POLITECNICO DI TORINO

Master Degree course in Mechatronic Engineering
A.Y. 2022/2023

Politecnico
di Torino

Master's Degree Thesis

COLLABORATIVE ROBOTICS: HUMAN-ROBOT INTERACTION THROUGH GAZE TRACKING

Supervisors

Prof. Stefano MAURO

Dr. Matteo MELCHIORRE

Dr. Laura SALAMINA

Candidate

Alice GIAMBERTONE

December 2023

Table of Contents

Abstract

Collaborative robotics is an increasingly relevant field in industrial automation, that aims to improve close interaction between human operators and robots. The main goal of this thesis is to explore human-robot interaction through gaze tracking, by using a UR5 robot from Universal Robots with 6 degrees of freedom and the free toolkit known as OpenFace 2.0. The project was inspired by the previous hand-over work implemented by the DIMEAS group at the Polytechnic University of Turin with the UR3 but with a different approach, focusing on visual tracking rather than physical hand-over.
The OpenFace 2.0 toolkit played a key role in this work, due to its remarkable ability to extract facial parameters and gaze-related data. The gaze information extracted by OpenFace 2.0 were utilized to drive the robot end-effector towards the human's point of focus in real-time.
The implementation process encompasses control algorithms, system calibration, and the integration of gaze tracking software with the UR5. In order to assess the effectiveness of this implementation, preliminary testing phases were conducted in a virtual environment using the CoppeliaSim software. Subsequently, physical tests were performed in the laboratory with the UR5 robot to verify the ability of the system to track eye movements and to respond appropriately.

Chapter 1

Introduction

1.1 Collaborative robotics

Collaborative robotics, a rapidly advancing field within robotics, is revolutionizing the way humans and machines work together in shared spaces. This discipline aims to create a new paradigm in which robots and humans collaborate harmoniously, enhancing productivity, safety, and efficiency across various industries.
The meaning of collaborative robots, or "cobots", has developed different interpretations based on the specific context of their use. Generally, they can be described as robots specifically created to work alongside humans; additionally, a cobot can be seen as a robot designed for physical interaction with humans within a common workspace. Cobotics encompasses the science and methods associated with the design, creation, examination, and assessment of cobotic systems [1].
According to Vincentini [2], the core concept behind collaborative robotics is to aid individuals in performing tasks that are not achievable through more traditional methods. Collaborative robotics seeks to complement conventional robotics by enhancing human involvement in shared time and space. The convergence of safety features, physical Human-Robot Interaction (pHRI), and human activity distinguishes collaborative robotics from the broader concept of robotics, emphasizing the active engagement of humans on a functional level [2].

Human-Robot Collaboration (HRC) is a central aspect of collaborative robotics, defining the ability to perform complex tasks through direct interaction between humans and robots in two main modalities. The first, physical collaboration, involves deliberate contact and force exchange between humans and robots, thus enabling the anticipation of human motion intentions and appropriate robotic reactions. The second, contactless collaboration, operates without physical interaction, and it aims on enhancing the robot's environmental perception by using technologies such as

learning-based vision and virtual reality. The coordination of actions occurs through the exchange of information acquired through direct (gestures) or indirect communication methods (intentions recognition, eye gaze direction, facial expressions) [3, 1]. Furthermore, HRC systems can be distinguished in 'workplace sharing systems' and 'workplace and time-sharing systems'. In the 'workplace sharing systems', human and robot accomplish different tasks in the same workspace; in the 'workplace sharing systems' the coordination in space and time is required [4].

Collaborative robotics is reshaping industrial processes, elevating efficiency and precision, while also extending their reach across various sectors. They're revolutionizing healthcare, logistics, and everyday life, transcending their initial applications.
One notable example in the medical field is ROBERT (Figure 1.1), developed by Life Science Robotics [5], specifically engineered for lower limb rehabilitation and the mobilization of immobilized patients. Unlike conventional therapeutic approaches, ROBERT provides customized and specific solutions designed for each patient's unique needs. An additional advantage is its capacity to facilitate prolonged and repetitive intensive therapy sessions with minimal therapist intervention [6].

Figure 1.1: ROBERT, cobot in healthcare sector. Reprinted from [5].

In manufacturing, collaborative robots have the potential to revolutionize production processes. Indeed, these robots can perform intricate and repetitive tasks with precision and consistency (Figure 1.2), while human workers can focus on more complex, strategic, or decision-based responsibilities [7].

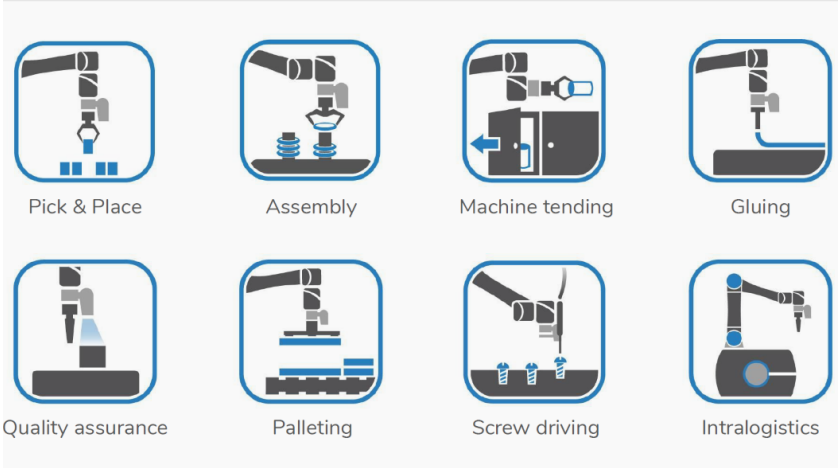


Figure 1.2: Application of cobots in manufacturing. Reprinted from [8].

Another example of how cobots can revolutionize the future manufacturing field is demonstrated by the work of Levratti et al. [9], which presented an innovative robotic assistant for tire workshops, capable of handling heavy wheels and transporting them to any location within the workshop. The robot proved versatile functionality by operating in several modes: independently through user recognition, responding to commands conveyed by gestures, and via tele-operation using a haptic interface.

Cobots stand as a foundational technology within industries, enabling businesses to promptly meet customer demands by adapting manufacturing processes through adaptable and nimble tools. This collaboration between humans and robots not only increases productivity but also enhances the overall quality of work by combining the dexterity and problem-solving skills of humans with the speed and accuracy of robotic systems. They serve as an integral component in the advancement of Industry 4.0, driving innovation and growth in the robotics sector [7].
The evolution of collaborative robotics goes beyond the mere deployment of machines: it signifies a transformation in our perception of the relationship between humans and technology; it's about creating a symbiotic partnership in which robots support and boost human abilities. This model, focused on fostering more efficient, safer, and highly innovative workspaces, distinctly separates collaborative robotics from traditional robotic systems.
The collaborative aspect of cobots allows for a more flexible, intuitive, and adaptable nature, driving the necessity for a new generation of robotics that can seamlessly integrate into human-centered workflows.

Hence, for a comprehensive grasp of collaborative robots and their significance, it's fundamental exploring the landscape of industrial robotics, thus providing the principles for understanding their distinctive characteristics.

1.2 Robots & Cobots

Industrial robots and cobots share a common ground as they are both employed in the realms of production and automation; both are integral components of modern manufacturing, contributing to increased efficiency, precision, and automation of repetitive tasks. Their utilization spans diverse industries, including automotive, electronics, food production, and more [10]. However, despite their shared purpose in enhancing industrial processes, industrial robots and cobots diverge significantly in their design and interaction with human workers.

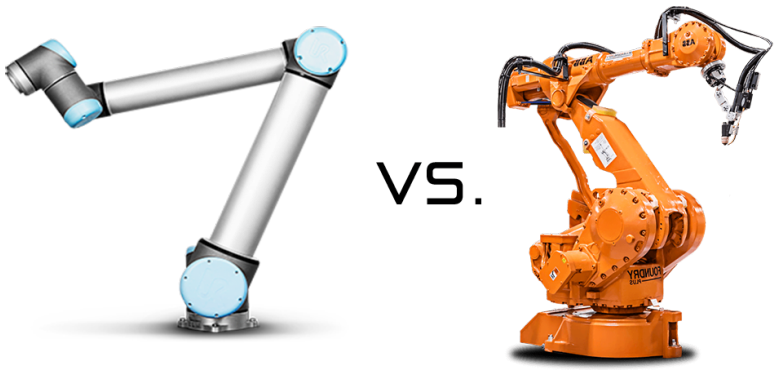


Figure 1.3: Cobot vs Robot.

The main distinction between cobots and traditional robots lies in their intended function: cobots are designed to collaborate with humans rather than entirely supplant them. For instance, as stated in studies [11, 10, 12], traditional industrial robots operate autonomously, carrying out tasks that replace human intervention. These robots often handle heavy and unwieldy equipment, such as large welding tools, executing their functions swiftly and with exceptional precision. Conversely, a cobot typically assists a human operator in various tasks, undertaking functions that might be risky, physically taxing, or monotonous if performed solely by an employee. Moreover, cobots often contribute to more intricate tasks that cannot be entirely automated, like managing wires within an appliance.

Robots are typically confined within safety enclosures in their own working space; fencing or cages are used to prevent accidental human entanglement with the moving parts of these robots. On the contrary, cobots are specifically designed to operate in close proximity to humans in the same space and prioritize human safety in their design, incorporating inherent safety measures compliant with specified safety standards for human interaction [11, 7].

Industrial robots are larger, heavier, and often fixed or anchored to the floor, intended to remain stationary once activated. This immobility is also deemed a safety measure, guaranteeing the robot's stability regardless of the speed and force required for its assigned tasks. Conversely, cobots are notably lighter, providing increased mobility and facilitating their movement within the industries [11].

Another evident contrast between robots and cobots lies in the complexity of programming required for their operation. For instance, cobots are easy to program, thus providing exceptional flexibility, and enabling them to accomplish a wide array of tasks. Their intuitive and user-friendly programming interfaces facilitate swift and efficient reprogramming to suit various tasks or environment. Industrial robots, instead, are comprised of intricate programming systems that demand a higher level of technical expertise for operation [11, 10, 13].

Lastly, industrial robots are faster and have a greater reach volume, making them preferable for high-volume processes with minimal variation. On the other hand, cobots, due to their emphasis on safety and collaboration, operate at a slower pace, yet excel in tasks requiring higher accuracy and precision [13].

The main features that differentiate robots and cobots are listed in the Table 1.1 below.

Table 1.1: Scheme of the main characteristics of robots and cobots in comparison.

Feature	Industrial Robots	Cobots
Intended function	-Autonomous -Replace human tasks	-Collaborate -Assist in human tasks
Interaction with humans	-Confined -Safety enclosures	-Close proximity -Prioritize safety
Size and mobility	-Larger -Fixed	-Lighter -Maneuverable
Programming complexity	-Complex -High technical expertise	-Easy -User-friendly -High flexibility
Speed and reach volume	-Faster -Greater reach -High volume	-Slower -Emphasis on safety -Precision

Navigating through these differences, the inherent variety and complexity of industrial robotic technologies become apparent. To round out the picture, the distinctive features of cobots will be explored, with a focus on their ability to incorporate advanced sensors and cutting-edge technologies based on Artificial Intelligence (AI).

1.3 Characteristics of cobots and technological advancements

The development and integration of cobotic systems continue to progress, driven by advancements in artificial intelligence, machine learning, and sensor technologies. This emerging technology aims to harness the strengths of both humans and robots, fostering a cooperative environment where each complements the other's ability. As technological advancements propel the field of collaborative robotics forward, cobots are evolving to become more sophisticated, adaptable, and capable of a diverse range of tasks in various environments [10]. Notable progress in enhancing their capabilities is observed in three key characteristics.

- **Integration of artificial intelligence:** cobots are increasingly incorporating AI technologies to enhance their decision-making capabilities. As reported in this study [10], artificial intelligence and robotics allowed the discovery of innovative solutions for challenges faced by businesses of various sizes and across diverse industries. AI-powered robots play a pivotal role in bridging the gap between

humans and technology, addressing issues, and adjusting business strategies to meet evolving customer expectations. The integration of machine learning, a subfield of AI, is crucial for AI robots, allowing them to continually enhance their performance over time. Robots utilizing machine learning can develop new learning methods and capabilities by leveraging contextual knowledge acquired through experience and real-time data.

- **Enhanced sensory systems:** the evolution of cobots is marked by the incorporation of sophisticated sensor technologies, including vision systems, tactile sensors, and environmental perception capabilities [13]. These advanced sensors play a central role in empowering the cobots interaction with their surroundings, in a more intelligent and efficient way. This heightened awareness and responsiveness contribute significantly to creating a safer working environment, thus avoiding the need for rigid physical barriers between humans and robots. This breakthrough in safety measures expands the horizons for collaborative robots. Hence, the integration of enhanced sensory systems not only encourages a more secure coexistence between humans and robots but also unlocks numerous opportunities for innovative and collaborative solutions in several sectors.
- **Human-Robot Collaboration:** technological advancements are fostering more seamless collaboration between humans and cobots. Enhanced communication interfaces and intuitive control mechanisms contribute to a harmonious interaction between operators and cobots, further solidifying their role in collaborative manufacturing environments.

After reviewing the advanced features of cobots, it is fascinating to investigate how these features can be translated into practical applications, and a notable example of this is represented by the hand-over. The sensors and environmental perception capabilities that characterize cobots manifest tangibly during the hand-over.

1.4 Human-Robot hand-over

In the human-robot collaboration field, the hand-over plays a fundamental role in both industrial and domestic environments [14]. It pertains to the process of exchanging objects or information between a giver and a receiver (Figure 1.4), necessitating effective communication and mutual comprehension between the participating entities. According to Kruger et al. [4], the hand-over can be considered as a 'workplace and time-sharing' task, since coordination in space and time is required, as well as human and robot work in the same space performing common tasks.

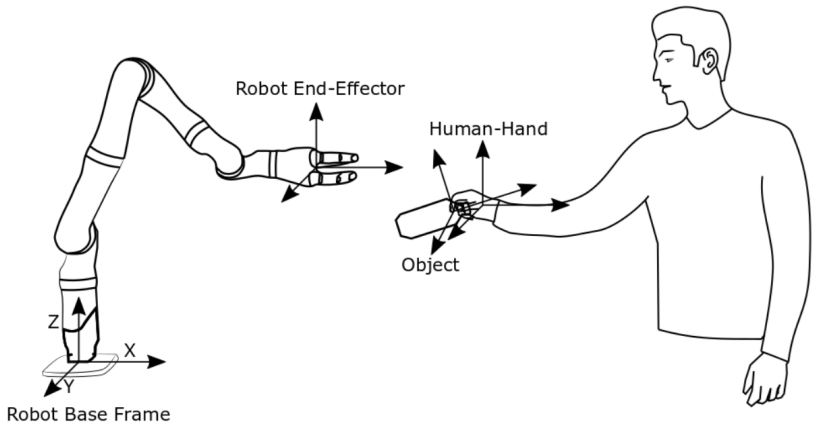


Figure 1.4: Human-Robot object hand-over; human as giver and robot as receiver. Reprinted from [14].

The hand-over conventionally encompasses three phases: (i) a reach phase where both giver and receiver extend their arms toward the hand-over location, (ii) a transfer phase during which the object moves from the giver to the receiver, and (iii) a retreat phase as the actors conclude the interaction and the robot goes in home position [15].
In order to understand collaborative interaction, several features of human-robot hand-over have been examined, including aspects such as motion planning, grasping techniques and the psychological impact on the human worker. Particularly, the focus on the psychological impact aims to identify the characteristics that contribute to a more fluent and natural experience for humans.

As described in Kshirsagar et al. [14], current algorithms for human-robot hand-over can be classified into two categories: offline and online. Most of the previous research on human-robot hand-over has developed an offline approach, where the object position and robot movements are planned before the hand-over begins. These

procedures do not consider the observed behaviour of the human during the hand-over, thus requiring the human adaption to the robot predetermined actions [14]. Regarding the offline control strategy, Waldhart et al. [16] developed a planner that addresses the problem of planning between humans and robots in constrained environments. The planner efficiently calculates sequences of hand-overs in advance of the actual execution. Moreover, it's important to note that this approach does not consider real-time behaviour or adaptation based on the observed behaviour of the human during the hand-over [16].

In contrast, online control strategies enhance the dynamic robot response to real-time data and the adaption of the human's actions during the hand-over. The robot can determine the object position by utilizing continuous data derived from specific sensors, which captures the movements of the human operator [17]. Wang et al. proposed a novel approach to control object hand-over by using a wearable sensory system. This approach enables humans to control the hand-over process effectively and naturally by recognizing human hand-over intentions. The wearable sensory system captures the motion information of the human's arm and hand, that are subsequently processed to recognize the human's hand-over intentions [18].

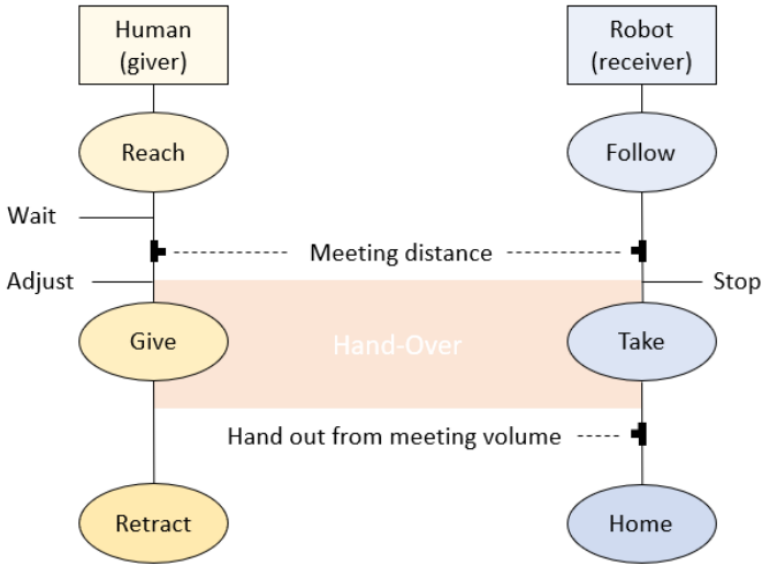


Figure 1.5: Structure of hand over task for human and robot. Reprinted from [19].

Alternative control strategies facilitate the robot in tracking a dynamic target, making real-time adjustments to the trajectory for precise hand-over execution. In their work [19], Scimmi et al. developed a vision-based control architecture able to direct the robot end-effector towards the right hand of the human, thus serving as dynamic

target point. The hand-over control strategy relies on two primary algorithms: the duplex-Kinect algorithm and the hand-following algorithm (Figure 1.5). In the latter, two crucial volumes have been defined – a stopping volume and a meeting volume – governing robot movements and ensuring a secure and efficient hand-over. The robot halts within the stopping volume once the Tool Center Point (TCP) reaches the designated space surrounding the target. After that, the robot is authorized to resume movement only when the operator withdraws their hand following the completion of the hand-over task [19].
In their subsequent study [17], the hand-over process was expedited by incorporating the prediction of the position of the human's hand.

Several research, including the one conducted by Strabala et al. [20], have demonstrated that a thorough understanding of cognitive dynamics can significantly enhance the effectiveness of hand-over. The authors proposed a comprehensive approach wherein a robot not only manages physical coordination, encompassing tasks like approaching, reaching, and transferring control, but also integrates social-cognitive coordination. Insights derived from human-human hand-overs, underscore the utilization of both physical and social/cognitive cues for seamless coordination. Non-verbal signals like eye gaze and body orientation act as indicators of readiness to initiate an hand-over. The outcomes of Strabala et al.'s study provide valuable guidelines for the Human-Robot Interaction (HRI) field [20]. Designers are encouraged to implement human-like gestures and cues, fostering seamless hand-overs.

In the context of human-robot collaboration, hand-over has been examined as a fundamental skill that necessitates effective communication and mutual understanding among the involved entities. However, beyond the physical coordination of hand-overs, there is a growing need to delve into the cognitive dynamics involved in such interactions.

1.5 Cognitive human-robot interactions

Nonverbal communication, which includes eye contact, gestures, and body language, is crucial for human interactions, particularly during hand-over. These cues are essential for coordinating actions, indicating the intent to engage in a hand-over, and facilitating a seamless interaction process [21, 22].
To optimize Human-Robot Interaction in industrial settings, the robot must be capable of handling a wide range of behaviours and actions, encompassing voices, gestures, and facial expressions, thus facilitating smooth and safe collaborations [1].

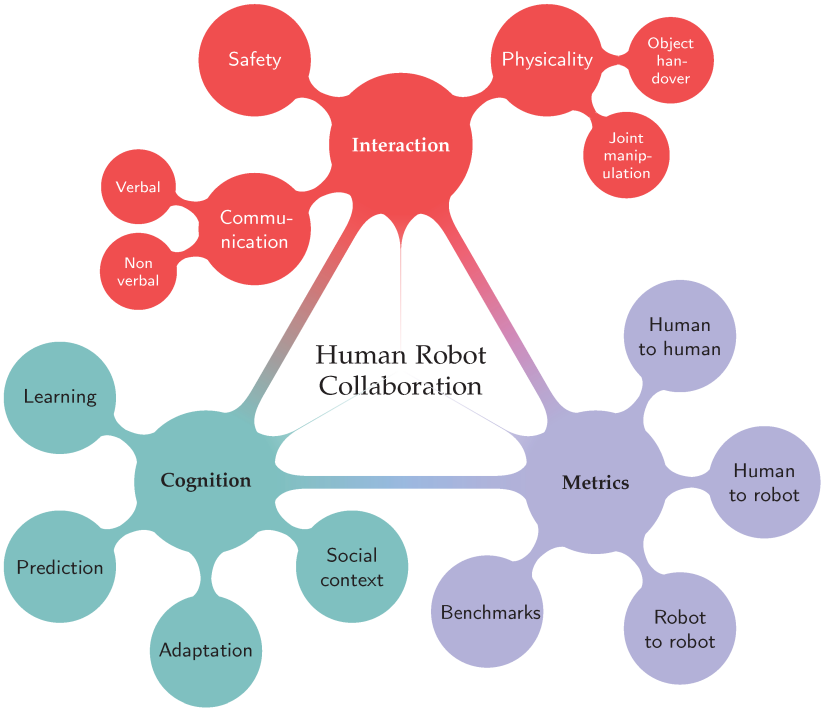


Figure 1.6: Map of Human Robot Collaboration as integration of interaction, cognition and metrics. Reprinted from [23].

Moreover, various cognitive dynamics play a crucial role in shaping the interaction between humans and robots. These cognitive aspects influence human-robot interaction and must be considered to ensure effective communication and congenial collaboration [1]:

- **Voice commanding:** advanced speech recognition allows the robot to comprehend verbal instructions, thus improving communication and coordination. A study presented in [24] explores the integration of natural language capabilities and responsive behaviour in robots, aiming to make them more social. This enhancement contributes to improve a natural, friendly, and efficient interaction in the human-robot relationship.

- **Face recognition:** recognizing facial expressions enables the robot to adjust its emotional response during hand-overs, enhancing user understanding and trust.

- **Action recognition:** refers to the process of identifying and understanding human actions by artificial intelligence or image processing systems. It is often associated with computer vision and human behaviour analysis.

- **Gesture recognition:** involves the identification and interpretation of human gestures by computer systems. These gestures can include movements of the hands, arms, face, or other body parts. The study in [25] concentrates on defining a set of gestures for communication between humans and robots, specifically in the context of automotive manufacturing. The results highlight a preference for gesture communication in industrial settings. These gestures are carefully defined, taking into consideration the needs of both the worker and the robot.
- **Social gaze:** refers to the use of gaze as a communicative signal within social interactions, including human-robot interaction. In this context, social gaze indicates how a robot's gaze can influence and enhance communication and understanding during interactions with human users. Fischer et al investigated the effects of gaze towards the human tutor in a human-robot assembly scenario. The results showed that social gaze serves as an indicator of the robot's affordance. Moreover, individuals in the social gaze condition engaged with the robot more quickly and felt a greater sense of responsibility for task performance [26].

Lastly, the authors in [1] noted that humans naturally interact with the world using multiple resources simultaneously. Consequently, to facilitate Human-Robot Interaction in such systems, it is essential to integrate various modalities with high-level interfaces.

Among all cognitive interactions, social gaze stands out as a pivotal element, especially in the context of hand-overs. In particular, the use of gaze information during hand-overs can be useful to indicate the destination or the object to be transferred, thereby facilitating a clear understanding of intentions, and thus simplifying the interaction between the robot and the human agent.

1.5.1 Potential of gaze tracking in hand-over

Several studies [27, 21, 28, 29, 22] have highlighted the effectiveness of integrating the gaze tracking into the hand-over process, showcasing its ability to signal intentions even before the actual transfer event.

Kshirsagar et al. in [27] focused on the gaze behaviours of the robot when it is receiving an object from a human. The primary objective was to scrutinize how these varying gaze behaviours influenced the human's perception, specifically in terms of liking, anthropomorphism, and the temporal aspects of the hand-over process. Notably, the study revealed that a "transition gaze", characterized by the robot initially directing its gaze at the giver's face and subsequently at the giver's hand, was

consistently perceived as more anthropomorphic and likable by human participants. Interestingly, despite these perceptual effects on likability and anthropomorphism, the study found no conclusive evidence suggesting that the robot's gaze had a significant impact on the initiation time of the human's hand-over [27].

Another research [28] explores how safety in the process of passing an object from a robot to a human can be improved by considering non-verbal cues from the human. It focuses on making this exchange more natural and safer by using cues like human gaze and attention. They developed a two-layer system: the first layer is focused on the physical aspects of the hand-over, while the other observing the human's attention. By integrating cues such as gaze into the robot's decision-making process, the success rate of these hand-overs significantly increased, making the system safer compared to a basic version. Hence, this study highlights the importance of understanding human behaviour and intentions during object hand-overs; by incorporating these human-like elements into a robot's decision-making, it improves safety and trust in interactions between humans and robots [28].

The study [29] delves into the dynamics that make the hand-over from a robot to a human as natural as possible, with a particular emphasis on the robot's gaze direction during the action. The configuration in which the giver looks at the object initially, then, as their arm moves, begins to look at the receiver and maintains this position until the end (referred to as "OR"), has been identified as the most natural and effective. The key findings emphasize that when the robot focuses its gaze on the object during delivery, it appears more attentive and engaged. Additionally, establishing eye contact during the hand-over makes the robot more sociable and communicative. Interestingly, no significant differences were observed based on whether the giver was human or robotic [29].

The study conducted by Moon et al. [22] revealed also positive implications of integrating gaze tracking, specifically through shared attention, during the hand-over process from a robot to a human. First of all, participants demonstrated quicker reach for the object when the robot directed its gaze towards the intended hand-over location, compared to scenarios with no gaze cues. Furthermore, subjects perceived the hand-over as more natural, indicative of timing, and preferable when the robot provided visual cues of shared attention [22].

The reported studies consistently demonstrate that integrating gaze tracking into the hand-over scenario positively influences various aspects, including anthropomorphism, likability, safety, the perceived naturalness and efficiency. The findings collectively underscore the significant role of gaze tracking in signalling intentions,

improving safety through non-verbal cues, and making the hand-over more natural and effective. Therefore, it can be confidently asserted that gaze tracking has the potential to be a valuable tool for optimizing the hand-over process between robots and humans.

1.6 Aim of the work

This work thesis is inspired by the prior project, 'Experimental Real-Time Setup for Vision Driven Hand-Over with a Collaborative Robot' [19], conducted by the DIMEAS department at the Polytechnic University of Turin. However, compared to the previous mentioned project, it adopts a distinct approach, placing a heightened emphasis on gaze tracking in contrast to traditional physical hand-over.
Building upon earlier research that established a positive correlation between gaze tracking and successful hand-overs, this thesis delves into a detailed analysis of gaze tracking dynamics. For instance, the integration of gaze tracking into hand-over process can effectively improve the understanding of visual inputs and human gaze directions, aspects that could further enhance human-robot collaboration and the smoothness of the hand-over.

While the primary focus remains on gaze tracking, the study is motivated by the interesting outcomes of previous investigations, highlighting the crucial role of gaze tracking in optimizing hand-over interactions. In doing so, this work not only contributes to the current understanding of gaze-assisted hand-overs, but also lays the foundation for potential advancements and future developments in this evolving field.

The research methodology encompasses a systematic exploration of gaze tracking systems, with a particular emphasis on Remote Eye Gaze Trackers (REGT). This comprehensive analysis will delve into the software and hardware components associated with REGTs.
In this context, several software and integrated systems will be evaluated, ultimately leading to the choice of OpenFace 2.0, a facial behavior analysis toolkit designed for computer vision and machine learning researchers. OpenFace 2.0, renowned for its accuracy in facial landmark detection, head pose estimation, facial action unit recognition, and eye-gaze estimation, will serve as the foundation for subsequent implementation of the gaze tracking code.

The MATLAB gaze tracking code will function as a bridge between user requirements and OpenFace 2.0 functionalities. The architecture will prioritize real-time accessibility to gaze tracking data by executing OpenFace 2.0 in the background. Simultaneously, a separate module will dynamically retrieve and process data, employing a dual-loop mechanism for accuracy verification.
The gaze tracking code will be constituted by two main loops: the first loop will verify the creation of the OpenFace 2.0 output file, while the second one will be dedicated to dynamically retrieving and processing data. This will involve ensuring data integrity and extracting only essential information for subsequent analyses. In fact, it will handle the data obtained from OpenFace 2.0, manipulating and managing only the

necessary information, as the OpenFace-generated file contains numerous data points that are irrelevant to the analysis. Hence, the two-loop design will optimize efficiency by minimizing unnecessary verification checks after the file detection.

To assess the feasibility and effectiveness of the implemented gaze control, preliminary tests will be conducted in a simplified environment using CoppeliaSim, a simulation platform. During this initial phase, a basic scene with a sphere and a plane will be simulated. In particular, tests will include first unidirectional gaze control along x axis, fixing y and z coordinates of the sphere, then the unidirectional control along y-axis, fixing x and z coordinates and the last test will involve the bidirectional control in xy plane. Gaze data, particularly gaze angles provided by OpenFace 2.0 (gaze_angle_x and gaze_angle_y), that represent the left-right and up-down gaze directions in radians in camera coordinates, will be used to map and control the movement of the sphere accurately. This mapping ensures that changes in the user's gaze direction will be translate into corresponding movements of the simulated sphere.

The preliminary tests performed with the sphere in simulated environment will set the stage for the next phase of this research leveraging gaze-driven control for the subsequent interaction with the UR5 robot. Indeed, the attention will turn towards the main features of UR5 robot and of its components, leading to a complete overview of the robot functionalities that will be then controlled by the developed eye gaze tracking algorithm. Before the transition to the physical robot, the same preliminary tests performed on the sphere will be conducted on the simulated UR5 using the CoppeliaSim simulation environment.
For the control of UR5 robot, both in simulation and in real-world scenarios, the previously implemented gaze tracking code (control only along the x and y axes with z fixed) will be integrated along with a path planning algorithm, that will consider the gaze position as a dynamic target. Together, they can be referred as a gaze-following algorithm. The gaze-following algorithm, managing feedback data received from the UR5 controller and gaze data, will then send the set of joint velocities to the UR5 controller, considering the inverse Jacobian and the linear velocity of the TCP. The linear velocity will be computed by considering the distance between the TCP and the target, with a smooth profile, that is faster when far from the target and decreasing as it approaches the target.

The behavior of the UR5 robot controlled with the gaze path algorithm will be tested, first considering the original gaze signal, and then by applying two types of filters to the gaze signal: a moving mean filter with window sizes of 5 and 10.
The execution of these tests will lead to an in-depth examination of the robot's ability to follow the user's gaze.

Chapter 2

Gaze tracking systems

In the human-computer interaction and visual analysis, gaze tracking refers to the technology and techniques employed to monitor and interpret the direction of a person's eye gaze. It plays a crucial role in predicting real-time human visual focus by capturing visual information from the user's face and eyes. The direction of eye gaze serves as a reflection of human attention and interest in the surrounding environment.

This technology not only measures the user's attention on any object but also aims to understand the user's desires and needs [30]. Techniques for gaze tracking and estimation have been extensively researched across diverse fields, including usability research, marketing, psychology, entertainment, and gaming.
For instance, in the field of advertising, the utilization of eye gaze allowed for an in-depth exploration of customer interest, facilitating the development of more captivating advertisements. In Human-Computer Interaction, eye gaze emerged as a potential alternative or supplementary input method, surpassing traditional tools like mouse and keyboard [31, 32].

The fundamental principle of eye tracking involves the detection of users' eye movements during interaction, achievable through different tools such as webcams, infrared cameras, and head-mounted displays. The acquired data unveils valuable insights into user behaviour, such as attention focus on specific screen elements [31].

Eye gaze tracking applications can be classified into diagnostic and interactive. In diagnostic applications, eye gaze data are employed as quantitative proof of the user's visual and attentional processes. On the other hand, interactive applications utilize eye gaze data to engage with or respond to the user based on the analysis of observed eye movements [33].

2.1 Evolution of eye research

As reported in [34], advancements in eye research and systems have progressed through four distinct periods (Figure 2.1), each discernible by the type of data and their characteristics:

- **First period (1879-1920):** was distinguished by the fundamental examination of the eye structure and movement.
- **Second period (1930-1950):** witnessed the advent of eye oculography, which entailed measuring and recording the eye position and movement. This process involved the use of invasive instruments that required physical contact with the user, such as contact lenses, electrodes, or head-mounted devices.
- **Third period (1970-1998):** represented a significant advancement in hardware processors and image processing techniques. This progress facilitated the evolution of non-intrusive gaze tracking applications, relying on vision-based approaches that utilize cameras to capture eye images. These applications aimed to estimate the drop point of visual attention, commonly referred to as the point of gaze.
- **Fourth period (2000-):** witnessed an exploration of REGT (remote eye gaze tracking) system capabilities, delving further into innovative applications.

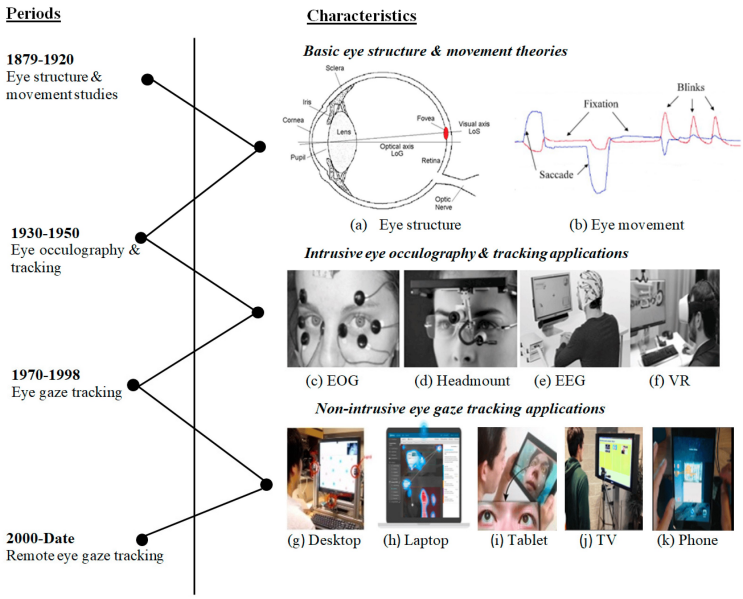


Figure 2.1: The evolution of eye research from 1879 until today. Reprinted from [34].

2.1.1 Intrusive & remote tracking methods

Traditionally, methods for eye gaze tracking have leaned towards intrusive techniques, necessitating physical contact through tools like contact lenses, electrodes, or head-mounted devices. In contrast, non-intrusive or remote approaches primarily rely on vision-based methods, utilizing cameras to capture eye images [33].

Intrusive eye gaze tracking methods, renowned for their heightened accuracy, encompass techniques such as electro-oculography, involving sensors around the eyes to measure the electric field generated during eye movement. However, these intrusive approaches demand direct contact, leading to interference. As technology advances toward intelligent systems, the focus expands beyond accuracy to include user experience. Consequently, non-intrusive gaze tracking systems, referred to as remote eye gaze trackers , emerge as the preferred choice [34].

The adoption of REGTs prioritizes user comfort and facilitates quicker setup, enabling extended system usage compared to intrusive methods. While REGTs are less precise, they provide unobtrusive alternatives, particularly suitable in scenarios where wearing devices proves inconvenient or impractical [33, 30].

The distinction between intrusive and remote systems underlines the inherent trade-offs between precision and user convenience. Although intrusive methods excel in precision, they may inconvenience users in specific contexts. On the other hand, remote eye trackers, while less precise, offer unobtrusive solutions, proving advantageous in scenarios prioritizing user comfort and mobility, such as in mobile robots or automotive applications. Integrating a remote eye tracker into a mobile robot's visual system has the potential to expand its working area, overcoming limitations in device mobility [33].

2.1.2 Feature based & appearance based methods

In REGT systems, the software component includes two main approaches: *features-based method* and *appearance-based method* (Figure 2.2).

Gaze estimation methods utilizing extracted local features such as contours, eye corners, and reflections from the eye image are classified as **feature-based methods**. These approaches focus on the identification of specific features within the eye region, making use of machine vision techniques. The detection of elements like pupil and glints, particularly under active light models, is relatively straightforward in these methods.

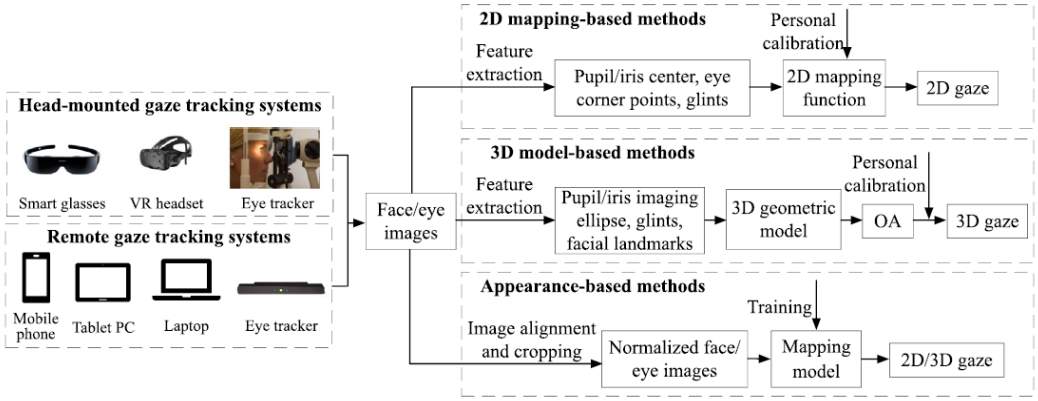


Figure 2.2: Gaze estimation methods . Reprinted from [30].

Moreover, these features can be formally linked to gaze, considering factors related to the geometry of the system and eye physiology. For these reasons, feature-based methods have gained popularity and become the predominant approach for gaze estimation [34].

Within the category of feature-based methodologies, two distinct paradigms emerge: the interpolation-based (regression-based) and the model-based (geometric) approaches.

- **Interpolation-based** techniques presuppose a parametric or nonparametric correlation from image attributes to gaze coordinates. These methodologies, steering clear of the direct computation of the point where gaze direction intersects with the observed object, leverage specific parametric structures such as polynomials or nonparametric configurations like neural networks [32].

- **Model-based** methodologies directly ascertain gaze direction from eye features through a geometric model of the eye. The point of gaze is approximated by the intersection of the gaze direction with the object in focus [32].

Appearance-based methods shift the focus from specific features to the overall appearance of the eye region. Utilizing machine learning or deep learning algorithms, these methods analyze factors such as eye color, texture, and pixel intensity patterns. During the training phase, subjects gaze at known locations on the screen, generating data for training the model to associate specific eye appearances with corresponding gaze directions. This learned information enable the model to predict a person's gaze based on the overall appearance patterns of their eyes. Various algorithms, including

genetic algorithms, Bayesian classifiers, support vector machines, and artificial neural networks, can be employed for feature extraction and mapping gaze points [34, 32].

2.1.3 Hardware components of REGT system

Gaze tracking technology relies on a sophisticated hardware setup designed to capture, process, and interpret visual information from users' faces and eyes. The key hardware components of a Remote Eye Gaze Tracking system include [34]:

- **High-resolution cameras**:
  High-resolution cameras are integral to the hardware configuration, enabling the capture of detailed facial and eye images. The quality of these cameras directly influences the precision of gaze tracking, as they play a crucial role in recording subtle eye movements and expressions.

- **Infrared illuminators**:
  Infrared illuminators enhance the eyes visibility, especially in varying lighting conditions. This component is essential for ensuring accurate tracking, as it provides illumination that is imperceptible to the human eye but enhances the contrast of eye features for the cameras.

- **Device interface**:
  The device interface serves as the interaction point between the subject and the tracking software. It facilitates seamless communication, allowing users to engage with the system effortlessly. This component is crucial for a user-friendly experience during data collection and analysis.

- **Illumination variability**:
  REGT systems can be categorized as either active or passive based on their illumination strategy. Active systems incorporate their light sources, such as infrared illuminators, while passive systems rely on ambient lighting. The choice between these approaches depends on factors like environmental conditions and specific use cases.

The precision and effectiveness of a REGT system hinge on the seamless integration and optimal functioning of these hardware components. High-quality cameras, infrared illuminators and a user-friendly interface collectively contribute to the system's ability to accurately detect and interpret users' eye movements [34].

Despite active research endeavours, eye detection and tracking persist as challenging tasks due to unique issues such as eyelid occlusion, variations in eye size, reflectivity,

head pose, and other factors contributing to the complexity of the task [32]. Therefore, there isn't a single affordable solution that is able to meet all the characteristics, but there are several devices and software for gaze tracking.

2.2 Evaluation of open source and commercial eye tracking systems

After examining the features of eye tracking systems in general, it is crucial to explore open source solutions available and through paid packages. This evaluation focuses on software and eye tracking systems accessible via the internet or through specific purchase plans. In this context, numerous software solutions, both free and paid, have been scrutinized to identify the optimal solution for the project.

2.2.1 Eye tracking software

During the research, various open source eye tracking software options were explored. The evaluation of these open source tools was based on criteria such as accuracy, accessibility, user-friendliness, and compatibility with the project's requirements.

- **GazeParser** [35]: is an open source library for low-cost gaze tracking and data analysis. Despite its good accuracy in determining gaze direction, GazeParser was not chosen due to its requirement for a head and chin rest to function optimally. In my work, a non-intrusive approach allowing head movement was preferred.

- **EyeTab** [36]: is tailored for portable devices like tablets, eliminating the need for external and costly hardware, such as cameras or infrared illuminators, thus ensuring a non-intrusive user experience. However, it's important to note that the tests were conducted at a fixed distance of 20 cm. In scenarios involving hand-overs, EyeTab's effectiveness might be limited.

- **OpenGazer** [37]: has some interesting specifications but might not be ideal for a hand-over system as it requires a webcam and only operates on Linux. Additionally, accuracy might be limited due to its exclusive support for webcam. Moreover, the need for specific programming and Linux skills could limit accessibility to non-expert users.

- **OpenEyes** [38]: it's a software that allows eye tracking with both webcams and infrared camera. It requires MATLAB but it doesn't provide a data analysis option.

- **ITU gaze tracker** [39]: is tailored to work with webcams with infrared lighting. Written in C#, it necessitates some technical proficiency for configuration. Its primary function involves eye-controlled cursor navigation, often in conjunction with a typing application for visual typing.
- **GazeTracking** [40]: is a Python library that provides a webcam-based eye tracking system. It gives in real time the exact position of the pupils and the gaze direction.
- **OpenFace 2.0** [41, 42]: is an open-source tool capable of facial landmark detection, head pose estimation, facial action unit recognition, and eye-gaze estimation with available source code for both running and training the models. The output includes these detections and estimations, which can be saved to disk or sent via network in real-time, making it efficient for various applications. In addiction, it can run also with a simple webcam.
- **GazeSense** [43]: is an eye tracking software implemented by Eyeware that tracks eye movements and gaze direction in real time without the need of an expensive hardware. In fact, it can work with 3D cameras or webcams.

2.2.2 Integrated eye tracking systems

Alongside open source softwares, commercial eye tracking systems were considered. These integrated systems combine both hardware and software components to offer a cohesive and efficient eye tracking experience. The assessment included considerations of features, functionality, and overall suitability for the project's objectives.

Tobii [44] is a global leader in eye tracking and a pioneer of attention computing. Two Tobii's integrated systems were evaluated.

- **Tobii Pro Glasses 3** [45]: wearable eye tracker designed to capture what the wearer is viewing while providing robust and accurate eye tracking data. Tobii Pro Glasses 3 supports live view of the scene camera video; an additional data channels provide complete eye tracking information (2D gaze, 3D gaze, gaze origin, gaze direction and pupil diameter.
- **Tobii Pro Spark** [46]: is a compact, high-performance screen-based camera. Sophisticated image processing algorithms identify relevant features, including the eyes and the corneal reflection patterns. Combined with its dedicated software Tobii Pro Lab and Tobii Pro SDK, it's possible to record data, analyze and visualize eye tracking data, such as 3D eye coordinates, raw data, pupil data.

2.2.3 Trade-off analysis of eye tracking solutions

After evaluating numerous eye-tracking software and integrated systems, the focus centered on comparing the final software solutions GazeTracking, OpenFace 2.0, GazeSense and the integrated systems Tobii Pro Glasses 3 and Tobii Pro Spark. The distinctive features considered for each of these solutions are tabulated in Table 2.1.

Table 2.1: Comparison between several eye tracking systems, considering different features.

Gaze tracking system	Sampling rate	Accuracy [Hz]	Operating distance [cm]	Output	Language	Price
Gaze Tracking	Variable	Undisclosed	30-60	Gaze direction Pupil position	Python	Free
OpenFace 2.0	Variable	Mean error 9°	30-100	Gaze direction Eye landmarks	MATLAB C++ Python	Free
GazeSense	10-90	1.5°	30-80	Gaze direction Pupil origin	Python C++	2000 €
Tobii Pro Glasses 3	50 or 100	0.6°		2D,3D gaze Gaze origin Gaze direction	Python Javascript HTML	20000 €
Tobii Pro Spark	33 or 60	0.45°	45-90	Gaze origin Gaze direction	Python MATLAB C	10000 €

Following the evaluation of various eye tracking solutions, a crucial trade-off was necessary to reach a final decision.

Despite both Tobii products demonstrated an exceptionally high level of accuracy, their cost remains considerably high. Conversely, OpenFace 2.0, although exhibiting a higher error rate, provides an open-source platform primarily in MATLAB. This resonates with my chosen programming language and ensures the extraction of all essential information, thereby influencing the decision in its favor.

2.3 OpenFace 2.0

OpenFace 2.0 is a facial behaviour analysis toolkit for computer vision and machine learning researchers. It offers accurate facial landmark detection, head pose estimation, facial action unit recognition, and eye-gaze estimation. It is designed to address the challenges in facial behaviour analysis, such as real-time performance, robustness to variations in lighting and pose, and accuracy in detecting subtle facial expressions. OpenFace 2.0 is an open-source project that is freely available for research purposes.

- **OpenFace 2.0 pipeline**

OpenFace 2.0 uses a pipeline (Figure 2.3) of core technologies for facial behaviour analysis, including facial landmark detection and tracking, head pose estimation, eye-gaze tracking, and facial action unit recognition. The system is designed to work in real-time, with all of the modules working together to provide accurate and reliable results [41].

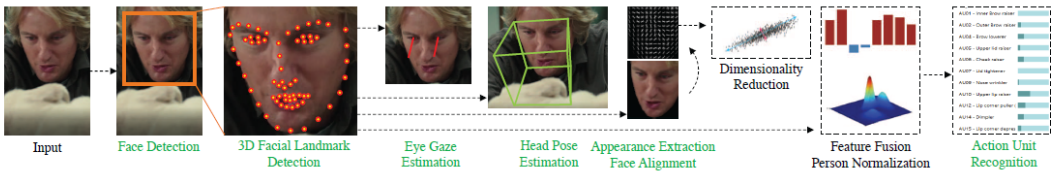


Figure 2.3: OpenFace 2.0 pipeline, including: landmark detection, head pose and eye gaze estimation, facial action unit recognition. The outputs in green can be saved to disk or sent via network in real-time. Reprinted from [41].

Facial landmarks are extracted in OpenFace 2.0 through an advanced methodology known as the Convolutional Experts Constrained Local Model (CE-CLM), to precisely identify key points such as eyes, nose, and mouth. The extraction process involves two main components: the Constrained Local Model (CLM), utilizing a statistical model, comprehends general variations in facial landmark shapes; the Convolutional experts (CE), convolutional neural networks, enhance the recognition of localized variations in facial appearance, focus on specific regions of the face to improve accuracy.
Head pose estimation is achieved by combining the facial landmarks with a 3D face model.
To estimate the eye gaze, a Constrained Local Neural Field (CLNF) landmark detector is employed. This detector identifies key landmarks such as eyelids, iris, and the pupil. The detected pupil and eye location are used to compute the eye gaze vector for each eye. The computation involves firing a ray from the camera origin through the center of the pupil in the image plane. The point of intersection of this ray with the eyeball sphere is then calculated. This intersection provides the pupil's location in 3D camera coordinates. The gaze vector is subsequently estimated as the vector from the 3D eyeball center to the pupil location [41].

- **Interface**

OpenFace 2.0 offers a user-friendly interface that makes it easy to use for both novice and expert users. There are two main ways of using OpenFace 2.0: Graphical User Interface (for Windows), and command line (for Windows, Ubuntu, and Mac OS

X). The system can operate on real-time data video feeds from a webcam, recorded video files, image sequences, and individual images. It is also possible to integrate OpenFace 2.0 in any C++, C, or MATLAB based project, thanks to the availability of the source code. Another huge advantage about OpenFace 2.0 is that it does not require specialized hardware and can run from a simple webcam [41].

- **Output file**

The output file generated by OpenFace 2.0 is a structured .CSV file capturing a range of facial behaviour analysis metrics. Each row in the file corresponds to a specific frame in the analyzed video, and the columns provide detailed information about various facial features and actions. Here's an overview of the key components [42]:

- **interface:** the frame number in the video sequence.
- **face_id :** indicates whether the face tracker detected a single face (0) or multiple faces (1) in the frame.
- **timestamp:** the time in seconds from the start of the video to the moment the frame was processed.
- **confidence:** the confidence level of the face tracker, typically around 0.98.
- **success:** binary indicator (1 or 0) signaling whether the face tracker successfully detected a face in the frame.
- **gaze_0_x, gaze_0_y, gaze_0_z:** eye gaze direction vector in camera coordinates for eye 0 (leftmost eye in the image), normalized.
- **gaze_1_x, gaze_1_y, gaze_1_z:** eye gaze direction vector in camera coordinates for eye 1 (rightmost eye in the image), normalized.
- **gaze_angle_x, gaze_angle_y:** eye gaze direction in radians in camera coordinates, averaged for both eyes. Provides a more convenient format than gaze vectors. gaze_angle_x indicates left-right gaze, and gaze_angle_y indicates up-down gaze.
- **eye landmarks:** multiple columns capturing the x, y, and z coordinates of 55 individual eye landmarks for each eye.

Additional available data, not used in this work, are the following:

- **Head pose:** position and rotation of the head relative to the camera are available in the output file. These parameters provide information about the orientation of the head in the 3D space.
- **Facial landmarks:** landmarks of the entire face are included in the output file. These landmarks represent key points on the face, aiding in detailed facial analysis.
- **Facial action units (AUs):** information related to Facial Action Units, which represent facial muscle movements associated with different expressions, is present in the output file.

While the .CSV output file contains additional valuable data points such as head pose, facial landmarks encompassing the entire face, and information about Facial Action Units (AUs), it's essential to clarify that these specific parameters have not been actively utilized. The primary focus has centered around gaze-related parameters, specifically eye gaze vectors and angles.

Chapter 3

Implementation of gaze tracking and sphere control in CoppeliaSim

The goal was to assess the feasibility and effectiveness of gaze-driven interactions within a controlled environment, exploring the practical implementation of gaze tracking through OpenFace 2.0 and its integration with CoppeliaSim, a simulation environment for real-time control of a simulated sphere. Motivated by the diverse applications of gaze tracking, particularly in human-computer interaction and assistive technologies, this investigation aimed to evaluate the accuracy and responsiveness of gaze tracking in controlling simulated objects. The choice of a simulated sphere as the testing ground provides a fundamental application to gauge the capabilities of gaze tracking and validate the viability of the concept.

3.1 Integration of OpenFace 2.0

OpenFace 2.0 incorporates gaze tracking functionality as part of its comprehensive facial behaviour analysis pipeline. This feature enabled tracking of eye movements, thus providing valuable data for understanding visual behaviour patterns.
The initial step involved the download and the installation of OpenFace 2.0 from its GitHub repository. This process grants access to a versatile toolkit that features a user-friendly graphical user interface (GUI) and extends functionality through MATLAB scripting. Moreover, the inclusivity of a graphical interface and the possibility to run code in MATLAB empowers users to tailor their interaction level to specific needs. The GUI of OpenFace 2.0 served as a central hub for initiating and managing gaze tracking tasks, thus providing an intuitive platform for users to interact with the

software, and offering real-time visualization of facial landmarks, gaze vectors, and other relevant metrics. Understanding the graphical interface is vital for both novice users exploring basic functionalities and advanced users customizing gaze tracking for specific research goals.

Moreover, OpenFace 2.0 supports MATLAB scripting, offering flexibility to manipulate and analyze gaze tracking data programmatically. MATLAB scripts can extract, process, and visualize data generated by OpenFace 2.0, creating a seamless bridge between OpenFace 2.0 functionalities and the specific analytical needs of users. By leveraging MATLAB, researchers and developers can tailor the gaze tracking process to project requirements and conduct in-depth analyses of obtained data.

Beyond its fundamental capabilities, OpenFace 2.0 package includes supplementary tools extending beyond mere experimentation. This comprehensive offering encompasses not only core functionalities, but also provides codes essential for training, enhancing its utility for broader and more specialized research projects. These codes serve as a foundation, ready for customization and modification to meet the unique demands of diverse research endeavours.
This integration approach positions OpenFace 2.0 not just as a tool for gaze tracking but as a comprehensive resource for facial behaviour analysis, catering to a spectrum of research requirements, from real-time applications to advanced experimental setups.

3.1.1 Camera calibration

In the pursuit of precision in gaze tracking, a pivotal stage was the calibration of the tracking camera, specifically the Orbbec Astra Pro. This calibration process went beyond the simple setup, as it involved a fine-tuning of the camera's intrinsic parameters to optimize it for the best accurate gaze estimation.

The calibration process unfolded through a sequence of image captures where known patterns are presented to the camera. These patterns served as reference points, allowing the calibration algorithm to precisely adjust the camera's settings. The correction for lens distortions is a critical aspect, as it ensures that the captured images align with the ideal optical model, thus enhancing the overall accuracy of gaze estimation. During calibration, the algorithm worked to refine intrinsic parameters such as focal length, fx and fy, and principal point coordinates, cx and cy. These parameters (Table 3.1), collectively defined the internal geometry of the camera and played a crucial role in accurately translating pixel coordinates to real-world coordinates. The iterative refinement of these parameters contributed to the creation of a more precise and reliable mapping between the visual input and gaze estimations.

Table 3.1: Intrinsic parameters of Astra Orbbec.

Intrinsic parameters	Values
fx	602.4
fy	601.2
cx	349.8
cy	252.5

3.2 Implementation of gaze tracking code

The gaze tracking code written in MATLAB was the bridge between user requirements and OpenFace 2.0 functionalities. Its architecture was designed to seamlessly integrate essential parameters for effective gaze tracking.

Background execution for immediate data capture
In the pursuit of real-time accessibility to crucial gaze tracking data, a strategic approach was employed. Recognizing the necessity for immediate access to data rather than awaiting post-execution file retrieval, the solution involved the execution of OpenFace 2.0, specifically the *FeatureExtraction.exe* executable, in *background*. The executable FeatureExtraction.exe operates seamlessly in background, leveraging a system command enriched with essential parameters. These parameters included the unique camera device number, finely calibrated intrinsic camera parameters, and specifications for file paths related to background execution and output directories.

Simultaneous data processing
Concurrently, in a separate section of the code, another module is dedicated to the retrieval and processing of the data generated by OpenFace 2.0. This approach enabled the system to function dynamically, with real-time data acquisition and parallel data processing for immediate accessibility.

The real-time data processing module worked through a dual-loop mechanism, designed for dynamic functionality, immediate data accessibility, and accuracy verification. This module is pivotal in ensuring that the system seamlessly captures and processes gaze tracking data generated by OpenFace 2.0.

First loop - file creation verification
The initial loop was intended to verify the creation of the OpenFace 2.0 output file, a .CSV file. Since the execution of the FeatureExtraction.exe executable required few seconds to initialize the camera window and generate the file, this loop monitored the file path to confirm its existence. The system patiently waited until the file was

created in the predefined output directory.

Second loop - dynamic data retrieval and processing
Upon confirmation of file creation, the system went on to the second loop, where the focus shifted to the dynamic retrieval and to the processing of real-time data. This loop ensured against potential timing discrepancies between MATLAB's execution speed and the writing speed of the .CSV file.
Within the second loop, careful attention was paid to ensuredata integrity. The system validated the reading of the last processed line of the .CSV file, verifying its uniqueness and preventing the inadvertent provision of duplicate data. This verification step added an extra layer of accuracy to the process.

During this loop, the system extracted and stored the relevant gaze tracking information: specific columns containing pertinent data were identified and processed, whereas extraneous information present in the .CSV file were discarded. This meticulous extraction ensured the use of only essential captured data by the system, for subsequent analyses.

The decision to implement two distinct loops was a deliberate optimization strategy that provided the advantage to prevent unnecessary continuous verification checks once the file was detected, streamlining the real-time data processing, and maximizing overall system efficiency.

Additional data handling: while the primary focus was on gaze-related metrics, the script initially included data manipulation for eye contours to obtain the pupil's origin in camera coordinates. However, this part was eventually excluded from the final implementation.

This MATLAB script, tailored to the specific requirements of the study, played a crucial role in translating the capabilities of OpenFace 2.0 into a real-time gaze tracking application. Its meticulous structure vouched efficient data handling and processing, contributing to the study's overall success in utilizing gaze tracking for interactive applications.

3.3 CoppeliaSim environment

CoppeliaSim, previously known as V-REP (Virtual Robot Experimentation Platform), stands out as a versatile and scalable robot simulation framework [47, 48]. It facilitates the rapid and precise simulation of complex physical scenarios using a robust physical engine. Within a simulation scene, various objects such as shapes and vision sensors are hierarchically arranged in a tree structure.

CoppeliaSim is designed with a versatile architecture, it supports multiple programming techniques, offering flexibility in executing control code. The particularly relevant paradigms for this integration are:

- **Embedded scripts:** these scripts, written in Lua, are attached to scene objects as child scripts. The main simulation loop, a Lua script, handles general functionality and calls child scripts according to the scene hierarchy. Child scripts, attached to specific objects, manage particular aspects of the simulation. This modular and distributed nature makes embedded scripts a powerful tool.
- **Remote application programming interface (API) clients:** the remote API interface allows external entities to interact with CoppeliaSim via socket communication. It includes server services and clients written in languages like C/C++, Python, Java, MATLAB, and Urbi. This interface facilitates remote function calling and fast data streaming.
- **Add-ons:** similar to embedded scripts, add-ons are supported via Lua scripts and can function as standalone utilities or regularly executed code.
- **Plug-ins:** these serve as a simulator customization tool, registering custom Lua commands and extending the functionality of simulation models or objects.

To facilitate a seamless integration between OpenFace 2.0 gaze tracking and CoppeliaSim, a MATLAB script acted as the intermediary connecting the two environments. This integration enabled the real-time control of a simulated sphere, driven by gaze input.
The integration progress started with the establishment of a connection between MATLAB and CoppeliaSim through the Remote API, *'remApi('remoteApi')'*. This bidirectional communication enabled the exchange of commands and data between the two platforms. MATLAB employs specific API functions provided by CoppeliaSim to facilitate this interaction, thus ensuring a smooth and synchronized workflow.

An essential step in the interaction was obtaining the object handle for the simulated sphere in CoppeliaSim. This handle was crucial for MATLAB to exert control over the sphere's movements within the simulation.
This function, called, *'simxGetObjectHandle'*, granted to MATLAB the necessary access and control over the specified object, in this case, the simulated sphere.

3.3.1 Control of the sphere

Once the connection was established, a simulation scene (Figure 3.1) was set up within CoppeliaSim to create a controlled environment. The depicted scene was intentionally kept minimal, featuring only the simulated sphere and a floor for spatial

reference. This simplistic setup was useful as an initial step to assess the viability of gaze-based control without introducing unnecessary elements that might distract from the primary objective.

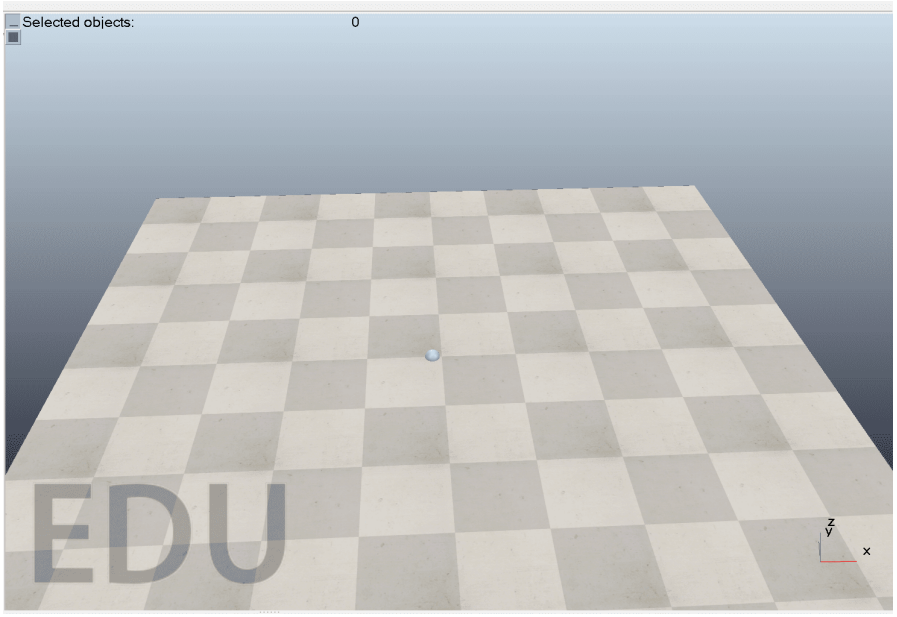


Figure 3.1: CoppeliaSim scene of the sphere.

Gaze-driven sphere control

The MATLAB script, empowered with OpenFace 2.0 gaze tracking functionalities, took the decisive role in directing the simulated sphere based on gaze input. Through real-time processing of gaze data, the script was effectively able to translat the acquired information into meaningful actions within the simulation environment.

Gaze data, particularly the gaze angles, were mapped to control the movement of the sphere. This mapping ensured the accurate translation of the changes in the user's gaze direction into corresponding movements of the simulated sphere. The precision of this mapping was important for simulating the user's visual interactions faithfully. The integration established a dynamic feedback loop where the user's gaze influenced the position of the sphere in real-time. This real-time interaction created a responsive experience within the simulated environment.

To implement the mapping between gaze angles and sphere movement, the script utilizes API functions provided by CoppeliaSim. The function *'simxSetObjectPosition'* sends position information to CoppeliaSim, adjusting the sphere's position based on the mapped gaze angle. The adaptability of the MATLAB script, coupled with the versatility of CoppeliaSim, allowed the exploration of different scenarios, parameters, and experimental setups. This adaptability was important for tailoring the integration

to specific research objectives and user interactions.

In summary, the integration between OpenFace 2.0 gaze tracking and CoppeliaSim, facilitated by specific API functions, provided a platform for evaluating the feasibility and effectiveness of gaze-driven interactions in a controlled virtual environment. The robust features of both OpenFace 2.0 and CoppeliaSim, combined with API functions for precise control, contributed to create a dynamic and versatile experimental setup for gaze tracking applications.

3.3.2 Gaze-driven sphere control experiments

A series of experiments were conducted to further explore the versatility and feasibility of the integration between OpenFace 2.0 gaze tracking and CoppeliaSim. These experiments aimed to assess the capability of the system in controlling the simulated sphere's movement along specific axes, providing insights into the potential applications of gaze-driven interactions.

Unidirectional gaze control of x-axis
The initial experiment involved exclusively varying the x-coordinate of the simulated sphere based on the user's gaze. The unidirectional control experiment specifically focused on manipulating the sphere's position along the x-axis while maintaining the y and z coordinates fixed at zero. This approach provided valuable insights into the system's performance in gaze-driven planar movements, offering a controlled scenario for evaluating the precision and effectiveness of the gaze tracking system.
By focusing solely on the horizontal movement along the x-axis, the goal was to evaluate the accuracy and responsiveness of the gaze tracking system in a simplified scenario. The mapping between the x-axis of CoppeliaSim and the gaze x-axis involves establishing a relationship between the horizontal gaze direction (gaze_angle_x) and the movement of the sphere along the x-axis in the CoppeliaSim simulation.
It was configured to match changes in gaze direction from right to left (positive for gaze_angle_x) to the same movements of the sphere in CoppeliaSim from right to left along the x-axis. In other words, when the user shifted his gaze to the left, the sphere was expected to move to the left in the simulation, and vice versa. The mapping between gaze angles, specifically gaze_angle_x, and the x-coordinate of the sphere demonstrated promising results, showcasing the system's ability to translate gaze data into precise control.

Unidirectional gaze control of y-axis
Extending the experimentation from the success of the x-axis scenario, the subsequent phase focused on isolating the vertical movement along the y-axis. The y-coordinate of the simulated sphere was manipulated based on the user's gaze, with the y and

z coordinates held constant. This experiment aimed to validate the system's performance in a unidirectional vertical movement scenario, assessing its capability to precisely control the sphere's position along the y-axis. The mapping for the y-axis involved establishing a relationship between the vertical gaze direction (gaze_angle_y) and the corresponding movement of the sphere along the Y-axis in the CoppeliaSim simulation. In this configuration, a positive change in gaze direction (gaze_angle_y) indicated a downward movement of the sphere, while a negative change represented an upward movement.

Bidirectional gaze control, x and y-axes
Expanding the range of the experiments, the gaze-driven control was extended to both the x and y axes simultaneously. This configuration allowed the simulated sphere to move freely within the xy plane, responding to changes in both gaze_angle_x and gaze_angle_y. The successful execution of bidirectional control emphasized the integration's capability to manage multiple parameters concurrently.

3.4 Why OpenFace 2.0 and CoppeliaSim integration

To sum up, the integration of OpenFace 2.0 and CoppeliaSim provided a platform for real-time control of a simulated sphere based on gaze input. Motivated by the broader applications of gaze tracking, particularly in human-computer interaction and assistive technologies, the experiments aimed to assess the accuracy and responsiveness of gaze tracking in controlling simulated objects. The choice of a simulated sphere as the testing ground allowed to gauge the capabilities of gaze tracking and validate the viability of the concept before transitioning to more complex scenarios.

The iterative process involved the integration of OpenFace 2.0, camera calibration, and the development of a MATLAB script for real-time gaze tracking. This script, optimized for immediate data capture and processing, ensured efficient utilization of OpenFace 2.0's capabilities. The subsequent integration with CoppeliaSim enabled the translation of gaze data into meaningful actions within a simulated environment. The unidirectional and bidirectional gaze control experiments provided valuable insights into the system's performance, demonstrating its ability to precisely manipulate the simulated sphere along specified axes.

Looking ahead, this groundwork sets the stage for the next phase of this research leveraging gaze-driven control for interaction with a UR5 robot. The success of these initial experiments underlined the potential of gaze tracking as a reliable and intuitive input method for human-robot interaction scenarios.

Chapter 4

Collaborative robot UR5

The UR5 robotic arm was utilized for this research, therefore it can be very useful to provide an overview of its characteristics and its mathematical model.
The UR5 collaborative robot (Figure 4.1) from the Danish company Universal Robots is a part of the CB series, which also includes the UR3 and UR10 robots, respectively smaller and bigger. The nomenclature of the robots corresponds to their payload capacity in kilograms, specifically 3 kg, 5 kg, and 10 kg for UR3, UR5 and UR10, respectively.

Figure 4.1: UR5 robot from Universal Robot. Reprinted from [49]

Collaborative robots of Universal Robots offer numerous advantages, including:

- **Ease of programming:** programming becomes more accessible and intuitive thanks to the simplicity and effectiveness of the human-robot interface, comprised of a touch screen known as the teach pendant.

- **Flexibility:** collaborative robots are recognized for their straightforward programming and quick transition between different tasks. They are small and lightweight, and can be mounted on tabletops, desks, or carts for easy mobility.
- **Collaboration capability:** cobots are designed to share a workspace with humans, making automation easier.
- **Compatibility with various peripheral devices:** can integrate a diverse range of peripherals, including force sensors, cameras, and other customized tools. This allows for increased adaptability and versatility in industrial applications.
- **Quality:** performing repetitive tasks can be monotonous and lead to a decline in focus for technicians and students over extended periods of the same process. Cobots, boasting high repeatability, excel in executing intricate assembly tasks with precision and handling material loading and unloading consistently.

This combination of features makes Universal Robots robots an appealing choice in various contexts, offering user-friendly operation, operational flexibility, the ability to collaborate directly with human operators, and the option for customization through the addition of specialized peripherals.

The main components of the Universal Robots UR5 are the following:

- **Control unit:** contains the ESD board and all ports for cable connections.
- **Teach pendant:** used for programming the robot's movements, it is the interface between the robot and the operator.
- **UR5:** the robotic arm.

4.1 Control unit

The control unit (Figure 4.2) plays a crucial role in defining the robot's path and overseeing its movements. It consists of the following elements:

- **Central Processing Unit (CPU):** functions as the core intelligence of the robot, boasting Ethernet and USB connectivity.
- **Safety Management Module:** handles all incoming and outgoing signals within the control unit and facilitates connections to peripheral devices.
- **USB Hub:** encompasses the entire software, housing the Linux operating system, the Polyscope programming interface, and user-generated programs.

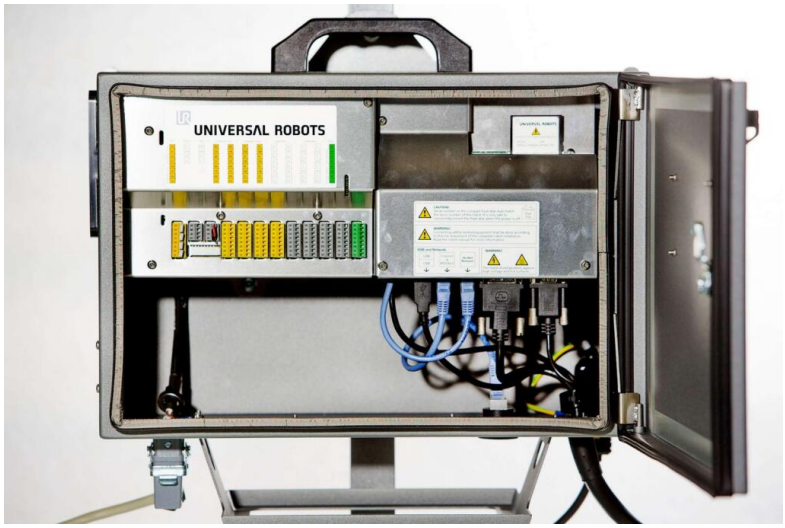


Figure 4.2: The control unit components of the UR5.

The control unit of the robot is responsible for managing connections and facilitating communication via TCP/IP (Transmission Control Protocol/Internet Protocol). The utilization of TCP enables reliable communication between the robot and other devices, supports remote control operations and allows the transmission of critical data. The connection between the two components is established through communication sockets, software abstractions designed to leverage standard and shared APIs for transmitting and receiving data over a network.
In this process, the application code of a given program accesses the communication channel through a designated port, executed at 125 Hz, facilitating the communication between processes operating on two physically separate machines. Furthermore, to ensure effective communication, it is necessary to specify a static IP address for the robot, which can be obtained in its settings section.

4.2 Teach pendant

The teach pendant, as illustrated in Figure 4.3, primarily consists of a 12” touchscreen hosting the programming user interface, PolyScope. It facilitates programming and commanding the robot through code input, as well as verifying the correct execution of programs, indeed if an error occurs, it is visually presented on the screen and categorized based on its nature.

Additionally, the teach pendant incorporates three physical buttons. The On/Off button is responsible for system power management, encompassing the functions of

turning the system on, off, and restarting it. In case of emergencies, the emergency button can be activated, triggering the robot brakes, and disconnecting the power supply for safety measures. At the rear of the teach pendant, the freedrive button allows the operator to manually move the robot to desired positions. It requires continuous pressure for its activation and proves particularly beneficial during the definition of waypoints.

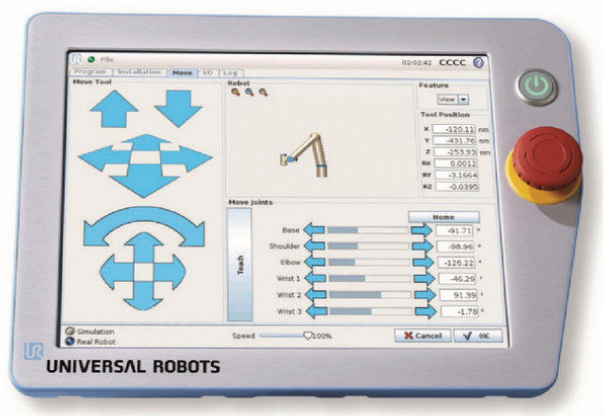


Figure 4.3: The teach pendant of the UR5 robot. Reprinted from [50]

PolyScope serves as the graphical programming interface accessible through the teach pendant. This Linux-based platform streamlines the programming process by providing an intuitive interface for users to input code, command the robot, and ensure the accurate execution of programs. The visual nature of PolyScope, characterized by icons and graphical representations, enhances the user experience, making it user-friendly even for those without extensive coding expertise.

PolyScope also supports a code-free programming approach, allowing users to program the collaborative robot using both basic and advanced commands simplifying the overall programming workflow. Among the basic commands "Move" introduces joint configuration movements. It can take the form of *'MoveJ'*, where the path is not crucial, and the joints reach the position simultaneously, representing the fastest movement. On the other hand, *'MoveL'* is employed for linear movements in the operational space. In *'MoveP'*, a linear movement with constant TCP speed can be achieved, and an option to incorporate circular trajectory mode is available.
Among the advanced commands are logical blocks, including *'if...else'*,*'loop'* or *'switch'* providing a more sophisticated level of control and automation in the programming process.

4.3 Robotic arm

The UR5 robot's mechanical structure consists of links and joints that form an open kinematic chain with six rotational joints connecting one link to another. The geometric configuration differs for each link, and their combination defines the positions and orientations achievable by the Tool Center Point (TCP). An End Effector (EE), such as a gripper or tool, can be attached to the Tool Center Point to carry out specific tasks or operations at the extremity of the robotic arm.

The UR5 is an anthropomorphic robot with a non-spherical wrist configuration, featuring six revolute joints that emulate the structure of the human arm. As illustrated in Figure 4.4, the first three joints are named *base*, *shoulder* and *elbow*; the base serves as the mounting point for the robot, while the shoulder and elbow joints rotate relative to the base. The subsequent three joints represent the wrist and are labelled *wrist 1*, *wrist 2* and *wrist 3*.

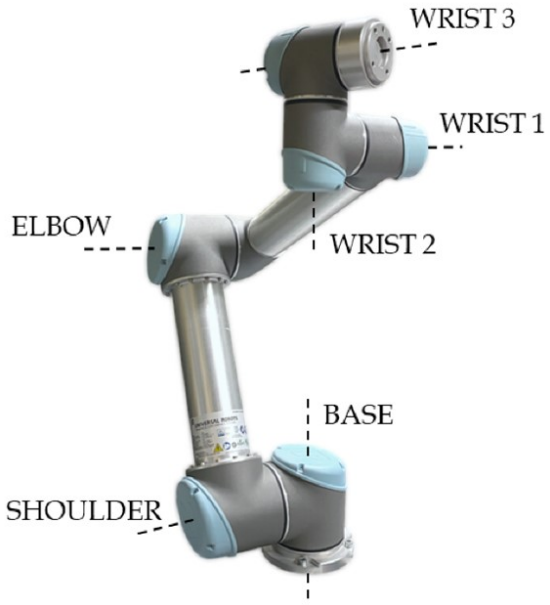


Figure 4.4: Graphical representation of the 6 joints of the collaborative robot UR5. Reprinted from [51]

Unlike many manipulators, the UR5 does not present a spherical wrist, thus all six joints play an active role to both the translational and rotational movements of its end effector [52].
A spherical wrist typically refers to a configuration where the three wrist joints axis intersect at a single point, allowing for a high degree of freedom and flexibility in orientation.

The technical specifications of the UR5 provided by Universal Robots in the user manual [49] are detailed in Table 4.1.

Table 4.1: Technicaal specification of UR5 robot.

Weight	18.4 kg
Maximum payload	5 kg
Reach	850 mm
Joint ranges	±360°
Joint max speed	180 °/s
TCP max speed	1 m/s
Repeatability	±0.1 mm
Degree of freedom	6 rotating joints
Communication	TCP/IP, Ethernet socket & Modbus TCP
Programming	Polyscope graphical user interface
IP classification	IP54
Power consumption	200 W
Temperature	0-50°C
Power supply	10-240 VAC, 0-50 Hz
Operating life	35.000 hours

Among the notable features, the repeatability of 0.1 mm stands out. Repeatability, in this context, refers to the robot's ability to consistently return to a specific position with a high degree of precision. Specifically, the robot can reliably reproduce the same movement or reach the same point in space with a maximum deviation of 0.1 mm from the intended position.

Another significant characteristic, typical of cobots, is its lightweight design of only 18.4 kg. This feature distinguishes the robot for its flexibility and ease to be mounted on both horizontal and vertical surfaces.

The *workspace* of a robot denotes the physical and geometric region where a robotic system can carry out tasks. It is defined by the allowable positions and orientations that the robot's end-effector can achieve, providing insights into the robot's capabilities and limitations.

The UR5's workspace is spherical, extending up to 850 mm from the robot's base. However, as specified in the UR5 manual [49], the optimal workspace is recommended to be limited to 750 mm.

Universal Robots, therefore, describes three distinct scenarios in which the robot may encounter challenges reaching a particular position, either because it is physically impossible for the robot to attain the required position or because it is not feasible to transition to that position from the robot's current joint configuration, the so-called singularities.
Singularities in the context of robotics refer to configurations where the mobility of the robot structure is reduced. They represent particular points within a robot's workspace where the robot loses one or more degrees of freedom, in fact, when the TCP of a robot moves into or near a singularity, it can result in the robot moving in an unexpected manner [53]. Singularities pose challenges to the smooth and predictable movement of the robot, requiring careful consideration and planning in robotic system design and operation.

The first scenario in which a singularity can be found, indicated by Universal Robots, occurs when the robot operates in the area beyond the recommended reach but still within the maximum working area (depicted in grey in Figure 4.5 a). While most positions can be reached, constraints on the tool orientation exist because the robot may physically be unable to reach far enough in certain situations.

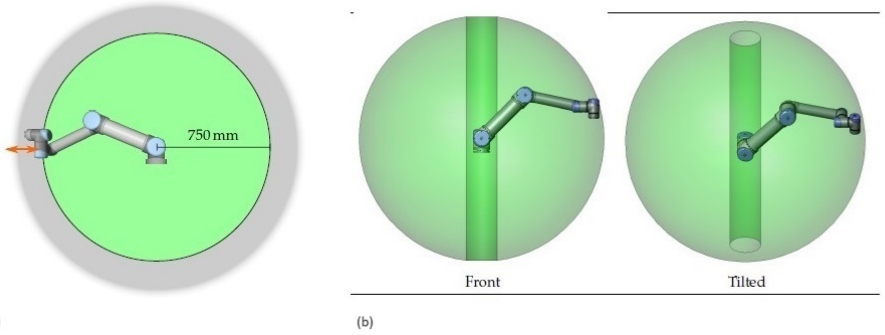


Figure 4.5: (a) Maximum, 850 mm, and recommended workspace, 750 mm, of UR5. **(b)** Cylindrical volume of 200 mm above and below the robot base. Moving the tool close to the cylindrical volume should be avoided. Reprinted from [49]

The second scenario advises against executing robot movements in the column directly above and below the robot base (Figure 4.5 b). This precaution is crucial because the arrangement of joints on the robot arm may render many positions and orientations physically unreachable. The proximity to this region can induce

rapid joint movements, even when the tool is moving slowly, leading to operational inefficiencies and complicating risk assessments.

The third configuration occurs when the shoulder, elbow, and wrist 1 joints rotate in the same plane. Aligning wrist joint 2 with this plane at 0° or 180° constrains the robot's range of movements.

4.4 Mathematical model and kinematics of the UR5 robot

Having explored the physical characteristics and capabilities of the UR5, it becomes crucial to delve into its mathematical model to fully understand how the robot moves in space. Direct kinematics plays a pivotal role in this analysis, as it allows the translation of joint angles into precise positions and orientations of the end effector.

As mentioned before, a robotic manipulator can be conceptualized as a kinematic chain, consisting of interconnected rigid bodies (links) linked by revolute or prismatic joints. The kinematic chain starts with one end constrained to a base, and the other end is equipped with an end-effector. The overall movement of the structure is achieved by combining the elementary motions of each link in relation to the previous one. To manipulate an object effectively in space, it is necessary to define the position and orientation of the end-effector. The primary objective of direct kinematics is to calculate the end-effector's pose based on the joint variables [53].

4.4.1 Denavit-Hartenberg convention

The Denavit-Hartenberg (D-H) convention provides a standardized approach to characterize the geometric configuration of robotic arms. This convention simplifies the mathematical representation of direct kinematics by assigning specific values to parameters such as link lengths, joint angles, and link offsets in a systematic manner.

In the D-H standard convention (Figure 4.6), the n joints are numbered from 1 to n, and links are numbered from 0 to n, starting from the base. The link *n+1* represents the Tool Center Point at the end of the robot.
Each joint is associated with a joint variable denoted as q_i, where i is the index of the joint connecting the *i-th* link to the next one. The rotation axis z_i is aligned with the axis of joint *i+1*.
For each link i, four parameters are defined: a_i, α_i, d_i and θ_i. These parameters are associated with the *i-th* link and joint and are commonly referred to as D-H parameters. They represent the kinematic characteristics of the *i-th* link in the manipulator:

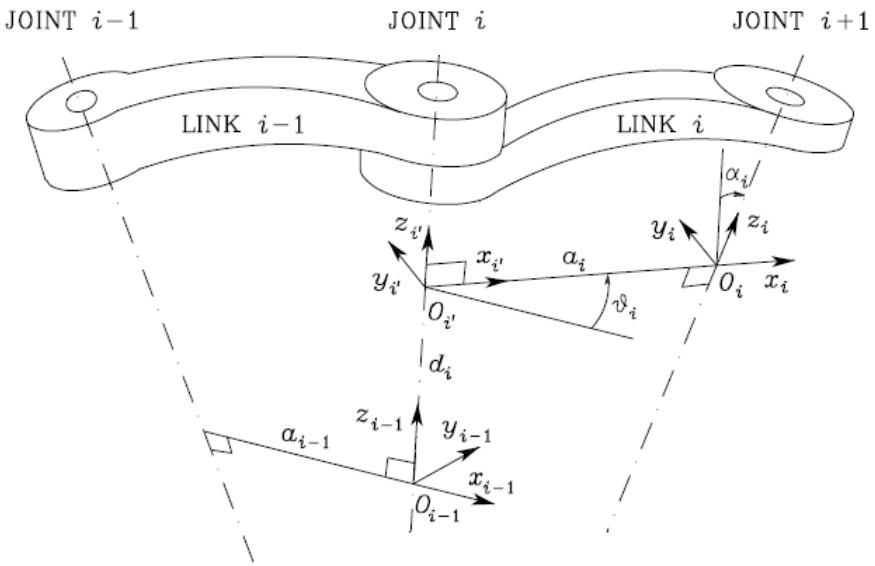


Figure 4.6: Standard Denavit Hartenberg convention representation. Reprinted from [53]

- **Translation distance d_i:** distance between axes x_i and x_{i-1} measured along the positive direction of z_{i-1}.
- **Offset distance a_i:** distance between axes z_i and z_{i-1} measured along the positive direction of x_i.
- **Twist angle α_i:** angle between axes z_{i-1} and z_i. It is the counterclockwise angle about the x_i, around which the z_{i-1} axis must rotate to align with the z_i.
- **Joint angle θ_i:** angle between axes x_{i-1} and x_i. It is the counterclockwise angle about the z_i, around which the x_{i-1} axis must rotate to align with the x_i.

Starting from the four D-H parameters, it is possible to construct the 4x4 rototranslation homogeneous matrix between the reference system *i-1* and *i*:

$$A_{i-1}^{i} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cdot\cos(\alpha_i) & \sin(\theta_i)\cdot\sin(\alpha_i) & a_i\cdot\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cdot\cos(\alpha_i) & -\cos(\theta_i)\cdot\sin(\alpha_i) & a_i\cdot\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

The Denavit-Hartenberg parameters provided by Universal Robots for the UR5 are presented in Table 4.2.

Table 4.2: D-H standard parameters of UR5 robot.

Joint	d_i [m]	a_i [m]	α_i [rad]	θ_i [rad]
Base	0.089159	0	$\pi/2$	q_1
Shoulder	0	-0.425	0	q_2
Elbow	0	-0.39225	0	q_3
Wrist 1	0.10915	0	$\pi/2$	q_4
Wrist 2	0.09465	0	$-\pi/2$	q_5
Wrist 3	0.0823	0	0	q_6

The variable q denotes the degrees of freedom of the robot arm, which vary based on the configuration of the robot arm at a specific moment in the trajectory.
With these parameters, the robot can be studied in any configuration, as they are solely dependent on the geometry and thus remain constant, except for the different θ values representing the individual degrees of freedom of the robot.

4.4.2 Inverse kinematics

While direct kinematics deals with determining the end-effector's pose from joint angles, inverse kinematics addresses the reverse problem, calculating the joint configuration required to achieve a specific end-effector pose. In the context of the UR5, this is essential for tasks where precise positioning of the robot's end effector is crucial.

4.4.3 Control law and Jacobian inverse

To execute complex tasks with the UR5, a control law based on the inverse kinematics algorithm was employed. This algorithm utilizes the Jacobian matrix and its inverse to find the joint velocities corresponding to desired end-effector velocities.
The Jacobian matrix J is a fundamental component in relating joint velocities $\dot{q}$ to end-effector velocities $\dot{x}$. For a robot with six revolute joints as the UR5, the Jacobian matrix J is constructed by concatenating the individual geometric Jacobians of each joint:

$$J = \left[J_1, J_2, J_3, J_4, J_5, J_6\right] \tag{4.2}$$

Each J_i is computed using the Equation 4.3:

$$J_i = \begin{bmatrix} z_{i-1} \times (p_e - p_{i-1}) \\ z_{i-1} \end{bmatrix} \tag{4.3}$$

where [53]:

- z_{i-1} is given by the third column of the rotation matrix R^0_{i-1}
- p_e is given by the first three elements of the fourth column of the transformation matrix A^0_e
- p_{i-1} is given by the first three elements of the fourth column of the transformation matrix A^0_{i-1}

The control law leverages the inverse of the Jacobian matrix J^{-1} to calculate joint velocities $\dot{q}$ corresponding to desired end-effector velocities $\dot{x}$. The relationship is expressed as:

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_n \end{bmatrix} = J^{-1} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix} \tag{4.4}$$

This allowed the robot to dynamically adjust its joint velocities in real-time, facilitating precise control of end-effector motion. The Jacobian inverse played a crucial role in achieving adaptability and responsiveness to varying operational requirements.

Chapter 5

Gaze-based control of UR5 robot

As outlined in Chapter 3, preliminary experiments were conducted in a controlled environment to assess the viability of gaze tracking using OpenFace 2.0 and its integration with CoppeliaSim. In the controlled setting, a simulated scenario was build up where a sphere within CoppeliaSim responded to gaze inputs, thus offering a simplified yet effective environment to evaluate the capabilities of the gaze tracking system.
The results obtained from the executed experiments were promising, demonstrating the successful translation of gaze data into precise control of the simulated sphere. Hence, the positive outcomes laid the groundwork for progressing from simulated objects to more intricate interactions with the UR5 robotic arm.

5.1 Control strategy overview

The gaze-based control algorithm described in Section 3.2 is seamlessly integrated into the external controller, implemented in MATLAB, which runs on a dedicated PC. This controller acted as the core processor for handling gaze data from the vision system and feedback from the UR5 controller.

The path planning algorithm employed adopts a methodology similar to that presented in previous cited articles [19] and [17]. While [19] and [17] consider as dynamic target the virtual hand, so an extension of the operator's hand in the xy-plane, in this approach the operator's point of focus was exploited as the dynamic target, indicating their gaze direction.

The UR5's workspace is defined as 1 m, corresponding to the maximum workspace of the UR5, 850 mm, plus the presence of the tool.
The control approach involved three spheres: the workspace sphere *Vw* with radius equal to 1m, the stopping volume *Vs* with radius *rif_stop* and the meeting sphere *Vm* with radius *rip_stop_rip*.
The *Vw* sphere, centered on the shoulder joint, triggered the robot's movement towards the target, initiating when the gaze point enters this 1m sphere. Specifically, this happen when the norm of the distance between the point of gaze and the position of the shoulder joint was less than 1 m.
The *Vs* sphere ensured that the robot stops when the TCP enters this volume around the target. This occurs precisely when the norm of the distance between the target and the TCP point is less than or equal to the radius rif_stop.
The *Vm* sphere enabled the robot to move again only when the TCP is in the stopping volume, and the target is outside this sphere.

The control loop represented in Figure 5.1, operated as follows: the gaze-following algorithm received as input the gaze information from the camera and feedback data from the UR5 controller. The whole code runs in 20 ms and the feedback data are sent from the robot with a frequency of 125 Hz through a TCP/IP network communication.

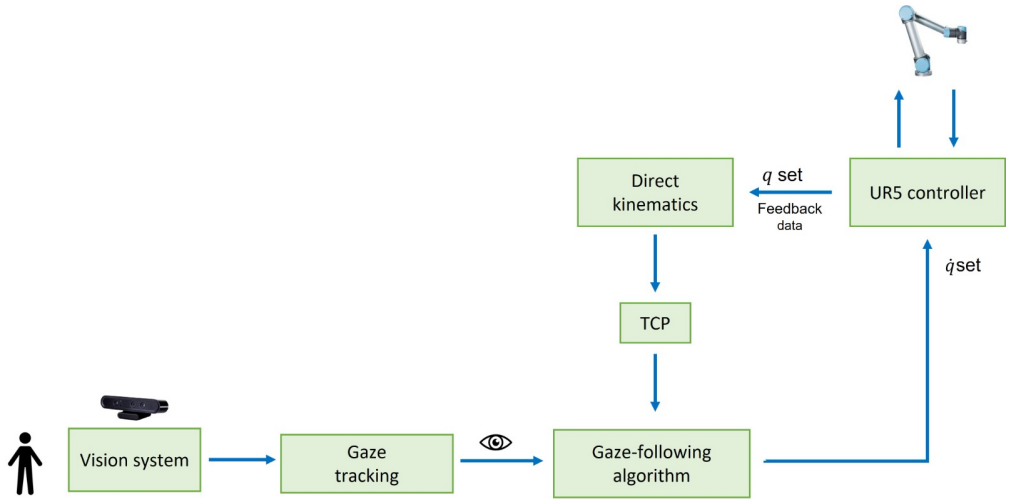


Figure 5.1: Scheme of the control loop adopted to manage the UR5 robot movements.

Utilizing the feedback data sent by the UR5, which includes the current joint positions, direct kinematics was applied to calculate the TCP position and orientation of the robot.
The TCP position and gaze point together served as inputs to the gaze-following algorithm for further processing. In fact, the position of the gaze target determined

the linear velocity of the TCP, calculated by considering the distance between the TCP and the target.
Moreover, as depicted in Figure 5.2, a smooth profile for the TCP linear velocity was adopted, allowing for a gradual and faster TCP velocity when the target was farther away, reaching a maximum velocity set to 0.35 m/s. The velocity then decelerated as the target approached, thus creating a smooth and responsive movement.

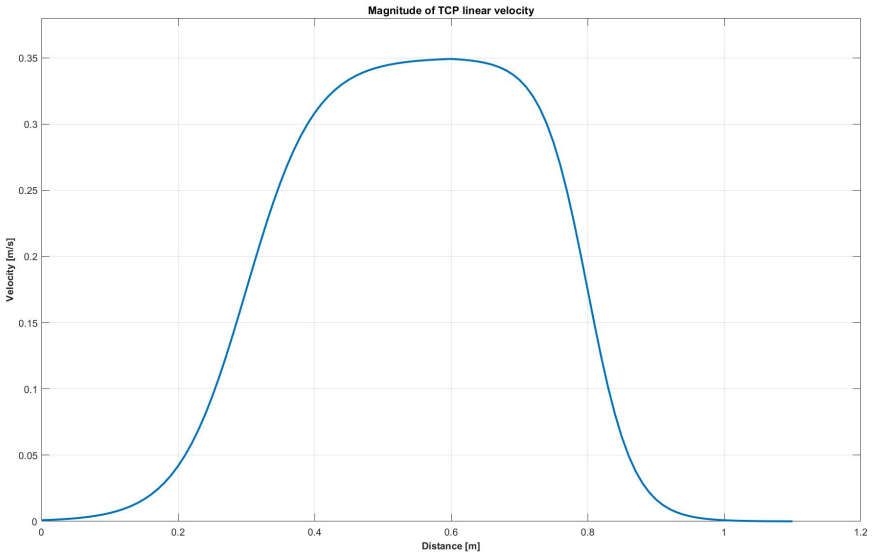


Figure 5.2: TCP linear velocity with respect to the distance between the gaze point and the TCP. The maximum TCP velocity was set to 0.35 m/s.

Additionally, the Jacobian matrix was computed using the positions information from the feedback data. Then, the joint velocities $\dot{q}$ were obtained by multiplying the inverse Jacobian with the operative space velocity vector, as expressed in Equation 5.1.

$$\dot{q} = J^{-1} \cdot [v_{TCP}; \omega_{TCP}] \tag{5.1}$$

The set of joint velocities $\dot{q}$ were then transmitted to the UR5 controller for movement execution.

5.2 Simulated test

The simulated test aimed to validate the effectiveness of the gaze-based control algorithm in a controlled environment before its application to the physical UR5

robot. The simulation setup integrates CoppeliaSim for visualization and interaction, in conjunction with URSim, a software tool that emulates Polyscope, the native software of the UR5 robot.

Within the simulated environment, the gaze-based control algorithm was tested. The scenario, represented in Figure 5.3, involved the UR5 robotic arm positioned on a virtual workbench, thus mirroring a real-world setting. The workspace, defined as a sphere with a radius of 1 meter, was centered on the shoulder joint, ensuring a confined testing area. In this outline, the virtual sphere represented the gaze point, and the main goal of the UR5 was to accurately follow it within the defined workspace.

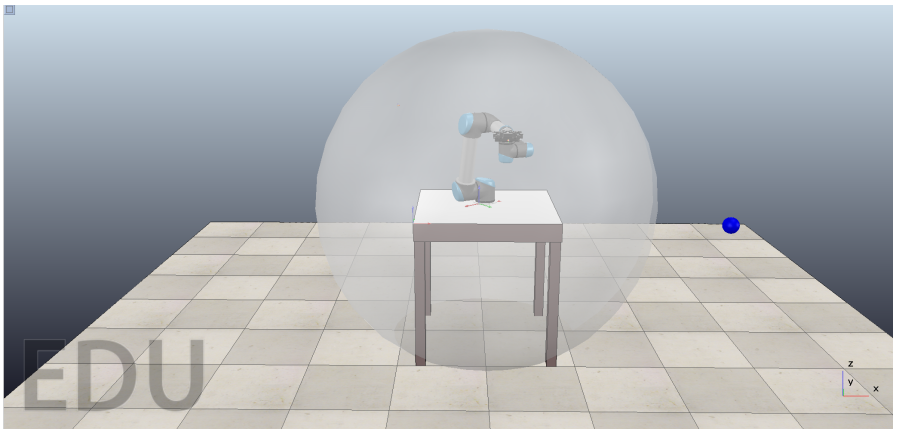


Figure 5.3: CoppeliaSim scene of robot UR5 positioned in the workbench in home configuration and its workspace area.

The Robotics Toolbox 10, developed by Peter Corke, was employed to model the UR5 robot and provided essential functions for creating a UR5 model, simulating its movements, and conducting several kinematic analyses.
Subsequently, the controller established a dual connection with both the UR5 simulator, URSim, through the IP address, and with CoppeliaSim through Remote API functions. To gain control over the simulated UR5 robot and the virtual sphere representing the gaze point, specific API functions provided by CoppeliaSim were used. The function *'simxGetObjectHandle'* was utilized to manage both the UR5 joints and the gaze point sphere.
Feedback data received from URSim, including joint positions of the simulated UR5, were transmitted to the simulated UR5 robot in CoppeliaSim by using the *'simxSetJointPosition'* function. Then the TCP position was calculated through direct kinematics with the Robotics Toolbox function *'ur5.fkine'*, that was used to assess the distance between the TCP and the gaze point sphere, considering predefined spheres as explained before in paragraph 5.1.

To simulate the movement of the gaze point in the virtual environment, the function 'simxSetObjectPosition' was employed to set the position of the gaze point sphere in CoppeliaSim.

Building on the successful simulation experiments described in Section 3.3.3, the same tests were performed to validate the gaze-based control algorithm with the UR5 robotic arm. The objective was to manipulate the UR5 by moving a simulated sphere through the gaze, initially controlled only along the x-axis, then the y-axis, and subsequently in the xy-plane. These tests aimed to assess the system's performance before the transition to physical trials with the actual UR5 robotic arm.

Control along x-axis:
In this test, the UR5 was observed to accurately reach the specified point indicated by the user's horizontal gaze direction (gaze_angle_x). The simulation results showcased the precise translation of gaze data into controlled motion along the x-axis in CoppeliaSim. The UR5 exhibited responsiveness in following a unidirectional path, reflecting the effectiveness of the gaze-based control system.
Control along y-axis:
Building upon the success of the x-axis scenario, the focus was shifted to vertical movement along the y-axis. The UR5 successfully translated changes in vertical gaze direction (gaze_angle_y) into movements along the y-axis of the sphere in CoppeliaSim. This test aimed to validate the system's performance in scenarios involving unidirectional vertical movements.
Control along xy plane:
In this test, the UR5 presented its ability to reach the gaze-directed target point within the xy plane. By setting a predefined value for the z-coordinate, the UR5 translated gaze data into motion, allowing it to span the entire xy plane. This test represented a comprehensive scenario where the UR5 responded dynamically to gaze input in a two-dimensional space.

With the successful demonstration of gaze-based control in simulated environments, the next phase will involve the transition to physical trials with the actual UR5 robotic arm.
Indeed, the subsequent physical trials will aim to validate and refine the system's performance in diverse real-world settings, in order to better confirm its robustness and precision in guiding the UR5.

5.3 Experimental validation with real UR5 robot

To validate the effectiveness of the gaze-based control strategy, experimental tests have been carried out.

Experimental setup

The experimental setup included a PC, the UR5 robot, the Astra Orbbec camera and a router. The PC served as the external robot controller, managing both the data acquired from the camera and the feedback received from the UR5 controller, as well as sending commands to the UR5.
The base of the UR5 robot was fixed on a workbench, the camera was placed in front of the manipulator at a distance of 1.3 m from the robot's base. The tests were carried out considering the operator sitting at a distance of 0.8 m from the camera.

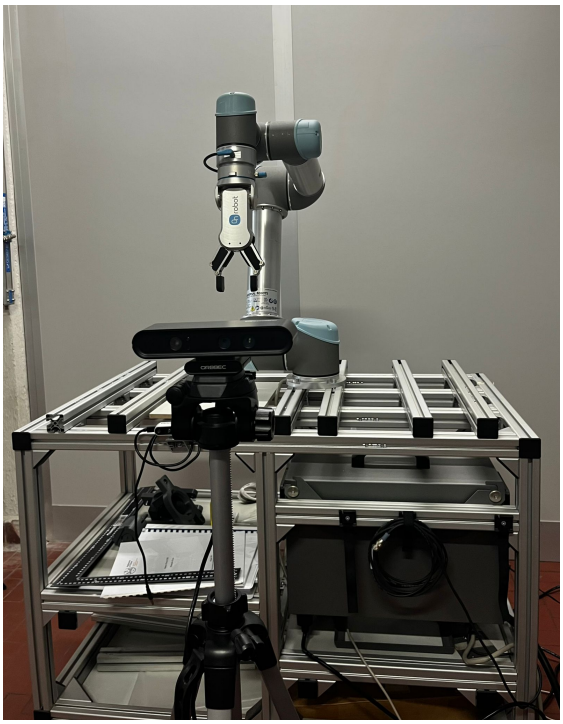
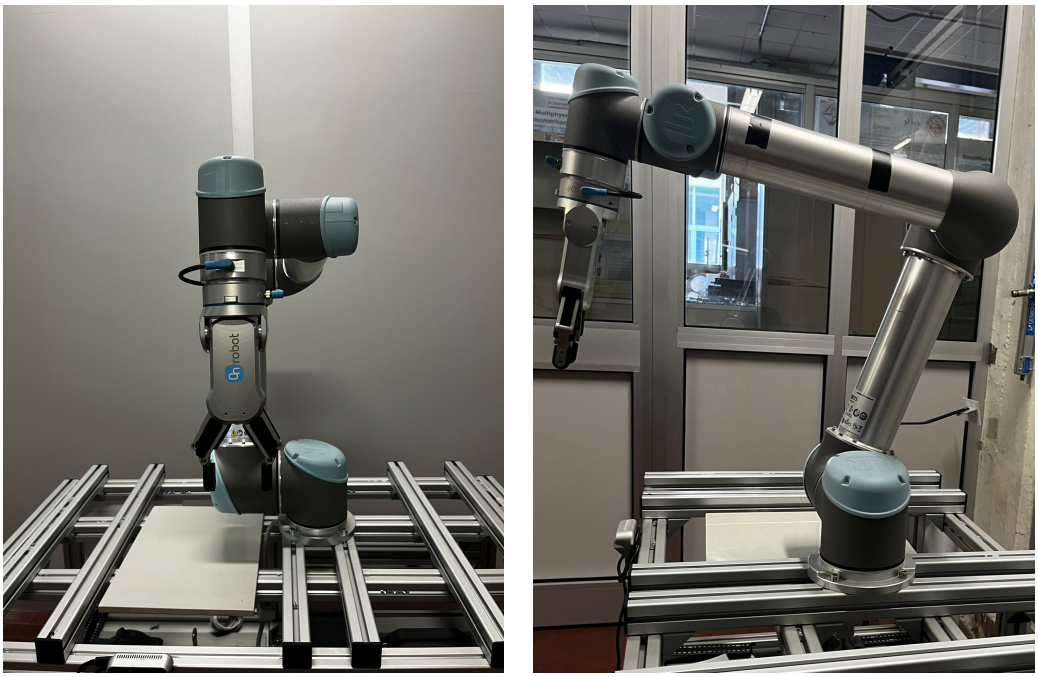


Figure 5.4: Experimental setup: UR5 manipulator fixed in the workbench and the Astra Orbbec camera in front of it

Through the camera, the scene was captured, and facial features of the operator, including gaze direction, were tracked though the gaze tracking software OpenFace 2.0. These information were processed on the PC using an algorithm running in the MATLAB environment. It's worth noting that was made the decision to control the robot's motion exclusively in the xy plane. Therefore, the gaze point was calculated by considering its projection into the xy plane, representing both horizontal and vertical displacements; as part of this decision, the gaze point on the z-axis was fixed at a constant value.

The external controller also received feedback data from the UR5 controller. Consequently, this feedback data, along with the gaze position, served as input for the gaze-based control, generating a set of joint velocities for the UR5, which were then sent to the UR5 controller.

The UR5 was initially in the predefined home configuration, with joint angles set equal to [-180°, -70°, -100°, -90°, 90°, 0°] (Figure 5.5).



(a) **(b)**

Figure 5.5: Home position configuration set for the UR5 robot. **(a)** Frontal view **(b)** Lateral view

When the gaze point approches the UR5's workspace, the manipulator begin to move and to track the target by shifting the TCP to the indicated point. The robot stops when the distance between the TCP and the gaze point reaches 40 mm. If the gaze point moves outside the UR5's workspace sphere, the UR5 returns to the home configuration.

Three types of tests were conducted in the experiment. In the first test, the original gaze signal was considered. In the second and third tests, the signal was filtered by using the 'moving mean filter', that is a type of filter used to reduce noise or random variations in time-series data. It calculates the mean of a certain number of consecutive data points in a moving window and uses this mean as the filtered value.

The moving window shifts along the time series, and the mean is constantly updated. In the tests, the gaze trajectory involved tracing a rectangle.

- ***Test a): original gaze signal***
  The first test focused on the original gaze signal. Considering the trajectory traced by the gaze and that of the robot (Figure 5.6), it was observed that the gaze point was tracked by the robot until it became stationary. At that point, the point of gaze was progressively approached by the robot, up to the pre-defined stop distance. Additionally, from the trajectory, the presence of noise in the gaze signal was observed.

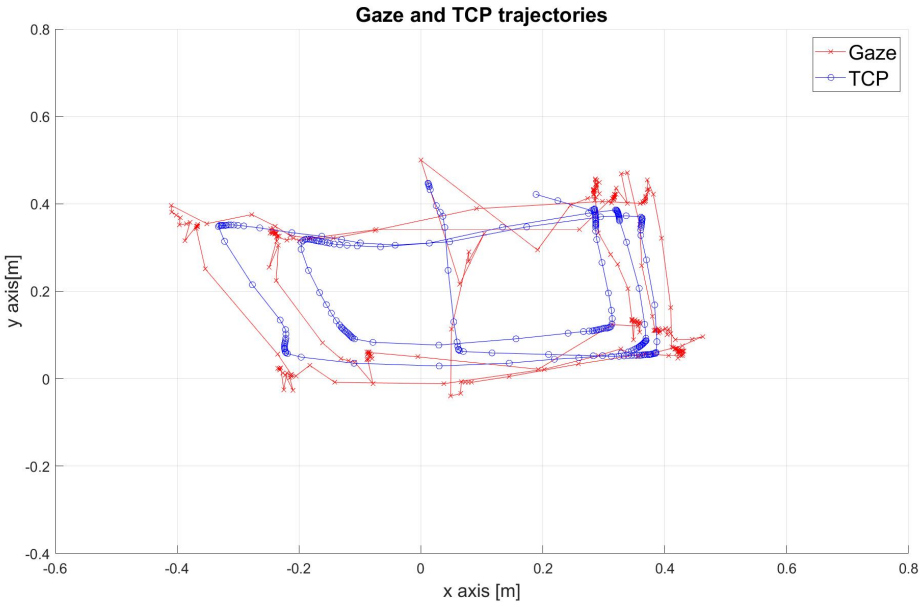


Figure 5.6: Original gaze signal case. Trajectories of gaze (in red) and of TCP (in blue) in xy plane.

Further analisys involved a detailed examination of two separate time-based graphs for x and y, (Figure 5.7), revealing pronounced noise, particularly along the y-axis. Additionally, an average delay of approximately 1 second was observed.

Upon closer examination of the x and y graphs, it was possible to observe the trajectory of the traced rectangle. Specifically, constant segments of x corresponded to varying segments of y, and vice versa, where constant segments of y matched varying segments of x. For instance, from 95.7 s to 116.7 s, x has remained approximately constant while y exhibited variation.

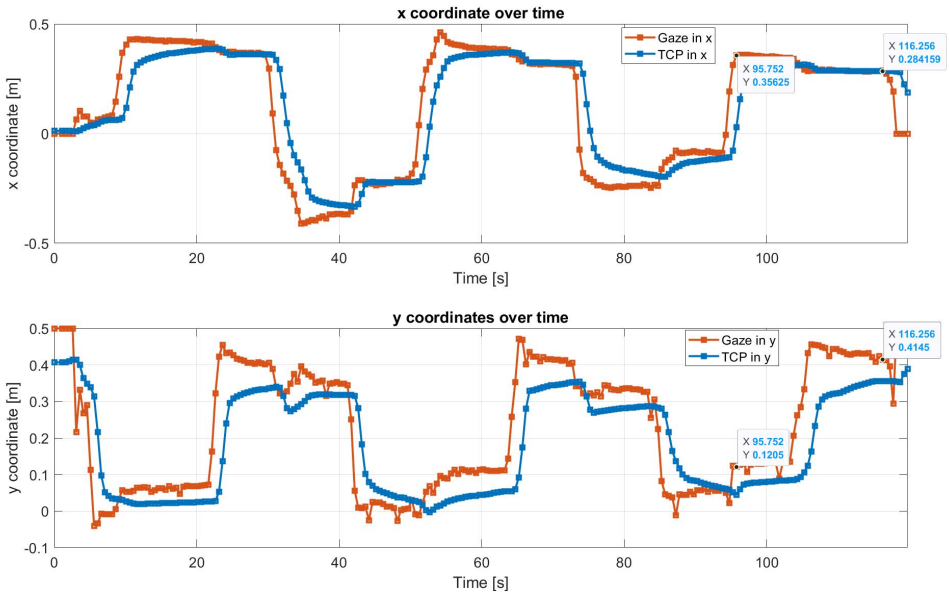


Figure 5.7: Original gaze signal case. x and y trajectories over time. In red the gaze trajectory and in blue the TCP trajectory.

From the linear velocity profile of the TCP represented in Figure 5.8, it was observed, as expected, that the velocity is higher when a greater distance between the gaze and the TCP is recorded. It decreases as it approaches the target, reaching zero velocity at the stop distance.

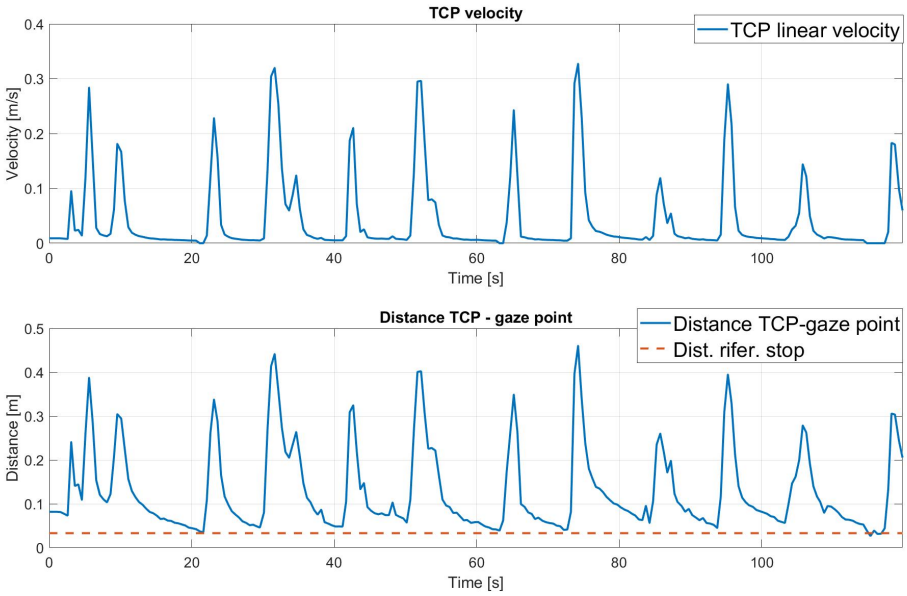


Figure 5.8: Original gaze signal case. Norm of the TCP linear velocity over time and related distance gaze point-TCP.

- ***Test b): filtered gaze signal (filter size 5)***
  In the second test the gaze signal was filtered by using the moving mean filter with a window of size 5. Examining the trajectory depicted in Figure 5.9, it is evident that the robot effectively tracked the gaze point until it came to a stop. Subsequently, the robot gradually approached the gaze point, reaching the predefined stop distance. It is noteworthy that, in contrast to the original gaze test, a less noisy gaze signal was observed in this test.

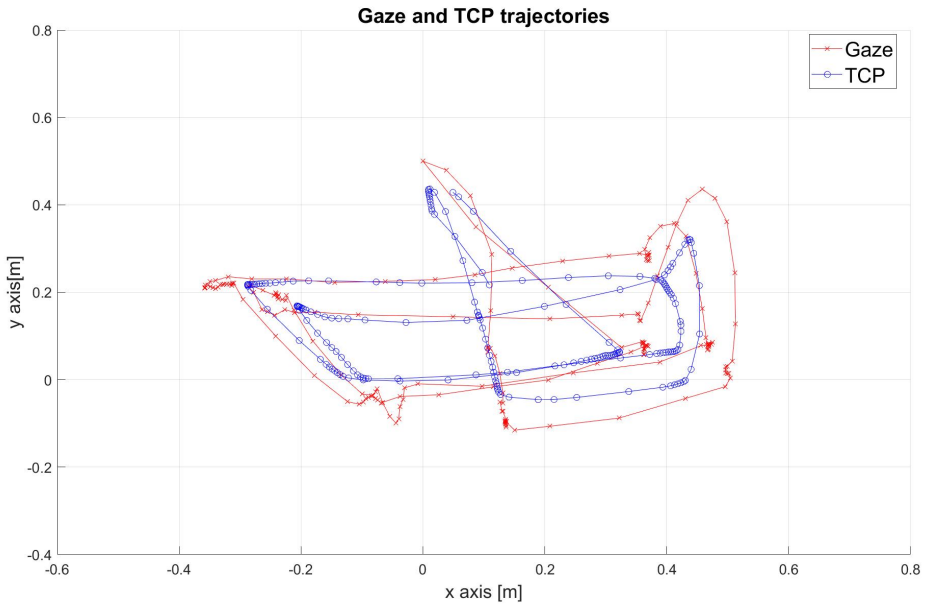


Figure 5.9: Filtered gaze signal case (filter size 5). Trajectories of gaze (in red) and of TCP (in blue) in xy plane.

In-depth analysis was extended to a meticulous examination of separate time-based trajectories for x and y, illustrated in Figure 5.10. The findings revealed a notably less noisy signal, with x showing greater improvement. Delving into the rectangle's trajectory, a distinctive symmetry emerged between x and y. Specifically, when x held constant, y exhibited variation, and vice versa. Moreover, attributable to the filtering process, a computed time delay of approximately 3 seconds was observed between the gaze and the TCP.

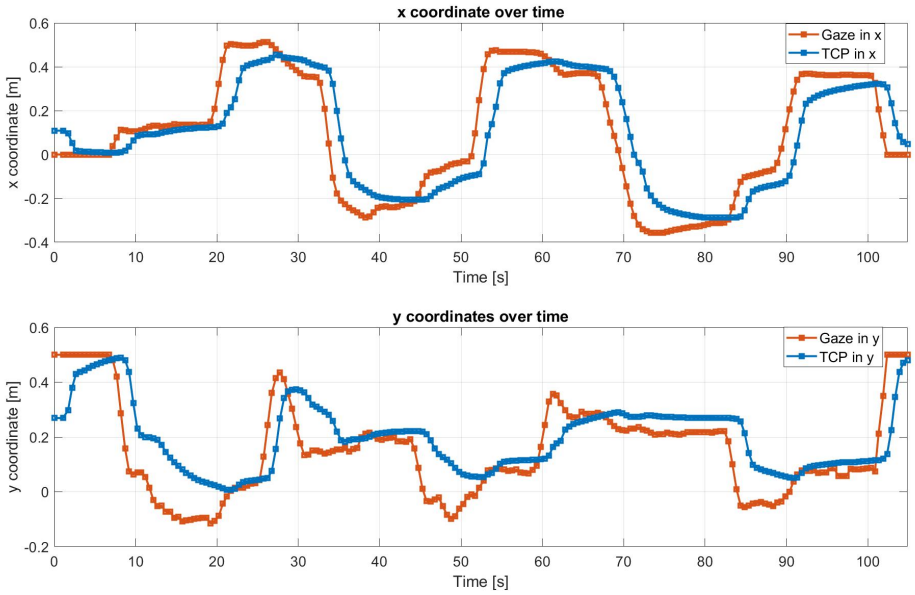


Figure 5.10: Filtered gaze signal case (filter size 5) of x and y trajectories over time. In red the gaze trajectory and in blue the TCP trajectory.

Examining the linear velocity profile of the TCP illustrated in Figure 5.11, as expected, it was observed that the velocity is higher when a larger distance between the gaze and the TCP is registered. It progressively diminishes as it approaches the target, culminating in zero velocity at the predetermined stop distance.

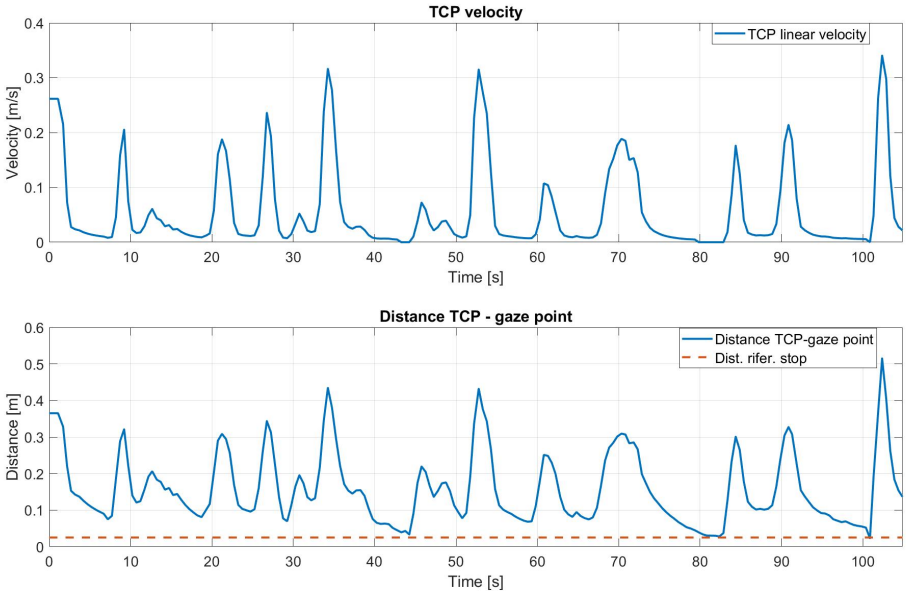


Figure 5.11: Filtered gaze signal case (filter size 5).Norm of the TCP linear velocity over time and related distance gaze point-TCP.

- ***Test c): filtered gaze signal (filter size 10)***
  Afterward, the gaze signal was further examined using a moving mean filter with a larger window size, specifically equal to 10. Analyzing the trajectories on the xy plane (Figure 5.12), it was observed that the robot faithfully tracks the gaze path. Moreover, the xy trajectory graph revealed a less noisy gaze signal.

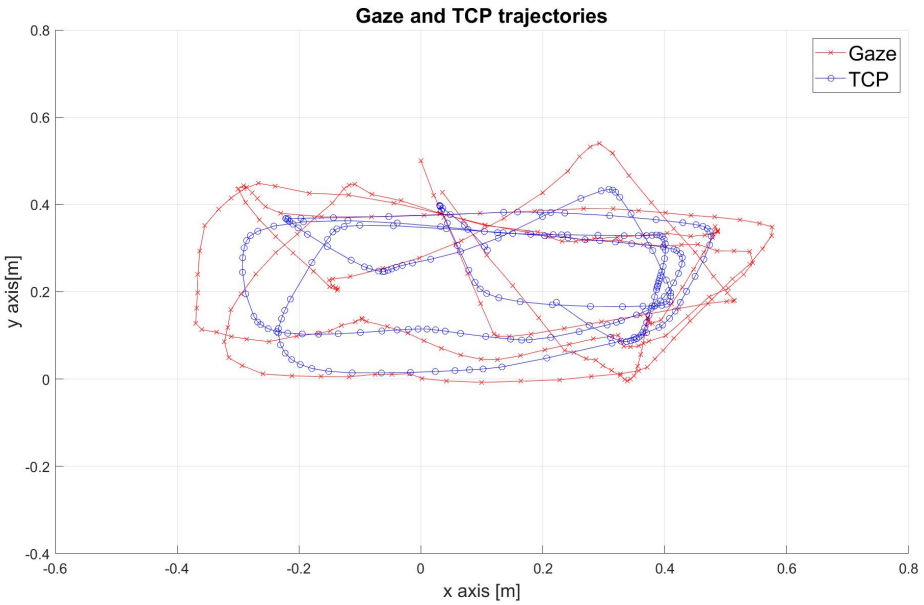


Figure 5.12: Filtered gaze signal case (filter size 10).Trajectories of gaze (in red), and of TCP (in blue) in xy plane.

Upon closer inspection, temporal graphs for x and y were assessed individually, as depicted in Figure 5.13. Due to the use of the wider filter, a delay of approximately 4 seconds was recorded. Moreover, due to the filter, it was also observed that some points, especially on the y-axis, (for instance at 29 s), were not consistently followed by the robot, transitioning to the subsequent ones.

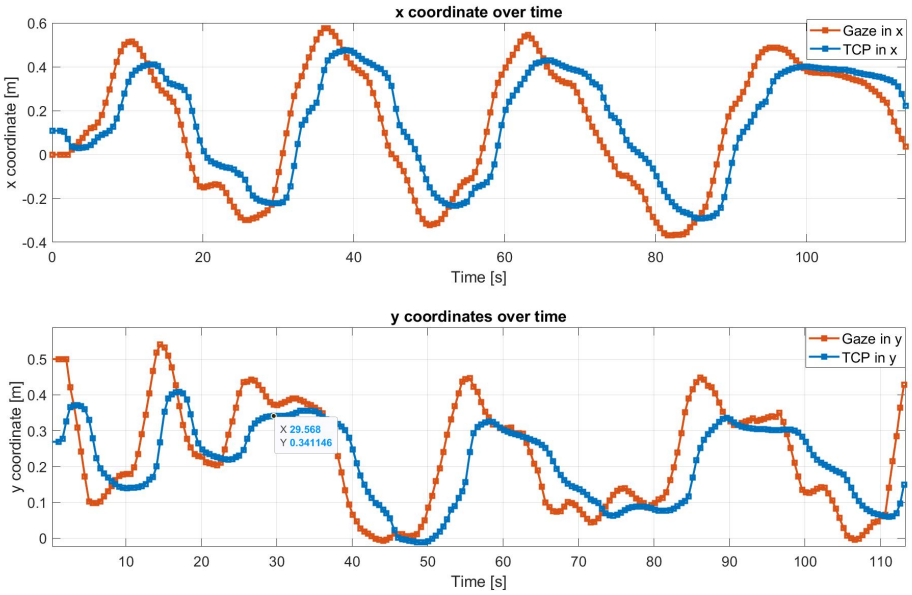


Figure 5.13: Filtered gaze signal case (filter size 10) of x and y trajectories over time. In red the gaze trajectory and in blue the TCP trajectory.

Among the three conducted tests, the filter with a window size of 10 emerged as the least favorable option. If it rendered the gaze signal noise-free, providing a smooth signal, it also introduced a significant delay, thus, causing the robot to deviate from perfect gaze tracking.
Actually, considering the requirement for real-time application, the use of the original gaze appeared to be the optimal choice, surpassing the option with a filter size of 5. In fact, from an accurate analysis of the obtained graphs, it seems that the TCP followed the gaze more accurately compared to the other two scenarios.
From a collaborative standpoint as well, the approach with the original gaze signal was preferred because of its heightened responsiveness to gaze movements.

Chapter 6

Conclusions and future developments

In this work, the control of the UR5 robot through gaze tracking was successfully implemented.
To reach this aim, Openface 2.0, a freely available software on Github useful for tracking facial features,was chosen for gaze tracking. The gaze tracking code was developed in the MATLAB environment, utilizing the Openface 2.0 software to track gaze and extract necessary information.
The feasibility of the gaze tracking code was initially assessed for a simple application, by using the CoppeliaSim simulation platform. For instance, the implemented algorithm was exploited to control the movement of a sphere on a plane.
To guarantee the robustness of the code, several tests were conducted: firstly the sphere's movement was controlled only along the horizontal axis or along the vertical one; then a comprehensive test was carried out, by involving the control of the sphere's movement across the entire xy plane, in order to map the sphere's motion to the user's gaze direction.

This systematic testing approach in the simulated environment aimed to validate the effectiveness of the gaze tracking code. By assessing the sphere's response to different gaze directions and movements, the study ensured the reliability of the gaze tracking algorithm before its application to more complex tasks, such as the control of the UR5 robot. Therefore, these tests effectively demonstrated the algorithm's effectiveness and reliability, providing a solid foundation for the subsequent transition to the core goal of the study: controlling the UR5 robot through user's gaze.

The experimental setup for the control of the UR5 robot included a PC as an external controller, the UR5 robot, a router, and the Astra Orbbec camera for data acquisition. Within the external controller, the gaze following algorithm code ran in 20 ms in

MATLAB environment; the algorithm used is a combination of the gaze tracking code, responsible for the acquisition and manipulation of gaze data, and the path-following algorithm. The latter shares a logic similar to that implemented in [19], but with the innovation of using gaze as the dynamic target for controlling the UR5 robot's movements along the xy plane.
Three tests were conducted to validate the efficacy of the gaze tracking following-algorithm applied to the UR5 robot. The first test considered the unfiltered gaze signal, while the other two tests employed a moving average filter with window sizes of 5 and 10, respectively. The results showed that the robot successfully followed the user's gaze, with a delay of 1 s in the first case, 3 s in the second, and 4 s in the third.

Among the three tests conducted, the filter with a window size of 10 was found to be the least favourable option. Despite effectively eliminating gaze signal noise and producing a smooth signal, it introduced a significant delay, leading to deviations from accurate gaze tracking. Considering real-time application as the key discriminating factor, the unfiltered gaze signal emerged as the optimal choice, also surpassing the performance of the filter with a size of 5.

Although encouraging results have been achieved in this thesis work, several solutions could be implemented in the future, in order to make gaze tracking as a cutting-edge method for robot control, useful in different sectors of the industry.
Hence, in terms of future developments, there is undoubtedly the need to allow the control of the UR5 along the depth of the defined workspace.
Additionally, another key point should be the enhancement of the gaze tracking algorithm in order to reduce the latency and to improve the robot's accuracy in executing more precise movements.
Finally, given the numerous studies that confirm the improvements in gaze tracking for hand-over tasks, a potential development could involve the integration of the gaze tracking code to enhance performance in other tasks, such as hand-overs, thus aiming for faster and more natural human-robot interaction.

Bibliography

[1] Maoudj Hentout Mustapha. «Human–robot interaction in industrial collaborative robotics: a literature review of the decade 2008–2017». In: *Advanced Robotics* 33(15-16) (2019), pp. 764–799 (cit. on pp. 1, 2, 10–12).

[2] Vicentini. «Collaborative robotics: a survey». In: *Journal of Mechanical Design* 143 (2021) (cit. on p. 1).

[3] Li Yang Zhou. «Collaborative robot dynamics with physical human–robot interaction and parameter identification with PINN». In: *Mechanism and Machine Theory* 189 (2023) (cit. on p. 2).

[4] A. Verl J. Krüger T.K. Lien. «Cooperation of human and machines in assembly lines». In: *CIRP Annals* 58(2) (2009), pp. 628–646 (cit. on pp. 2, 8).

[5] *Robot-assisted rehabilitation – ROBERT® and KUKA facilitate mobilization*. URL: `https://www.kuka.com/en-us/industries/solutions-database/2019/08/robert-from-life-science-robotics`. (accessed: 13.11.2023) (cit. on p. 2).

[6] Palmieri Chiriatti Bottiglione. «Manipulability optimization of a rehabilitative collaborative robotic system». In: *Machines* 10(6) (2022), p. 452 (cit. on p. 2).

[7] Singh Rab Javaid Haleem. «Significant applications of Cobots in the field of manufacturing». In: *Cognitive Robotics* 2 (2022), pp. 222–233 (cit. on pp. 2, 3, 5).

[8] *Mobile Cobots - Dimalog*. URL: `https://www.dimalog.com/mobile-cobots/`. (accessed: 13.11.2023) (cit. on p. 3).

[9] Fantuzzi Secchi Levratti De Vuono. «TIREBOT: A novel tire workshop assistant robot». In: *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)* (2016), pp. 733–738 (cit. on p. 3).

[10] Karna Vishnu Vardhana Elamvazuthi Borboni Alberto Reddy. «The Expanding Role of Artificial Intelligence in Collaborative Robots for Industrial Applications: A Systematic Review of Recent Works». In: *Machines* 11(1) (2023), pp. 2075–1702 (cit. on pp. 4–6).

[11] Fahad Sherwani and B.S.K.K. Ibrahim Muhammad Mujtaba Asad. «Collaborative Robots and Industrial Revolution 4.0 (IR 4.0)». In: *2020 International Conference on Emerging Trends in Smart Technologies (ICETST)* (2020), pp. 1–5 (cit. on pp. 4, 5).

[12] Vasu Srinadh Patil Swapnil. «Advances and perspectives in collaborative robotics: a review of key technologies and emerging trends». In: *Discover Mechanical Engineering* 2(1) (2023), p. 13 (cit. on p. 4).

[13] Pellegrini Taesi Aggogeri. «COBOT Applications—Recent Advances and Challenges». In: *Robotics* 12 (2023), p. 79 (cit. on pp. 5, 7).

[14] H. Kress-Gazit A. Kshirsagar and G. Hoffman. «Specifying and Synthesizing Human-Robot Handovers». In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2019), pp. 5930–5936 (cit. on pp. 8, 9).

[15] Hoffman Edan Faibish Kshirsagar. «Human preferences for robot eye gaze in human-to-robot handovers». In: *International Journal of Social Robotics* 14 (2022), pp. 995–1012 (cit. on p. 8).

[16] R. Alami J. Waldhart M. Gharbi. «Planning handovers involving humans and robots in constrained environment». In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2015), pp. 6473–6478 (cit. on p. 9).

[17] Mauro Melchiorre Scimmi. «Vision-based control architecture for human–robot hand-over applications». In: *Asian Journal of Control* (2020) (cit. on pp. 9, 10, 47).

[18] Chen Zhang Wang Li Diekel. «Controlling object hand-over in human–robot collaboration via natural wearable sensing». In: *IEEE Transactions on Human-Machine Systems* 49 (2018), pp. 59–71 (cit. on p. 9).

[19] Mauro Scimmi Melchiorre. «Experimental Real-Time Setup for Vision Driven Hand-Over with a Collaborative Robot». In: *2019 International Conference on Control, Automation and Diagnosis (ICCAD)* (2019), pp. 1–5 (cit. on pp. 9, 10, 15, 47, 63).

[20] Forlizzi Strabala Lee Dragan. «Toward seamless human-robot handovers». In: *Journal of Human-Robot Interaction* 2 (2013), pp. 112–132 (cit. on p. 10).

[21] Pardi Ortenzi Cosgun. «Object Handovers: A Review for Robotics». In: *IEEE Transactions on Robotics* 37(6) (2021), pp. 1855–1873 (cit. on pp. 10, 12).

[22] Gleeson Moon Troniak. «Meet me where i'm gazing: how shared attention gaze affects human-robot handover timing». In: *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction* (2014), pp. 334–341 (cit. on pp. 10, 12, 13).

[23] Santos Castro Silva. «Trends of Human-Robot Collaboration in Industry Contexts: Handover, Learning, and Metrics». In: *Sensors* 21 (2021), p. 4113 (cit. on p. 11).

[24] Li Niculescu Banchs. «Why Industrial Robots Should Become More Social: On the Design of a Natural Language Interface for an Interactive Robot Welder». In: *International Conference on Social Robotics* (2014), pp. 276–278 (cit. on p. 11).

[25] Robertson Barattini Morand. «A proposed gesture set for the control of industrial collaborative robots». In: *012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication* (2012), pp. 132–137 (cit. on p. 12).

[26] Kirstein Fischer Jensen. «The effects of social gaze in human-robot collaborative assembly». In: *Social Robotics: 7th International Conference, ICSR* (2015), pp. 204–213 (cit. on p. 12).

[27] Christian Kshirsagar Lim. «Robot gaze behaviors in human-to-robot handovers». In: *IEEE Robotics and Automation Letters* 5 (202), pp. 6552–6558 (cit. on pp. 12, 13).

[28] Pipe Grigore Eder. «Joint action understanding improves robot-to-human object handover». In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2013), pp. 4622–4629 (cit. on pp. 12, 13).

[29] Clodic Gharbi Paubel. «Toward a better understanding of the communication cues involved in a human-robot object transfer». In: *24th IEEE international symposium on robot and human interactive communication (RO-MAN)* (2015), pp. 319–324 (cit. on pp. 12, 13).

[30] Yang Liu Chi. «In the eye of the beholder: A survey of gaze tracking techniques». In: *Pattern Recognition* (2022) (cit. on pp. 17, 19, 20).

[31] Qiang Ji Kang Wang. «Real time eye gaze tracking with kinect». In: *2016 23rd International Conference on Pattern Recognition (ICPR)* (2016), pp. 2752–2757 (cit. on p. 17).

[32] Ji Qiang Hansen Dan Witzner. «In the eye of the beholder: A survey of models for eyes and gaze». In: *IEEE transactions on pattern analysis and machine intelligence* 32 (2009), pp. 478–500 (cit. on pp. 17, 20–22).

[33] Marcio R.M. Mimica Carlos H. Morimoto. «Eye gaze tracking techniques for interactive applications». In: *Computer Vision and Image Understanding* 98(1) (2005), pp. 4–24 (cit. on pp. 17, 19).

[34] Wang Shehu Shehi. «Remote Eye Gaze Tracking Research: A Comparative Evaluation on Past and Recent Progress». In: *Electronics* 10 (2021), pp. 2079–9292 (cit. on pp. 18–21).

[35] Hiroyuki Sogo. «GazeParser: an open-source and multiplatform library for low-cost eye tracking and analysis». In: *Behavior Research Methods* 45(3) (2013), pp. 684–695 (cit. on p. 22).

[36] Bulling Wood Erroll. «Eyetab: Model-based gaze estimation on unmodified tablet computers». In: *proceedings of the symposium on eye tracking research and applications* (2014), pp. 207–210 (cit. on p. 22).

[37] Emil-Mari Nel. *Opengazer: open-source gaze tracker for ordinary webcams*. URL: https://www.inference.org.uk/opengazer/. (accessed: 13.11.2023) (cit. on p. 22).

[38] *openEyes – Eye tracking systems used for casino gaming, paysafecard payments and more*. URL: https://thirtysixthspan.com/openEyes/. (accessed: 13.11.2023) (cit. on p. 22).

[39] *Gaze Tracker - Gaze Tracking Library*. URL: https://github.com/devinbarry/GazeTracker. (accessed: 13.11.2023) (cit. on p. 23).

[40] Antoine Lamé. *Gaze Tracker - webcam-based eye tracking system*. URL: https://github.com/antoinelame/GazeTracking. (accessed: 13.11.2023) (cit. on p. 23).

[41] Yao Chong Lim Tadas Baltrušaitis Amir Zadeh. «OpenFace 2.0: Facial Behavior Analysis Toolkit». In: *IEEE International Conference on Automatic Face and Gesture Recognition* (2018) (cit. on pp. 23, 25, 26).

[42] Tadas Baltrušaitis. *OpenFace 2.2.0: a facial behavior analysis toolkit*. URL: https://github.com/TadasBaltrusaitis/OpenFace. (accessed: 13.11.2023) (cit. on pp. 23, 26).

[43] *GazeSense - Eye Tracking Software for Webcams & 3D Sensor*. URL: https://eyeware.tech/gazesense-eye-tracking-software-for-webcams-3d-sensor/. (accessed: 13.11.2023) (cit. on p. 23).

[44] *Tobii*. URL: https://www.tobii.com/. (accessed: 13.11.2023) (cit. on p. 23).

[45] *Tobii Pro Glasses 3*. URL: https://www.tobii.com/products/eye-trackers/wearables/tobii-pro-glasses-3. (accessed: 13.11.2023) (cit. on p. 23).

[46] *Tobii Pro Spark*. URL: https://www.tobii.com/products/eye-trackers/screen-based/tobii-pro-spark#overview. (accessed: 13.11.2023) (cit. on p. 23).

[47] Freese Rohmer Singh. «V-REP: A versatile and scalable robot simulation framework». In: *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2013), pp. 1321–1326 (cit. on p. 31).

[48] *Coppelia Robotics*. URL: `https://www.coppeliarobotics.com/`. (accessed: 17.11.2023) (cit. on p. 31).

[49] Universal Robots. *UR5 user manual* (cit. on pp. 36, 41, 42).

[50] Webb Watanabe Tang. «The Design and Evaluation of an Ergonomic Contactless Gesture Control System for Industrial Robots». In: *Journal of Robotics* 2018 (2018) (cit. on p. 39).

[51] Bertolino Raviola Guida. «A Comprehensive Multibody Model of a Collaborative Robot to Support Model-Based Health Management». In: *Robotics* 12(3) (2023) (cit. on p. 40).

[52] Saba Kebria Parham. «Kinematic and dynamic modelling of UR5 manipulator». In: *2016 IEEE international conference on systems, man, and cybernetics* (2016), pp. 4229–4234 (cit. on p. 40).

[53] Villani Siciliano Sciavicco. «Robotics.Modelling, Planning and Control». In: *Springer London* 1 (2008), pp. XXIV, 632 (cit. on pp. 42–44, 46).