning or the incorporation of session-related features may prove valuable in enhancing the classifier's performance in real-world, dynamic environments.

The findings from the "Leave-One-User-Out Test" highlight the importance of personalized modeling approaches to account for user-specific patterns in the hand-object recognition system. In order to enhance the classifier's performance, it may be necessary to consider user-specific fine-tuning that can help the model better capture the nuances of new users. This personalized training can lead to better convergence and performance. This reflects the

importance of actively collecting data from new users in a systematic way to adapt the model. In line with this, the implementation of a system that allows for continuous model updating as new user data becomes available can be foreseen, i.e., the model can adapt and improve over time.

In the previous tests, the training times for both architectures were recorded, which are displayed in Table 4.16. The results consistently showed that the Transformer required approximately ten times more training time compared to the CNN. This is due to the fact that the CNN is a relatively simpler architecture even if the number of parameters is significantly higher in the CNN (156 644 compared to 16 384). This information in relevant in a real-time implementation where the training time can be a deciding factor.

Table 4.16: Average training times comparison

Test	CNN	Transformer
Full Dataset	126.85 s	1350.53 s
Session-Based	119.25 s	1323.65 s
Full User Dataset	41.73 s	625.90 s
Session-Based User1	42.96 s	517.36 s
Leave-One-User-Out	96.53 s	995.17 s

CHAPTER 5

Conclusion and Future Work

5.1 Discussion

This document studies the problem of anticipating human actions in collaborative environments with the goal of developing an anticipatory robot controller for an assembly task. Looking at previous work found in the literature, there is a clear predominance of perception using RGB cameras with different ways of preprocessing the captured images, particularly with libraries that can detect keypoints in an image such as skeleton joints which are very important to detect human poses. Then, supervised learning techniques are used to predict the action being made and associate that information with the following action.

The approach adopted in this work was to perform action anticipation by recognizing the object being grasped by the user from the configuration of the user's hand. To support it, an infrastructure in ROS was developed connecting a collaborative robot and two cameras to a robot controller. From the controllers tested, the rule-based provided the most flexibility by allowing to easily switch the rules used, which associate the state of the environment to a particular action.

The proposed DL-based framework for hand-object recognition relies on the MediaPipe Hands model to predict the hand keypoints and a multi-class classifier that uses them to predict the grasped object. The study focused on the classifier's generalization ability, remarking on the importance of an effective evaluation before the system is applied in real-world scenarios. Throughout the experiments, variations in performance were observed, particularly in scenarios involving session-based testing and user-specific adaptation. The main results emphasize the importance of continuous model monitoring, retraining, and active data collection to enable the classifier to generalize effectively across diverse user behaviors and grasping patterns. Personalized modeling approaches and fine-tuning strategies may be useful to address the challenges at hand. In this context, careful consideration of dataset dimensionality is essential to optimize model performance and facilitate meaningful insights from the data. The findings offer valuable insights into the factors influencing the performance of the classifier and the implications for real-world applications.

5.2 Future Work

This study presents a proof-of-concept about how to perform action anticipation from a different type of information and the associated limitations. This work sets the stage for ongoing improvements with the ultimate goal of delivering effective solutions for a wide range of applications. Moving forward, there are three main topics of research:

- **More Data**: test the current architectures with a bigger dataset which can have more people from different sex and/or age groups, can have more objects, can be from different keypoint detection frameworks, or can be from different camera angles.
- **Adaptability Strategies**: explore advanced techniques to further enhance the adaptability and generalization capabilities of the hand-object recognition system, namely by exploring data from human grasping databases like [95].
- **Real-Time Integration**: although not tested in a specific example, most of the infrastructure needed to proceed to real-time integration was implemented. However, a real-time application of this work could prove to be an interesting research topic when dealing with the limitations of the system described in Section 4.7. In particular, using the models in this work alongside an object recognition model that classifies directly from images could provide a consistent object detection since they complement each other.

5.3 Contributions

This work provided the following contributions:

- P. Amaral, F. Silva, V. Santos, «Recognition of Grasping Patterns using Deep Learning for Human-Robot Colaboration», Sensors (in press).
- Pedro Amaral, Recognition of Human Grasping Patterns for Intention Prediction in Collaborative Tasks, Github Repository, `https://github.com/pedromiglou/MRSI_Thesis_Action_Anticipation`.
- Pedro Amaral, Human Grasping Patterns for Object Recognition, Kaggle Dataset, `https://www.kaggle.com/datasets/pedromiglou/human-grasping-patterns-for-object-recognition`.

References

[1] R. Rosen, *Anticipatory Systems: Philosophical, Mathematical and Methodological Foundations*. Elsevier, 1985, pp. 339–347, ISBN: 9780080311586. DOI: 10.1016/C2009-0-07769-1.

[2] A. Louie, «Robert rosen's anticipatory systems», *Foresight*, vol. 12, R. Miller, Ed., pp. 18–29, 3 Jun. 2010, ISSN: 1463-6689. DOI: 10.1108/14636681011049848. [Online]. Available: https://www.emerald.com/insight/content/doi/10.1108/14636681011049848/full/html.

[3] R. Poli, «The many aspects of anticipation», *Foresight*, vol. 12, R. Miller, Ed., pp. 7–17, 3 Jun. 2010, ISSN: 1463-6689. DOI: 10.1108/14636681011049839. [Online]. Available: https://www.emerald.com/insight/content/doi/10.1108/14636681011049839/full/html.

[4] S. Robla-Gómez, V. M. Becerra, J. R. Llata, E. González-Sarabia, C. Torre-Ferrero, and J. Pérez-Oria, «Working together: A review on safe human-robot collaboration in industrial environments», *IEEE Access*, vol. 5, pp. 26 754–26 773, 2017. DOI: 10.1109/ACCESS.2017.2773127.

[5] V. Villani, F. Pini, F. Leali, and C. Secchi, «Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications», *Mechatronics*, vol. 55, pp. 248–266, Nov. 2018, ISSN: 09574158. DOI: 10.1016/j.mechatronics.2018.02.009. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0957415818300321.

[6] A. Ajoudani, A. M. Zanchettin, S. Ivaldi, A. Albu-Schäffer, K. Kosuge, and O. Khatib, «Progress and prospects of the human-robot collaboration», *Autonomous Robots*, vol. 42, Jun. 2018. DOI: 10.1007/s10514-017-9677-2.

[7] E. Matheson, R. Minto, E. G. G. Zampieri, M. Faccio, and G. Rosati, «Human-robot collaboration in manufacturing applications: A review», *Robotics*, vol. 8, p. 100, 2019.

[8] S. Kumar, C. Savur, and F. Sahin, *Survey of Human-Robot Collaboration in Industrial Settings: Awareness, Intelligence, and Compliance*, 2021. DOI: 10.1109/TSMC.2020.3041231.

[9] A. Castro, F. Silva, and V. Santos, «Trends of human-robot collaboration in industry contexts: Handover, learning, and metrics», *Sensors*, vol. 21, p. 4113, 12 Jun. 2021, ISSN: 1424-8220. DOI: 10.3390/s21124113. [Online]. Available: https://www.mdpi.com/1424-8220/21/12/4113.

[10] G. Michalos, N. Kousi, P. Karagiannis, *et al.*, «Seamless human robot collaborative assembly–an automotive case study», *Mechatronics*, vol. 55, pp. 194–211, 2018.

[11] S. Papanastasiou, N. Kousi, P. Karagiannis, *et al.*, «Towards seamless human robot collaboration: Integrating multimodal interaction», *The International Journal of Advanced Manufacturing Technology*, vol. 105, pp. 3881–3897, 2019.

[12] G. Hoffman, «Evaluating fluency in human–robot collaboration», *IEEE Transactions on Human-Machine Systems*, vol. 49, no. 3, pp. 209–218, 2019. DOI: 10.1109/THMS.2019.2904558.

[13] L. Rozo, H. Ben Amor, S. Calinon, A. Dragan, and D. Lee, «Special issue on learning for human–robot collaboration», *Autonomous Robots*, vol. 42, Apr. 2018. DOI: 10.1007/s10514-018-9756-z.

[14] J. Jiao, F. Zhou, N. Z. Gebraeel, and V. Duffy, «Towards augmenting cyber-physical-human collaborative cognition for human-automation interaction in complex manufacturing and operational environments», *International Journal of Production Research*, vol. 58, no. 16, pp. 5089–5111, 2020.

[15] G. Hoffman and C. Breazeal, «Cost-based anticipatory action selection for human–robot fluency», *IEEE Transactions on Robotics*, vol. 23, pp. 952–961, 5 Oct. 2007, ISSN: 1552-3098. DOI: 10.1109/TRO.2007.907483. [Online]. Available: https://ieeexplore.ieee.org/document/4339531/.

[16] A. M. Williams, «Perceiving the intentions of others: How do skilled performers make anticipation judgments?», *Progress in brain research*, vol. 174, pp. 73–83, 2009.

[17] C.-M. Huang and B. Mutlu, «Anticipatory robot control for efficient human-robot collaboration», in *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, IEEE, Mar. 2016, pp. 83–90, ISBN: 978-1-4673-8370-7. DOI: 10.1109/HRI.2016.7451737. [Online]. Available: http://ieeexplore.ieee.org/document/7451737/.

[18] N. F. Duarte, M. Raković, J. Tasevski, M. I. Coco, A. Billard, and J. Santos-Victor, «Action anticipation: Reading the intentions of humans and robots», *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4132–4139, 2018.

[19] C.-M. Huang, S. Andrist, A. Sauppé, and B. Mutlu, «Using gaze patterns to predict task intent in collaboration», *Frontiers in psychology*, vol. 6, p. 1049, 2015.

[20] O. C. Görür, B. Rosman, F. Sivrikaya, and S. Albayrak, «Social cobots: Anticipatory decision-making for collaborative robots incorporating unexpected human behaviors», in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, 2018, pp. 398–406.

[21] G. Gkioxari, R. Girshick, P. Dollár, and K. He, «Detecting and recognizing human-object interactions», in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8359–8367.

[22] A. A. Malik, «Robots and covid-19: Challenges in integrating robots for collaborative automation», unpublished, Jun. 2020.

[23] WiredWorkers, *Cobots*, Last accessed 3 January 2023. [Online]. Available: https://wiredworkers.io/cobot/.

[24] D. Mukherjee, K. Gupta, L. H. Chang, and H. Najjaran, «A survey of robot learning strategies for human-robot collaboration in industrial settings», *Robotics and Computer-Integrated Manufacturing*, vol. 73, p. 102 231, Feb. 2022, ISSN: 07365845. DOI: 10.1016/j.rcim.2021.102231. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0736584521001137.

[25] F. Semeraro, A. Griffiths, and A. Cangelosi, «Human–robot collaboration and machine learning: A systematic review of recent research», *Robotics and Computer-Integrated Manufacturing*, vol. 79, p. 102 432, Feb. 2023, ISSN: 07365845. DOI: 10.1016/j.rcim.2022.102432. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0736584522001156.

[26] C. Deans, «Biological prescience: The role of anticipation in organismal processes», *Frontiers in Physiology*, vol. 12, Dec. 2021, ISSN: 1664-042X. DOI: 10.3389/fphys.2021.672457. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fphys.2021.672457/full.

[27] V. Braitenberg, *Vehicles: Experiments in Synthetic Psychology*. The MIT Press, Feb. 1986, ISBN: 9780262521123.

[28] D. Carneiro, F. Silva, and P. Georgieva, «Robot anticipation learning system for ball catching», *Robotics*, vol. 10, p. 113, 4 Oct. 2021, ISSN: 2218-6581. DOI: 10.3390/robotics10040113. [Online]. Available: https://www.mdpi.com/2218-6581/10/4/113.

[29] Z. Wang, A. Boularias, K. Mülling, B. Schölkopf, and J. Peters, «Anticipatory action selection for human–robot table tennis», *Artificial Intelligence*, vol. 247, pp. 399–414, Jun. 2017, ISSN: 00043702. DOI: 10.1016/j.artint.2014.11.007. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0004370214001398.

[30] N. Sebanz, H. Bekkering, and G. Knoblich, «Joint action: Bodies and minds moving together», *Trends in Cognitive Sciences*, vol. 10, pp. 70–76, 2 Feb. 2006, ISSN: 13646613. DOI: 10.1016/j.tics.2005.12.009. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1364661305003566.

[31] C. Canuto, P. Moreno, J. Samatelo, R. Vassallo, and J. Santos-Victor, «Action anticipation for collaborative environments: The impact of contextual information and uncertainty-based prediction»,

Neurocomputing, vol. 444, pp. 301–318, Jul. 2021, ISSN: 09252312. DOI: 10.1016/j.neucom.2020.07.135. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0925231220317719.

[32] H. Gammulle, S. Denman, S. Sridharan, and C. Fookes, «Predicting the future: A jointly learnt model for action anticipation», in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2019, pp. 5561–5570, ISBN: 978-1-7281-4803-8. DOI: 10.1109/ICCV.2019.00566. [Online]. Available: https://ieeexplore.ieee.org/document/9009844/.

[33] Y. Wu, L. Zhu, X. Wang, Y. Yang, and F. Wu, «Learning to anticipate egocentric actions by imagination», *IEEE Transactions on Image Processing*, vol. 30, pp. 1143–1152, 2021, ISSN: 1057-7149. DOI: 10.1109/TIP.2020.3040521. [Online]. Available: https://ieeexplore.ieee.org/document/9280353/.

[34] C. Rodriguez, B. Fernando, and H. Li, «Action anticipation by predicting future dynamic images», in *Computer Vision – ECCV 2018 Workshops*, Springer, 2019, pp. 89–105, ISBN: 9783030110147. DOI: 10.1007/978-3-030-11015-4_10. [Online]. Available: http://link.springer.com/10.1007/978-3-030-11015-4_10.

[35] A. Furnari and G. M. Farinella, «Rolling-unrolling lstms for action anticipation from first-person video», *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, pp. 4021–4036, 11 Nov. 2021, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2020.2992889. [Online]. Available: https://ieeexplore.ieee.org/document/9088213/.

[36] G. J. Maeda, A. Maloo, M. Ewerton, R. Lioutikov, and J. Peters, «Anticipative interaction primitives for human-robot collaboration», in *AAAI Fall Symposium - Technical Report*, 2016, ISBN: 9781577357759.

[37] D. Moutinho, L. F. Rocha, C. M. Costa, L. F. Teixeira, and G. Veiga, «Deep learning-based human action recognition to leverage context awareness in collaborative assembly», *Robotics and Computer-Integrated Manufacturing*, vol. 80, p. 102 449, Apr. 2023, ISSN: 07365845. DOI: 10.1016/j.rcim.2022.102449. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0736584522001314.

[38] S. Tortora, S. Michieletto, F. Stival, and E. Menegatti, «Fast human motion prediction for human-robot collaboration with wearable interface», in *2019 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, IEEE, Nov. 2019, pp. 457–462, ISBN: 978-1-7281-3458-1. DOI: 10.1109/CIS-RAM47153.2019.9095779. [Online]. Available: https://ieeexplore.ieee.org/document/9095779/.

[39] P. Schydlo, M. Rakovic, L. Jamone, and J. Santos-Victor, «Anticipation in human-robot cooperation: A recurrent neural network approach for multiple action sequences prediction», IEEE, May 2018, pp. 1–6, ISBN: 978-1-5386-3081-5. DOI: 10.1109/ICRA.2018.8460924. [Online]. Available: https://ieeexplore.ieee.org/document/8460924/.

[40] W. Wang, X. Peng, Y. Su, Y. Qiao, and J. Cheng, «Ttpp: Temporal transformer with progressive prediction for efficient action anticipation», *Neurocomputing*, vol. 438, pp. 270–279, May 2021, ISSN: 09252312. DOI: 10.1016/j.neucom.2021.01.087. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0925231221001697.

[41] R. D. Geest, E. Gavves, A. Ghodrati, Z. Li, C. Snoek, and T. Tuytelaars, «Online action detection», unpublished, Apr. 2016. [Online]. Available: http://arxiv.org/abs/1604.06506.

[42] Z. Zhang, G. Peng, W. Wang, Y. Chen, Y. Jia, and S. Liu, «Prediction-based human-robot collaboration in assembly tasks using a learning from demonstration model», *Sensors*, vol. 22, p. 4279, 11 Jun. 2022, ISSN: 1424-8220. DOI: 10.3390/s22114279. [Online]. Available: https://www.mdpi.com/1424-8220/22/11/4279.

[43] K. Kuutti *et al.*, «Activity theory as a potential framework for human-computer interaction research», *Context and consciousness: Activity theory and human-computer interaction*, vol. 1744, pp. 9–22, 1996.

[44] G. Taubin and D. B. Cooper, «Object recognition based on moment (or algebraic) invariants», in *Geometric Invariance in Computer Vision*. Cambridge, MA, USA: MIT Press, 1992, pp. 375–397, ISBN: 0262132850.

[45] F. Mindru, T. Moons, and L. Van Gool, «Color-based moment invariants for viewpoint and illumination independent recognition of planar color patterns», in *International Conference on Advances in Pattern Recognition*, S. Singh, Ed., London: Springer London, 1999, pp. 113–122, ISBN: 978-1-4471-0833-7. DOI: 10.1007/978-1-4471-0833-7_12.

[46] M. Sarfraz, «Object recognition using moments: Some experiments and observations», in *Geometric Modeling and Imaging–New Trends (GMAI'06)*, 2006, pp. 189–194. DOI: 10.1109/GMAI.2006.39.

[47] X. Wu, D. Sahoo, and S. C. Hoi, «Recent advances in deep learning for object detection», *Neurocomputing*, vol. 396, pp. 39–64, 2020, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.01.085. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231220301430.

[48] I. Barabanau, A. Artemov, E. Burnaev, and V. Murashkin, «Monocular 3d object detection via geometric reasoning on keypoints», in *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2020) - Volume 5: VISAPP*, INSTICC, SciTePress, 2020, pp. 652–659, ISBN: 978-989-758-402-2. DOI: 10.5220/0009102506520659.

[49] S. Ren, K. He, R. Girshick, and J. Sun, «Faster r-cnn: Towards real-time object detection with region proposal networks», *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 39, no. 06, pp. 1137–1149, Jun. 2017, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2016.2577031.

[50] F. Zhuang, Z. Qi, K. Duan, *et al.*, «A comprehensive survey on transfer learning», *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2021. DOI: 10.1109/JPROC.2020.3004555.

[51] C. Zimmermann, T. Welschehold, C. Dornhege, W. Burgard, and T. Brox, «3d human pose estimation in rgbd images for robotic task learning», in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia: IEEE Press, 2018, pp. 1986–1992. DOI: 10.1109/ICRA.2018.8462833. [Online]. Available: https://doi.org/10.1109/ICRA.2018.8462833.

[52] D. Rato, M. Oliveira, V. Santos, M. Gomes, and A. Sappa, «A sensor-to-pattern calibration framework for multi-modal industrial collaborative cells», *Journal of Manufacturing Systems*, vol. 64, pp. 497–507, 2022, ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2022.07.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612522001182.

[53] S. Qi, X. Ning, G. Yang, *et al.*, «Review of multi-view 3d object recognition methods based on deep learning», *Displays*, vol. 69, p. 102 053, 2021, ISSN: 0141-9382. DOI: https://doi.org/10.1016/j.displa.2021.102053. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141938221000639.

[54] Y.-W. Chao, Y. Liu, X. Liu, H. Zeng, and J. Deng, «Learning to detect human-object interactions», in *2018 ieee winter conference on applications of computer vision (wacv)*, IEEE, 2018, pp. 381–389.

[55] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, «Openpose: Realtime multi-person 2d pose estimation using part affinity fields», *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, pp. 172–186, 1 Jan. 2021, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2019.2929257. [Online]. Available: https://ieeexplore.ieee.org/document/8765346/.

[56] S. Liu, H. Jiang, J. Xu, S. Liu, and X. Wang, «Semi-supervised 3d hand-object poses estimation with interactions in time», in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 687–14 697.

[57] S. Gupta and J. Malik, «Visual semantic role labeling», *arXiv preprint arXiv:1505.04474*, 2015.

[58] B. Zhuang, Q. Wu, C. Shen, I. Reid, and A. van den Hengel, «Hcvrd: A benchmark for large-scale human-centered visual relationship detection», in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[59] H. S. Koppula and A. Saxena, «Anticipating human activities using object affordances for reactive robotic response», *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 1, pp. 14–29, 2015.

[60] B. Hayes and J. A. Shah, «Interpretable models for fast activity recognition and anomaly explanation during collaborative robotics tasks», in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 6586–6593.

[61] A. Furnari and G. M. Farinella, «What would you expect? anticipating egocentric actions with rolling-unrolling lstms and modality attention», in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6252–6261.

[62] B. Xu, J. Li, Y. Wong, Q. Zhao, and M. S. Kankanhalli, «Interact as you intend: Intention-driven human-object interaction detection», *IEEE Transactions on Multimedia*, vol. 22, no. 6, pp. 1423–1432, 2019.

[63] D. Roy and B. Fernando, «Action anticipation using pairwise human-object interactions and transformers», *IEEE Transactions on Image Processing*, vol. 30, pp. 8116–8129, 2021.

[64] J. Fan, X. Fan, F. Tian, *et al.*, «What is that in your hand? recognizing grasped objects via forearm electromyography sensing», *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 4, pp. 1–24, 2018.

[65] B. Paulson, D. Cummings, and T. Hammond, «Object interaction detection using hand posture cues in an office setting», *International journal of human-computer studies*, vol. 69, no. 1-2, pp. 19–29, 2011.

[66] R.-D. Vatavu and I. A. Zaiţi, «Automatic recognition of object size and shape via user-dependent measurements of the grasping hand», *International Journal of Human-Computer Studies*, vol. 71, no. 5, pp. 590–607, 2013.

[67] T. Feix, J. Romero, H.-B. Schmiedmayer, A. M. Dollar, and D. Kragic, «The grasp taxonomy of human grasp types», *IEEE Transactions on human-machine systems*, vol. 46, no. 1, pp. 66–77, 2015.

[68] C. L. MacKenzie and T. Iberall, *The grasping hand*. Elsevier, 1994.

[69] T. Feix, I. M. Bullock, and A. M. Dollar, «Analysis of human grasping behavior: Object characteristics and grasp type», *IEEE transactions on haptics*, vol. 7, no. 3, pp. 311–323, 2014.

[70] S. Puhlmann, F. Heinemann, O. Brock, and M. Maertens, «A compact representation of human single-object grasping», in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2016, pp. 1954–1959.

[71] S. Betti, G. Zani, S. Guerra, U. Castiello, and L. Sartori, «Reach-to-grasp movements: A multimodal techniques study», *Frontiers in psychology*, vol. 9, p. 990, 2018.

[72] I. Egmose and S. Køppe, «Shaping of reach-to-grasp kinematics by intentions: A meta-analysis», *Journal of Motor Behavior*, vol. 50, no. 2, pp. 155–165, 2018.

[73] D. Valkov, P. Kockwelp, F. Daiber, and A. Krüger, «Reach prediction using finger motion dynamics», in *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–8.

[74] M. Quigley, K. Conley, B. Gerkey, *et al.*, «Ros: An open-source robot operating system», in *ICRA workshop on open source software*, Kobe, Japan, vol. 3, 2009, p. 5.

[75] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, «Robot operating system 2: Design, architecture, and uses in the wild», *Science Robotics*, vol. 7, no. 66, eabm6074, 2022. DOI: 10.1126/scirobotics.abm6074. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074.

[76] Orbbec, *Astra pro plus*, Last accessed 30 March 2023. [Online]. Available: https://shop.orbbec3d.com/Astra-Pro-Plus.

[77] U. Robots, *The ur10e*, Last accessed 9 May 2023. [Online]. Available: https://www.universal-robots.com/products/ur10-robot/.

[78] U. Robots, *Robotiq 2f-140*, Last accessed 24 October 2023. [Online]. Available: https://www.universal-robots.com/plus/products/robotiq/robotiq-2f-140/.

[79] WiredWorkers, *Ur10e*, Last accessed 9 May 2023. [Online]. Available: https://shop.wiredworkers.io/en_GB/shop/universal-robots-ur10e-87.

[80] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, «Hand keypoint detection in single images using multiview bootstrapping», unpublished, Apr. 2017. [Online]. Available: http://arxiv.org/abs/1704.07809.

[81] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, «Openpose: Realtime multi-person 2d pose estimation using part affinity fields», unpublished, Dec. 2018. [Online]. Available: http://arxiv.org/abs/1812.08008.

[82] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, «Convolutional pose machines», unpublished, Jan. 2016. [Online]. Available: http://arxiv.org/abs/1602.00134.

[83] S. Kreiss, L. Bertoni, and A. Alahi, «Openpifpaf: Composite fields for semantic keypoint detection and spatio-temporal association», unpublished, Mar. 2021. [Online]. Available: http://arxiv.org/abs/2103.02440.

[84] S. Kreiss, L. Bertoni, and A. Alahi, «Pifpaf: Composite fields for human pose estimation», unpublished, Mar. 2019. [Online]. Available: http://arxiv.org/abs/1903.06593.

[85] C. Lugaresi, J. Tang, H. Nash, *et al.*, «Mediapipe: A framework for perceiving and processing reality», in *Third workshop on computer vision for AR/VR at IEEE computer vision and pattern recognition (CVPR)*, vol. 2019, 2019.

[86] F. Zhang, V. Bazarevsky, A. Vakunov, *et al.*, «Mediapipe hands: On-device real-time hand tracking», *arXiv preprint arXiv:2006.10214*, 2020.

[87] Google, *Mediapipe*, Last accessed 13 October 2023, 2023. [Online]. Available: https://developers.google.com/mediapipe.

[88] I. H. Sarker, «Machine learning: Algorithms, real-world applications and research directions», *SN Computer Science*, vol. 2, p. 160, 3 May 2021, ISSN: 2662-995X. DOI: 10.1007/s42979-021-00592-x. [Online]. Available: https://link.springer.com/10.1007/s42979-021-00592-x.

[89] K. Fukushima, «Neocognitron: A hierarchical neural network capable of visual pattern recognition», *Neural Networks*, vol. 1, no. 2, pp. 119–130, 1988, ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(88)90014-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0893608088900147.

[90] M. Z. Alom, T. M. Taha, C. Yakopcic, *et al.*, «A state-of-the-art survey on deep learning theory and architectures», *Electronics*, vol. 8, p. 292, 3 Mar. 2019, ISSN: 2079-9292. DOI: 10.3390/electronics8030292. [Online]. Available: https://www.mdpi.com/2079-9292/8/3/292.

[91] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, «Attention is all you need», G. I., F. R., W. H., *et al.*, Eds., Cited by: 31224; Conference name: 31st Annual Conference on Neural Information Processing Systems, NIPS 2017; Conference date: 4 December 2017 through 9 December 2017; Conference code: 136033, vol. 2017-December, Neural information processing systems foundation, 2017, pp. 5999–6009. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85043317328&partnerID=40&md5=3e5a5c2b862c8979ffea845bb707b3c3.

[92] A. Kapoor, A. Gulli, S. Pal, and F. Chollet, «Deep learning with tensorflow and keras - third edition», in 3rd ed. Packt Publishing, Oct. 2022, ch. 6.

[93] G. Amprimo, G. Masi, G. Pettiti, G. Olmo, L. Priano, and C. Ferraris, «Hand tracking for clinical applications: Validation of the google mediapipe hand (gmh) and the depth-enhanced gmh-d frameworks», *arXiv preprint arXiv:2308.01088*, 2023.

[94] G. Amprimo, C. Ferraris, G. Masi, G. Pettiti, and L. Priano, «Gmh-d: Combining google mediapipe and rgb-depth cameras for hand motor skills remote assessment», in *2022 IEEE International Conference on Digital Health (ICDH)*, IEEE, 2022, pp. 132–141.

[95] A. Saudabayev, Z. Rysbek, R. Khassenova, and H. A. Varol, «Human grasping database for activities of daily living with depth, color and kinematic data streams», *Scientific data*, vol. 5, no. 1, pp. 1–13, 2018.