Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Mechanical Engineering (38th Cycle)

Effective and safe framework for human-robot interaction

Michele Polito
* * * * * *

Supervisors
Prof. Stefano Pastorelli, Supervisor
Prof. Laura Gastaldi, Co-Supervisor

Politecnico di Torino
January 31, 2026

I hereby declare that, the contents and organisation of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

…….....

Michele Polito

Turin, January 31, 2026

Summary

Human–robot collaboration requires robotic systems capable of operating safely and efficiently in close proximity to human operators, while adapting to dynamic and uncertain environments. Although current collaborative robots comply with international safety standards, their capabilities are often limited to conservative and reactive behaviors, which restrict the exploitation of true human–robot synergy. This doctoral thesis addresses these limitations by proposing an integrated, human-centered framework that tightly couples perception, safety assessment, and motion planning to enable adaptive and context-aware collaboration.

The main contributions of this work focus on the perception and cognition layers of the interaction loop. A multimodal human arm tracking system is developed by fusing wearable inertial measurement units with an event-based vision sensor through a Kalman filter–based framework. The complementary characteristics of inertial sensing and event-driven vision enable accurate and low-latency upper-limb tracking while reducing computational load. Experimental results demonstrate improved accuracy and stability compared to unimodal tracking approaches.

The same inertial sensing infrastructure is further exploited to assess the operator's motion safety state. A machine learning–based classifier is trained to detect abrupt and potentially hazardous movements in real time, allowing the system to distinguish between nominal task execution and safety-critical behaviors. The network achieves recognition accuracy above 93% during testing. When integrated into the complete system, the detection pipeline enables timely safety responses, including robot stopping actions, with an overall reaction time below 0.5 s.

At the planning level, a reinforcement learning–based framework is proposed to generate human-aware robot motions. Two learning environments are designed:

a joint-control environment, used to progressively train the robot to accurately follow target trajectories, and a path-planning environment, in which the agent learns to generate collision-free Cartesian paths in the presence of a human operator and environmental constraints. Safety, kinematic feasibility, and collision avoidance are embedded directly into the reward functions, allowing the learned policies to internalize safety constraints rather than relying on external supervisory logic. The path planner proved to be an effective approach for generating collision-free trajectories with a success rate exceeding 90% within the specified workspace.

All components are integrated within a modular ROS 2–based architecture that enables real-time communication between perception, safety assessment, and motion planning modules. The complete framework is validated on a representative collaborative assembly task, where the robot and the human operator share the workspace. The experimental results demonstrate that the proposed architecture can safely manage human presence, react to abrupt operator movements, and generate adaptive robot trajectories consistent with the interaction context.

Overall, this thesis shows how multimodal perception, data-driven safety assessment, and reinforcement learning–based planning can be cohesively integrated to move beyond purely reactive safety mechanisms, contributing toward adaptive and robust human–robot collaboration.

Acknowledgment

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Collaborative Robotics

The current economic paradigm is increasingly oriented toward mass-customization of products and highly flexible production systems, diverging from the traditional model of mass production, which was characterized by high specialization and limited adaptability. The previous industrial era relied heavily on automation and industrial robots. These machines are designed for high-speed motion, large payloads, and wide workspaces. Such features enable precise and rapid execution of repetitive tasks typical of mass production, but they do not allow for the simple reconfiguration of production systems that is necessary in modern flexible manufacturing.

Moreover, traditional industrial robots cannot operate in close proximity to human workers. For safety reasons, they are physically separated from human workspaces. This segregation leads to several drawbacks: inefficient use of factory floor space, production interruptions whenever human intervention on the line is required, and low responsiveness to unexpected events. These limitations have motivated the emergence of collaborative robotics.

Collaborative robots (cobots) are designed to share the same workspace with human operators, and even allow direct physical contact. This paradigm is particularly well suited for small and medium-sized enterprises, where production flexibility is crucial. Although collaborative robots have been available for several years and can now be purchased commercially (Table 1), their global market penetration remains relatively limited. According to the International Federation of Robotics, 541000 new industrial robots were installed worldwide in 2023, marking the third consecutive year above half a million units. However, only around 54000 of these installations (10%) were collaborative robots, as shown in Figure 1, while traditional robots accounted for the remaining 90% [1].

Table 1 Some of the available cobot in the market

Manufacturer	Model	DOF	Communication Protocol	Programming Mode
Universal Robots	UR-series	6	Ethernet/IP, ModbusTCP, Fieldbuses	Polyscope pendant; URScript; APIs; ROS/ROS2
FANUC	CR-7iA / CRX-10iA	6	Ethernet, I/O, Vision connectors	Teach pendant touchscreen
Omron	TM5-700	6	Ethernet, Modbus TCP/RTU, RS232, Fieldbuses	TMflow flowchart-based programming
KUKA	LBR iiwa	7	Ethernet (Sunrise Cabinet)	Lead-through; KUKA Sunrise Cabinet; co-manipulation; simulation
KUKA	LBR iisy	6	KR C5 micro controller with Ethernet; torque sensor feedback; IP30/IP54 protection	Drag-and-drop via iiQKA.OS; manual guiding (smartPAD pro); lead-through (KUKA AG, NGEN IT LTD.)
UFactory	xArm 7	7	Ethernet; Wi-Fi; USB; Modbus-TCP	Drag-and-Drop UI; Python/C++; SDK; ROS/ROS2
Elephant Robotics	Mercury A1	7	CAN Bus; Wi-Fi; Network; Bluetooth; USB	myPanel touch UI; Python, C++; ROS/MoveIt integration
ABB	GoFa (CRB 15000 series)	6	Ethernet/fieldbus, OmniCore Iontroller	Lead-through; Wizard; FlexPendant UI
ABB	YuMi (IRB 14000 dual-arm)	14	Ethernet; integrated IRC5 controller	Lead-through; RAPID via FlexPendant & RobotStudio
Comau	MyCo Series	6	TCP/IP, ModbusTCP, Profinet, Ethernet/IP	Graphical Programming, Remote calling interface, ROS/ROS2
Doosan	A-Series, E-Series M-Series, H-Series,	6	ModbusTCP, ModbusRTU, PROFINET IO, EtherNet/IP, TCP/IP, RS-232/RS-422/RS-485 using USB to Serial converter	Built-in Teach Pendant, DART-Platform, DART-Studio, App Builder, ROS/ROS2
Elite Robots	CS-Series	6	RS485, Ethernet TCP/IP, Modbus TCP/RTU, Ethernet/IP, Profinet	Graphical interface, Teach pendant , SDK, Python, Java, C++,C#,ROS7ROS2, Moveit
Jaka	Zu-Series, Pro-Series	6	TCP/IP,Modbus TCP, Modbus RTU, Profinet, Ethemet/IP	Graphical programming, and freedrive programming
Kinova	Gen3-Series	6/7	Modbus	Web-based application, Python, C++, Matlab, ROS, ROS2
Techman	TM-Series	6	RS-232/RS-422/RS-485, Ethernet, Modbus TCP/RTU	TMflow, TMscript, TMcraft

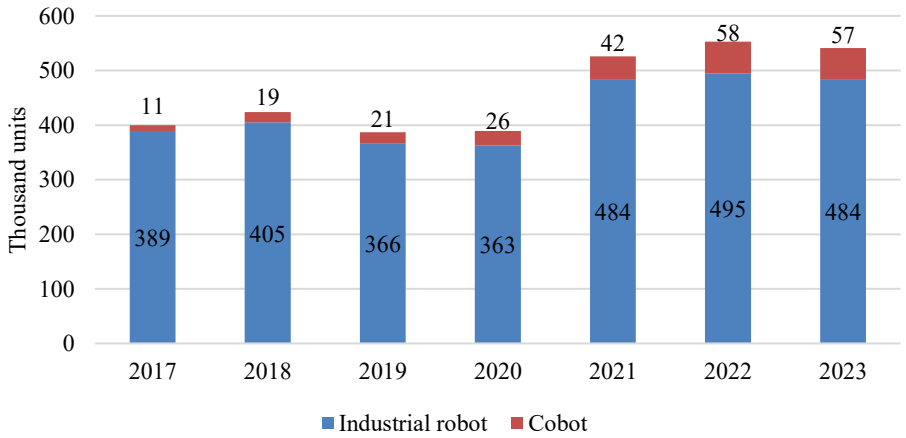

Figure 1 Annual installation of industrial and collaborative robots per year [1]

Nevertheless, the impact of collaborative robotics on companies that adopt it has been positive, with reported improvements in performance [2]. In practice, however, most current industrial applications of cobots replicate the same tasks traditionally performed by standard industrial robots, leveraging primarily the elimination of physical barriers and the consequent space savings. This elimination of workspace separation is possible because collaborative robots are intrinsically safe, designed in accordance with international safety standards such as ISO 10218 and ISO/TS 15066, which specify requirements for power- and force-limited operation and safe human-robot interaction. A second motivation for the adoption of collaborative robots lies in their inherent ease of use. Modern collaborative robots are typically equipped with intuitive graphical interfaces and support programming by direct teaching, whereby the operator manually guides the robot to the desired configurations, or by means of simple input devices such as joysticks. This significantly reduces the complexity of robot setup and task reconfiguration, lowering the barrier to entry and limiting the need for highly specialized programming operator. As a result, collaborative robots can be more readily integrated into existing production environments and can be reprogrammed directly by shop-floor operators, thereby enhancing flexibility and facilitating wider industrial adoption. Despite these advances, the broader potential of collaborative robotics is still underexploited [3]. A likely explanation is the insufficient maturity of key enabling technologies required for truly effective human–robot collaboration.

To fully unlock the potential of collaborative robotics, robots must demonstrate advanced environmental perception, robust decision-making, and adaptive planning capabilities.

1.2 Scope

Building upon the need for advanced perception, robust decision-making, and adaptive planning in collaborative robotics, this thesis adopts a structured perception–decision–action paradigm as its conceptual foundation. As illustrated in Figure 2, effective human–robot collaboration relies on the continuous perception of a shared environment that explicitly observable includes a human operator, the interpretation of the acquired information to select appropriate actions, and the execution of those actions, which in turn modify the environment and influence subsequent interactions with the operator.

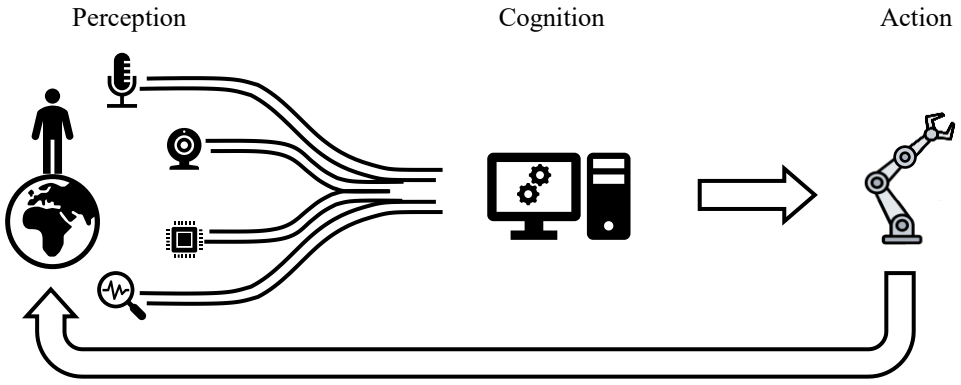

Figure 2 Perception–decision–action loop underlying human–robot collaboration: multimodal perception of the shared workspace informs cognitive processes for safety assessment and decision-making, which drive human-aware motion planning and robot execution

From a perception standpoint, the accurate and robust representation of the workspace is a critical requirement. In dynamic collaborative scenarios, this increasingly calls for the integration of heterogeneous sensing modalities capable of capturing both the kinematic state of the human operator and the surrounding environment. The sensory data must then be processed and interpreted to extract meaningful information for higher-level reasoning and decision-making.

Within this context, the primary objective of this thesis is to advance the state of the art in collaborative robotics by focusing on the perception and cognition layers of the interaction loop. To this end, a human-centred framework is developed that combines operator tracking, motion safety assessment, and human-aware motion planning. Specifically, a multimodal operator tracking system is proposed, based on the fusion of heterogeneous sensors, namely wearable inertial measurement units and an event-based camera monitoring the workspace. A dedicated sensor fusion algorithm is designed to integrate these complementary sources of information, providing a robust and continuous estimation of the operator's motion.

The same inertial sensing infrastructure is further exploited to assess the safety state of the operator's movements. Machine learning-based algorithms are trained to recognize motion patterns associated with potentially hazardous or abrupt behaviors, enabling an informed evaluation of the interaction context.

Finally, the information provided by the perception and safety assessment modules is leveraged to generate context-aware robot actions. In this regard, a learning-based motion planner is developed, capable of producing collision-free trajectories with respect to both the robot itself and the human operator. By means of reinforcement learning, safety and kinematic constraints are directly internalized within the planning policy, allowing the collaborative robot to adapt its behavior to the presence of the operator.

Overall, the scope of this thesis is to demonstrate how tightly coupled perception, cognition, and planning modules can enable more natural, adaptive, and effective human–robot collaboration, moving beyond reactive safety mechanisms toward genuinely human-aware robotic systems.

1.3 International Standard

The primary reference standards are ISO 10218-1:2011 “Robots and robotic devices — Safety requirements for industrial robots Part 1: Robots” and ISO 10218-2:2011 “Robots and robotic devices — Safety requirements for industrial robots Part 2: Robot systems and integration”, which define general safety requirements for robots and integrated robotic systems, complemented by ISO/TS 15066:2016 “Robots and robotic devices — Collaborative robots”, which provides specific guidelines for the design and application of collaborative robots.

ISO 10218 is divided into two parts. ISO 10218-1 addresses the intrinsic safety requirements of the robot itself, including actuation controls, stop functions, speed monitoring, singularity protection, and axis limitations. Within this framework, the concept of collaborative operation is introduced, describing a condition in which a robot specifically designed for collaboration works directly alongside a human operator in a shared workspace. ISO 10218-2 broadens the perspective by focusing on the integration of robotic systems. It establishes criteria for risk assessment, installation, commissioning, and the protective measures required during programming, maintenance, and repair. In particular, it emphasizes the need for system design to account for the working environment, robot trajectories, safety devices, and interactions with other machinery.

The publication of ISO/TS 15066:2016 marked a decisive step in the development of collaborative robotics, as it introduced operational guidelines and quantitative values for managing physical interactions between humans and robots. This technical specification clarifies that safety cannot rely exclusively on physical barriers, as in traditional robotic systems, but must instead be ensured through a combination of intrinsic design, advanced control systems, and biomechanical limitations. A central aspect of the specification is the definition of acceptable force and pressure thresholds for physical contact with the human body, differentiated by anatomical region. This allows objective parameters to be set within which a robot can operate without causing injury in the event of collision.

The standards outline four clearly defined modes of collaborative operation, each with its own safety architecture and control logic. The first is the safety-rated monitored stop, where the operator may only access the robot’s workspace when the robot is fully stopped. The control system remains active, but all motion ceases when human presence is detected, and restart requires deliberate operator action. This configuration provides a high level of safety but does not enable true simultaneous collaboration, as human and robot alternate in their tasks.

The second mode, hand guiding, introduces more direct interaction. Here, the operator physically leads the robot’s movements using a manual guidance device such as a teach pendant, joystick, or force-sensitive handle mounted on the robot arm. Safety is ensured through enabling devices with three-position switches, which allow motion only when deliberately engaged. Speed and force are limited at all times. Hand guiding is common in trajectory teaching, intuitive

programming, and assembly tasks where the robot is manipulated as a tool rather than functioning autonomously.

A further level of collaboration is achieved with speed and separation monitoring (SSM). In this mode, robot and human share the same workspace, with safety ensured by real-time distance monitoring. The robot operates at maximum speed when no operator is nearby but progressively slows as the operator approaches, stopping entirely if a minimum safety distance is breached. Implementing SSM requires advanced sensors such as laser scanners, depth cameras, and certified vision systems. This mode reduces downtime compared to a monitored stop but still does not enable full interaction, as human and robot may work simultaneously in the same area but not on the same object.

The power and force limiting (PFL) mode allows continuous physical contact between robot and operator. Robots designed for PFL incorporate intrinsic safety features such as torque-limited joints, rounded edges, lightweight structures, and real-time force monitoring. ISO/TS 15066 establishes precise quantitative thresholds for acceptable forces, pressures, and kinetic energies on different regions of the human body, distinguishing between quasi-static contacts, where an operator may become trapped, and transient contacts, where the body can naturally withdraw from the impact. PFL thus embodies the core principle of collaborative robotics, enabling genuine simultaneous cooperation between human and machine. However, this capability comes at the expense of robot performance, as speed and payload must remain within the biomechanical limits defined by the standard. Unlike other modes, PFL relies on the robot's intrinsic design rather than active environmental monitoring, which means it ensures baseline safety but lacks adaptability to varying operational contexts. As a result, while PFL makes collaboration physically possible, it does so by trading flexibility and efficiency for guaranteed compliance with safety constraints.

Taken together, the four modes illustrate a progression from conservative protective solutions, such as monitored stops, to advanced and immersive collaboration based on force and power limitation. Each mode is suitable for specific applications, and the choice depends on risk assessment, task requirements, and robot design. Importantly, operator safety is not achieved through a single measure but through a multilayered approach combining intrinsic design, protective devices, redundant control systems, and continuous validation.

In conclusion, ISO 10218 and ISO/TS 15066 form the essential foundation for the safe design and implementation of collaborative applications. However, regulatory compliance is a necessary but not sufficient condition for effectiveness in industrial contexts. The true value of collaborative robotics will depend on the ability of control systems to move beyond strict compliance, incorporating adaptive logic that can dynamically select the most appropriate mode of collaboration. At the same time, advances in perception and human–machine interfaces will be crucial to ensure that robots correctly interpret operator intentions and respond with predictable, coherent behaviors. Only through this synergy between normative frameworks, advanced control technologies, and

human-centered perception can collaborative robotics achieve its promise of improving productivity while preserving safety and quality of work.

ISO 10218 update 2025

The ISO 10218:2025 “Robotics — Safety requirements” standard represents a significant update to the regulatory framework for industrial and collaborative robotics. One of the most notable changes is the integration of the technical specification ISO/TS 15066, whose guidelines are no longer voluntary but now form binding requirements. The 2025 revision also reorganizes and clarifies the concepts surrounding collaborative robotics. In particular, the term collaborative robot has been removed, as it was considered misleading, and replaced with the broader notion of robotic application. This shift emphasizes that collaboration is not an intrinsic property of the robot itself but rather a characteristic of the overall application, which may rely on multiple systems working together to ensure safety and effective cooperation.

Risk assessment requirements have also been expanded and now explicitly mandate consideration of different interaction types, distinguishing between intentional and accidental contacts through structured methodologies. The revised standard introduces dedicated sections on the design of collaborative layouts, the prevention of entrapment, and the limitation of movements within shared workspaces. In addition, it formalizes requirements for certified safety functions capable of dynamically managing parameters such as speed, acceleration, force, and power. Another important innovation is the introduction of the principle of safe transition between collaborative and non-collaborative modes, which must be governed by validated control functions.

Cybersecurity is given a prominent role in the 2025 update. In collaborative scenarios, the ability to manipulate safety parameters remotely presents critical risks, making robust cyber-protection mechanisms a necessary component of compliance. The revision also recognizes the increasing prevalence of multi-robot cells and applications involving multiple operators, extending the scope of collaboration to more complex and dynamic industrial environments. Furthermore, ISO 10218:2025 introduces a new classification system for robots, dividing them into two distinct classes (Class I and Class II), each with differentiated safety requirements. To be designated as a Class I robot, compliance must be demonstrated through a specific validation test.

Overall, the 2025 revision shifts the focus of the standard from the mere safe coexistence of humans and robots to a more integrated vision of collaborative applications, in which operators and robotic systems share tasks within dynamically managed workspaces. By formalising requirements for adaptive safety functions, multi-robot layouts, cybersecurity, and structured treatment of intentional and unintentional contacts, ISO 10218:2025 explicitly supports scenarios where robots are aware of, and responsive to, human presence rather than simply constrained by it. The methods developed in this work, from upper-limb motion tracking and abrupt-movement detection to reinforcement learning–

based path planning, are aligned with this evolution, as they address key enabling technologies for context-aware, adaptive, and user-centred collaboration. In this sense, the proposed framework contributes to make in practice the new standard's principles, with the broader goal of enhancing the quality, intuitiveness, and safety of human–machine interaction in industrial environments.

Chapter 2

State of the Art

2.1 Human robot collaboration

Human–robot collaboration (HRC) describes all those situations in which robots and human operators are no longer separated by physical barriers but instead share the same workspace. In this context, direct physical contact between human and robot is possible and, in some cases, even necessary. It is important to emphasize, however, that the presence of a collaborative robot does not by itself guarantee meaningful collaboration. Different forms of interaction can be distinguished according to the degree of integration between the two agents, ranging from minimal coexistence to highly reactive cooperation [4].

At the lowest level, coexistence occurs when the human and the robot operate

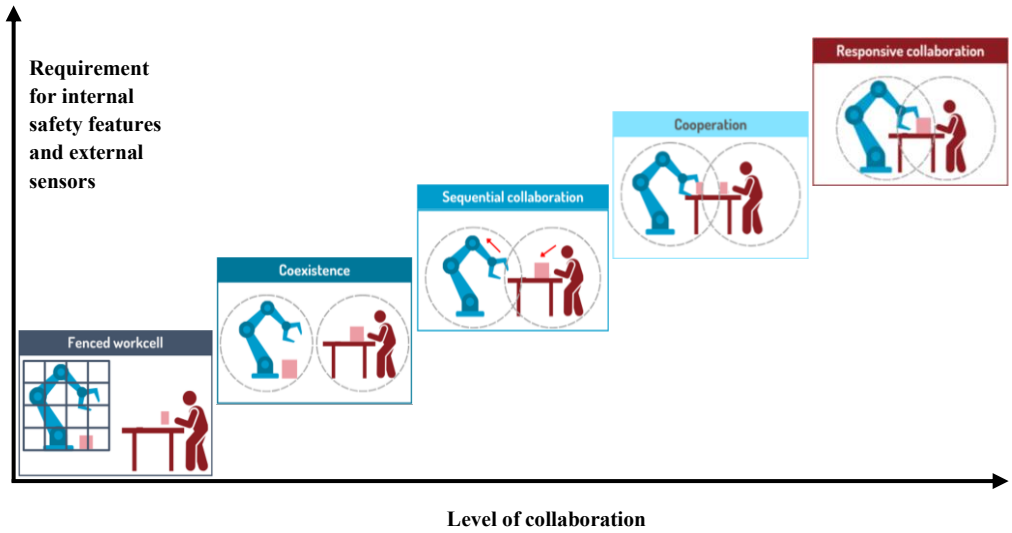

Figure 3 Level of collaboration in human robot interaction. [4]

in parallel but in distinct areas of the workspace, without overlapping tasks. A more integrated form, known as sequential collaboration, emerges when both share the same workspace but alternate their interventions on the task, with the robot and the operator acting at different times. Cooperation represents a further step, where human and robot work simultaneously on the same object, requiring higher synchronization and mutual awareness. The most advanced mode is reactive collaboration, in which the robot continuously interprets the operator's actions and intentions and adapts its behavior in real time. These categories, summarized in Figure 3, provide a conceptual framework to classify collaborative scenarios.

As the degree of collaboration increases, the requirements imposed on the robot also become more demanding. At basic levels, such as coexistence, it is generally sufficient to comply with established safety standards, which ensure that the system remains safe even in the event of physical contact. As collaboration evolves towards cooperation and reactivity, however, robots must integrate advanced sensing and cognitive abilities in order to perceive their environment, interpret human behaviors, and respond effectively.

Perception constitutes the first layer of these capabilities. Robots rely on a variety of sensors to capture information from their surroundings: monocular and stereo cameras are the most common, but complementary technologies such as LiDAR, ultrasonic devices, inertial measurement units, capacitive systems, and tactile sensors designed as artificial skins are increasingly adopted. These sensors enable the robot to detect the presence and position of objects, monitor the operator's location, and track human motion patterns.

Yet perception alone is not enough. For effective collaboration, sensory data must be transformed into meaningful understanding through cognitive processes. In structured environments with limited variability, this can be achieved through deterministic approaches, such as finite state machines or optimization algorithms that map predefined stimuli to robot actions. In more dynamic and uncertain contexts, however, the limitations of these approaches become evident, and more sophisticated methods are required. Neural networks and other artificial intelligence techniques allow robots to learn from data and to generalize to new situations, thereby supporting flexible and adaptive behaviors. The recent development of large language models has further expanded the potential of machine cognition, enhancing the robot's ability to interpret context, infer operator intentions, and handle increasingly complex interactions.

Concrete examples illustrate how the balance between perception and cognition depends on the collaboration level. In coexistence, simple solutions such as proximity sensors that halt or slow the robot when a human enters its working area are usually sufficient. In sequential collaboration, the robot must be able to recognize the different phases of a process and adapt to variations introduced by the operator, but without the need to interpret every single human action. In cooperative or reactive modes, by contrast, the robot must continuously monitor, interpret, and adapt to human movements and intentions. Without these

abilities, collaboration risks becoming ineffective or even counterproductive, potentially hindering rather than supporting the operator's performance.

The following chapters will explore in greater depth the sensing modalities used in collaborative robotics and the computational frameworks that enable robots to interpret and respond to complex human behaviors in shared workspaces.

2.2 Perception and Cognition

Effective human–robot collaboration relies fundamentally on perception. A robot cannot collaborate safely or efficiently without the ability to sense and interpret its environment [5]. Perception provides the basis for spatial awareness, recognition of human presence, monitoring of movements and gestures, and detection of objects and obstacles. It is therefore not only a prerequisite for ensuring safety, but also a key enabler of adaptability, contextual understanding, and the anticipation of human intentions [6].

A wide variety of sensors have been developed and deployed to equip robots with these perceptual capabilities. In the context of HRC, the main categories include optical sensors, inertial (motion-based) sensors, acoustic sensors, tactile sensors, and bionic sensors [6, 7], Figure 4. Within each category, multiple technologies exist that share the same physical principle but differ in terms of resolution, range, robustness, and integration complexity. No single type of sensor can capture the full richness of a collaborative workspace. Each has strengths and limitations depending on the application. Optical systems, for example, are well suited for human pose estimation and object recognition, but they may be sensitive to occlusions and lighting. Inertial sensors are robust and lightweight, but they typically require fusion with other modalities to achieve accurate spatial tracking. Tactile and capacitive sensors provide local, high-fidelity information about physical interaction, but cannot describe the global environment. For this reason, HRC applications often rely on multimodal perception, combining different sensors to obtain a more complete and reliable representation of the shared workspace [6, 8, 9]. The following sections will examine these sensor categories in detail, discussing their technical characteristics and typical applications in collaborative robotics. Each subsection will also highlight recent

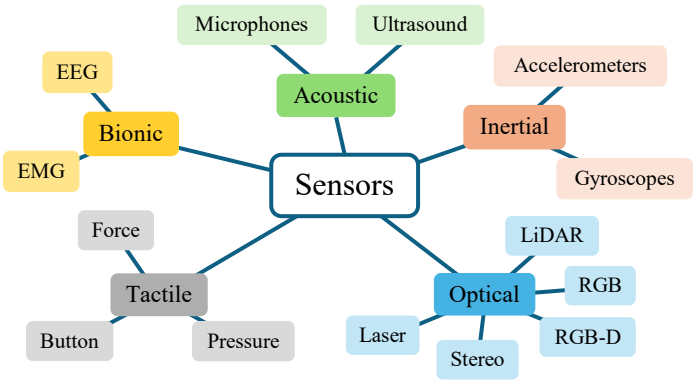

Figure 4 Sensor taxonomy employed in HMI

research contributions and case studies, illustrating how these technologies support different levels of human–robot collaboration.

Optical-based sensor

Optical sensors represent the broadest and most widely adopted class of sensing technologies in collaborative robotics [5]. This category encompasses a wide range of devices, including monocular, stereo, RGB-D, and event-based cameras, as well as active-light systems such as Light Detection and Ranging (LiDAR), infrared sensors, and laser range finders. Their applications extend from simple presence or proximity detection to complex three-dimensional reconstructions of the environment. Among these, cameras remain the most frequently used in HRC, as they provide rich visual information that supports detailed representations of the shared workspace [10]. Data from cameras have enabled the development of numerous algorithms for tracking, recognition, and 3D reconstruction. The main limitations of optical sensors include susceptibility to occlusions, sensitivity to ambient lighting conditions, and the large amount of data generated, which requires significant computational resources compared with other sensing modalities. Figure 5 shows the different optical-based sensing modalities together with representative references in which each technology is employed.

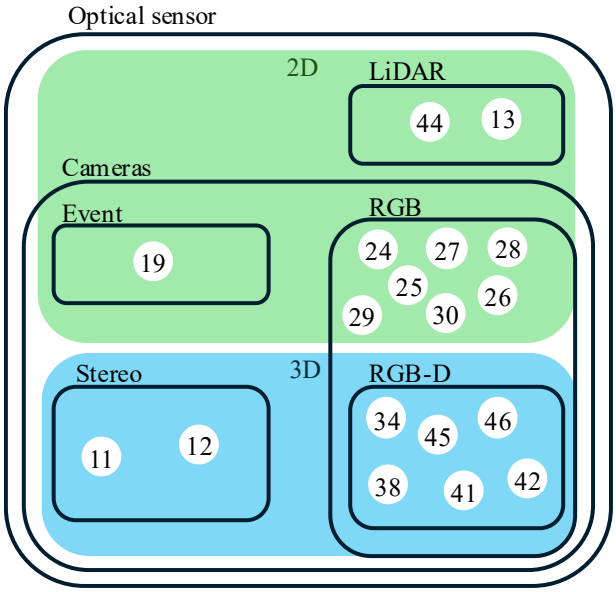

Figure 5 Classification of the optical-based sensing approaches employed in human–robot interaction, the numbers in the circles indicate the corresponding references in the bibliography

RGB cameras, in particular, detect visible light using a lens to focus radiation onto an optical sensor, typically based on a CMOS array with color filters. The resulting output is a two-dimensional color image of the observed environment. Key performance parameters are resolution and frame rate. Higher resolution increases spatial detail but also computational load, while higher frame rate improves temporal resolution and reduces motion blur at the cost of greater data throughput. The choice of camera properties therefore depends strongly on the

application context: for example, high-resolution sensors are preferable for object recognition and scene mapping, whereas high frame rates are more critical for dynamic tasks such as gesture recognition or collision avoidance.

RGB-D cameras are optical sensors capable of simultaneously capturing both color information and depth data, making them one of the most versatile devices for human–robot collaboration. An RGB-D system combines a traditional RGB camera with a depth-sensing unit, which can be based on structured light or time-of-flight (ToF). The RGB module acquires a standard color image, while the depth module provides per-pixel distance measurements, generating a 3D point cloud aligned with the visual scene. This fusion enables the robot to perceive not only the appearance of objects but also their spatial geometry and relative position in the environment. Typical performance parameters include depth resolution, maximum operating range, field of view, and frame rate, all of which influence accuracy and applicability. RGB-D cameras are widely used for human detection, gesture recognition, workspace mapping, object tracking, and collision avoidance.

In human-tracking applications, RGB-D cameras are commonly used within a perception pipeline that combines dense visual and depth information to reconstruct the spatial configuration of the operator. A typical pipeline starts with the synchronized acquisition of RGB and depth frames, followed by intrinsic and extrinsic calibration, RGB-depth registration, depth filtering, hole filling, and temporal smoothing. These preliminary stages are necessary because the raw depth map is affected by measurement noise, missing values, and distance-dependent uncertainty. The human body is then extracted from the scene through background subtraction, depth-based segmentation, point-cloud clustering, or learning-based person detection. Once the operator has been isolated, anatomical landmarks are estimated either by a vendor-specific body-tracking SDK or by a dedicated pose-estimation network. In many implementations, 2D keypoints are first detected in the image plane and then projected into three-dimensional space using the corresponding depth values and the intrinsic parameters of the camera. The resulting joint positions are finally refined through skeleton fitting, biomechanical constraints, and temporal filtering in order to reduce jitter, compensate for temporary missing detections, and enforce plausible limb configurations.

The execution of this pipeline requires several conditions to be satisfied. The operator must remain sufficiently visible within the field of view, the depth map must provide reliable measurements in correspondence with the relevant body segments, and the computational platform must be able to process dense RGB-D frames together with the pose-estimation and filtering algorithms. These requirements are particularly important when tracking the upper limb, where self-occlusions, fast motions, and partial visibility of the elbow and wrist can strongly affect the reliability of the reconstructed skeleton. In controlled conditions, RGB-D systems can achieve centimetre-level tracking accuracy, with multi-camera approaches reporting upper-limb errors of approximately 5 cm [14], while single-camera or more challenging dynamic configurations often lead to errors in the range of approximately 8–18 cm for body keypoints [15, 16]. From a temporal

perspective, RGB-D systems are generally compatible with real-time operation, but their effective update rate is commonly constrained by the frame-based acquisition paradigm and by the computational cost of the tracking pipeline. Although some RGB-D sensors can acquire depth data at higher rates depending on the selected resolution and operating mode, many human-tracking implementations operate at approximately 30 Hz, which is also the typical frame rate adopted by several Kinect, Azure Kinect, and Intel RealSense-based studies [15–18]. More complex pipelines, especially those relying on deep learning models, multi-camera fusion, or full-body model fitting, may require GPU acceleration to maintain real-time performance. Consequently, RGB-D-based systems are well suited for many monitoring and interaction tasks, but their responsiveness is ultimately limited by the need to acquire and process complete image and depth frames at each time step. From an economic point of view, RGB-D cameras represent a mature and commercially available technology. Devices such as Intel RealSense, Azure Kinect-derived sensors, Orbbec cameras, and similar mass-market depth cameras are widely accessible, relatively compact, and available at competitive prices compared with high-end motion-capture systems or scientific event-based cameras. Nevertheless, in human–robot collaboration, some limitations must be considered. RGB-D tracking is sensitive to occlusions, unfavorable viewpoints, reflective or transparent surfaces, distance-dependent depth degradation, and interference between multiple active depth sensors. Furthermore, the dense nature of RGB-D data introduces a non-negligible computational cost. These aspects can reduce tracking reliability and increase latency in dynamic collaborative workspaces, where the perception system must promptly react to human movements.
However, limitations include sensitivity to ambient light conditions, reduced accuracy at longer ranges, and performance degradation in the presence of reflective or transparent surfaces.

Event cameras, also referred to as Dynamic Vision Sensors (DVS), represent a class of bio-inspired visual sensors whose operating paradigm fundamentally differs from that of conventional frame-based cameras. Rather than acquiring complete images at fixed sampling intervals, event cameras operate asynchronously and measure local temporal variations of scene illumination. This sensing principle is inspired by the biological visual system, where neural activity

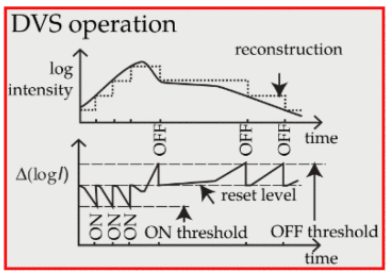

Figure 6 Operating principle of a Dynamic Vision Sensor (DVS). ON and OFF events are generated when changes in logarithmic light intensity exceed predefined contrast thresholds, with the reference level reset after each event to enable continuous asynchronous sensing [19].

is triggered primarily by changes in visual stimuli rather than by static intensity levels. Each pixel in an event camera functions independently and continuously monitors the logarithmic intensity of the incoming light. An event is generated whenever the accumulated change in log-intensity exceeds a predefined contrast threshold. This mechanism is illustrated in Figure 6, which depicts how variations in the logarithmic light intensity over time lead to the generation of discrete events. When the change reaches a positive threshold, an ON event is produced, indicating an increase in brightness; conversely, when the negative threshold is crossed, an OFF event is generated, corresponding to a decrease in brightness. After an event is triggered, the internal reference level of the pixel is reset, allowing it to respond to subsequent intensity variations [19].

This threshold-based operation implies that event cameras do not encode absolute brightness values but instead respond exclusively to temporal contrast changes. As a result, pixels observing static regions of the scene remain silent, while dynamic edges, motion boundaries, and flickering patterns naturally generate activity. The asynchronous and event-driven nature of the sensor allows each pixel to react immediately to changes in the visual stimulus, without being constrained by a global exposure time or readout cycle

The output of an event camera is therefore not a sequence of images, but a stream of discrete events ordered in time, how it is illustrated in Figure 7b-d. Each event is commonly represented as a tuple $e_i = (x_i, y_i, t_i, p_i)$, where (x_i, y_i) denotes the spatial coordinates of the pixel, t_i is a high-resolution timestamp, typically with microsecond precision, and $p_i \in \{+1, -1\}$ denotes the polarity of

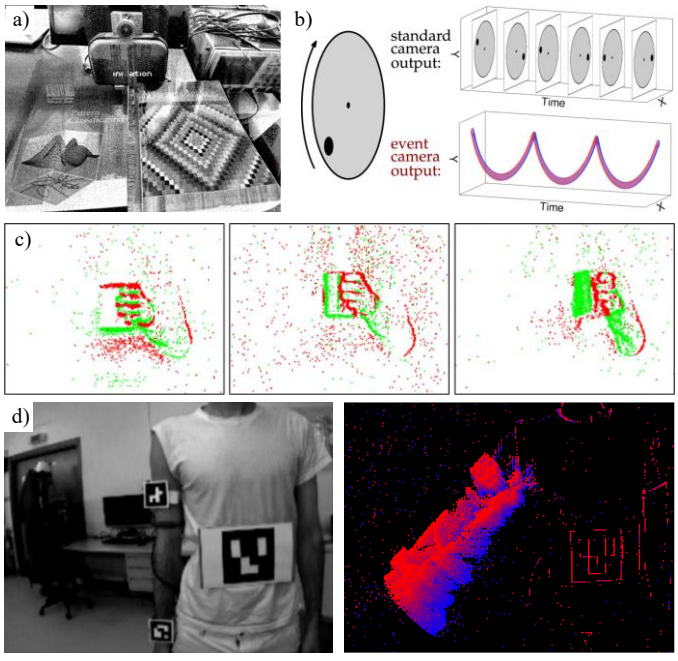

Figure 7 Examples of event-based representations. (a) Accumulated event frame over a 33 ms time window [21]. (b) Comparison between conventional frame-based output and event camera output for a rotating stimulus as spatio-temporal representation (red: ON events, blue: OFF events) [22]. (c) Event-based representation with a fixed event count during a pick-and-place task (green: ON, red: OFF) [23]. (d) Comparison between a standard image and an accumulated event frame, where color encodes event age (red: recent events, blue: older events).

the event, corresponding to ON or OFF transitions. This representation inherently couples spatial and temporal information, providing a fine-grained description of scene dynamics [20].

A natural way to interpret event data is through a space–time representation, in which events are embedded in a three-dimensional domain defined by spatial coordinates and time. As illustrated in Figure 7b, a moving stimulus generates structured spatio-temporal patterns, often appearing as continuous trajectories in the (x, y, t) space. In this representation, motion is implicitly encoded in the temporal ordering of events, without requiring explicit frame differencing or optical flow computation.

Although event cameras are not designed to produce images, event streams can be projected into image-like representations for visualization or algorithmic convenience. A common approach consists in accumulating events over a short temporal window, typically on the order of a few milliseconds, to generate event frames or event count images. In these representations, pixel values reflect the number of events or the balance between ON and OFF polarities observed within the selected time interval. Figure 7c shows an example of such a reconstruction, where the contours of objects and their motion patterns emerge clearly despite the sparse nature of the data. A representation closer to conventional image-based vision can be obtained by accumulating events over a finite temporal window and assigning each pixel an intensity value proportional to the number of events generated at that location within the selected interval modulated by a time-dependent decay factor. This approach enables the reconstruction of visually interpretable scene representations, in which object contours and structural details become clearly distinguishable, as illustrated in Figure 7a.

While conventional cameras uniformly sample the visual scene in time, event cameras provide a sparse, asynchronous, and temporally precise description of visual changes. This sensing paradigm enables extremely low latency, high temporal resolution, and efficient data usage, laying the foundation for robust perception in highly dynamic environments. These properties are especially relevant for robotic applications involving fast motion, continuous interaction, and real-time decision-making, which are further discussed in the following sections. Overall, event cameras introduce a paradigm shift in visual sensing, offering a efficient, and temporally precise alternative to conventional frame-based vision [20].

LiDAR sensors are active optical systems that estimate the geometry of the environment by emitting laser pulses (or modulated beams) and measuring the time required for the reflected signal to return to the receiver. This principle enables the construction of dense three-dimensional point clouds that describe the spatial arrangement of objects with high precision. The performance of a LiDAR system is determined by several key parameters. The operating range defines both the minimum and maximum distances at which objects can be detected, typically extending from a few centimeters up to several hundred meters depending on the device. The angular resolution specifies the smallest angular separation between

distinguishable objects, with values between 0.1° and 0.5° being common in robotic and industrial applications. The field of view describes the angular coverage of the sensor; while many systems offer 360° in the horizontal plane, the vertical coverage varies considerably and strongly influences the completeness of the environmental model. The scan rate and point density indicate how many points per second are acquired and how frequently the environment is updated, directly impacting both accuracy and computational requirements. Finally, accuracy and precision quantify the deviation between the measured and true distance, with typical errors ranging from a few millimeters to several centimeters depending on target reflectivity and environmental conditions. In collaborative robotics, LiDAR sensors are widely employed for workspace mapping, obstacle detection, human tracking, and as complementary sources in multimodal sensor fusion frameworks. Their ability to provide accurate global information makes them valuable in ensuring safety and adaptability, particularly in tasks where continuous monitoring of human presence is required. Nevertheless, several limitations must be acknowledged. LiDAR performance degrades in the presence of dust and reflective or transparent surfaces often introduce measurement errors or signal loss. Moreover, the generation and processing of dense point clouds impose significant computational, and bandwidth demands, potentially increasing latency in real-time applications.

Optical sensors have enabled the development of a wide range of data processing algorithms for contextual understanding in human–machine interaction. Significant efforts have been directed toward human detection and pose estimation, as locating and identifying the human operator represents the first step in interpreting intentions. One of the earliest approaches to detect humans was face detection [24–29], followed by color segmentation. Face detection enables the identification and localization of human faces within an image or video stream. One of the most established approaches relies on the extraction of discriminative features, such as intensity contrasts between bright and dark regions corresponding to the eyes, nose, and mouth. These features are then processed by classification algorithms, most notably the Viola–Jones detector [24], to determine the presence of a face. More recent methods employ convolutional neural networks (CNNs), which automatically learn relevant features and achieve greater robustness to variations in illumination, pose, and scale. These techniques are primarily applied to RGB images, while in systems based on RGB-D sensors, depth information is often integrated to improve detection accuracy and reduce false positives. Segmentation is a fundamental technique in robotic vision systems used to separate regions of interest, such as a person’s body or hands, from the background. In optical sensing applications, this process can be implemented using RGB cameras, where color thresholds in HSV or YCrCb spaces are applied to detect skin tones or colored objects. Although color-based segmentation is computationally efficient and conceptually simple, it remains highly sensitive to illumination changes and chromatic variability across individuals or environments. For this reason, it is often combined with depth cues or morphological operations to achieve more robust performance in HRC

scenarios. Several studies in the field of human–robot collaboration have employed the color segmentation approach [30–34]. Machine learning-based methods have also become increasingly prevalent in Human machine interaction (HMI) and HRC applications. Region-based Convolutional Neural Networks (R-CNN) [35] and Single Shot Detectors (SSD) [36] are commonly employed to directly localize regions of interest such as the face, hands, or eyes within an image. R-CNN methods perform object detection through a two-stage process, first generating region proposals and then classifying each region, achieving high accuracy at the cost of computational complexity. In contrast, SSD frameworks use a single-stage architecture that predicts object locations and classes simultaneously, allowing real-time performance with slightly reduced precision. Both approaches have been widely adopted in various human detection and gesture recognition systems [37, 38]. Deep Learning is also commonly adopted in human pose estimation, the most employed networks are Convolution Pose Machines [39] and OpenPose [40]. These networks extract keypoints and skeletons from 2D images reconstructing the pose of one or multiple people in a frame. Human motion tracking represents a further development in visual perception for HRI and HRC, enabling continuous monitoring of human movement over time rather than isolated detection or pose estimation. While pose estimation identifies the instantaneous configuration of the body, motion tracking associates these configurations across frames to infer trajectories and dynamic behavior. Techniques commonly used include Kalman filters [37, 41], particle filters [42], optical flow [43] and visual SLAM (Simultaneous Localization and Mapping) [44] approaches, which integrate temporal and spatial cues to maintain consistent identification of human targets. In RGB and RGB-D applications, these algorithms often combine visual and depth data to improve robustness against occlusions and illumination changes. Recent implementations also integrate deep learning-based tracking with probabilistic models [45, 46], achieving reliable prediction of human motion paths in collaborative workspaces. This capability allows robots to anticipate human trajectories, optimize task sharing, and ensure safety during close-proximity interaction. Another application enabled from visual system is social classification. Social classification extends visual perception beyond physical detection to the interpretation of human affective and social states. In HRC, this process involves analyzing cues such as facial expressions, gaze direction, body posture, and head orientation to infer emotional conditions, engagement, and intention to interact. Techniques range from feature-based classifiers, using descriptors like Gabor filters or histogram of oriented gradients, to deep neural networks capable of learning multimodal representations from visual and audio data. More advanced models, including CNNs and RNNs trained on affective datasets, enable robots to adapt behavior dynamically by recognizing user states such as satisfaction, hesitation, or focus

Motion-based sensors

Motion sensors constitute a broad class of devices designed to detect and quantify movement. In the context of human–robot collaboration, they are primarily used for motion tracking, gesture recognition, and operator localization [47]. The most common types include accelerometers, gyroscopes, and magnetometers, which together form the core of inertial measurement units (IMUs).

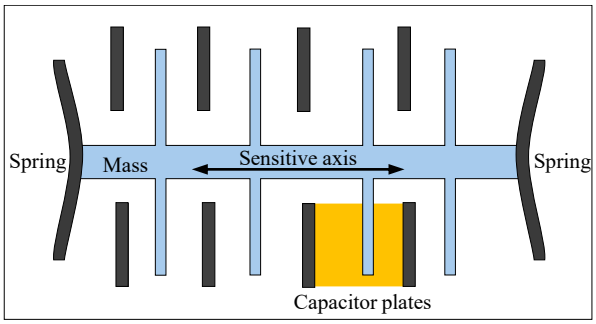

Figure 8 MEMS accelerometers design

Accelerometers measure the linear acceleration of the body to which they are attached. The most widely adopted accelerometers in robotics are based on micro-electromechanical systems (MEMS) technology, owing to their compact size, low cost, and low power consumption. Their operating principle relies on a proof mass connected to a fixed frame by elastic elements, typically modeled as springs. When an external acceleration is applied along the sensor's sensitive axis, the mass is displaced proportionally to the magnitude of the acceleration. This displacement is converted into an electrical signal, most commonly through capacitive sensing, allowing the acceleration to be measured with high precision.

Gyroscopes, on the other hand, measure the angular velocity of a body around a defined sensitive axis. MEMS gyroscopes operate based on the Coriolis effect, which arises when a moving mass experiences a force proportional to its velocity and the applied angular rate. Structurally, a MEMS gyroscope consists of a proof mass suspended by springs that allow motion along two orthogonal

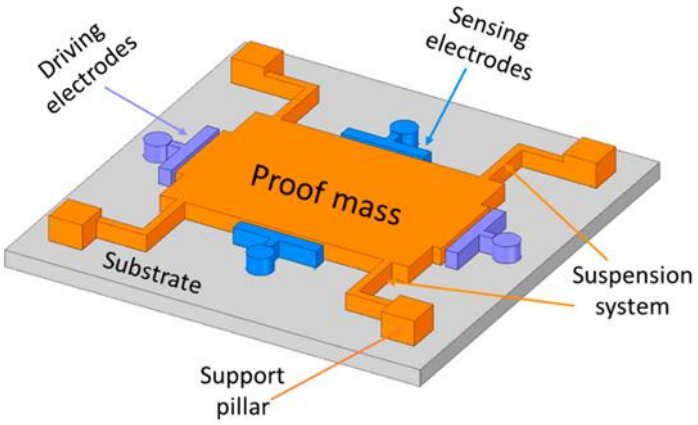

Figure 9 Gyroscope MEMS architectures [48]

directions. A controlled oscillation with fixed amplitude and frequency is applied along one of these axes, known as the drive axis. When the sensor undergoes angular rotation, the Coriolis force acts on the moving mass, inducing a secondary oscillation along the orthogonal direction, referred to as the sense axis. This motion is then detected and is proportional to the angular velocity applied to the sensor. The simplest schematic of MEMS structure is depicted in Figure 9.

Magnetometers are sensors designed to measure magnetic fields and their variations over time. Several technologies exist, but in collaborative robotics, the most commonly used are MEMS-based Hall-effect magnetometers. These devices exploit the Lorentz force acting on moving charge carriers within a conductor to determine both the direction and polarity of the magnetic field. In practical applications, magnetometers are often employed to detect the Earth's magnetic field, providing a global reference for orientation when combined with accelerometers and gyroscopes in an IMU. However, in industrial environments, their use is limited because the presence of metallic structures and motorized machinery distorts the local magnetic field, preventing the magnetometer from serving as a reliable global reference.

Inertial sensors are typically used in combination to form an Inertial Measurement Unit. A 6-DoF IMU consists of a three-axis accelerometer and a three-axis gyroscope, while a 9-DoF IMU also integrates a three-axis magnetometer, the latter can be also identified as Magnetic-Inertial Measurement Unit, MIMU. These systems provide real-time measurements of a body's linear acceleration and angular velocity, from which velocity, position, and spatial orientation can be estimated. The fusion of signals from the individual sensors, commonly performed through algorithms such as Kalman or complementary filters, enables a consistent and stable estimation of motion states. However, due to noise and integration drift, IMUs are prone to cumulative errors over time and are often combined with external references, such as optical or magnetic sensors, to improve accuracy [49].

Within the domain of collaborative robotics, IMUs are increasingly employed to enhance human–robot interaction due to their compactness, low weight, high sampling frequency, reduced data throughput, and high operator acceptability. These wearable sensors enable precise acquisition of acceleration and angular velocity data, thus supporting accurate human motion analysis even in complex industrial environments, free from the spatial and lighting constraints typical of optical motion capture systems [50].

IMU-based algorithms in human–robot collaboration can be broadly classified according to their functional objective, ranging from low-level state estimation to high-level gesture recognition and intention prediction.

At the most fundamental level, orientation and motion state estimation is addressed through quaternion-based representations combined with kinematic models of the human body or limbs. These approaches rely on the fusion of accelerometer and gyroscope data to estimate orientation and angular motion, while mitigating sensor noise and integration drift. Probabilistic filtering techniques, such as Kalman filters and particle filters, are widely adopted for this

purpose, providing statistically optimal estimates under uncertainty and enabling real-time performance in wearable and robotic systems [47, 49, 51–53].

Probabilistic filtering addresses the problem of estimating the state of a system from a sequence of time-varying measurements affected by statistical noise and modeling inaccuracies. Both the system inputs and the sensor observations contribute information about the system dynamics, while simultaneously introducing uncertainty due to process noise and measurement errors. These filters combine these two sources of information through a recursive procedure composed of two main steps. In the prediction (or estimation) step, the system state at the current time instant is inferred from the state estimate at the previous time step using a model of the system dynamics. In the subsequent update step, this predicted state is corrected by incorporating new sensor measurements, yielding an improved and statistically consistent estimate of the current system state [47]. Other approaches rely on the integration of kinematic information to directly estimate the orientation and position of the sensors [47, 52]. In human motion tracking, the measurements provided by IMUs are often integrated with a kinematic model of the human body in order to refine the estimation of sensor orientation and position. The kinematic constraints associated with the body segments to which the sensors are attached are exploited to enforce biomechanical consistency and reduce estimation errors [47].

Beyond state estimation, IMU-based sensing has been extensively employed for gesture recognition and motion prediction, which are essential for understanding human intent in collaborative tasks. Early machine learning approaches, such as Linear Discriminant Analysis (LDA), have demonstrated strong performance in classifying structured industrial motions, including pick-and-place actions of the upper limbs, with recognition rates exceeding 95% [54]. LDA is a supervised learning technique used for classification, which searches for a linear projection of the data that maximizes class separability under the assumption of Gaussian class distributions with shared covariance. More recently, deep learning techniques have gained prominence. Long Short-Term Memory (LSTM) networks have been used to capture temporal dependencies in motion sequences, while Convolutional Neural Networks (CNNs) have shown high accuracy in extracting discriminative features from inertial signals [55–57]. Transferable CNN architectures trained on wrist-mounted IMU data, for instance, have achieved recognition accuracies of up to 99.4% across multiple gesture classes [55].

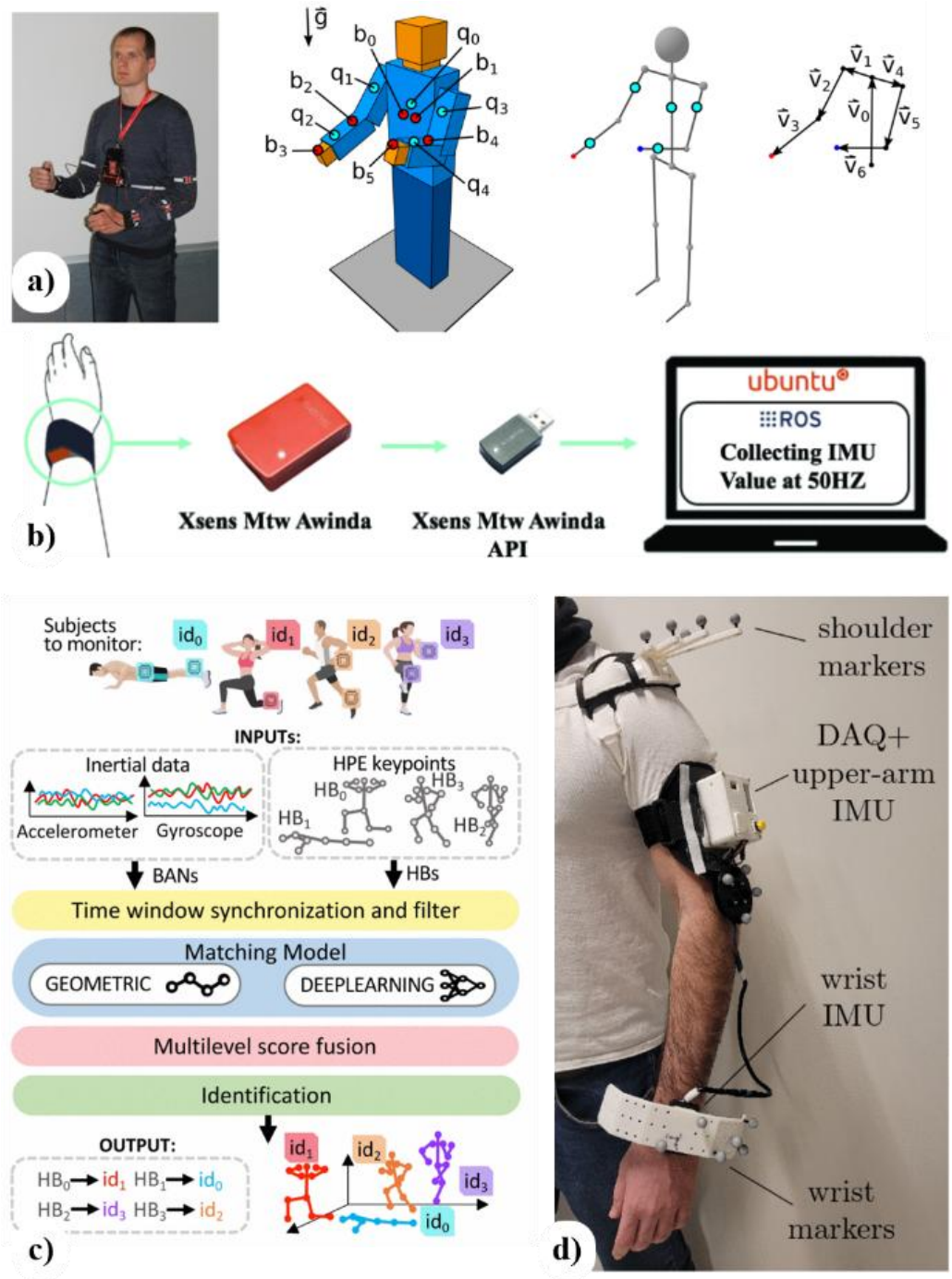

Figure 10 Different application with IMUs sensors, a) Pose estimation of human upper limb [53], b) gesture recognition with CNN Network [55], c) User identification and human tracking [51], d) Human arm predicting motion with LSTM [57].

Acoustic sensors

Acoustic sensors are devices designed to capture sound waves. Various types exist, but the most common in HMI applications are capacitive microphones. These sensors convert variations in air pressure into electrical signals. Their most widespread application is speech recognition; however, they have also been employed for object or body-part localization [7, 8] and for monitoring muscle fatigue [58]. Through these sensors, it becomes possible to use natural language to interact directly with robots, thereby improving both the efficiency and intuitiveness of collaboration.

From an algorithmic standpoint, acoustic perception relies on a combination of signal processing, machine learning, and deep learning techniques to extract and classify sound features. Traditional approaches such as Mel-Frequency Cepstral Coefficients (MFCCs), Fast Fourier Transform (FFT), and Linear Predictive Coding (LPC) are commonly used to preprocess sound data and to represent acoustic signatures in a compact spectral form suitable for machine learning. Based on these representations, algorithms including Hidden Markov Models (HMMs), Gaussian Mixture Models (GMMs), and Dynamic Time Warping (DTW) have historically been applied to speech segmentation and recognition tasks. More recently, deep neural networks (DNNs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs) have largely replaced classical models in industrial applications due to their superior capacity to generalize from complex and noisy datasets [8].

Several studies have exploited acoustic information to directly command robots to perform specific actions [59–62]. Other works have focused on interpreting the speaker's emotional state or the intention to interact [63, 64]. Some studies have also addressed speaker localization using acoustic sensor arrays [65]. Although this form of localization is minimally invasive and natural, its accuracy remains lower than that of other sensing technologies [8].

Tactile sensors

Tactile sensors provide fundamental information about pressure, force, vibration, texture, and temperature, all of which are essential for dexterous robotic manipulation and for establishing effective HMI. In recent years, remarkable progress has been achieved thanks to advances in flexible materials, micro–electro–mechanical systems, and nanofabrication techniques. These developments have allowed the realization of sensors that are lightweight, highly sensitive, and mechanically compliant, capable of adapting to curved or deformable surfaces and mimicking the properties of human skin [66–68].

Among the various tactile sensing technologies, resistive, capacitive, piezoelectric, and triboelectric sensors are the most widely employed. Resistive tactile sensors operate by converting mechanical deformation into a change in electrical resistance. They are commonly made of conductive elastomers or

nanocomposite materials such as graphene, carbon nanotubes, or conductive polymers embedded in flexible matrices like polydimethylsiloxane (PDMS). These sensors combine simplicity and low cost with high sensitivity and broad dynamic range, although they often suffer from hysteresis and thermal instability. Capacitive tactile sensors, instead, detect changes in capacitance between two electrodes separated by a deformable dielectric layer. Microstructuring of the dielectric, by introducing domes, pyramids, or porous geometries, significantly increases sensitivity and enables decoupling of normal and tangential forces. Such devices are characterized by low power consumption, excellent repeatability, and compatibility with CMOS integration, which allows the creation of high-density tactile arrays with spatial resolutions on the order of millimeters [66].

Piezoelectric sensors exploit the electric potential generated by certain crystalline materials, such as ZnO, PZT, or PVDF, when subjected to mechanical stress. These sensors are self-powered and particularly suitable for dynamic signal detection, including vibrations and transient contacts, although they are less effective for static load measurement. Triboelectric sensors share the property of self-sufficiency but rely on the contact and separation of materials with different triboelectric polarities to generate electrical signals. Their inherent capability to sense both contact and proximity makes them especially attractive for energy-autonomous wearable systems and artificial electronic skins. Other types, including optical and magnetic sensors, have also found applications in tactile sensing. Optical devices use variations in light intensity or interference patterns to measure deformation, offering immunity to electromagnetic interference and very high resolution. Magnetic sensors, based on the Hall effect or giant magnetoresistance, enable non-contact detection of forces and three-axis decoupling. More recently, hybrid tactile systems have emerged that combine multiple sensing principles, capacitive, resistive, and piezoelectric, within a single platform, allowing simultaneous measurement of pressure, temperature, and strain while improving robustness and accuracy [67].

The information generated by tactile sensors is inherently complex, often consisting of high-dimensional, nonlinear, and noisy data streams. To extract meaningful information, a series of signal interpretation algorithms is required, ranging from classical signal processing techniques to advanced artificial intelligence approaches. At the preprocessing level, filtering and normalization are essential to remove noise and drift. Temporal and frequency-domain analyses, often employing Fast Fourier Transform or Discrete Wavelet Transform, are used to identify characteristic patterns associated with slip detection, texture recognition, or dynamic contact events [67]. Dimensionality reduction techniques such as Principal Component Analysis and Linear Discriminant Analysis simplify the interpretation of large tactile datasets obtained from dense sensor arrays, improving computational efficiency and classification accuracy [68].

Machine learning has become a central tool in tactile signal interpretation. Supervised algorithms like Support Vector Machines (SVMs), Random Forests, and k-Nearest Neighbors (k-NN) are applied to classify materials, textures, or grasp states, often achieving accuracy exceeding 90%. More complex

spatiotemporal models, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), allow end-to-end learning from raw tactile sequences to infer physical properties such as object stiffness or surface curvature. When labeled data are not available, unsupervised learning methods such as clustering or self-organizing maps are used to explore novel object categories and tactile features. Reinforcement learning represents an emerging frontier in this field, where tactile feedback directly informs the optimization of control policies during robotic manipulation, allowing autonomous adaptation to changing contact conditions.

The progress in tactile sensing technologies and algorithms has enabled a wide range of practical applications in HMIs and robotic perception. In the context of human–machine interfaces, flexible tactile sensors are being embedded in wearable devices such as gloves, sleeves, and soft patches to capture body motion, gestures. These systems enable intuitive control of prosthetic limbs, assistive robots, and exoskeletons.

In robotics, tactile sensing has become indispensable for improving object perception and manipulation. High-resolution tactile arrays allow robots to infer local surface properties such as roughness, hardness, and thermal conductivity, while machine learning models assist in recognizing object identity and pose [67]. Tactile feedback enables the detection of incipient slip and the control of grasp stability through dynamic adaptation of grip forces. MEMS-based force and pressure sensors integrated into robotic fingers, palms or grippers provide tri-axial measurement capabilities that are essential for dexterous manipulation and delicate handling of fragile objects [66]. The combination of tactile sensing and compliant actuation has also advanced the development of soft robotics, where deformable manipulators equipped with distributed tactile skins can adaptively interact with unstructured environments.

Biosignal sensors

Bionic sensors enable the direct translation of human physiological or biomechanical activity into electrical signals that can be interpreted and acted upon by machines, forming an essential bridge between biological and artificial intelligence. Among the various types of bionic sensors, bioelectrical and biomechanical modalities have attracted particular attention.

Bioelectrical sensors such as electroencephalography (EEG) and electromyography (EMG) are the most established technologies in this field. EEG sensors measure the brain’s cortical activity through electrodes positioned on the scalp, typically detecting low-amplitude signals in the microvolt range that correspond to motor imagery or cognitive intention. These signals capture the earliest representation of a user’s motor command before any physical motion occurs, making them especially valuable for applications that require anticipatory control or cognitive-state monitoring. EMG sensors, by contrast, detect the electrical potentials generated by muscle fibers during contraction. They provide a direct indication of the level of muscular activation and the nature of the motor

task being executed. Surface EMG (sEMG) systems, in particular, have become widespread due to their non-invasive nature, low cost, and suitability for wearable configurations [69].

Beyond electrical sensing, biomechanical sensors such as mechanomyography (MMG) and force myography (FMG), play an increasingly significant role in interpreting human motion. MMG and FMG detect mechanical vibrations and force variations associated with muscle activity. These sensors enhance robustness in dynamic environments where electrical biosignals may suffer from interference or motion artifacts [69].

The interpretation of these biosignals relies heavily on advanced signal processing and machine learning algorithms. Since EEG and EMG are inherently non-linear, non-stationary, and subject to considerable inter-individual variability, their processing demands both robust feature extraction and adaptive classification frameworks. For EMG, time-domain features such as root mean square, mean absolute value, waveform length, and integrated EMG (iEMG) are widely used to quantify the intensity and complexity of muscle activity [70]. These metrics are computationally efficient and correlate strongly with muscle force. EEG signals, in contrast, are typically analyzed in the frequency domain through power spectral density, empirical mode decomposition, discrete Wavelet Transform, and wavelet packet decomposition [71].

At the classification stage, support vector machines (SVM) remain one of the most effective models for biosignal interpretation due to their ability to handle high-dimensional, non-linear feature spaces with limited training data. Fahmi et al. reported an accuracy of 83.5% in predicting grip force levels using an SVM with a Gaussian kernel [72], while Cao et al. achieved comparable results in discriminating left and right motor imagery using EEG data [73]. Neural network models have also gained prominence; backpropagation networks and multilayer perceptrons can learn complex mappings between biosignal features and motion classes, offering superior performance in multi-gesture recognition. Cao and colleagues demonstrated this approach in real-time hand gesture classification based on EMG, achieving over 93% mean accuracy [73]. More recently, probabilistic methods such as Gaussian process regression (GPR) have been introduced to estimate continuous human motion trajectories and predict future intentions, as in the hierarchical human–robot collaboration framework developed by Meng et al. [74]. The integration of these models into real-time systems has been supported by optimized data pipelines and low-latency communication architectures, ensuring that recognition and control remain synchronous with human actions.

The practical applications of bionic sensors extend across several domains, ranging from assistive and rehabilitative robotics to industrial teleoperation and cognitive human–robot collaboration. In human–machine interaction, these sensors allow users to control robotic manipulators, prosthetic limbs, and exoskeletons intuitively through their own physiological activity. Cao et al. successfully demonstrated EEG–EMG-based control of a multi-degree-of-freedom robotic arm over a Bluetooth interface, enabling real-time feedback and

remote interaction [73]. Fahmi et al. applied EMG-driven control to modulate grip force in a KUKA LBR iiwa manipulator, creating a bio-inspired grasping behavior that mimicked human hand dynamics [72]. Such interfaces are essential not only for physical assistance but also for ensuring safety and adaptability in shared workspaces where humans and robots operate in close proximity.

In the rehabilitation context, bionic sensors provide a basis for adaptive therapy and motor recovery. Meng et al. demonstrated that hierarchical motion-intention prediction can synchronize robotic assistance with user movement in collaborative tasks such as co-manipulation, reducing human effort and increasing task efficiency [74]. These findings highlight the potential of predictive biosignal interpretation to enable proactive robot behavior that anticipates rather than merely responds to user commands. Bimbraw emphasized that when latency is minimized and predictive algorithms are applied, robots gain a quasi-perceptual capability to infer and respond to user intentions in real time [75].

2.3 Multi-modal perception

Multimodal perception refers to the integration of heterogeneous sensory data to create a coherent, redundant, and context-rich representation of the environment. Unlike unimodal systems, which depend on a single sensing principle, multimodal frameworks combine complementary inputs to overcome the intrinsic limitations of individual sensors [5]. In human–robot collaboration, this approach allows robots to perceive, interpret, and predict human behavior with greater accuracy and robustness, thereby enabling safe and fluid cooperation in dynamic industrial contexts [9]. Some examples of multimodal system is depicted in Figure 12.

The foundation of multimodal perception lies in sensor fusion; the process through which heterogeneous data streams are aligned, combined, and interpreted as a single, consistent perception of the environment. Modern fusion methods increasingly rely on machine learning. Deep neural networks, recurrent architectures, and attention-based models can learn shared latent representations between modalities, identifying cross-domain patterns that classical rule-based methods cannot capture [9].

The shift from unimodal to multimodal perception provides both quantitative and qualitative advantages. The first and most evident benefit is complementarity: different sensors capture distinct dimensions of information, and their integration fills the perceptual gaps left by each individual modality [5]. A second benefit is redundancy, which increases fault tolerance, if one sensor fails or becomes unreliable, others can maintain system awareness. A third advantage lies in robustness to uncertainty: fusion frameworks combine data probabilistically, filtering out noise and measurement bias, which leads to more stable and trustworthy decisions [6]. Moreover, multimodal integration allows robots to interpret high-level phenomena such as human engagement, fatigue, or intent, rather than merely detecting positions or movements [8].

Despite these advantages, multimodal perception introduces notable challenges. Synchronizing heterogeneous data streams with different sampling rates, resolutions, and coordinate frames remains complex. The fusion process demands substantial computational resources, which can limit real-time performance, particularly in embedded or edge-computing scenarios [6, 9]. Furthermore, the absence of standardized benchmarks and datasets for industrial collaboration makes it difficult to compare or validate fusion methods [8].

In recent years, however, it has become increasingly evident that, despite these challenges, multimodal perception is attracting significant attention within the research community. As reported by Wang et al. [6], the number of studies employing multimodal approaches has steadily increased, confirming that the benefits of this paradigm outweigh its current limitations. A growing number of works now demonstrate how sensor fusion can enhance robustness, interpretability, and human-centered responsiveness across different application domains of human–machine interaction.

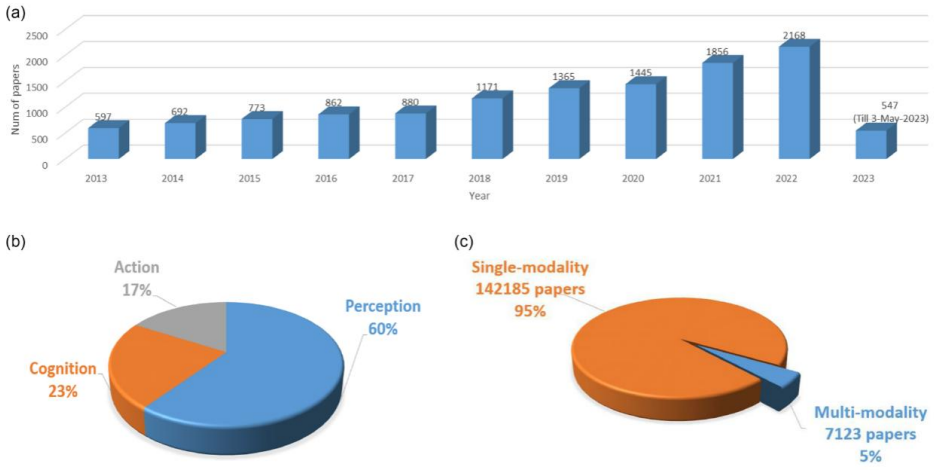

Figure 11 Number of studies related to multimodal human machine -robot interaction [6]

Several representative examples from the literature highlight these benefits. Many multimodal systems rely on a vision sensor as the primary modality, complemented by secondary sources that provide additional spatial or semantic information. For instance, Wang et al. [76] proposed a gesture recognition architecture that fuses visual data from a camera with deformation signals obtained from soft, stretchable sensors mounted on the user's fingers. The visual and somatosensory data were jointly processed through neural networks, enabling the system to achieve substantially higher recognition accuracy than unimodal visual or tactile approaches alone. This example demonstrates how fusing spatial and proprioceptive cues can capture both external appearance and internal deformation during motion, leading to richer gesture representation.

Similarly, Lee et al. [77] combined stereo vision and inertial data to reconstruct three-dimensional hand motion with improved robustness to occlusion and interference. Their visual–inertial fusion method maintained accurate tracking even when one modality became unreliable, such as during hand–object contact or

partial visibility. This type of integration is particularly relevant in collaborative robotics, where hands and tools often move unpredictably within shared workspaces.

Another field where multimodal fusion has shown remarkable improvements is speech recognition. Song et al. [78] developed an audio–visual speech recognition system based on a sparse transformer network that combines acoustic and lip-motion information. The fusion of audio and video streams significantly reduced word error rates, demonstrating that combining modalities can provide complementary information in noisy environments, an important feature for industrial or service robots operating amid background sound.

Finally, Ngai et al. [79] explored emotion recognition using multimodal fusion of heterogeneous biosignals. In their study, EEG data were integrated with RGB or depth images using convolutional neural networks. Both combinations improved recognition accuracy compared to single-modality models, confirming that emotional state estimation benefits from combining internal physiological cues with external facial expressions.

These examples collectively illustrate how multimodal perception not only enhances robustness and accuracy but also expands the range of perceptual capabilities available to robotic systems. From recognizing gestures and tracking motion to interpreting speech and emotional cues, sensor fusion provides a unified mechanism for bridging low-level sensory information with high-level semantic understanding.

In summary, the evolution from unimodal to multimodal perception marks a decisive step toward more intelligent and adaptive human–robot interaction. The fusion of vision, inertial, tactile, and physiological modalities allows robots to develop a richer understanding of both the environment and the human operator, paving the way for safe, intuitive, and context-aware collaboration. This progression also sets the foundation for the methodology developed in the following chapters, which focuses on multimodal tracking through the fusion of event-based vision and inertial sensors, aiming to exploit the complementary characteristics of these sensing modalities to achieve high-speed, low-latency perception in dynamic scenarios.

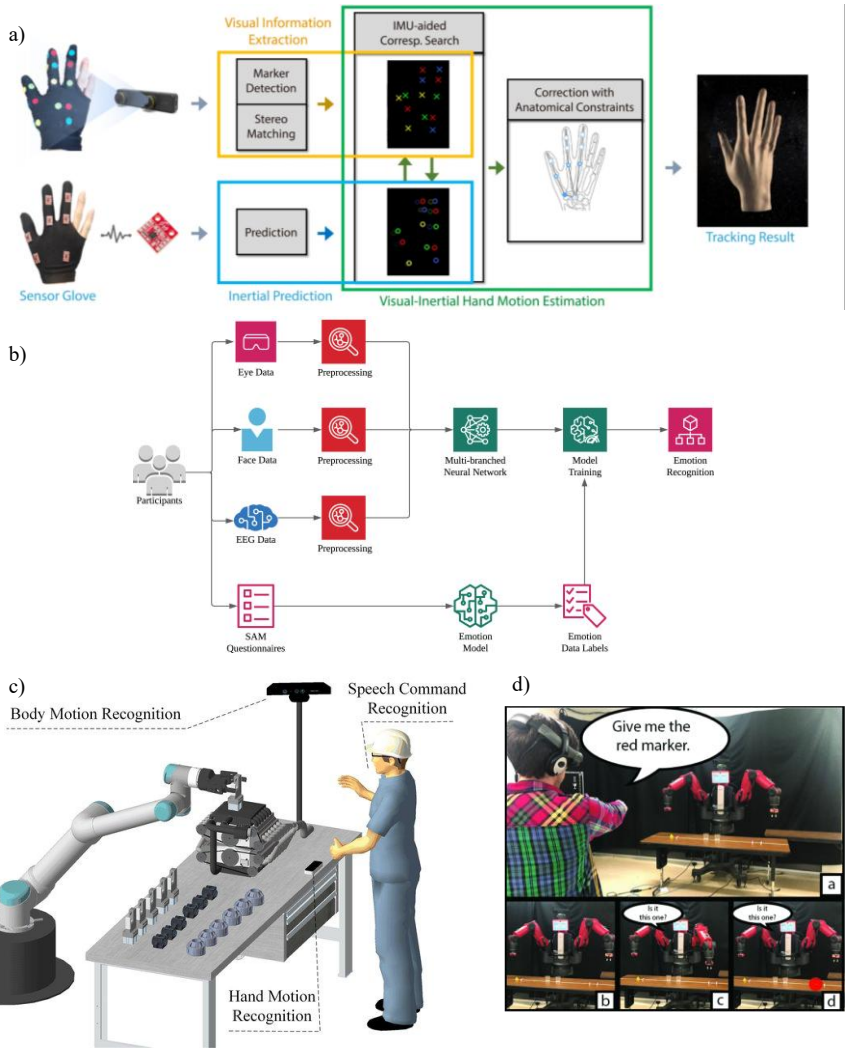

Figure 12 Representative applications of multimodal perception in human–machine and human–robot interaction. a) Visual–Inertial Skeleton Tracking combining a sensor glove equipped with multiple IMUs and passive visual markers with a head-mounted stereo camera [77]. b) Multimodal emotion recognition pipeline integrating eye data and EEG signals with facial information to improve recognition accuracy [79]. c) Multimodal human–robot collaboration scenario enabled by the integration of speech command recognition, hand motion recognition, and full-body motion recognition [61]. d) Multimodal interaction via speech, eye gaze, and pointing gestures .

Chapter 3

Human arm tracking

Accurate and reliable perception of human motion is a fundamental requirement in modern human–robot collaboration scenarios, where humans and robots share the same workspace and cooperate on common tasks. In such settings, the robot must continuously perceive the operator's movements to ensure safety, anticipate intentions, and adapt its behavior in real time. Among the various body parts involved in interaction, the upper limb plays a central role, as it is directly responsible for manipulation, gestural communication, and physical interaction with the environment.

Tracking the human arm with sufficient accuracy and low latency enables a wide range of collaborative functionalities, including collision avoidance [80], motion prediction [81], gesture-based control [35], and cooperative task execution [77]. However, achieving robust arm tracking in dynamic and unstructured environments, such as industrial workspaces, remains a challenging problem. Existing solutions often face a trade-off between accuracy, responsiveness, robustness, and practical deployability, which limits their applicability outside controlled laboratory conditions [82].

Conventional vision-based systems based on standard frame cameras can provide detailed spatial information, but their use in collaborative robotic scenarios presents several limitations. Frame-based cameras operate by acquiring full images at fixed intervals, which introduces inherent latency due to exposure time and sequential processing stages, including image preprocessing, feature extraction, and pose estimation. Even at relatively high frame rates, the end-to-end latency of the perception pipeline can become significant, reducing the system's ability to promptly react to human motion [20].

Moreover, frame-based cameras continuously acquire large amounts of visual information, most of which is irrelevant for motion tracking. This redundancy leads to high bandwidth and storage requirements and increases computational load, as additional processing is required to isolate the arm and extract motion-related features. These aspects are particularly critical in embedded and edge

robotic systems, where computational resources and power availability are limited. In addition, standard cameras are sensitive to changes in illumination, reflections, and contrast variations, which are common in industrial environments and can degrade the reliability of visual perception [20].

Inertial Measurement Units represent a complementary sensing modality, as they provide high-frequency, occlusion-free motion information that is independent of lighting conditions. However, inertial tracking alone suffers from cumulative drift and sensitivity to dynamic accelerations, which progressively degrade positional accuracy over time. As a result, unimodal tracking approaches, whether vision-based or inertial, are generally insufficient to meet the requirements of real-world human–robot collaboration [49].

RGB-D cameras represent one of the most widely adopted sensing solutions for human tracking, since they provide synchronized color and depth information and enable direct three-dimensional reconstruction of the operator's body configuration. As discussed in Chapter 2, RGB-D-based tracking typically relies on a dense processing pipeline that includes depth acquisition, filtering, RGB-depth registration, human segmentation, pose estimation, skeleton fitting, and temporal smoothing. Although this approach can achieve centimetre-level accuracy in controlled conditions and benefits from the maturity and relatively low cost of commercial depth cameras, its effectiveness in dynamic human–robot collaboration scenarios remains constrained by different factors. The tracking performance depends on the visibility of the relevant body segments, the quality of the depth map, the camera viewpoint, and the robustness of the pose-estimation algorithm. Moreover, the need to process full RGB-D frames at each acquisition step introduces a non-negligible computational cost and limits the temporal responsiveness of the perception system, whose update rate is commonly tied to frame-based acquisition frequencies. These limitations are particularly relevant for upper-limb tracking, where fast movements, self-occlusions, and partial visibility of the elbow and wrist may degrade the reliability of the reconstructed pose. Therefore, while RGB-D cameras remain a mature and effective solution for many monitoring and interaction tasks, alternative sensing strategies are needed when low latency, reduced data redundancy, and responsiveness to human motion are required.

Event-based vision sensors offer an alternative sensing paradigm that addresses several of the aforementioned limitations of standard cameras. Instead of capturing full image frames, each pixel in an event camera asynchronously reports changes in brightness as soon as they occur. This results in very high temporal resolution and low latency, while significantly reducing data redundancy and bandwidth requirements. By focusing exclusively on changes in the scene, event cameras naturally emphasize motion-related information, making them particularly suitable for real-time perception. Furthermore, their high dynamic range allows reliable operation under challenging lighting conditions.

Despite these advantages, monocular event cameras do not provide direct depth information and rely on the temporal and spatial structure of events to infer motion. This limitation motivates their integration with complementary sensing

modalities. In this context, sensor fusion with inertial measurements represents a promising solution, as IMUs provide direct 3D orientation information that can compensate for the missing depth cues of event-based vision.

To address these challenges, this chapter presents a multimodal human arm tracking framework that combines data from two inertial sensors and a single event-based vision sensor through a Kalman filter–based fusion strategy. By exploiting the complementary characteristics of inertial sensing and event-based vision, the proposed system aims to achieve accurate, low-latency estimation of elbow and wrist positions using a compact and computationally efficient sensor setup suitable for human–robot collaboration.

The chapter introduces the complete tracking pipeline, including the kinematic modeling of the human arm, the event-based optimization strategy, the sensor fusion algorithm, and an experimental validation. Through this approach, the chapter demonstrates how combining inertial sensing with event-based vision can provide a practical, robust, and scalable solution for upper-limb tracking in collaborative robotic applications.

3.1 System overview

The proposed system integrates two complementary sensing modalities, inertial and event-based vision, to achieve accurate and low-latency tracking of the human right upper limb, the overall architecture is illustrated in Figure 13. It is composed of two MEMS-based Inertial Measurement Units mounted on the operator's upper arm and forearm, and a DAVIS240C event-based vision sensor positioned to include the operator within its field of view. A fiducial ArUco marker is attached to the torso to establish a common reference frame between the camera and the inertial sensors, ensuring that all measurements are expressed in a shared coordinate system.

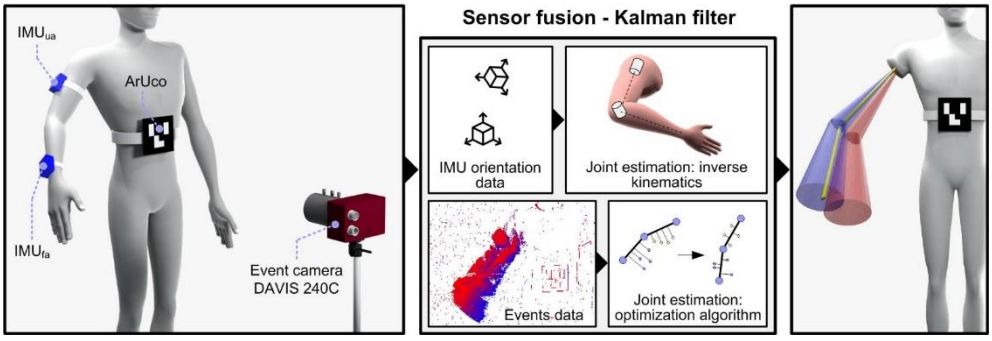

Figure 13 Overview of the proposed joint tracking system. Human arm joints value are estimated using data from two IMUs and an event camera and then fused together with a Kalman filter. In the right image the fusion process is depicted, in blue the posture estimated with the IMUs and its covariance, in red the posture estimated with the event camera and its covariance and, lastly, in yellow the fused pose.

Each sensing modality provides an independent estimation of the arm configuration based on the kinematic model of the upper limb. Exploiting this model allows the system to incorporate anatomical motion constraints and to define a consistent reference framework both for vision-based estimation and for the subsequent sensor fusion stage. The IMU subsystem computes the joint angles by means of an inverse kinematics algorithm using orientation data from the two inertial units. Conversely, the vision subsystem processes the asynchronous events captured by the event camera to extract motion-related features and computes an optimized configuration of the arm through a cost minimization procedure that aligns the kinematic model with the detected events.

The two pose estimates are then combined through a Kalman filter, which merges the high-frequency and continuous nature of inertial data with the spatial accuracy and low-latency characteristics of event-based vision. This integration mitigates the weaknesses of each modality, drift and cumulative error in the IMUs, and partial observability in the event camera, resulting in a stable and precise estimation of the wrist and elbow positions.

Compared with conventional RGB-D or optical tracking systems, the proposed approach offers several advantages. Event cameras generate data only when brightness changes occur, meaning that only relevant motion information is acquired. In contrast, RGB-D cameras must first capture full-frame 3D data and then extract the region corresponding to the human arm, introducing additional computation and delay. By processing only the required information, the event-based vision system significantly reduces both computational load and end-to-end latency. Furthermore, the use of IMUs provides high-frequency motion data that can be acquired and processed synchronously with the event stream, enabling a reactive and responsive tracking system suitable for dynamic human–robot interaction.

Finally, the sensor fusion approach allows the system to achieve accuracy levels comparable to state-of-the-art multimodal tracking methods, while maintaining reduced computational complexity and faster response times.

3.2 Human arm kinematic model

The kinematic model of the human upper limb provides the mathematical and biomechanical foundation for both the inertial and vision-based tracking subsystems. It enables a consistent representation of the arm's spatial configuration while enforcing realistic motion constraints derived from human anatomy. In this work, the model is defined with reference to the right upper limb, which is consistently considered throughout the tracking, modeling, and experimental validation phases. The model captures the essential movements of the shoulder and elbow through a simplified yet effective structure that balances computational efficiency and biomechanical plausibility.

As illustrated in Figure 14, the proposed kinematic model represents the human upper body using three main segments: the torso, the upper arm, and the forearm. The motion of the arm can be described with 5 degrees of freedom

(DoF): three at the shoulder (elevation plane, elevation angle, and humerus rotation) and two at the elbow (flexion–extension and forearm pronation–supination). To formulate the arm motion as a serial kinematic chain composed exclusively of revolute joints, additional virtual links are introduced. In particular, three virtual bodies are inserted between joints 1-2, 2-3, and between joints 4 and 5. These virtual links do not correspond to physical anatomical segments and have no associated length; instead, they are introduced to decouple the rotational degrees of freedom and to allow the superposition of reference frames. This modeling choice enables each rotational motion to be associated with a distinct joint while preserving a clear and structured kinematic formulation.

In addition to the reference frames associated with the physical and virtual links, an auxiliary reference frame, denoted as frame w, is introduced to represent the wrist location. Frame w is rigidly attached to the forearm and is defined through a fixed transformation with respect to frame 5, corresponding to a translation along the forearm axis with magnitude equal to the forearm length.

With this representation, the shoulder motion is described by three rotational degrees of freedom, while the elbow contributes two degrees of freedom. The overall kinematic structure allows the shoulder to be modeled as an equivalent spherical joint and the elbow as an equivalent universal joint

To describe the model the Modified Denavit–Hartenberg (MDH) convention is adopted. Each joint rotation is denoted as q_i, corresponding to rotation about the local z_i axis.

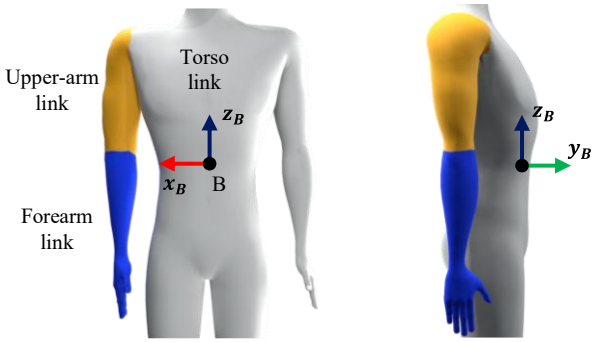

Figure 14 Human right arm model. The model is composed of three links, Torso (white), Upper-arm (gold) and Forearm (blue) and five joints, three for the upper arm motion and two for the forearm.

The model is defined with respect to a common base reference frame, denoted as frame B, which is rigidly attached to the torso and assumed to remain stationary during operation. All reference frames positions and orientations are expressed relative to this base frame, ensuring a consistent coordinate reference for both sensing modalities and for the sensor fusion process.

To ensure adaptability, the model allows for customized segment lengths corresponding to each operator's anthropometric measurements. This personalization increases the accuracy of spatial reconstruction and enables adaptation to different users. To maintain computational simplicity, some approximations are introduced. In particular, the shoulder, elbow, and wrist centers are assumed to be collinear. This reduces anatomical fidelity slightly but

preserves sufficient realism for motion-tracking applications while simplifying anthropometric measurements. The model is defined in the anatomical reference position (Figure 15), where the operator stands upright with the torso aligned to the vertical axis and the palms facing inward. This standardized configuration defines the zero reference posture, from which all joint rotations are measured. The choice of this non-natural posture helps the operator concentrate during calibration, improving repeatability. Figure 15 also illustrates the complete set of reference frames expressed with respect to this reference position, allowing the relative orientations and spatial locations of all coordinate systems to be clearly understood.

Overall, this model serves a dual purpose: it acts as a geometric framework for forward and inverse kinematics calculations, and as a biomechanical constraint that ensures the estimated poses remain consistent with feasible human motion. The following section details the definition of the coordinate frames, the MDH parameterization, and the forward kinematic formulation used to compute the spatial position and orientation of each joint.

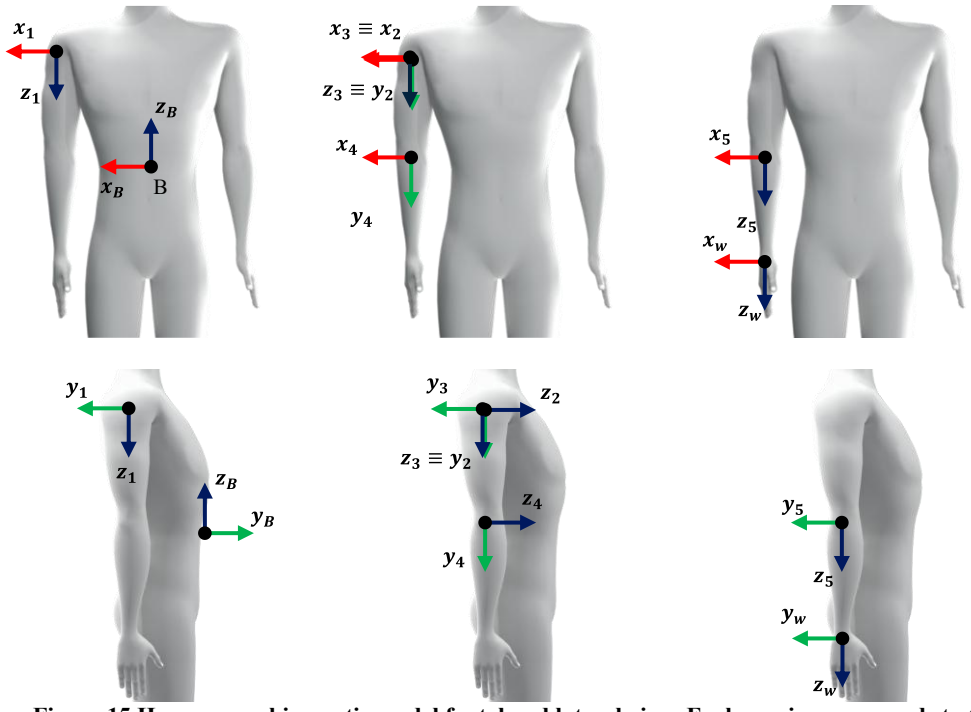

Figure 15 Human arm kinematic model fontal and lateral view. Each z-axis corresponds to the rotation axis of its associated joint

The relationship between coordinate frames is defined using rotation and transformation matrices. The transformation matrix will be denoted as ${}^{?}T_{!}$, where the right subscript (!) refers to the described frame and the left superscript (?) indicates the reference frame in which the transformation is expressed. Each transformation matrix belongs to the Lie group $SE(3)$. Similar to the transformation matrix, rotation matrix will be denoted with ${}^{?}R_{!}$ and they belong to the Lie group $SO(3)$

The torso reference frame (B) is positioned in the central abdominal area, with the x_B and z_B axes lying parallel to the anatomical frontal plane. The z_B axis is directed vertically upward, the x_B axis is horizontal pointing the right side. The y_B axis completes a righthanded coordinate system. The references system 1 (associated with the first rotation axis) is located at the shoulder center, with the z_1 axis directed vertically, pointing downward, while the x_1 axis is horizontal again pointing to the right. The homogeneous transformation between the base references frame and the reference frame 1 is expressed as:

$$ {}^{B}T_1 = \begin{pmatrix} \mathrm{c}(q_1) & -\mathrm{s}(q_1) & 0 & x_{sh} \\ -\mathrm{s}(q_1) & -\mathrm{c}(q_1) & 0 & y_{sh} \\ 0 & 0 & -1 & z_{sh} \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (1) $$

where q_1 is the rotation value of the first joint and (x_{sh}, y_{sh}, z_{sh}) are the position vector components from the base references frame to the references frame 1 expressed in the base reference frame. The remaining transformations are described by the MDH parameter $\alpha_{i-1}, a_{i-1}, d_i, \theta_i$ summarized in Table 2.

Table 2 Denavit–Hartenberg Parameter

Link	α_{i-1}	a_{i-1}	d_i	θ_i
2	90°	0	0	q_2
3	-90	0	0	q_3
4	90	0	Upper-arm length	q_4
5	-90	0	0	q_5
w	0	0	Forearm length	0

For each pair of consecutive reference frame, the homogeneous transformation is given by Eq 2.

$$ {}^{i-1}T_i = \begin{pmatrix} \mathrm{c}(\theta_1) & -\mathrm{s}(\theta_1) & 0 & a_{i-1} \\ \mathrm{s}(\theta_1)\,\mathrm{c}(\alpha_{i-1}) & \mathrm{c}(\theta_i)\,\mathrm{c}(\alpha_{i-1}) & -\mathrm{s}(\alpha_{i-1}) & -d_i\,\mathrm{s}(\alpha_{i-1}) \\ \mathrm{s}(\theta_1)\,\mathrm{s}(\alpha_{i-1}) & \mathrm{c}(\theta_i)\,\mathrm{s}(\alpha_{i-1}) & \mathrm{c}(\alpha_{i-1}) & d_i\,\mathrm{c}(\alpha_{i-1}) \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2) $$

By chaining the individual transformations, the forward kinematics can be written as:

$$ {}^{B}T_j = \prod_{i=1}^{j} {}^{i-1}T_i(q_i) \quad (3) $$

In particular:

$$^{B}T_3 = {}^{B}T_1\ {}^{1}T_2\ {}^{2}T_3 \tag{4}$$

$$^{3}T_5 = {}^{3}T_4\ {}^{4}T_5 \tag{5}$$

$$^{B}T_w = {}^{B}T_3\ {}^{3}T_5\ {}^{5}T_w \tag{6}$$

It is important to note that reference frame 3 is associated with the upper-arm link and its position corresponds to the shoulder, while reference frame 5 is associated with the forearm link and its position corresponds to the elbow, meanwhile frame w is associated with the forearm but located at the wrist.

3.3 IMU-based tracking system

The IMU-based tracking subsystem estimates the angular configuration of the human upper limb using orientation data acquired from the two IMUs. One IMU is mounted on the upper arm and the other on the forearm. This configuration allows the system to capture the three-dimensional orientation of each limb segment and, through the inverse kinematics formulation, compute the joint angles defined in the kinematic model described in the previous section.

At each time step, each IMU provides its orientation with respect to its own global reference frame. This global frame is not known a priori and is assigned autonomously at the sensor's initialization; therefore, the global frames of the two IMUs are not necessarily aligned or congruent. The reference frame associated with the IMU on the upper arm is denoted as ua, and that of the forearm as fa, while the corresponding global frames are indicated as G_1 e G_2 respectively. Accordingly, the orientations measured by the sensors can be expressed as $^{G_1}R_{ua}$ and $^{G_2}R_{fa}$. To make these data usable within the kinematic model, the orientations must be expressed in appropriate reference frames. The following assumptions are therefore introduced:

The orientations $^{3}R_{ua}$ and $^{5}R_{fa}$ between each IMU and the related rigid body are constant, and they are set during mounting.

The initial orientations of the IMUs with respect to the torso frame B, denoted as $^{B}_{0}R_{ua}$ and $^{B}_{0}R_{fa}$, are known. The initial state is denoted with the left subscript 0.

Under these assumptions, it is possible to compute the transformations that relate the base reference frame to the two global IMU frames, as reported in Eqs. (7) and (8). By combining all these transformations, the sensor orientations can be expressed in the appropriate coordinate systems required for inverse kinematics, specifically: the orientation of the upper-arm link with respect to the base frame, and the orientation of the forearm link with respect to frame 3, as defined in Eqs. (9) and (10). The IMUs orientation with respect the different reference frames is depicted in Figure 16 in two different configurations, the anatomical reference posture and a generic one.

$$ {}^{B}_{0}R_{G_1} = {}^{B}_{0}R_{ua}\,{}^{ua}_{0}R_{G_1} \tag{7} $$

$$ {}^{B}_{0}R_{G_2} = {}^{B}_{0}R_{fa}\,{}^{fa}_{0}R_{G_2} \tag{8} $$

$$ {}^{B}R_3 = {}^{B}_{0}R_{G_1}\,{}^{G_1}R_{ua}\,{}^{ua}R_3 \tag{9} $$

$$ {}^{3}R_5 = {}^{3}R_B\,{}^{B}_{0}R_{G_2}\,{}^{G_2}R_{fa}\,{}^{fa}R_5 \tag{10} $$

Given the orientation matrices derived from the IMU data, the corresponding joint angles q_i are obtained by solving the inverse kinematics problem of the five-degree-of-freedom arm model. The orientation of the upper arm relative to the torso, ${}^{B}R_3$, provides the three shoulder joint angles q_1, q_2, q_3, while the orientation of the forearm relative to the upper arm, ${}^{3}R_5$, provides the two elbow joint angles q_4, q_5.

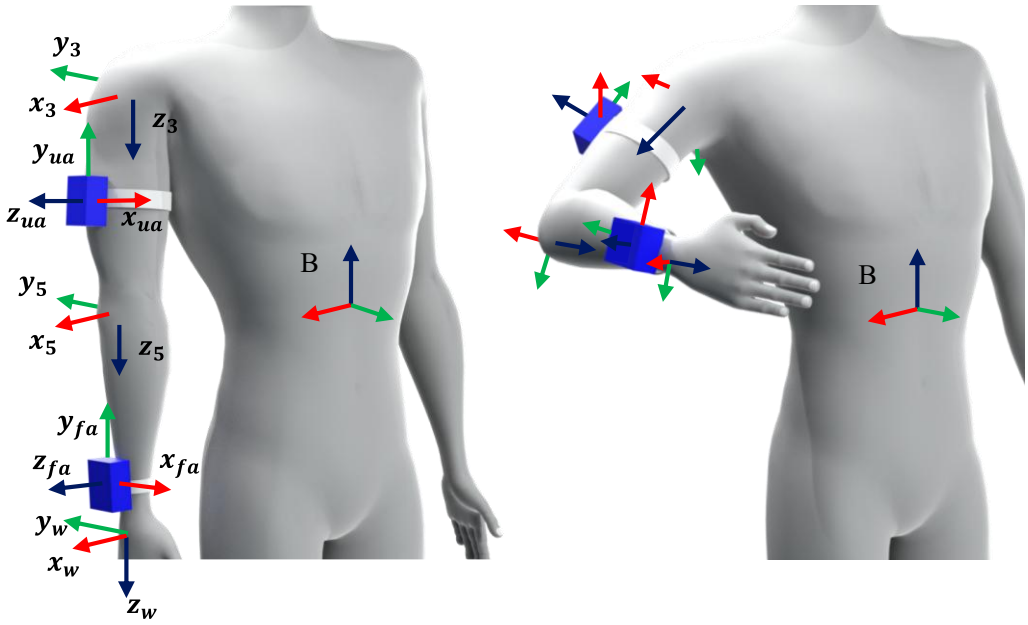

Figure 16 IMU and links orientation in the anatomical referances configuration and in a generic one.

The analytical relations used to compute the joint variables are reported in Eqs. (11)-(15) where $R(m,n)$ indicates the element of the rotation matrix in the m-th row and n-th column.

$$ q_{t,1} = \arctan2\left(\frac{{}^{B}R_3(2,3)}{\sin q_{t,2}}, -\frac{{}^{B}R_3(1,3)}{\sin q_{t,2}}\right) \tag{11} $$

$$ q_{t,2} = \pm\arccos\left({}^{B}R_3(3,3)\right) \tag{12} $$

$$ q_{t,3} = \arctan2\left(-\frac{{}^{B}R_3(3,1)}{\sin q_{t,2}}, \frac{{}^{B}R_3(3,2)}{\sin q_{t,2}}\right) \tag{13} $$

$$ q_{t,4} = \arctan2\left(-\,{}^{3}R_5(1,3),\ {}^{3}R_5(3,3)\right) \tag{14} $$

$$ q_{t,5} = \arctan2\left({}^{3}R_5(1,2),\ {}^{3}R_5(2,2)\right) \tag{15} $$

Among the two possible analytical solutions for each trigonometric inversion, only the configuration consistent with physiological constraints, such as the natural range of motion of the human shoulder and elbow, is selected. This ensures biomechanical plausibility of the estimated configuration. The resulting vector of estimated joint angles at time t is expressed as:

$$\mathrm{q}_{\mathrm{t}}^{\mathrm{IMU}} = [q_1, q_2, q_3, q_4, q_5] \tag{16}$$

This vector represents the complete pose of the arm according to the inertial measurements and is used as an independent input for the sensor fusion module.

3.4 Event-based tracking system

The event-based vision subsystem estimates the configuration of the upper limb by processing the asynchronous brightness-change data provided by an event camera. Unlike conventional frame-based cameras, which record entire intensity images at fixed intervals, the event camera generates data only when a pixel detects a change in brightness. Each pixel operates independently and outputs an event with its spatial coordinates (x_k, y_k), polarity (pol) and timestamp (t), defining the events as $\mathrm{e_k} = (x_k, y_k, pol_k, t_{sk})$. This sensing principle enables microsecond temporal resolution, low latency, and reduced data bandwidth, making event-based vision particularly suitable for real-time motion tracking in human–robot collaboration. The processing pipeline of the vision subsystem consists of four main stages: event filtering, undistortion, event assignment, and optimization of the arm configuration. Each step progressively refines the raw event data to obtain reliable geometric information that can be used for pose estimation with the kinematic model. A schematic representation of this pipeline is shown in Figure 17

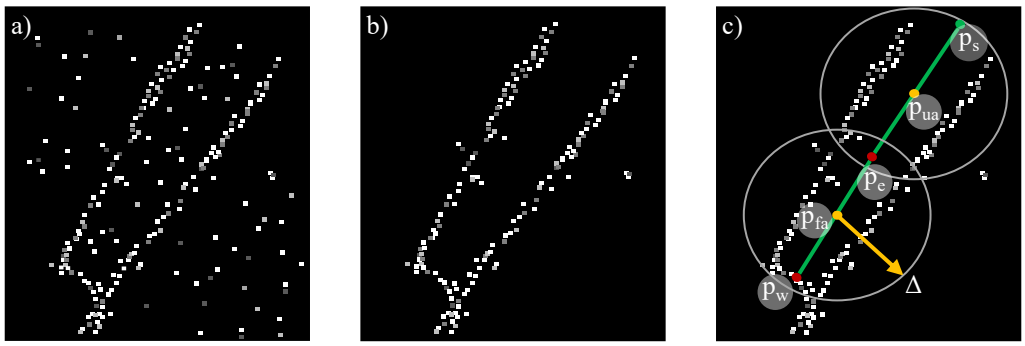

Figure 17 Illustration of event camera data processing steps. a) The original raw events are b) filtered and undistorted. c) The resulting processed events are assigned to the set of upper arm points or to the set of forearm points according to the area defined by landmarks on the kinematic model of the arm.

The first processing step implements the filtering of the raw data acquired from the event camera. This stage is crucial, as event cameras are inherently affected by background activity noise, caused by pixel leakage currents or threshold mismatches, which produce events that are not spatially or temporally correlated with actual motion. To remove this noise, a modified version of the Time Surface filtering method was employed [83], in which an Event-Count Surface (ECS) is used instead of the traditional time surface. The ECS relies on the observation that noise-related pixels are typically isolated in both space and time, whereas motion-generated events occur in spatially and temporally coherent clusters. Therefore, if a single event has no neighboring events within its vicinity, it is likely to originate from noise rather than real motion. To implement this principle, the ECS divides the pixel array into local cells of size $s \times s$, where $s = 2^b$. The resulting Event-Count Surfaces is defined as $S \in \mathbb{N}^{\frac{W}{s} \times \frac{H}{s}}$, where W and H are di camera resolution. Each cell stores the number of events detected within a short time window. The subsampled coordinates u_k and v_k of the cell to which each event belongs are computed as:

$$u_k = x_k \gg b;\ v_k = y_k \gg b \tag{17}$$

Where the operator "≫" denotes the right bit-shift, equivalent to dividing the event coordinate by s and keeping only the integer part.

A second offset surface S' is generated by shifting the subsampling grid by half a cell to improve spatial correlation between adjacent regions. The corresponding subsampled coordinates are calculated as:

$$u'_k = (x_k + b) \gg b;\ v'_k = (y_k + b) \gg b \tag{18}$$

An explanatory illustration of this offset sampling pattern is shown in Figure 18. An event is retained only if the number of neighboring events within its corresponding cells in both S and S' exceeds a predefined threshold n, as reported in (19).

$$S(u_k, v_k) > n \ \wedge\ S'(u'_k, v'_k) > n \tag{19}$$

This spatio-temporal correlation filter effectively suppresses isolated noise events while preserving those associated with coherent motion patterns. The resulting filtered event set, denoted $\mathcal{E}_f$, thus contains only meaningful motion information corresponding to the actual movement of the arm.

Following the filtering stage, the next steps of the tracking algorithm aim to refine the event data to accurately estimate the arm configuration. This process involves two main operations: undistortion of the accumulated events and assignment of each event to the corresponding arm segment, either the upper arm or the forearm. The undistortion procedure corrects the raw event coordinates

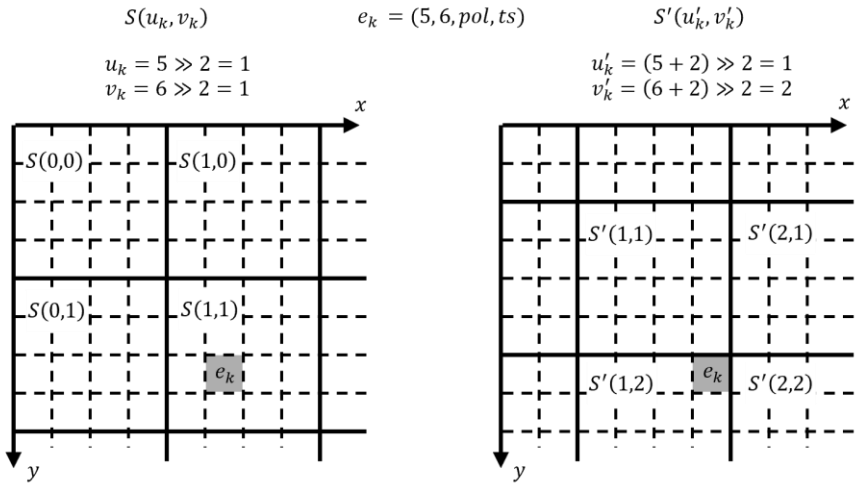

Figure 18 Example of event assignment on the Event-Count Surfaces. The figure illustrates how an event (e_k) with coordinate $x_k = 5$ and $y_k = 6$ is assigned to the corresponding subsampled cells in the surfaces S and S'.

using the intrinsic parameters of the event camera, denoted by the matrix K. Event assignment is then performed according to **Algorithm 1**, which exploits the previously estimated configuration of the kinematic model to guide the spatial separation of events. Using the joint configuration from the previous time step, q_{t-1}, the forward kinematics module computes the transformation matrices ${}^{B}T_1$, ${}^{B}T_4$, and ${}^{B}T_w$, corresponding to the positions of the shoulder, elbow, and wrist, respectively. These three-dimensional poses are projected onto the image plane using the camera parameters to obtain two-dimensional points representing the shoulder (p_s), elbow (p_e), and wrist (p_w) centres, as illustrated in Figure 19.

To assist in the assignment of events, the algorithm computes the geometric centers of the upper arm and forearm segments by averaging the projected joint positions. The point p_{ua} represents the midpoint between the shoulder and elbow, while p_{fa} corresponds to the midpoint between the elbow and wrist. For each detected event, the Euclidean distance between its coordinates and these segment centers is calculated. An event is classified as belonging to a segment if its distance from the corresponding center is below a predefined threshold Δ. This procedure results in two distinct event sets: $\mathcal{E}_{ua}$ for the upper arm and $\mathcal{E}_{fa}$ for the forearm. The threshold Δ defines the segmentation area around each center, represented by the circular regions shown in Figure 17(c). The separated event sets constitute the input for the subsequent optimization stage, which estimates the updated pose of the arm.

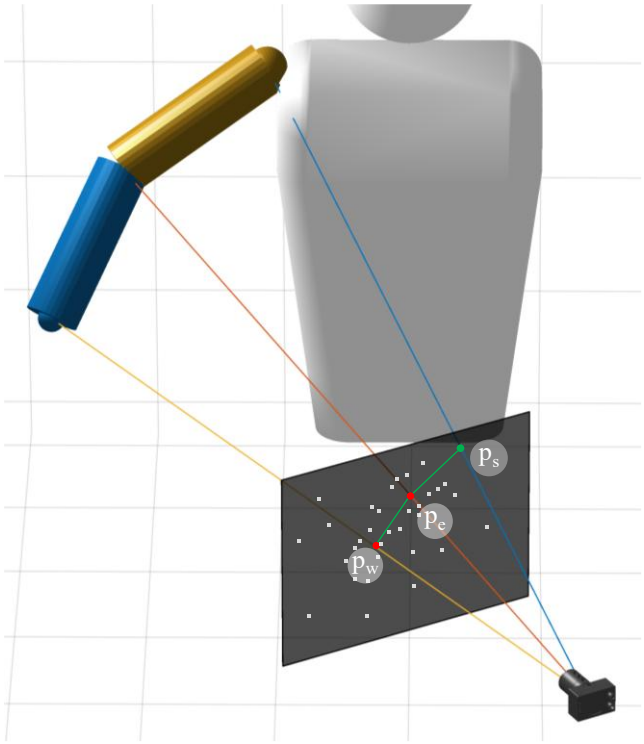

Figure 19 Explanatory illustration of the model projection in event camera plane

Algorithm 1 Undistortion and Assignment of Events

function Assignment ($\mathcal{E}_f$)

$\mathcal{E} \leftarrow Undistort(\mathcal{E}_f, K)$

${}^{B}T_1, {}^{B}T_4, {}^{B}T_w \leftarrow ForwardKinematics(q_{t-1})$

$p_s, p_e, p_w \leftarrow Project({}^{B}T_1, {}^{B}T_4, {}^{B}T_w, K)$

$p_{ua} \leftarrow avg(p_s, p_e)$ Upper arm center

$p_{fa} \leftarrow avg(p_e, p_w)$ Forearm center

$\mathcal{E}_{ua} \leftarrow \mathcal{E}[\, dist(p_{ua}, \mathcal{E}) < \Delta \,]$

$\mathcal{E}_{fa} \leftarrow \mathcal{E}[\, dist(p_{fa}, \mathcal{E}) < \Delta \,]$

return $(\mathcal{E}_{ua}, \mathcal{E}_{fa})$

The optimization procedure estimates the current arm configuration q_t^{evt} using a model-based approach that minimizes the distances standard deviations between the projected arm segments and the events detected by the camera. This approach relies on the assumption that correctly estimated arm poses generate

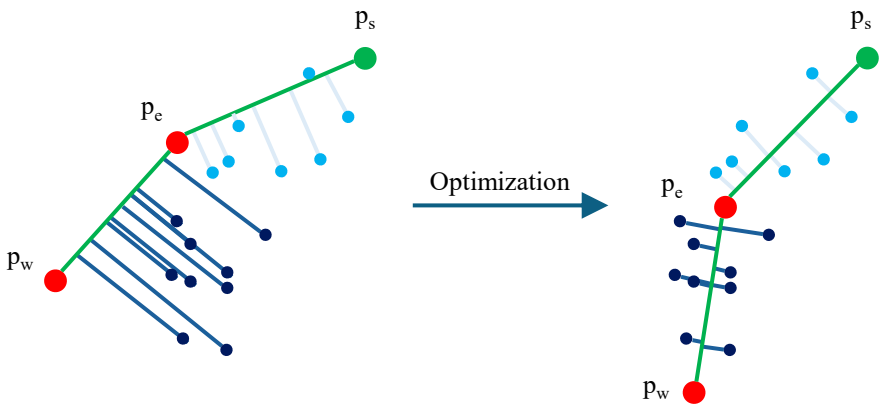

Figure 20 An illustrated example of the event optimization method. The joint configuration is changed until the two segments are in the pose that minimize the standard deviation of the distances between the segments and the events. The light and dark blue points represent events assigned to the upper arm and forearm, respectively.

event distributions that align spatially with the projected model segments, as illustrated in Figure 20. The optimization is further constrained by the physiological range of motion of the shoulder and elbow joints to ensure biomechanically valid configurations. Minimizing the standard deviation ensures that the estimated segment lies at a uniform distance from all the events assigned to it, resulting in an even spatial distribution of points along the modeled limb. In contrast, minimizing other metrics, such as the mean distance, would favor configurations where the segment aligns with the highest concentration of events, which, due to the event camera's nature, often corresponds to one of the arm's visual edges rather than its central axis.

The algorithm is executed only when the sets of events assigned to the upper arm ($\mathcal{E}_{ua}$) and forearm ($\mathcal{E}_{fa}$) contain a sufficient number of elements, depending on the lighting conditions and movement intensity. Three possible scenarios are implemented: full arm optimization, forearm-only optimization, and no optimization.

Arm optimization

When both $\mathcal{E}_{ua}$ and $\mathcal{E}_{fa}$ contain enough events, a complete optimization of all five joint angles is performed. A candidate joint configuration is first defined, and the corresponding homogeneous transformation matrices ${}^{B}T_1$, ${}^{B}T_4$, and ${}^{B}T_w$ are computed through the forward kinematics module. These transformations are then projected onto the image plane using the camera intrinsics K, yielding 2D positions for the shoulder (p_s), elbow (p_e), and wrist (p_w).

Two lines are subsequently defined in the image plane in implicit form $ax + by + c = 0$: one passing through $\mathrm{p_s}$ and $\mathrm{p_e}$ (upper arm) and the other through $\mathrm{p_e}$ and $\mathrm{p_w}$ (forearm). For each event assigned to a segment, the Euclidean distance to its corresponding line is calculated. These distances are collected in the sets $\mathcal{D}_{ua}$ and $\mathcal{D}_{fa}$, representing how well the events align with the projected model.

The optimization aims to find the configuration that minimizes the cost function:

$$\underset{\mathrm{q}_t^{evt}}{\operatorname{argmin}}(std(\mathcal{D}_{ua}) + std(\mathcal{D}_{fa}) + \|W(\mathrm{q}_t^{evt} - \mathrm{q}_{t-1})\|_2) \tag{20}$$

where the first two terms enforce spatial consistency by minimizing the dispersion of events around the predicted kinematic lines, and the third term acts as a temporal regularization factor that penalizes abrupt changes in joint angles. The weighting matrix W assigns different confidence levels to each joint according to its expected mobility and measurement reliability. The diagonal entries of W are empirically tuned, assigning smaller weights to the elbow joints to account for their higher estimation uncertainty.

This formulation ensures that the estimated configuration not only fits the event data but also evolves smoothly over time. The pseudocode of the proposed optimization procedure is reported in **Algorithm 2**.

Algorithm 2 Arm Optimization

$\mathrm{q}_t^{evt} = \underset{\mathrm{q}_t^{evt}}{\operatorname{argmin}}(\mathrm{OPTIMISE}(\mathrm{q}_t^{evt}, \mathrm{q}_{t-1}, \mathcal{E}_{ua}, \mathcal{E}_{fa}))$

function $\mathrm{OPTIMISE}(\mathrm{q}_t^{evt}, \mathrm{q}_{t-1}, \mathcal{E}_{ua}, \mathcal{E}_{fa})$

$\quad {}^{B}T_1, {}^{B}T_4, {}^{B}T_w \leftarrow ForwardKinematics(\mathrm{q}_t^{evt})$

$\quad \mathrm{p_s}, \mathrm{p_e}, \mathrm{p_w} \leftarrow Project({}^{B}T_1, {}^{B}T_4, {}^{B}T_w, K)$

$\quad a_1, b_1, c_1 \leftarrow line(\mathrm{p_s}, \mathrm{p_e})$ Upper arm line parameter

$\quad a_2, b_2, c_3 \leftarrow line(\mathrm{p_e}, \mathrm{p_w})$ Forearm line parameter

$\quad \mathcal{D}_{ua} \leftarrow dist(line(a_1, b_1, c_1), \mathcal{E}_{ua})$ Set of distances

$\quad \mathcal{D}_{fa} \leftarrow dist(line(a_2, b_2, c_3), \mathcal{E}_{fa})$

$\quad$ **return** $(std(\mathcal{D}_{ua}) + std(\mathcal{D}_{fa}) + \|W(\mathrm{q}_t^{evt} - \mathrm{q}_{t-1})\|_2$

No optimization

If the forearm event set $\mathcal{E}_{fa}$ contains enough points while the upper-arm set $\mathcal{E}_{ua}$ does not, and no substantial motion is detected, the system assumes that the arm has remained stationary. Consequently, the configuration is maintained as $\mathrm{q}_t^{evt} = \mathrm{q}_{t-1}$. This prevents the introduction of noise when visual data are insufficient for reliable estimation.

Forearm-only optimization

When $\mathcal{E}_{fa}$ contains sufficient events but $\mathcal{E}_{ua}$ does not, the optimization is restricted to the elbow joints, $q_{t,4}^{evt}$ and $q_{t,5}^{evt}$, while keeping the shoulder

configuration fixed. This approach assumes that no movement has occurred in the proximal joints. The corresponding objective function simplifies to:

$$\underset{\mathrm{q}_t^{evt}}{\operatorname{argmin}}(\ std(\mathcal{D}_{fa})\ +\ \|W(\mathrm{q}_t^{evt} - \mathrm{q}_{t-1})\|_2) \tag{21}$$

This reduced optimization follows the same structure as the full version but omits computations related to the upper arm. It therefore reduces computational complexity while maintaining continuity by leveraging the absence of events as implicit information that the upper arm has not moved.

The optimization is solved iteratively using a bounded gradient-descent method, with a maximum number of iterations set according to real-time constraints. This approach provides a good trade-off between computational efficiency and estimation accuracy. On average, the optimization converges within a few hundred iterations, keeping the overall processing time compatible with the high-frequency output of the event camera. The estimated configuration q_t^{evt} is then fused with the inertial estimation within the Kalman-based framework described in the following.

3.5 Sensor Fusion Framework

The fusion of the joint angle estimates obtained from the IMUs and the event-based vision subsystem is performed through a Kalman filter framework, which exploits the complementary characteristics of the two sensing modalities. While the IMUs provide high-frequency orientation data with potential drift, the event camera contributes spatially accurate but bidimensional measurements. The Kalman filter offers a method to integrate these heterogeneous data sources by recursively estimating the most probable joint configuration based on sensor

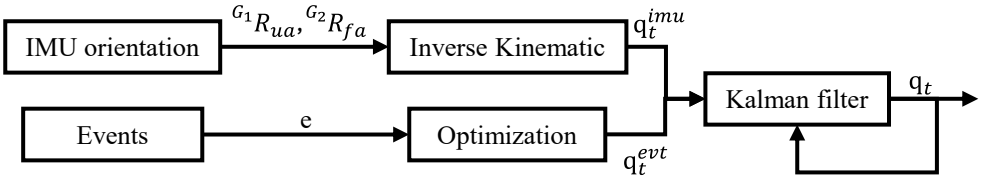

Figure 21 Kalman framework and logic flow

uncertainty and system dynamics. Figure 21 illustrates the overall fusion architecture.

The dynamic behavior of the system is modeled using a discrete linear state-space formulation. The state vector $\mathrm{q}_t \in \mathbb{R}^5$ represents the joint angles of the shoulder and elbow at time t. Its evolution is described by a first-order dynamic model:

$$\mathrm{q}_t = A\mathrm{q}_{t-1} + \mathrm{w}_t \tag{22}$$

where A is the state transition matrix and $\mathrm{w}_t \sim \mathcal{N}(0, Q_t)$ is the process noise, modeling random variations in motion and modeling uncertainty. Since the arm configuration changes gradually between consecutive steps, a quasi-static model is assumed with $A = I_5$.

The measurement model combines the IMU-based and event-based estimates into a single observation vector:

$$\mathrm{z}_t = H\mathrm{q}_t + \mathrm{r}_t \tag{23}$$

where H is the observation matrix, and $\mathrm{r}_t \sim \mathcal{N}(0, \Theta)$ represents measurement noise assumed to be Gaussian and uncorrelated between sensors.

At each time step, the observation vector $\mathrm{z}_t \in \mathbb{R}^{10}$is formed by stacking the two independent estimates:

$$\mathrm{z}_t = [\mathrm{q}_t^{imu}, \mathrm{q}_t^{evt}]^\top \tag{24}$$

where the first five components correspond to the IMU-based estimation and the last five to the event-based estimation.

To correctly weight each sensor contribution, the measurement noise covariance matrix Θ is structured in block-diagonal form:

$$\Theta = \begin{bmatrix} \Theta_{imu} & 0 \\ 0 & \Theta_{evt} \end{bmatrix} \tag{25}$$

- For the IMUs, the diagonal covariance matrix Θ_{imu} encodes higher uncertainty in the forearm joints (25°) compared to the upper arm (15°).
- For the event camera, the covariance matrix Θ_{evt} assumes a noise standard deviation of 5° for the shoulder joints and 10° for the elbow joints, reflecting reduced confidence in the estimation of distal joints.

The process noise covariance matrix Q_t represents the uncertainty of the motion model and is defined as a diagonal matrix with a baseline noise of 3°. This matrix is adaptively scaled based on the number of detected events: when more events are available, the model uncertainty increases to account for potentially faster and less predictable motions.

The recursive Kalman filter consists of two main phases: prediction and correction. Prediction step:

$$\hat{\mathrm{q}}_t = A\mathrm{q}_{t-1} \tag{26}$$

$$P_{t|t-1} = AP_{t-1|t-1}A^\top + Q_t \tag{27}$$

Correction step:

$$K_t = P_{t|t-1}H^\top\left(HP_{t|t-1}H^\top + \Theta\right)^{-1} \tag{28}$$

$$\mathrm{q}_t = \hat{\mathrm{q}}_t + K_t(z_t - H\hat{\mathrm{q}}_t) \tag{29}$$

$$P_{t|t} = (I_5 - K_t H)P_{t|t-1} \tag{30}$$

where K_t is the Kalman gain, which dynamically balances the influence of the predicted state and the incoming sensor measurements according to their uncertainties. The observation matrix $H = [I_5, I_5]^\top$ maps the five-dimensional state to the ten-dimensional measurement space, while I_5 denotes the 5×5 identity matrix.

In this implementation, the discrete time variable t is synchronized with the event camera's sampling rate, ensuring that all measurements are processed consistently at the highest temporal resolution available. Figure 22 illustrate how the sensor fusion works and its conceptual meanings.

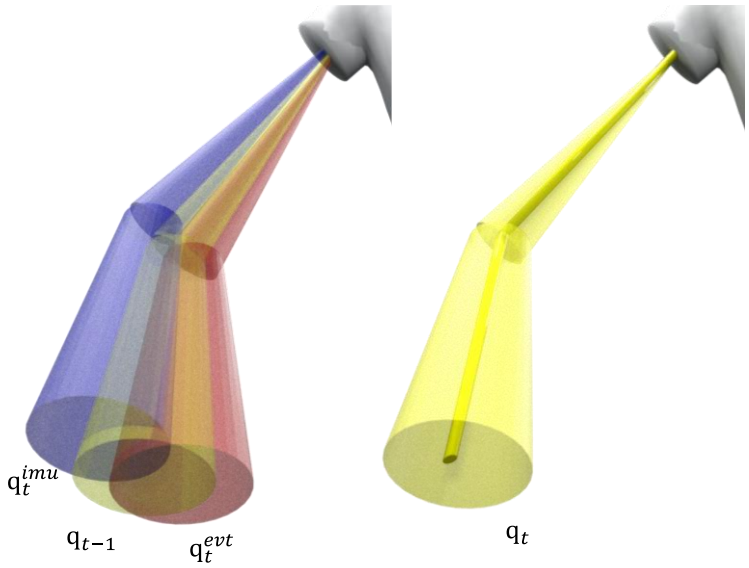

Figure 22 Conceptual illustration of the fusion. The left image illustrates the pose estimated by the IMUs, by the event camera and in the previous state estimation with their covariances, they are represented in blue, red and yellow, respectively. The right image shows the output of the sensor fusion, i.e. the fused pose with its covariance.

3.6 Experimental Validation

The experimental campaign was designed to comprehensively characterize the proposed tracking system, with the goal of assessing its accuracy, the factors influencing it, its reactivity to motion, robustness, computational cost, and overall technical feasibility, as well as identifying its limitations. The test scenarios were therefore selected to specifically challenge individual sensing modalities and evaluate the ability of the sensor fusion framework to compensate for their respective weaknesses.

For instance, motions performed along different directions were designed to test the camera's perception capabilities, while prolonged motion sequences were used to assess drift accumulation in the inertial sensors. The tests were conducted by two human operator and a mechanical arm analogue. The distinction between tests performed with human subjects and those conducted using a mechanical arm analogue allowed for the analysis of anthropometric uncertainty. While human limb dimensions cannot be precisely measured, the mechanical setup, with fixed, known segment lengths and joint geometries, enabled an assessment of the intrinsic accuracy of the system independent of anatomical variability.

Two sets of trials were performed by two human subjects (Figure 23b), while a third set was carried out using the mechanical arm analogue (Figure 23a). Each trial consisted of different types of movements. The motion types are illustrated in Figure 24.

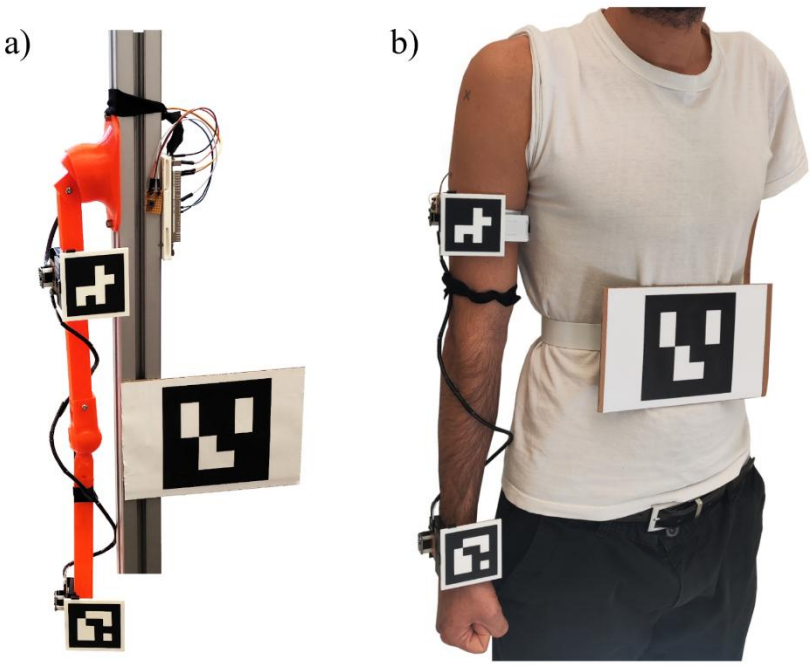

Figure 23 The wearable equipment setup on the a) analogue arm; b) human arm.

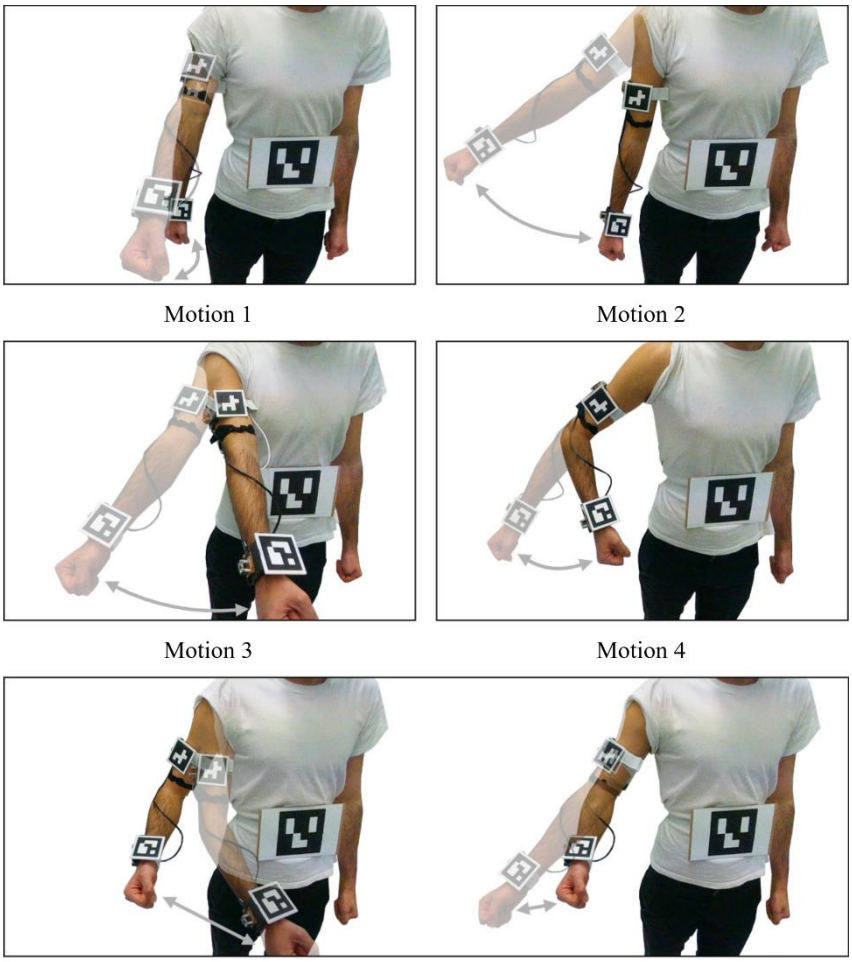

Figure 24 The five types of movements performed in the experimental trials.

Each motion was repeated three times by each subject. In addition, Subject 1 performed ten consecutive repetitions of the full motion set to evaluate drift accumulation over extended operation.

The experimental setup consists of a DAVIS240C event camera, an Intel RealSense D435i RGB-D camera, an Arduino Due microcontroller, two MPU6050 IMUs, three ArUco markers, and two computers. The core tracking system, comprising the event camera, the IMUs, the torso marker, and a primary computer, acquires and synchronizes the sensor data. Additionally, the microcontroller is equipped with a LED and a button used to synchronize the recordings of the tracking system and the validation system. Specifically, each time the button is pressed, the LED turns on, and the corresponding timestamp is recorded and transmitted to the primary computer. The secondary computer manages the RGB-D camera used for ground-truth acquisition via a Python script. The video acquisition is set with the highest sharpness setting, a frequency of 30 Hz and a resolution of 720 $\times$ 1280. Three ArUco markers with 4 $\times$ 4 bits are used in the system. One is placed on the torso of the operator and serves as the reference frame for the tracking system. This marker is twice the size of the others, measuring 120 mm, to compensate for the low resolution of the event

camera and to enhance detection accuracy. The remaining two ArUco markers on the arm measure 60 mm and are used for ground-truth tracking.

The acquisition and communication architecture was implemented in Robot Operating System 2 (ROS2). Figure 25 illustrates the node and topic structure adopted in this work.

The system includes the following ROS2 nodes:

- eCAM node – responsible for event data acquisition from the DAVIS240C, with an accumulation window of 10 ms;
- IMU node – running on the Arduino Due through Micro-ROS, publishing orientation data every 5 ms;
- Saving node – managing data storage for offline analysis.

This modular structure facilitates data synchronization and enables future extensions to additional sensing modalities.

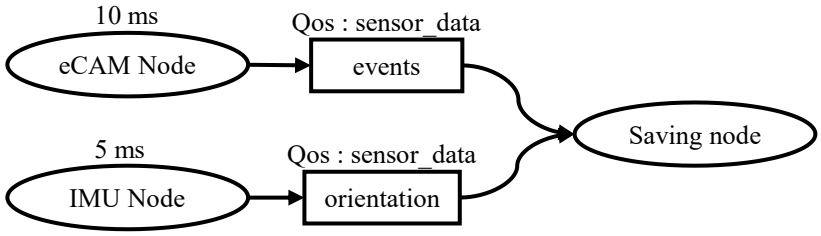

Figure 25 Tracking system ROS2 architecture

The event camera was positioned approximately one meter in front of the subject, with its optical axis slightly tilted respect the sagittal plane. The distance was chosen to balance two competing requirements:

1. ensuring that the full range of motion remained within the field of view,
2. maintaining sufficient spatial resolution of the events.

Increasing the camera-to-subject distance reduces event density, as each pixel corresponds to a larger area in 3D space, decreasing the number of brightness changes detected for the same physical displacement. The selected configuration ensured adequate spatial coverage and event density across all motion types.

A dedicated mechanical arm analogue was developed to provide a controlled environment for testing. The setup replicates the geometry of a human upper limb, including adjustable shoulder and elbow joints. The arm analogue was 3D printed in four different parts, one for the shoulder joint, one for the upper arm segment, one for the elbow joint and one for the forearm segment. This mechanical setup eliminates soft-tissue motion, and measurement uncertainty associated with human tests. Figure 26 shows the mechanical prototype.

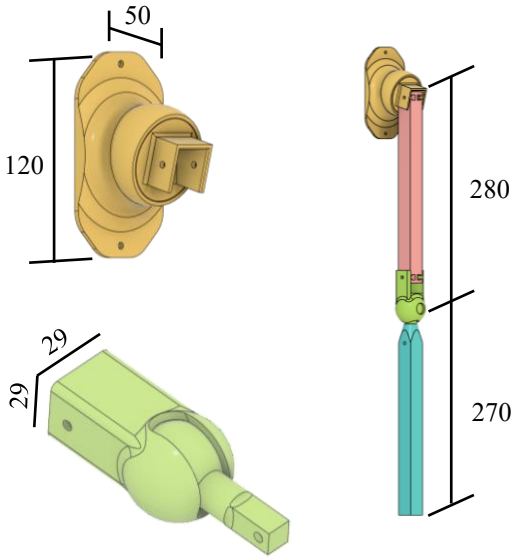

Figure 26 Mechanical arm analogue. The four parts composing the analogue are depicted in different color. In orange the shoulder joint, in pink the upper arm, in green the elbow and in blue the forearm. Relevant dimensions are reported in millimeters.

Experimental Protocol

To ensure consistency and repeatability, all trials followed a standardized procedure comprising the following steps:

1. Sensor mounting and alignment – IMUs and ArUco markers were securely attached and aligned with the limb segments.
2. Anthropometric measurement – segment lengths were measured for subject-specific kinematic model definition.
3. System calibration – the operator assumed the anatomical reference position for the initial camera calibration. A short reference video is captured using the greyscale video output from the event camera to locate the torso reference frame.
4. Data acquisition – sensors were activated, and synchronization was ensured via LED blink. The subject is first instructed to maintain the anatomical reference position, standing still with the palms facing inward for approximately five seconds. This static phase ensures proper initialization of all sensors and alignment of reference frames. Afterward, the subject proceeds to perform the prescribed movement sequence.
5. Offline processing and error evaluation – data were processed in MATLAB for pose estimation, ground-truth extraction, and error analysis.

After data acquisition, all processing was carried out offline to ensure high numerical accuracy and precise validation of the proposed tracking framework. At this stage of system development, the focus was placed on evaluating the performance and reliability of the multimodal algorithm rather than achieving real-time execution. The offline implementation allowed for detailed analysis, precise time synchronization, and the use of higher-precision numerical routines

without the computational constraints of online processing. Furthermore, the chosen validation setup, based on video acquisition and ArUco marker detection, was inherently unsuitable for real-time evaluation, reinforcing the need for an offline approach.

The recorded data streams from the IMUs, the event camera, and the RGB-D camera were processed in MATLAB. Image processing, including camera projection and ArUco marker detection and localization, was performed using the Computer Vision Toolbox. The post-processing workflow followed a structured procedure. First, the grayscale video from the event camera was analyzed to estimate the torso reference frame by detecting the ArUco marker attached to the subject's torso in each frame. The estimated poses were averaged across a short temporal window to obtain a stable reference orientation and position, computed respectively as the mean translation vector and the average quaternion [84]. The same procedure was applied to the upper arm and forearm markers in the RGB-D camera video. This calibration step established the spatial correspondence between the physical markers and the kinematic model's coordinate frames.

The multimodal tracking algorithm was then executed using synchronized and temporally aligned data. The initial orientation of each IMU, ${}^{B}_{0}R_{ua}$ and ${}^{B}_{0}R_{fa}$, was computed as the mean orientation within the interval between 2 s and 2.5 s after synchronization, ensuring stable reference conditions before motion started. The temporal resolution of the Kalman fusion was set equal to the event camera accumulation cycle, corresponding to 10 ms, while IMU readings sampled every 5 ms were averaged in pairs to maintain synchronization with the event stream. For each time step, the event-based optimization and the Kalman filter were applied to produce the fused joint configuration estimate, which was subsequently stored for comparison with the ground-truth data. For this set of trial the minimum number of events to perform the optimization algorithm was set to 20.

Ground-truth poses of the upper arm and forearm were extracted from the Intel RealSense camera recordings by detecting the corresponding ArUco markers in each frame. As a first step, the last occurrence of the synchronization LED blink was identified in the video, and its associated timestamp was stored to align the ground-truth data with the tracking system. Subsequently, the initial poses of the ArUco markers attached to the upper arm and forearm were averaged over the interval between 2 s and 2.5 s after synchronization, following the same procedure adopted for the torso reference estimation. This averaging was performed separately for position and orientation to obtain stable reference values.

The resulting mean poses were then used to compute the constant rigid transformations between the coordinate frames of the kinematic model and those of the ArUco markers. Once these reference transformations were defined, the entire video sequence was processed frame by frame to extract the pose of each marker over time. Finally, the tracking results and the ground-truth data are compared. In particular, the comparison focuses on the position of the ArUco markers placed on the upper arm and forearm. The forward kinematics is used to compute the pose ${}^{B}T_{4}$. The rigid transformation is then applied to account for the

offset between the elbow and the ArUco marker on the upper arm. A similar procedure is applied to compute the pose of the ArUco marker on the forearm. Since the acquisition frequencies of the tracking and ground-truth systems were different, with the ground-truth data acquired at a lower rate, the estimated trajectories were interpolated to match the timestamps of the ground-truth measurements, ensuring precise temporal alignment for error computation.

The system performance was evaluated using position metrics, as spatial accuracy directly reflects the system's effectiveness in collaborative robotic scenarios. Accurate positional estimation of the operator's limbs is a fundamental requirement for interaction safety and motion prediction in shared workspaces. Among the possible positional metrics, the Euclidean distance between the estimated and reference points was selected because of its straightforward physical interpretation and widespread use in motion-tracking performance analysis.

The accuracy of the tracking system was therefore assessed by calculating the mean absolute position error between the estimated and reference marker positions. For each time step t, the positional error was computed both along individual Cartesian axes and as the Euclidean norm of the difference between the estimated and ground-truth positions. The overall mean absolute position error was then derived as the average over the entire duration of the trial:

$$error = \frac{1}{N}\sum_{t=1}^{N}\left|h_t^{Ground\ truth} - h_t^{Fusion}\right| \tag{31}$$

where h represents either one of the Cartesian components x, y, z or the Euclidean distance, and N is the total number of samples. This formulation provides an intuitive and quantitative measure of spatial deviation that can be directly related to the physical motion of the arm. In the next section the Euclidian norm will be represented by the symbol $\|\cdot\|_2$.

3.7 Experimental Results and Discussion

This section presents and discusses the experimental results obtained from the validation campaign described in Section 3.6. The analysis focuses on evaluating the performance of the developed tracking system compared to the single modalities tracking, across different subjects, motion types, and testing conditions. The results provide a comprehensive assessment of the system's accuracy, robustness, and temporal stability, and allow for a critical discussion of the main factors influencing performance.

The reported outcomes were obtained using a set of parameters selected after multiple tuning iterations. Particular attention was devoted to the configuration of the Kalman filter, whose parameters, specifically the process and measurement noise covariance matrices, play a crucial role in determining the system's behavior. Several combinations were tested to identify the configuration that minimized the overall average error across all trials, rather than optimizing

performance for a single movement or subject. This strategy ensured that the final parameter set provided a balanced trade-off between responsiveness, stability, and accuracy under a wide range of operating conditions. The parameters adopted in this work are reported in Section 3.5.

The following discussion presents both qualitative and quantitative analyses of the results, highlighting how each sensing modality contributes to the final performance of the system and how the sensor fusion strategy enhances the overall tracking capability compared to individual sensors.

Figure 27 and Figure 28 show representative examples of the tracking performance for Motion 1 and Motion 2 performed by Subject 1. Each figure illustrates the estimated trajectories of the upper arm and forearm along the x, y, and z axes, together with the corresponding position errors. The plots compare the estimates obtained through the proposed sensor fusion approach, the IMU-only estimation, the event-based-only estimation, and the ground-truth data acquired through the Real-Sense camera. It is important to highlight that the tracking based solely on IMUs is independent from the Kalman filter, whereas the event camera-based estimation relies on the configuration estimated at the previous time step. This dependency stems from the need to assign each detected event to the upper arm or the forearm segment based on the previously estimated configuration. As a result, the configuration estimated using only event data differs from the configuration produced by the same events in the fusion approach. The graphs on the left correspond to the ArUco marker placed on the forearm, while those on the right represent the marker on the upper arm. These plots provide an overview of the system's capability to follow the real motion trajectory and reveal the distinct behavior of the three sensing modalities. As shown in the graph, the IMU-only tracking tends to overestimate the motion amplitude. This behavior is mainly attributed to the effect of linear accelerations on the orientation estimation. In contrast, the event camera alone is not capable of accurately tracking movements that occur along the optical axis. This is evident in Motion 1, which involves movement mainly along the optical axis of the camera. In this condition, the event-based-only approach shows considerably higher tracking errors compared to the other two methods. The loss of depth information inherent to the 2D projection of the event data limits the accuracy of the reconstruction. Nonetheless, even under this challenging condition, the sensor fusion approach maintains the lowest overall error, demonstrating the complementary nature of the two sensing modalities. A different behavior is observed for Motion 2, which occurs predominantly within the image plane and therefore generates a denser and more informative event distribution. In this case, the event-based estimation performs significantly better, particularly along the x and z directions, highlighting the accuracy of visual information in these axes. Although the result of sensor fusion is less accurate than that of the event camera, it is still more accurate than only using the inertial sensors.

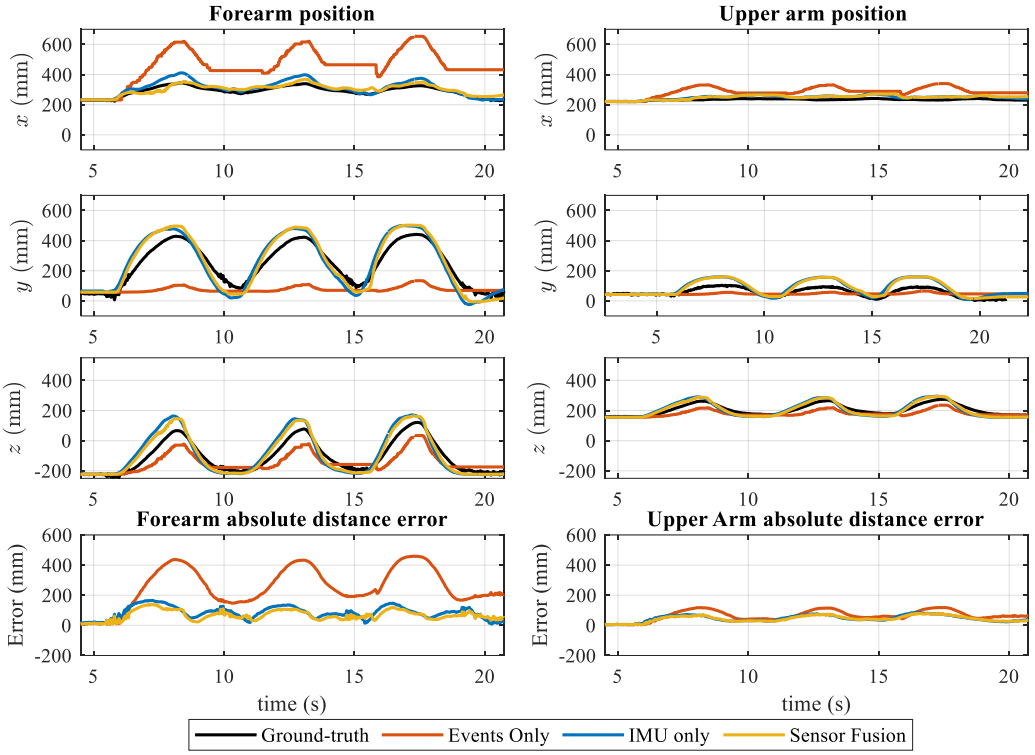

Figure 27 Position estimation from Motion 1 performed by Subject 1. The position in the x, y and z coordinates and the distance error of the sensor fusion results are compared with the IMU-only estimation, the events-only estimation and the ground-truth. The left and right graphs report the ArUco on the forearm and the ArUco on the upper arm, respectively.

Quantitatively, the comparison across all tested movements for Subject 1 confirms that the sensor fusion approach is beneficial. On average, the sensor fusion configuration consistently reduces the mean Euclidean error compared to the IMU-only case and, in most movements, also improves upon the event-only estimation. The adoption of sensor fusion led to a reduction in the mean absolute position error compared to the IMU-only estimation, ranging from approximately 20% to 7% for the forearm and up to 5% for the upper arm. When compared to the event-based-only estimation, the fusion approach achieved a decrease in error between 74% and 33% for all movements except Motion 2, where the error increased by about 24% for the forearm. For the upper arm, the error decreased by roughly 30% in two movements, remained unchanged or slightly higher in two others, and showed an 18% increase in Motion 4.

Table 3, Table 4 and Table 5 summarize the average absolute position errors obtained across all experimental trials for both subjects and the mechanical analogue. For the artificial arm, two of the five predefined motions were not performed due to mechanical constraints that interfered with video acquisition. Overall, the mean absolute position error are comparable to those observed in the state of the art, listed in Table 6. The proposed arm tracking system accuracy is equivalent to data of [85] or higher than data in [86], [87], confirming the validity of the proposed approach. Systems with lower tracking errors were tested in short periods of time [88], less than 2 s, or adopted deep learning algorithms [89].

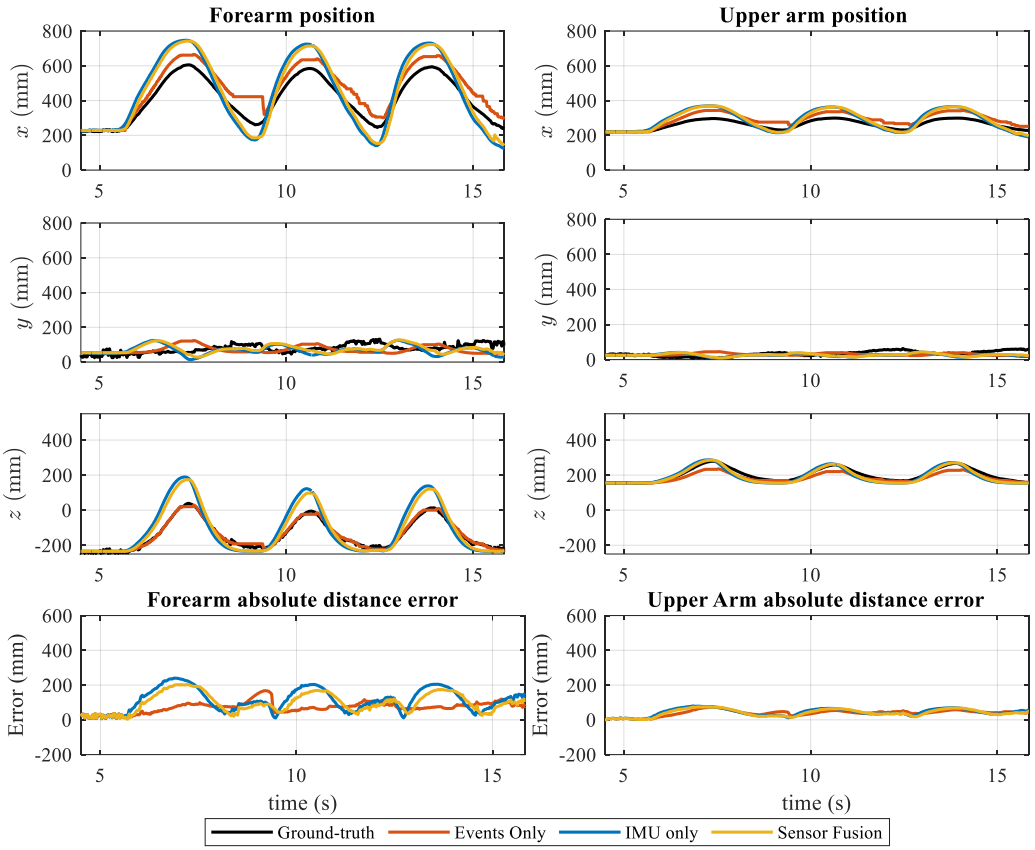

Figure 28 Position estimation from Motion 2 performed by Subject 1. The position in the x, y and z coordinates and the distance error of the sensor fusion results are compared with the IMU-only estimation, the events-only estimation and the ground-truth. The left and right graphs report the ArUco on the forearm and the ArUco on the upper arm, respectively

When observing the results obtained with the mechanical arm, a clear improvement in accuracy can be noted, particularly in Motion 2 and Motion 4, where the mean position errors for the forearm were 71 mm and 67 mm, respectively. This improvement is primarily attributed to the precise kinematic parameters and the absence of anthropometric uncertainty in the artificial setup. The mechanical model adheres more strictly to the assumptions used in the kinematic formulation, such as fixed segment lengths and perfect sensor alignment, which in human tests are difficult to guarantee.

Table 3 Average absolute position error for each movement performed by Subject 1, reported for the x, y, and z components, as well as for the Euclidean distance.

	Forearm (mm)				Upper arm (mm)			
	x	y	z	$\|\cdot\|_2$	x	y	z	$\|\cdot\|_2$
Motion 1	16	52	34	67	16	34	16	44
Motion 2	76	30	45	99	32	14	10	41
Motion 3	83	52	33	116	30	40	12	55
Motion 4	60	47	30	88	45	28	8	56
Motion 5	89	61	36	122	33	24	15	47

Table 4 Average absolute position error for each movement performed by Subject 2, reported for the x, y, and z components, as well as for the Euclidean distance.

	Forearm (mm)				Upper arm (mm)			
	x	y	z	$\|\cdot\|_2$	x	y	z	$\|\cdot\|_2$
Motion 1	37	57	70	113	12	26	12	37
Motion 2	56	83	35	130	21	46	12	58
Motion 3	43	53	43	95	18	17	18	34
Motion 4	24	72	37	96	15	64	17	71
Motion 5	58	67	37	108	27	37	14	55

Table 5 Average absolute position error for each movement performed by the Analogue Arm, reported for the x, y, and z components, as well as for the Euclidean distance.

	Forearm (mm)				Upper arm (mm)			
	x	y	z	$\|\cdot\|_2$	x	y	z	$\|\cdot\|_2$
Motion 1	17	61	44	87	6	27	15	33
Motion 2	49	33	23	71	18	14	4	26
Motion 3	-	-	-	-	-	-	-	-
Motion 4	35	40	31	67	11	13	4	20
Motion 5	-	-	-	-	-	-	-	-

Table 7 reports the mean absolute position error averaged over ten repetitions of each movement performed by Subject 1. In general, an increase in the overall error is observed compared to the single-trial experiments, with the exception of Motion 3. This behavior is likely due to drift accumulation in the IMU data, which becomes more pronounced during extended tracking sequences. Although assigning a greater weight to the event-based measurements in the fusion algorithm could potentially mitigate this drift, such an adjustment would progressively amplify the divergence between the configurations estimated from the IMUs and those inferred from the event data, ultimately leading to inconsistencies between the two sensing modalities. The event-based estimation cannot independently maintain an accurate pose estimate without a stable reference. Over-reliance on this modality is therefore not optimal, as the event camera cannot accurately track motion along the optical axis. The counter effect of that can be observed again in Motion 4, where the overall increase in error is mainly associated with a higher deviation along the y-axis. Since this movement does not involve significant displacement in that direction, the observed error is attributable to cumulative IMU drift, which the event-based measurements are unable to compensate. In contrast, the increased error observed in Motion 5 is likely caused by the intrinsic complexity of the 3D motion, which involves

simultaneous rotation and translation components that challenge both sensing modalities.

Table 6 Comparison of mean wrist and elbow position errors reported in the literature for different arm-tracking systems.

Sensor Modality	Wrist Mean Error (cm)	Elbow Mean Error (cm)
9-axis smartwatch IMU [89]	8.5	8.5
6-axis smartwatch IMU [86]	16.7	11.8
Smartwatch IMUs + Smartphone [87]	12	-
1 IMU + OptiTrack [88]	1.75	-
2 IMUs + 1 Potentiometer [85]	10	4
Ours - 2 IMUs + Event Camera	9.9	4.1

Table 7 Average absolute position error for each movement repeated 10 times performed by Subject 1, reported for the x, y, and z components, as well as for the Euclidean distance.

	Forearm (mm)				Upper arm (mm)			
	x	y	z	$\|\cdot\|_2$	x	y	z	$\|\cdot\|_2$
Motion 1	59	64	44	104	15	49	18	56
Motion 2	77	36	38	99	37	13	12	43
Motion 3	58	40	19	81	38	45	11	64
Motion 4	60	114	38	147	43	63	6	83
Motion 5	85	93	63	156	34	20	15	46

A closer examination of Table 3 reveals that Motions 3 and 5 consistently exhibit higher tracking errors compared to the others. These movements involve pronounced three-dimensional trajectories, where events generated by the upper arm and forearm often overlap in the image plane, resulting in partial occlusion and ambiguity in the 2D event distribution. This overlap complicates the optimization process, making it difficult for the event-based subsystem to accurately reconstruct the spatial configuration.

The results for Subject 2 (Table 4) show an overall error magnitude comparable to that of Subject 1, with some motions yielding slightly lower or higher values. In particular, Motion 2 exhibits a higher mean error, mainly concentrated along the y-axis. Motion 1 also shows a noticeable increase in error for Subject 2, characterized by overshoots along the x and z axes, the principal directions of motion. These overshoots suggest a systematic overestimation of displacement, likely caused by dynamic accelerations influencing the IMU orientation estimation.

The consistency of the results between subjects indicates that the proposed fusion framework generalizes well to different users, with performance mainly affected by small variations in sensor placement or individual motion characteristics. The mechanical arm results further support the conclusion that most of the residual errors observed in human trials arise from anatomical variability and minor alignment inaccuracies rather than limitations of the fusion algorithm itself.

For all the trial the assumption of a stationary torso is checked analyzing the movement of the torso ArUco. The standard deviation of the translational components remained below 5 mm in every experiment, indicating negligible torso displacement. Likewise, the orientation was found to be highly stable, with a standard deviation of less than 0.5 degrees, computed as the minimal rotation angle between the average orientation and each individual orientation extracted from the ground-truth data. These results validate the use of a fixed base reference frame in the kinematic model.

Another important aspect of the implemented system is its high acquisition rate and computational efficiency. Both the inertial and event-based sensors enable high-frequency data acquisition with limited processing overhead. The most computationally demanding operation within the algorithm is the event-based optimization; however, even when limiting the number of iterations to a maximum of 500 for full-arm optimization and 200 for forearm-only optimization, the system consistently achieves accurate tracking results. While additional optimization and code refinement would be required to achieve full real-time capability, the current implementation already provides a strong foundation for practical use in real-world human–robot collaboration scenarios.

The experimental results can also be interpreted in relation to RGB-D-based human-tracking solutions discussed in Chapter 2. RGB-D cameras provide direct depth information and are therefore well suited for full-body skeleton reconstruction and workspace monitoring. However, their perception pipeline typically requires different stages that are not trivial. These processing stages may introduce additional computational load and latency, especially when pose estimation is performed through deep learning-based methods or when multiple cameras are required to mitigate occlusions.

In contrast, the proposed system relies on sparse event-based visual information combined with high-frequency inertial measurements. From an accuracy perspective, the obtained mean position errors are comparable with several tracking systems reported in the literature.

From a temporal standpoint, the proposed architecture is advantageous because the event camera produces measurements only when visual changes occur, while the IMUs provide continuous high-frequency motion information. This reduces visual data redundancy and avoids the need to process complete depth frames at every time step. In the present implementation, the event-based optimization represents the most computationally demanding stage and was evaluated offline; nevertheless, accurate tracking was obtained with a bounded number of optimization iterations, indicating that the approach is suitable for

future real-time implementation after code optimization. Overall, the results support the use of IMU/event-camera fusion as a compact and reactive alternative to RGB-D-based tracking when the primary requirements are low latency, reduced data throughput, and responsiveness to upper-limb motion.

It is important to emphasize that the results presented in this work were obtained using a specific set of parameters for both data acquisition and the fusion algorithm, selected after many experimental tuning. Several configurations were tested before converging on the final set of parameters adopted in the experiments. Among the various parameters, two groups were found to be particularly critical: those related to the event-based pose optimization and those defining the Kalman filter.

The optimization parameters significantly influence the smoothness of the trajectories reconstructed from the event camera and their reactivity to new incoming events. Conversely, the parameters of the Kalman filter determine the degree to which the final fused estimate tends to follow either the IMU-based or the event-based measurements. Multiple experiments were conducted to explore the best trade-off, including tests with adaptive covariance matrices that varied according to the instantaneous state of the system. For instance, process and measurement noise were modelled as functions of the linear acceleration, under the assumption that high accelerations degrade IMU orientation accuracy.

However, this adaptive approache did not lead to a consistent overall improvement. In some test conditions they produced positive effects, while in others they introduced instability or error amplification. For this reason, a simpler, fixed-parameter configuration was ultimately selected, as it provided a more uniform and reliable performance across all movement types and test conditions, even if slightly less optimal in some specific cases.

Nevertheless, the use of adaptive covariance modelling should not be completely ruled out. These techniques demonstrated a noticeable impact on the tracking behavior and may prove beneficial once the current fusion methodology is refined. In particular, with the presently adopted fusion structure, it is not possible to deviate substantially from the pose estimated by the IMUs, which constrains the potential benefit of adaptive weighting strategies. This limitation and possible directions for overcoming it are further discussed in the next section.

3.8 Limitations

Despite the promising results obtained, several limitations of the current implementation remain and offer clear opportunities for future improvement. These limitations can be categorized as hardware-related, algorithmic, and methodological.

From a hardware perspective, the chosen sensing components impose significant constraints on performance. The DAVIS240C event camera, while well suited for prototyping, provides limited spatial resolution, which affects the accuracy of the torso reference frame estimation and reduces the number of detectable events for optimization. The low resolution also constrains the

experimental setup, requiring the subject to remain relatively close to the camera and limiting the naturalness of the movements. Employing a higher-resolution event camera would increase the event density and enhance the robustness of visual tracking even during rapid or large-scale motions. Similarly, the MPU6050 IMUs used in this study, although compact and cost-effective, lack the precision and noise stability of higher-end inertial sensors. Replacing them with more accurate modules would likely improve orientation estimation and reduce drift.

A further hardware-related limitation concerns the intrinsic sensing characteristics of the event-based camera. Unlike RGB-D cameras, the event-based sensor does not provide direct depth measurements. Consequently, depth-related ambiguities cannot be resolved at the sensor level and must instead be addressed indirectly through the kinematic model, inertial measurements, and the sensor fusion strategy. This aspect is particularly relevant for motions occurring along the optical axis, where the lack of direct depth information reduces the observability of the visual subsystem. In addition, although event cameras offer substantial advantages in terms of temporal resolution, data sparsity, and dynamic range, they are currently less widespread and generally more expensive than commercial RGB-D cameras. Their adoption is therefore still mainly confined to research and laboratory environments, which may limit their immediate transferability to industrial applications.

Another practical limitation concerns the wearable nature of the inertial sensing subsystem. Unlike purely optical solutions, IMU-based tracking requires the operator to wear sensors on the body, which may be undesirable in industrial contexts where non-intrusive perception systems are generally preferred. This represents a relevant trade-off between operator comfort and tracking robustness. However, wearable sensing technologies are now widely accepted in everyday applications, such as smartwatches and smartphones, and their miniaturization has significantly reduced their impact on user comfort. In industrial scenarios, IMUs can also be embedded into workwear, gloves, sleeves, jackets, or personal protective equipment, allowing the sensing system to be worn together with elements that are already part of the operator's equipment. From a functional perspective, this additional requirement is compensated by important advantages: IMUs provide high-frequency motion information, are independent of lighting conditions, and remain informative even when the arm is partially occluded from the camera.

From an algorithmic standpoint, the current system relies on the direct use of IMU-derived orientation estimates followed by inverse kinematics computation. While this approach simplifies implementation, it decouples the IMU estimation process from the Kalman filter state, leading to partial inconsistency between the inertial and event-based estimates. A more integrated strategy could involve fusing raw inertial measurements, accelerations and angular velocities, directly with the event-based visual data within the filtering process, choosing an early fusion approach. This would link the inertial information more tightly to the kinematic constraints of the model, potentially yielding smoother and more coherent estimates. Moreover, including additional parameters in the optimization

function to estimate depth could significantly reduce the error introduced by the 2D projection. This could be achieved by enforcing that the lengths of the upper-arm and forearm segments remain consistent with the spatial distribution of the acquired events.

Future work may explore the implementation of a nonlinear filtering scheme, such as an Extended Kalman Filter or Unscented Kalman Filter, capable of directly handling the nonlinear kinematic relations of the upper limb. Moreover, adaptive covariance strategies could be incorporated into the filter, dynamically adjusting sensor confidence based on event rate, motion acceleration, or signal quality, thereby improving robustness to changing motion conditions. From a methodological viewpoint, the simplified kinematic model used in this study, assuming perfect alignment and collinearity of the shoulder, elbow, and wrist centers, represents an additional source of systematic error. Extending the model to include sensor to joint calibration could enhance realism and adaptability.

Finally, from an application standpoint, the current implementation focuses on offline validation and analysis. A natural next step is the transition to real-time operation, to be utilized in human-robot collaboration application, where fast and reliable perception of human motion is essential for safety and coordination.

In summary, while the proposed system demonstrates competitive results respect to existing state-of-the-art methods, substantial room for improvement remains in both hardware and algorithmic design. Addressing these aspects will enable the development of a more scalable, accurate, and adaptable multimodal tracking framework suitable for deployment in real-world collaborative robotic environments.

Chapter 4

Abrupt Movements

Human–robot collaboration relies fundamentally on the robot's ability to interpret human motion in a timely and reliable manner. In typical industrial manipulation tasks, upper-limb movements exhibit a highly repeatable and controlled kinematic structure, reflecting the standardized nature of the workflow and the rhythmic organization of the operator's actions. Within these conditions, robots can exploit predictive models, human motion tracking, and intent estimation frameworks to ensure fluid and safe interaction. However, external disturbances, lapses of attention, unexpected environmental stimuli, and involuntary neuromuscular reactions may induce operator abrupt movements that deviate significantly from the expected kinematic pattern. These gestures are characterized by faster movement [90] and a non-periodic execution, thereby compromising the assumptions of predictability that underpin most collaborative control architectures [91].

From a safety perspective, abrupt movements represent a critical condition in shared workspaces. Because they occur unexpectedly, they reduce the available reaction time for the robot and increase the likelihood of unintended contact. In the context of HRC, where proximity between operator and robot is intrinsic, the ability to promptly detect these anomalous movements is essential to mitigate collision risks and preserve the operator's safety. Moreover, an abrupt movement performed by the operator can in turn induce unintended reactions in the robot, particularly in collaborative systems where human motion is directly perceived and incorporated into the robot control loop. Such interactions may lead to unstable or self-reinforcing behaviors, in which unintentional human reactions and robot responses mutually amplify each other, potentially resulting in hazardous situations. Abrupt movements therefore constitute a meaningful indicator of potentially unsafe conditions, and their rapid identification represents an enabling functionality for next-generation collaborative systems.

This need has become even more explicit with the recent revision of the regulatory framework governing collaborative robotics. The ISO 10218:2025

standard introduces a substantial update to safety requirements by mandating the capability to distinguish voluntary from involuntary physical contacts between human and robot. This distinction reflects a shift toward more refined and context-aware safety mechanisms, acknowledging that involuntary contacts typically arise from unintentional, unexpected, or reflexive human movements. Such events cannot be treated as routine interactions and must instead trigger specific protective actions. In this regard, the recognition of abrupt, non-voluntary gestures is directly aligned with the intent of the updated standard: a collaborative application must therefore be capable not only of limiting forces and monitoring separation, but also of distinguishing normal, intentional motion from atypical and sudden movements. Accordingly, the development of methods capable of differentiating between standard and abrupt upper-limb movements is not merely a research objective but a requirement emerging from both industrial needs and international safety regulations. Detecting the onset of involuntary motion provides the robot with essential contextual information to ensure appropriate reactions, enabling the system to transition to safer operational modes and preventing hazardous interactions. Within this framework, the present work addresses the technical challenges associated with recognizing abrupt movements in real time, leveraging wearable inertial sensing and deep learning techniques to enhance the safety of human-centered collaborative environments.

Although a wide range of sensing techniques and recognition methods has been proposed for human motion analysis, including gesture recognition, motion tracking, and intent estimation, these approaches generally assume that human movements are voluntary, coordinated, and temporally structured. Most existing systems are designed to classify deliberate gestures or reconstruct predictable kinematic patterns, and therefore rely on motion characteristics that differ substantially from those observed in abrupt, involuntary actions.

In contrast, abrupt movements are non-periodic, short in duration, and dominated by sudden variations in acceleration. These properties reduce the effectiveness of conventional tracking pipelines and make the resulting motion difficult to anticipate, limiting the applicability of intent-prediction models and impairing the responsiveness of collaborative controllers. Moreover, abrupt gestures exhibit high variability in their kinematic and dynamic profiles, execution patterns, and number of involved segments [92], making them inherently harder to model within traditional recognition frameworks.

Previous studies have primarily focused on characterizing the biomechanics and behavioral implications of impulsive or involuntary movements, rather than on their automatic recognition. Castellote et al. analyzed the voluntary reaction to startling auditory stimuli and demonstrated that such perturbations predominantly accelerate only the initial phase of the movement [93]. Kirschner and colleagues described involuntary motions as rapid, uncontrolled responses triggered by unexpected robot actions, showing that user training can mitigate their impact on safety [94].

Although these works provide insight into the characteristics, causes, and effects of abrupt movements, they do not address the development of algorithms

capable of promptly identifying such events. A systematic methodology for detecting abrupt movements in real time, suitable for integration into collaborative robotic systems, remains largely unexplored. In addition, existing datasets predominantly contain voluntary gestures and lack the variability required to train models capable of generalizing to involuntary motions triggered by unexpected stimuli. Consequently, there is a clear need for algorithms that can detect abrupt movements with low latency and sufficient robustness for safety-critical human–robot collaboration.

The objective of this chapter is to develop and validate a computational framework for the prompt recognition of abrupt upper-limb movements using wearable inertial sensing and deep learning techniques. The proposed system is designed to distinguish standard gestures from sudden actions that may compromise safety during human–robot collaboration. To this end, the chapter presents the design of a Long Short-Term Memory (LSTM) network capable of identifying abrupt movements within short temporal windows and under real-time constraints.

In the following sections, the development of the LSTM-based method for early recognition of abrupt movements is presented in detail, beginning with the data collection procedures required for network training, followed by the validation of the model, and concluding with the performance optimization for real-time application.

The work presented in this chapter represents an in-depth refinement, reorganization and extension of the work presented in [95–99]

4.1 Data collection

To train an LSTM network capable of reliably distinguishing between standard and abrupt upper-limb movements, it is essential to collect data that are representative of both motion classes. For this purpose, a dedicated acquisition campaign was designed to reproduce movement patterns typically observed in industrial environments, characterized by repetitive, rhythmical gestures, and to embed within them a set of controlled perturbations intended to elicit abrupt, non-volitional reactions. The resulting setup ensures the collection of high-quality inertial data suitable for both network training and validation under realistic yet controlled conditions. 60 participants were enrolled in the study ensuring variability and generalization. The dataset is public available at [100].

Experimental set-up

The experimental design is centered around a pick-and-place task, selected because it closely reflects operations common in industrial settings such as assembly, packaging, and material handling. These tasks are highly standardized and performed at a regular cadence, making them ideal for representing standard movements. To complement this, controlled sensory stimuli were introduced to induce sudden reactions resembling instinctive behaviors that may occur in real

working scenarios, for example, reaching out to prevent an object from falling or raising the arm in response to a startling noise. This combination of predictable task execution and unpredictable perturbations enables the collection of a dataset that captures the contrast between coordinated, intentional motion and abrupt, atypical responses.

Inertial data were acquired using wearable sensors positioned on the operator's upper limb. This sensing modality is minimally intrusive and capable of capturing both the smooth kinematics of standard gestures and the high-acceleration profiles typical of involuntary movements. The workstation, stimuli, and task structure were designed to ensure reproducibility across participants while allowing anthropometric adjustments to preserve natural execution.

The workstation used during the trials is illustrated in Figure 29 and consists of five stations, denoted S0 through S4. Station S0 serves as the pick-up location and contains a set of small balls used for the task. Stations S1, S2, and S3 are located on the same horizontal plane and each includes an array of holes where the ball can be released. Prior to the experiment, each participant selects one hole per station to match their anthropometry and ensure comfortable reachability.

Station S4 is positioned above the plane of the other stations, with a fixed elevation of 30 cm. Its horizontal distance from the operator is adjustable so that the movement required to reach it remains ergonomically appropriate. Unlike the other release stations, S4 contains a single hole.

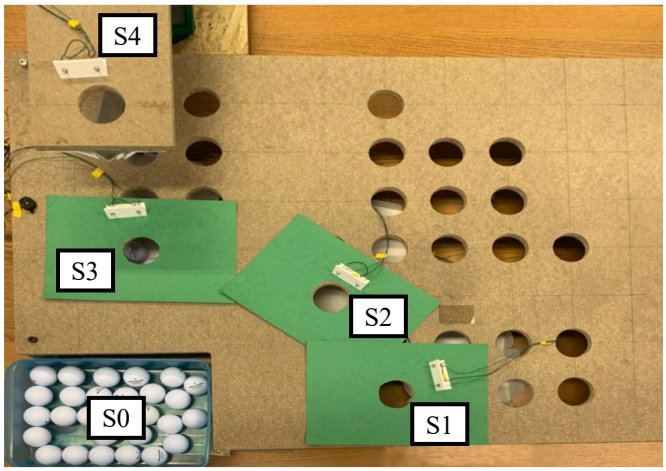

Figure 29 Workstation top view.

Each station is equipped with two LED indicators:

- a **green LED**, used to signal standard pick-and-place movements.
- a **red LED**, used as the visual stimulus to induce abrupt reactions.

To emphasize the distinction between the two conditions, the red indicator flashes when activated, while the green indicator remains steadily illuminated for half of the period associated with a single task cycle. A buzzer located near station S1 provides the auditory stimulus used to trigger abrupt movements of the second type.

During the experiment, the operator performs a sequence of 30 pick-and-place cycles. Each cycle begins with illumination of one of the green indicators, signaling the target station. The pace is fixed at 20 beats per minute, providing a consistent rhythmic structure typical of highly standardized industrial tasks. This regular cadence ensures that standard movements exhibit predictable kinematic profiles, forming a suitable baseline against which abrupt deviations can be identified.

Within the 30-cycle sequence, four perturbations are introduced at unpredictable times: two visual and two auditory stimuli. When a red visual stimulus is activated, the operator must immediately redirect the motion toward the newly illuminated station, thus producing a sudden change of trajectory. When an auditory stimulus is triggered, the operator is instructed to raise the arm as quickly as possible, mimicking a rapid protective or reflexive action. These two perturbation types were intentionally chosen to generate variability in the elicited reactions and to replicate plausible sudden events occurring in real collaborative environments.

Three configurations of the operator's posture were tested:

1. seated, facing the workstation, using the right arm;
2. seated, facing the workstation, using the left arm;
3. seated, positioned laterally to the workstation, using the left arm.

This variability ensures that the dataset includes different arm orientations and movement directions while maintaining the same underlying task structure. All upper-limb movements were recorded using the inertial sensors throughout the trial, enabling precise segmentation and labelling of both standard and abrupt gestures.

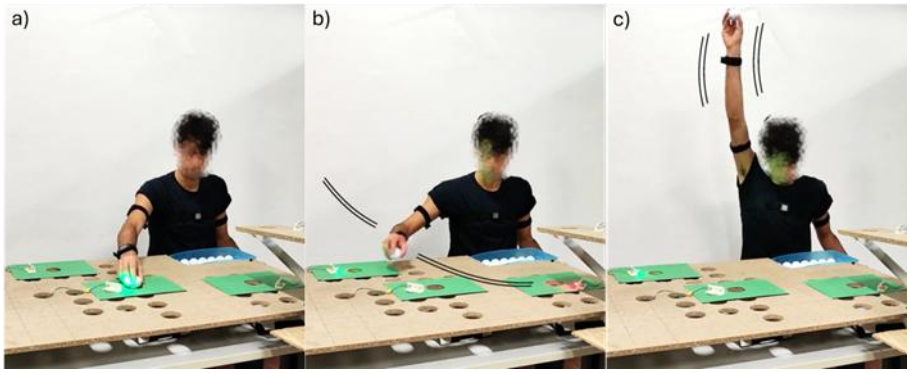

Figure 30 Task execution example, a) standard movements, b) visual stimulus movements response and c) auditory stimulus movement response.

Data collection was performed using APDM Opal MIMUs. Each Opal unit integrates a tri-axial accelerometer (±200 g), a tri-axial gyroscope (±2000 deg/s), and a tri-axial magnetometer (±8 Gauss), with a sampling frequency of 200 Hz. All MIMU data were streamed and recorded through a dedicated acquisition computer running APDM MotionStudio.

The delivery and timing of visual and auditory stimuli, both for standard and abrupt movements, were controlled by a microcontroller (Arduino Nano, processor ATmega328, clock speed of 16 MHz). The Arduino managed a set of eight LEDs (green indicators for standard tasks and red indicators for visual abrupt stimuli) and a buzzer used for generating the auditory stimulus. For each stimulus activation, the microcontroller transmitted a timestamp to a second computer, allowing precise association between stimulus events and inertial data.

To ensure consistent temporal alignment between the two systems, the Opal MIMUs were synchronized using the APDM synchronization box. At the beginning of each trial, the Arduino issued a hardware trigger to the synchronization unit, which in turn initiated the data acquisition procedure in MotionStudio. This mechanism guaranteed that all MIMU data were temporally aligned with the corresponding stimulus log.

To ensure that abrupt movements occurred during an active motor phase rather than at rest, visual and auditory stimuli were always triggered 500 ms after the activation of a green LED, ensuring that the participant was already engaged in a standard pick-and-place action when the perturbation occurred. The overall control logic of the stimulus-generation algorithm is illustrated in Figure 31, and a simplified pseudocode representation is provided in **Algorithm 3**.

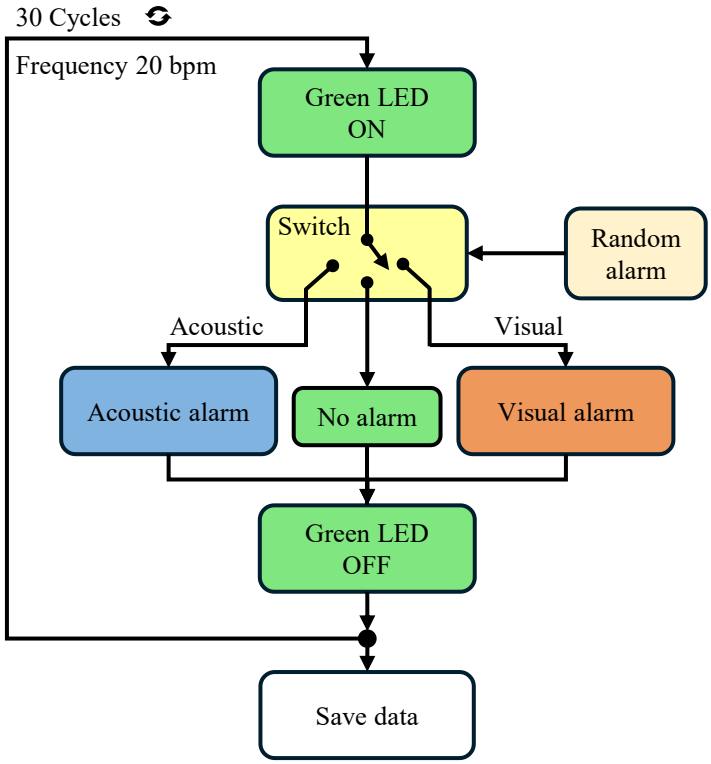

Figure 31 Logic flow of the microcontroller algorithm used to manage visual and auditory stimuli during the pick-and-place task. The Arduino controls the activation of green LEDs for standard movements and triggers either visual or acoustic alarms after a randomized selection, ensuring that abrupt stimuli occur during the ongoing motion phase.

The pseudocode details the logic implemented to generate the experimental movement sequence and to control the visual and acoustic stimuli during a trial.

In the first part, a set of constants is defined to characterize the experimental protocol. These include the total number of movements executed during a trial (*N_MOV*), the number of initial movements without abrupt stimuli (*N_INIT*), the duration of each movement cycle (*PERIOD_MS*), and the delay between the onset of a standard movement and the possible activation of an abrupt stimulus (*STIM_DELAY_MS*). These parameters define the temporal structure of the task and ensure repeatability across trials.

The main variables include the array *stationSeq*, which specifies the target station for each movement, and two arrays (*stimVis* and *stimAco*) that store the indices of the movements during which visual and acoustic stimuli are triggered, respectively.

The SETUP function is executed once at system startup. In this phase, the hardware is initialized by configuring the pins associated with the green and red LEDs, the buzzer, and the synchronization signal. Serial communication is opened to allow timestamp logging. The sequence of target stations (*stationSeq*) is then generated. Visual and acoustic stimuli are assigned randomly under predefined constraints: two stimuli of each type are scheduled, with one occurring in the first half of the trial and one in the second half, while ensuring that no stimuli are assigned within the initial non-abrupt movements. A final check is performed to avoid overlapping assignments; if two stimuli are associated with the same movement, the assignment procedure is repeated.

The LOOP function manages the actual execution of the trial. After receiving a start command, a synchronization pulse is sent to align the external acquisition systems. A for loop then iterates over all movements. For each movement, the corresponding green LED is activated to indicate the target station, and the event is logged with a timestamp. After a fixed delay, the algorithm checks whether the current movement is associated with an abrupt stimulus. If a visual stimulus is required, a red LED corresponding to a station different from the currently active green one is selected and blinked until the end of the movement cycle. If an acoustic stimulus is scheduled, the buzzer is activated for a predefined duration. All stimulus events are logged.

At the end of each movement period, all LEDs and the buzzer are turned off. This procedure is repeated until all movements in the trial have been executed.

Each trial followed a standardized sequence to ensure repeatability and consistent data quality:

1. **Participant Registration and Anthropometric Measurements**
   The participant completed a short form with basic information, and the lengths of the upper arm and forearm were measured. These measurements were later used to map sensor data to individual limb geometry and to assist in task configuration.
2. **Sensor Placement**

Five Opal IMUs were mounted on the participant as shown in Figure 32, two sensors per upper limb (upper arm and forearm), one sensor placed on the sternum. Although the classification task relied primarily on wrist-mounted data, additional sensors were included to ensure comprehensive recording.

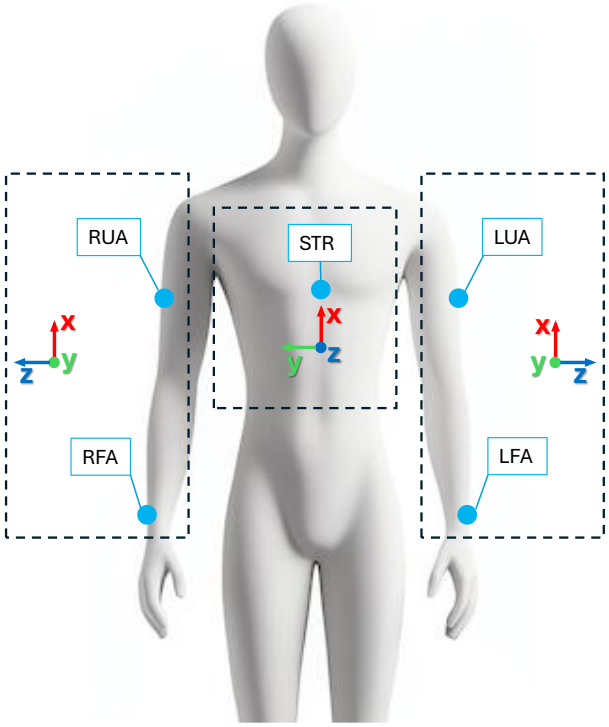

Figure 32 Placement of the inertial sensors on the operator's upper body. The abbreviations indicate the sensor locations: RUA (right upper arm), RFA (right forearm), STR (sternum), LUA (left upper arm), and LFA (left forearm). Each sensor is mounted according to the orientation defined by the local reference frame shown in the dashed box associated with each body segment.

3. **Participant Positioning and Task Familiarization**
   The subject was seated in front of the workstation and received instructions about the task. A short familiarization phase was carried out to allow the participant to adjust to the task rhythm and the station layout.
4. **Execution of the Three Trial Configurations**
   The participant performed the full task sequence in the three predefined configurations (two frontal and one lateral), with short pauses between conditions to reduce fatigue. During the trial, the first five movements were always standard, allowing the operator to establish a stable execution pattern before abrupt stimuli could occur.

All inertial data were recorded for each sensor unit, including the full set of measurements provided by the Opal devices, in particular linear acceleration components, angular velocity components, and sensor orientation. In addition, the microcontroller logged the temporal activation of both visual and acoustic stimuli together with the associated target stations. All these data, along with the trial configuration information, were stored for subsequent processing, segmentation, and preparation of the dataset used for network training and validation. Figure 33

shows an example of a complete acquisition obtained during one experimental trial. The upper plot reports the three components of linear acceleration (X, Y, and Z) measured by the IMU positioned on the operator's wrist. Superimposed on the inertial signals, vertical markers indicate the timing of the external events that structure the task execution. In particular, green vertical lines denote the activation of the green LEDs, corresponding to standard pick-and-place commands, while red vertical lines indicate the activation of visual alarm stimuli. Blue markers identify the activation of the acoustic alarm.

The lower plot provides a compact representation of the task sequence, showing the temporal order of the target stations associated with each green LED activation. Each marker corresponds to one station (S1–S4), allowing a direct association between the commanded task sequence and the recorded kinematic response. Acoustic alarm events are labelled as AA (Acoustic Alarm), since they are not associated with any specific station but instead require a generic reflexive reaction from the operator.

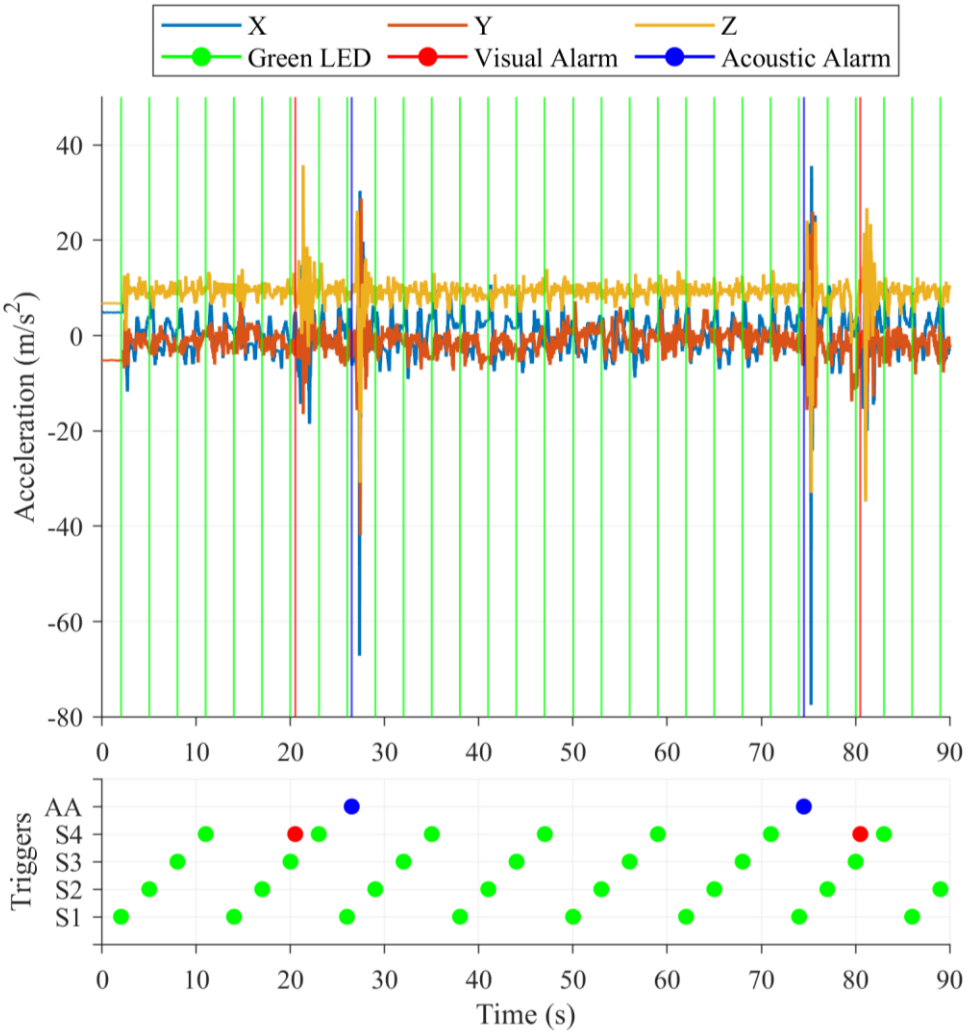

Figure 33 Example of data obtained in a trial

Algorithm 3 Movement generation code

```
CONSTANTS
   N_MOV      = 30
   N_INIT     = 5
   PERIOD_MS       = 3000      // 20 bpm
   STIM_DELAY_MS    = 500

VARIABLES
   stationSeq[1..N_MOV]   // target station for each movement
   stimVis[2]  // visual stimulus
   stimAco[2]  // acoustic stimulus
```

```
function SETUP()
        Configure pins: greenLED[], redLED[], buzzer, syncPin
        Open serial
        Generate stationSeq[]
        stimVis = [rand(N_INIT, N_MOV/2 +1), rand(N_MOV/2 +1,
        N_MOV),]
        stimAco = [rand(N_INIT, N_MOV/2 +1), rand(N_MOV/2 +1,
        N_MOV),]
        Check stimuli overlap, if overlap assign again
```

```
function LOOP()
        wait for start command
        send sync pulse on syncPin

        FOR m = 1 to N_MOV
          station = stationSeq[m]
          t0 = current_time_ms()

          turn ON greenLED[station]
          log("STD_ON", m, station, t0)

          wait STIM_DELAY_MS

          IF m ==  stimVis THEN          // visual
              abrupt_sta =  random station different from the current one
              blink redLED[ abrupt_sta ] until t0 + PERIOD_MS
              log("VISUAL", m, abrupt_sta, current_time_ms())
          ELSE IF m ==  stimAco THEN     // acoustic
              activate buzzer for fixed time
              log("AUDIO", m, station, current_time_ms())
          END

          wait until current_time_ms() >= t0 + PERIOD_MS
          turn OFF all LEDs and buzzer

        END
```

4.2 Data Statistics

The quality and structure of the acquired data play a central role in the performance and generalizability of any machine learning algorithm. For this reason, a preliminary analysis of the collected inertial signals was conducted before defining the model architecture and the training strategy. This initial investigation aimed to verify how the data were distributed, to assess whether the patterns associated with abrupt movements were statistically distinct from those generated during standard task execution, and to understand the intrinsic variability of each class.

A detailed examination of the abrupt movements was then carried out to characterize their temporal and kinematic properties, including their typical duration, amplitude, and signal morphology. This analysis provided insight into how abrupt reactions manifest in the inertial domain and clarified which signal components are most informative for classification.

Building on these observations, the segmentation strategy for the time-series data was progressively refined. The objective was to identify window lengths and overlap configurations that would preserve the discriminative characteristics of abrupt movements while reducing the amount of information needed for each prediction. This refinement is essential to achieve faster detection times and ensure compatibility with real-time constraints in collaborative robotics applications. The following sections detail the analyses performed on the dataset and their implications for the subsequent design of the LSTM-based recognition framework.

The initial analysis focused on the two kinematic signals most relevant for distinguishing standard and abrupt movements: the linear acceleration and the angular velocity measured by the MIMU positioned on the wrist of the active arm. All processing was performed in MATLAB. To avoid distortions related to sensor orientation and movement direction, the vector norm of both signals was computed, providing a scalar representation of their overall magnitude. An example of the resulting signals is shown in Figure 34.

Following preprocessing, the continuous recordings were segmented into temporal windows. Segmentation was aligned with the activation of the green LEDs, which define the onset of each pick-and-place action. Each window had a duration of 3 seconds, corresponding to the period of a single task cycle. Windows were then labelled as standard or abrupt depending on whether a visual or auditory stimulus was present within that interval. This procedure generated two classes:

- a **standard movement class**, containing 26 windows per trial
- an **abrupt movement class**, containing 4 windows per trial

For each window, the root mean square (RMS) value of the signal norm was computed to obtain a compact descriptor of the motion. These RMS values were then used to compare the two classes statistically.

As a first step, the normality of each group was assessed using MATLAB's *lillietest* function. Neither the standard nor the abrupt movement distributions passed the normality test; consequently, all subsequent comparisons between the two classes were performed using statistical tests suitable for non-normally distributed data. Mean and standard deviation were then computed for each class across the three test configurations. Both metrics showed clear differences between standard and abrupt windows, suggesting distinct underlying motion characteristics, Table 8. To confirm the statistical separation between the two groups, a Wilcoxon rank-sum test was performed, yielding *p*-values well below 0.01. The graphical representation of the data distribution is depicted in Figure 35 in an histogram chart.

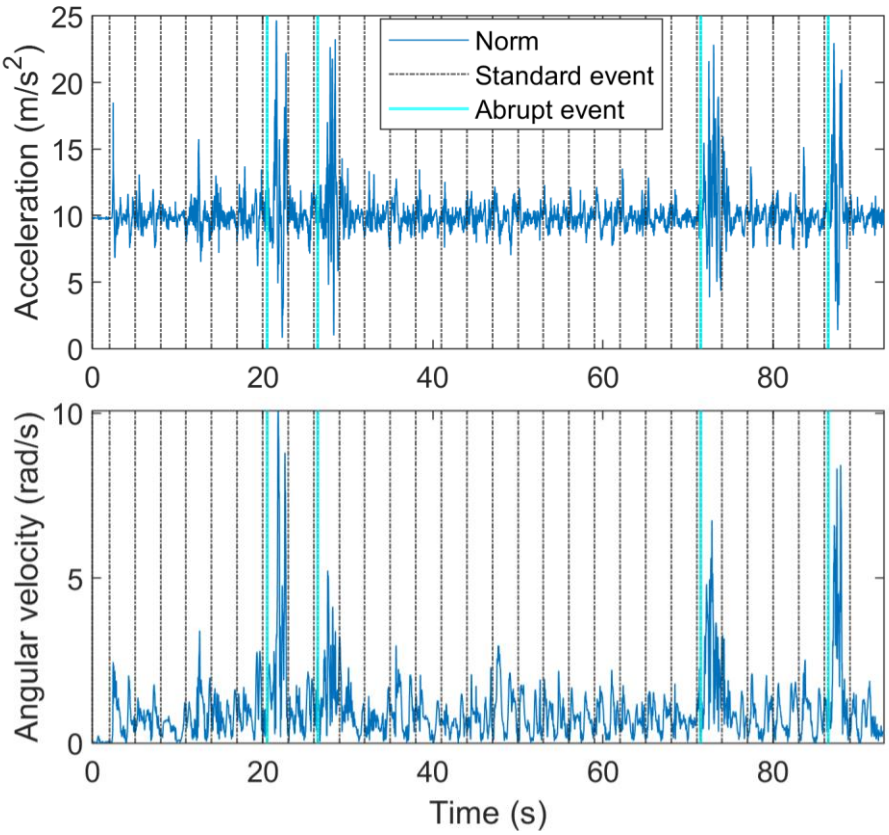

Figure 34 (Top) acceleration norm and (bottom) angular velocity norm recorded during a complete pick-and-place trial. Grey dashed lines indicate the onset of standard events, while cyan solid lines correspond to visual or auditory stimuli used to elicit abrupt movements.

Table 8 RMS values of accelerations and angular velocities averaged among windows (mean ± standard deviation), and p-values obtained from the Wilcoxon test between standard and abrupt values (indicate a statistically significant difference).**

	Trial 1		Trial 2		Trial 3	
	Standard	**Abrupt**	**Standard**	**Abrupt**	**Standard**	**Abrupt**
Acceleration (m/s^2)	9.96 ± 0.14	11.18 ± 1.17	9.98 ± 0.11	11.07 ± 1.01	9.96 ± 0.11	10.97 ± 0.91
p-value	<0.01**		<0.01**		<0.01**	
Angular velocity (rad/s)	1.29 ± 0.35	2.11 ± 0.73	0.98 ± 0.25	1.94 ± 0.74	1.12 ± 0.33	1.97 ± 0.68
p-value	<0.01**		<0.01**		<0.01**	

These preliminary analyses were initially conducted on raw acceleration signals, which include the gravitational component. Since gravity may partially

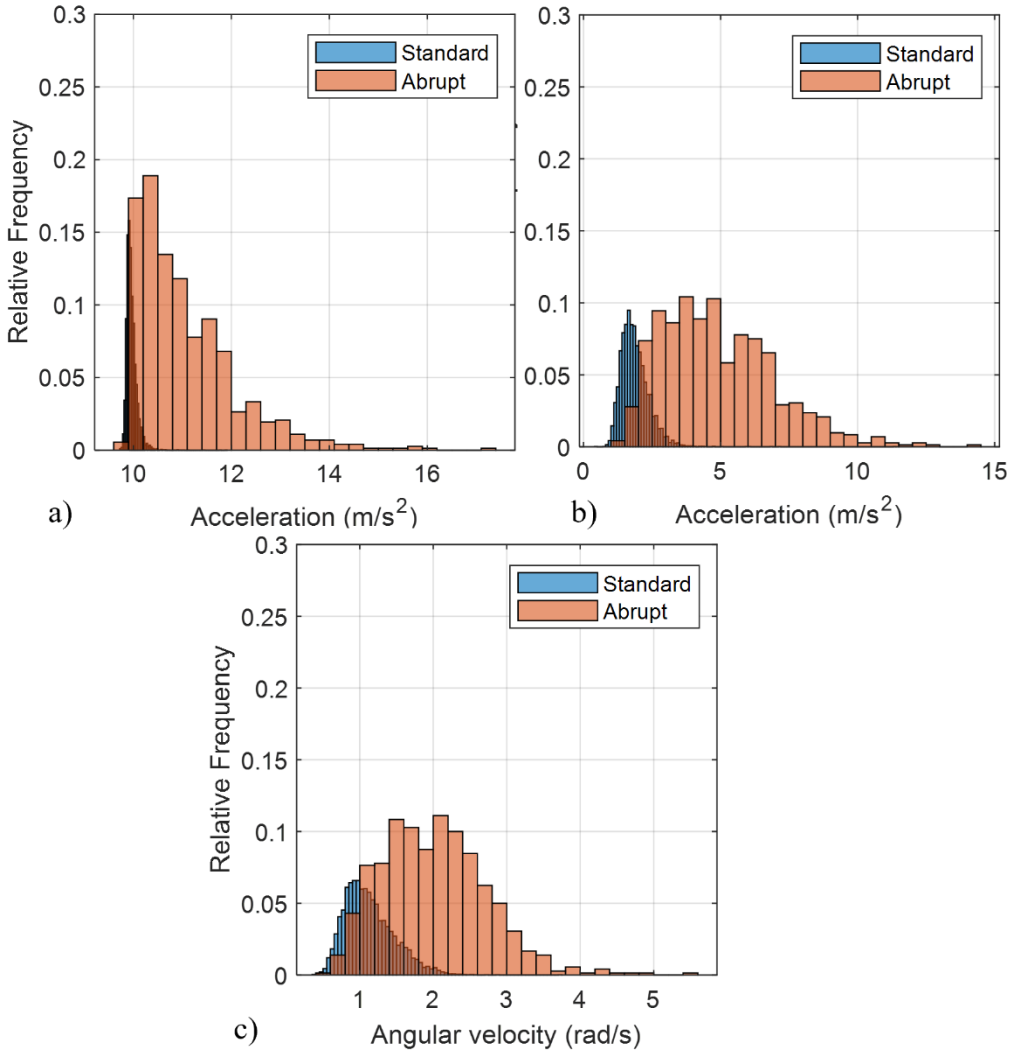

Figure 35 RMS distribution of: (a) gravity-compensated and (b) not compensate acceleration and (c) angular velocity for standard and abrupt movements. Bin widths were determined automatically using the Freedman–Diaconis rule.

obscure motion-related accelerations, the same statistical tests were repeated after removing the gravitational contribution using the orientation information provided by the MIMU. The separation between the two classes became even more pronounced when gravity was removed. Based on this observation, all subsequent analyses were performed using gravity-compensated acceleration signals.

To deepen the understanding of how standard and abrupt movements evolve over time, the acceleration and angular velocity signals were aligned and averaged across all windows belonging to each class. For each time point, the inter-window standard deviation was also computed, allowing the estimation of the typical variability within each movement category. The resulting representative profiles are shown in Figure 36.

The averaged signals reveal that standard movements exhibit a highly consistent temporal pattern: both acceleration and angular velocity remain relatively stable throughout the 3-second window, with limited dispersion across

repetitions. This behavior reflects the intrinsic repeatability of the pick-and-place task and confirms that standard actions are performed with controlled and predictable kinematics.

In contrast, abrupt movements display a markedly different behavior. Following the perturbation, the acceleration and angular velocity profiles show rapid increase in magnitude, fluctuations and substantially higher variability across trials. The dispersion band is notably wider, indicating that abrupt reactions vary considerably in amplitude, shape, and timing.

A key observation arises at approximately 0.5 s, which corresponds to the programmed delay between the activation of the green LED and the onset of an abrupt stimulus. Before this instant, the average profiles of abrupt and standard movements are almost indistinguishable, reflecting the fact that participants begin both movements in a similar manner. Immediately after the stimulus, however, the two trajectories diverge sharply: abrupt movements present a pronounced increase in magnitude and variability, while standard movements maintain their regular structure. The profiles gradually reconverge around 2 s, when both actions return to more stable phases.

These observations have direct implications for the segmentation strategy used in the classification framework. The discriminative features of abrupt movements are concentrated within a relatively narrow time interval following the stimulus, whereas the remainder of the 3-second window contains limited useful information. Moreover, using full-window segments during training would introduce large portions of standard movement into the abrupt class, diluting the informative content and potentially degrading classification accuracy.

For these reasons, the time windows were reduced and refined to isolate the portion of the signal where abrupt movements genuinely manifest. This refinement serves two fundamental objectives:

1. **Clear separation of classes during training.** Shorter windows allow the model to receive samples that contain exclusively abrupt or exclusively standard content. Including long windows that combine both behaviors would blur the distinction between classes and reduce the network's ability to learn discriminative patterns.
2. **Early detection capability.** An abrupt movement typically lasts approximately less than 1.5 s. Using windows longer than this duration would dilute the impulsive component with non-informative signal segments and delay recognition. Since the goal of this work is to detect an abrupt response at its onset, training the model on entire 3-second movements would produce a system capable of recognizing the gesture only after its full execution, defeating the purpose of prompt detection in collaborative robotics.

By narrowing the temporal window around the impulsive phase, the segmentation better captures the distinctive signatures of abrupt behavior while simultaneously enabling lower-latency inference. This adjustment is therefore essential for developing a recognition framework suitable for real-time operation and aligned with safety requirements in human–robot collaboration.

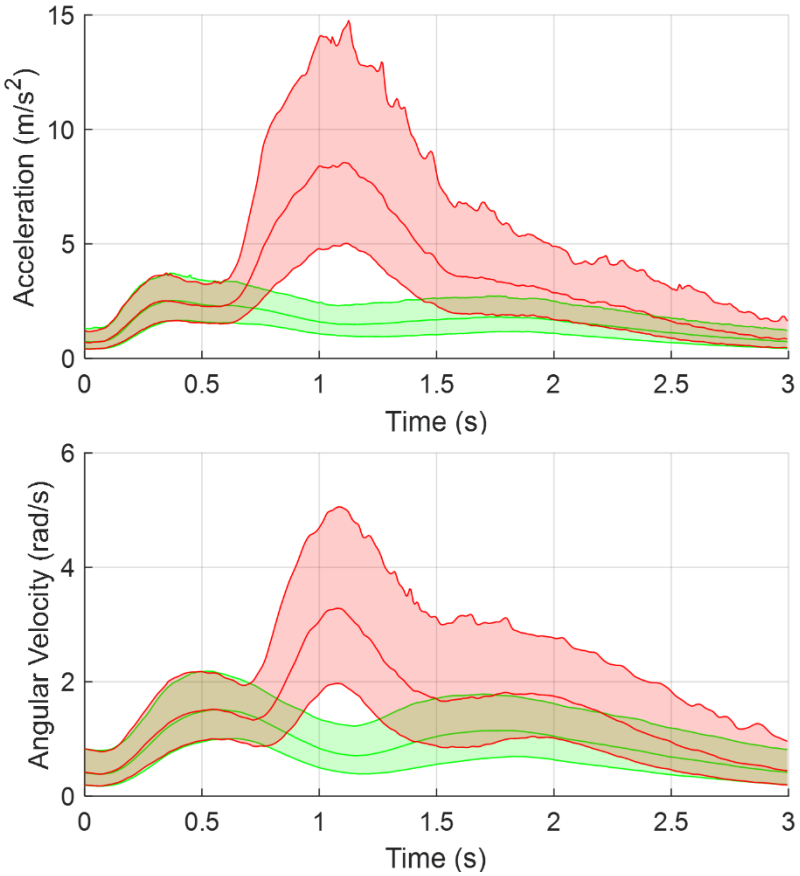

Figure 36 Average temporal profiles of (top) acceleration and (bottom) angular velocity for standard (green) and abrupt (red) movements. Solid lines represent the mean across all windows, while shaded areas indicate the inter-window standard deviation.

Following the temporal analysis of the complete 3-second windows, the same set of analyses was repeated using 0.5-second sub-windows. In this case, the original 3-second window was divided into six consecutive segments, which were kept separate to preserve the temporal structure of the full gesture and to evaluate how discriminative information evolves within the movement. For each sub-window, the same characteristic indicators, such as the RMS of the acceleration norm, were computed across all participants and all trials.

To quantitatively assess which sub-windows present statistically significant differences between abrupt and standard conditions, a Wilcoxon test was performed for each of the six 0.5-second segments. This analysis provides an objective criterion for identifying the time intervals that carry the most discriminative information and thus guides the definition of an optimized windowing strategy for subsequent model training. The *p*-values are reported in Table 9.

Table 9, *p*-values from the Wilcoxon test comparing standard and abrupt movements across the six 0.5-second sub-windows for both acceleration and angular velocity.

	Acceleration (*p*-value)	Angular Velocity (*p*-value)
Sub-window 1	0,61	0,79
Sub-window 2	<<0,01	<<0,01
Sub-window 3	<<0,01	<<0,01
Sub-window 4	<<0,01	<<0,01
Sub-window 5	<<0,01	<<0,01
Sub-window 6	0,005	0,06

Based on the statistical analyses described above, the dataset used for training the neural network was constructed using 0.5-second windows. Each window was assigned a binary label indicating whether it belonged to the *standard* class (label = 0) or to the *abrupt* class (label = 1). For abrupt movements, only sub-windows 2, 3, and 4 were labelled as abrupt, in accordance with the statistical evidence showing that these segments contain the most distinctive kinematic features. Although sub-window 5 also exhibited statistical differences, it was excluded because the distinction between classes diminishes in this interval and the gesture gradually transitions toward its final phase. Since the objective of this work is early detection, including later portions of the movement would not contribute to recognizing the onset of an abrupt reaction. Sub-windows 1, 5 and 6 obtained from the original 3-second abrupt windows were excluded from the dataset. Removing them provides a more conservative and homogeneous dataset, especially given the already large availability of standard samples.

The resulting dataset is structured as follows:

- 156 standard windows per trial,
- 3 trials per subject,
- 60 subjects,

leading to a total of $156 \times 3 \times 60 = 28080$ standard windows.

For abrupt movements:

- 12 abrupt windows per trial,
- 3 trials per subject,
- 60 subjects,

yielding $12 \times 3 \times 60 = 2160$ abrupt windows.

Each 0.5-second window contains 100 samples per feature, corresponding to the 200 Hz sampling rate. The two selected features are the norm of the linear acceleration and the norm of the angular velocity, yielding a two-channel temporal sequence for each window.

This dataset structure ensures a clear separation between classes, captures the temporal segment in which abrupt behavior manifests, and provides the neural network with an input format suitable for learning an accurate and prompt detection model.

4.3 Network Training

The recognition problem addressed in this work is inherently temporal: the distinction between standard and abrupt movements does not lie only in their instantaneous amplitude, but in the way acceleration and angular velocity evolve over time. Aggregate descriptors may mask short impulsive components or be biased by fast but voluntary actions. For this reason, a recurrent neural architecture was adopted, so that the classifier could exploit the full temporal history of each 0.5 s window rather than a single summary statistic.

Among recurrent models, a Long Short-Term Memory (LSTM) network was selected because of its proven capability to learn long- and short-range dependencies in time series while mitigating vanishing-gradient effects. This property is particularly suitable for the present application, where the network must discriminate between impulsive and non-impulsive behavior using the temporal evolution of two kinematic quantities (norm of linear acceleration and norm of angular velocity). The final architecture and its hyperparameters (number of hidden units, dropout rate, sequence length, and layer configuration) were defined after a series of hyperparameter tuning experiments aimed at maximizing validation performance while keeping the model complexity compatible with real-time constraints.

Each input sample is a 0.5 s time window, represented as a sequence of 100 time steps with two features per step. The network is composed of six layers, each with a specific role:

- **Sequence input layer.**

  The *sequenceInputLayer* receives the time series in the form of a 100×2 sequence and defines the dimensionality of the input (*inputSize = 2*). Its role is to pass the ordered temporal data to the recurrent part of the network without any modification of the feature space.

- **LSTM layer.**

  The *lstmLayer* contains 100 hidden units (*numHiddenUnits = 100*) and is configured with OutputMode = "last". This layer learns temporal patterns and dependencies across the 100 time steps, compressing the entire sequence into a single latent representation at the final time step. In this way, the network bases its decision on the temporal evolution of the signals rather than on instantaneous values.

- **Dropout layer.**

  The *dropoutLayer* applies a dropout rate of 0.5 to the output of the LSTM layer. Its purpose is to reduce overfitting by randomly deactivating half of the units during training, encouraging the network to learn more robust and distributed representations of the temporal patterns.

- **Fully connected layer.**

  The *fullyConnectedLayer* maps the latent representation provided by the LSTM (after dropout) to the output space of the classification problem. It contains two neurons (*numClasses = 2*), each corresponding to one of the target classes: standard movement and abrupt movement.

- **Softmax layer.**

  The *softmaxLayer* converts the raw outputs of the fully connected layer into normalized class probabilities. This layer ensures that the network output can be interpreted as the likelihood of each window belonging to the standard or abrupt class.

- **Classification layer.**

  The *classificationLayer* computes the loss function used during training (cross-entropy) and provides the interface between the network outputs and the training algorithm. It compares the predicted class probabilities with the ground-truth labels and drives the gradient-based optimization of all preceding layers.

A representation of the network structure is depicted in Figure 37

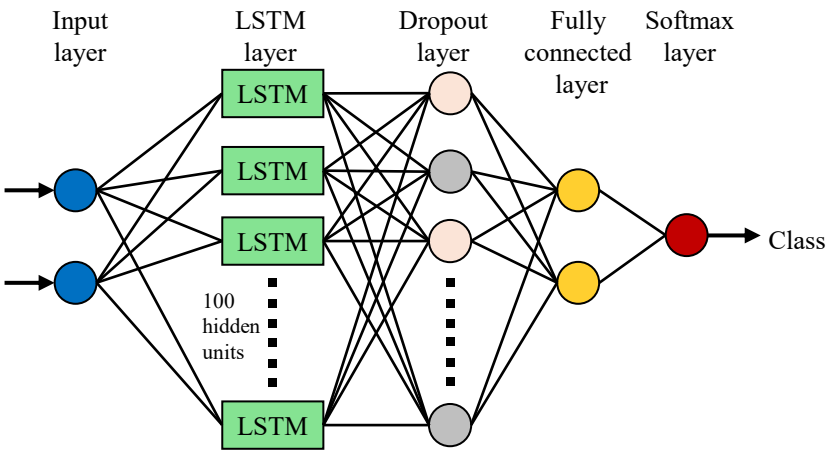

Figure 37 Network structure

This architecture allows the network to exploit the temporal structure of the kinematic signals while remaining sufficiently compact to be suitable for subsequent real-time implementation.

To ensure reliable learning and generalizable performance, the training data were carefully constructed and partitioned following established best practices in supervised time-series classification. Since the number of abrupt windows was substantially smaller than the number of standard windows, the abrupt class represented the limiting factor in defining a balanced learning set. To avoid biasing the classifier toward the majority class, 80% of the abrupt windows were selected for training, and an equal number of standard windows were randomly sampled. This sampling was performed while ensuring appropriate randomization across subjects and trial configurations, preventing the network from overfitting to specific individuals or execution patterns.

The remaining 20% of abrupt windows were reserved exclusively for testing. All standard windows not selected for training were similarly assigned to the test set. This strategy ensured that the test set remained substantially larger and more heterogeneous, providing a robust evaluation scenario that reflects the broad variability of standard behavior encountered in real applications.

A strict separation between training and testing is essential in machine learning to prevent information leakage: if samples too similar to the training data appear in the test set, the resulting accuracy may significantly overestimate the true generalization capability of the model. By enforcing this separation at the subject and trial level, the classifier is evaluated on data truly unseen during training, yielding a more realistic assessment of its performance for real-time deployment.

To further strengthen the reliability of the training process, a 5-fold cross-validation scheme was applied. In each fold, the balanced training set was divided into five partitions: four were used for model training and one for validation. This procedure was repeated five times so that each partition served as validation once. Cross-validation is particularly valuable in scenarios with class imbalance and inter-subject variability, as it reduces sensitivity to specific partitions of the data, mitigates overfitting, and provides a more stable estimate of model performance. The final validation accuracy was computed as the average across the five folds.

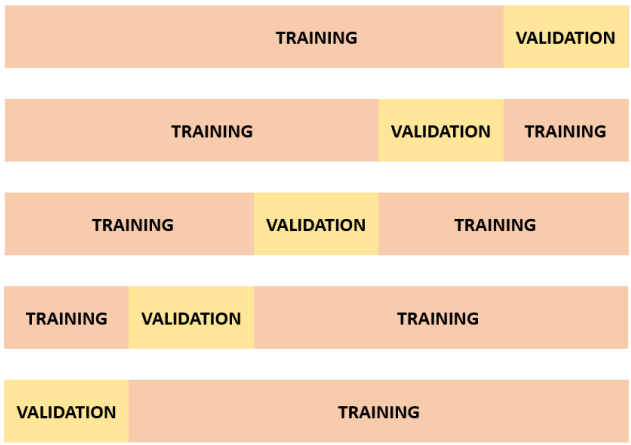

Figure 38 Schematic representation of the 5-fold cross validation training implementation

The training itself followed the architecture described in Section 4.4. Each fold was trained using the Adam optimizer, with a maximum of 20 epochs, a mini-batch size of 27, and gradient clipping set to a threshold of 2 to ensure stable convergence. The network weights were shuffled at every epoch to improve generalization, and validation was carried out at the end of each epoch using the fold-specific validation set. During training, only the last hidden state of the LSTM was used for classification, in accordance with the network configuration.

The performance of the trained LSTM network was evaluated using both the validation set and the independent test set. A standard confusion matrix was used to summarize the classification outcomes, and five quantitative metrics were computed: Accuracy, Precision, Recall, Specificity, and F1-Score. These metrics provide a comprehensive assessment of the classifier by capturing different aspects of its behavior, especially in the presence of class imbalance. A graphical interpretation of the classification outcome is depicted in Figure 39.

Accuracy represents the proportion of correctly classified samples over the total number of predictions. It expresses the overall correctness of the model.

Precision quantifies the proportion of samples classified as abrupt that are actually abrupt. High precision indicates a low rate of false positives, which is critical in safety-related applications, where unnecessary triggering of protective behaviors should be minimized.

Recall (or sensitivity) measures the proportion of true abrupt events that were correctly identified. A high recall ensures that the system seldom misses a true

abrupt motion, which is essential to guarantee timely reaction in collaborative scenarios.

Specificity reflects the proportion of standard movements correctly identified as non-abrupt. This metric complements recall by describing the model's ability to avoid misclassifying normal motions as abrupt.

F1-Score is the harmonic mean of precision and recall. It provides a balanced measure of performance, particularly useful when the two metrics must be simultaneously optimized. The complete formulas are reported in Eq. (32)-(36).

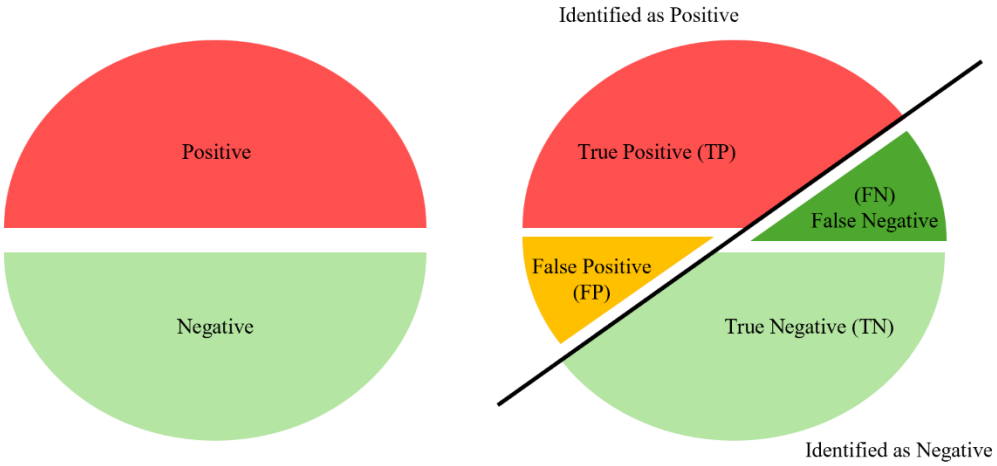

Figure 39 Graphical interpretation of classification outcomes and related performance metrics. The left diagram illustrates the two-class partition, while the right diagram shows how predicted and actual classes combine to form true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). These four components constitute the basis for computing accuracy, precision, recall, specificity, and F1-score.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{32}$$

$$Precision = \frac{TP}{TP + FP} \tag{33}$$

$$Recall = \frac{TP}{TP + FN} \tag{34}$$

$$Specificity = \frac{TN}{TN + FP} \tag{35}$$

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{36}$$

During training and validation, accuracy was used as the primary optimization metric.

After completing the training phase, the LSTM network achieved a final validation accuracy of 84.7%, indicating good discriminative capability in distinguishing between abrupt and standard movements. The corresponding confusion matrix is reported in Figure 40, the complete metric is reported in Table 10.

For the independent test phase, the network was evaluated using the remaining 20% of the abrupt windows not used during training, together with all

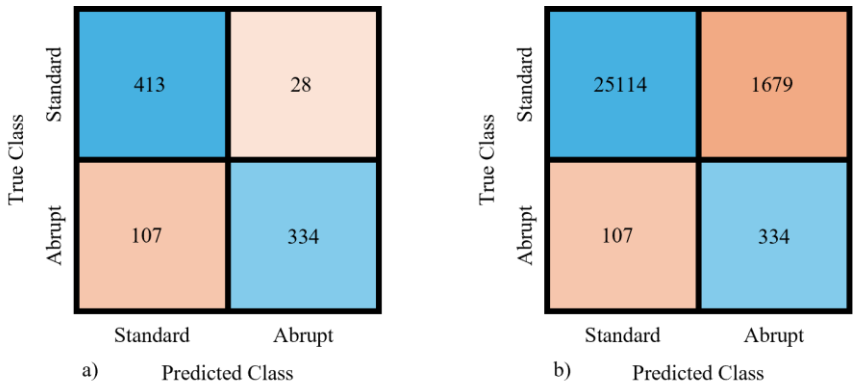

Figure 40 Confusion matrix obtained on the validation set a) and on the test set b).

the standard windows excluded from the training set. The quantitative results obtained on the test set are summarized in Table 11.

Table 10 Results metrics on the validation set

Accuracy	84,7 %
Precision	93,3 %
Recall	74,7 %
Specificity	93,7 %
F1 Score	83,2 %

Table 11 Results metrics on the test set

Accuracy	93,4 %
Precision	16,59 %
Recall	74,7 %
Specificity	93,7 %
F1 Score	27,2 %

The performance obtained on the independent test set highlights several important aspects of the classifier's behavior. As shown in the confusion matrix in Figure 40b, the network exhibits a substantial number of false positives, reflected in the low precision (16.59%). This corresponds to a large portion of standard movements incorrectly classified as abrupt (top-right cell of the matrix). Two main factors contribute to this behavior.

First, the test set is highly imbalanced, with the vast majority of windows belonging to the standard class. In such conditions, even a modest false-positive rate produces a large absolute number of misclassifications. Since the precision

metric is highly sensitive to the ratio between true positives and false positives, a small number of abrupt windows combined with many standard windows causes the precision to decrease rapidly.

Second, although standard movements are generally characterized by low variability, it is plausible that some 0.5 s windows classified as standard contain transient fluctuations or minor irregularities. These patterns may partially resemble the early phase of an abrupt reaction, leading the classifier to assign them to the abrupt class.

To contextualize the magnitude of the error, it is important to note that approximately 3.5 hours of recording were correctly classified as standard, whereas roughly 14 minutes were incorrectly classified as abrupt. Although this misclassification is not negligible, it represents a small fraction of the total duration of standard motion and was considered acceptable for a safety-oriented early-detection system, where prioritizing the avoidance of missed abrupt events makes false positives a safer and more conservative outcome.

Regarding the false negatives, the recall of 74.7% indicates that most abrupt movements were detected correctly, but a subset was missed by the classifier. A detailed inspection of these cases shows that the majority correspond to sub-window 4 of the abrupt sequence. This can be explained by the temporal dynamics of the reaction: for participants with faster response times, the impulsive component may occur entirely within sub-window 2 and 3, leaving sub-window 4 with only residual or returning-to-baseline motion, which is less distinguishable from standard behavior.

If sub-window 4 is excluded from the abrupt class definition, the number of missed abrupt movements decreases substantially, from the original value to 35 misclassified samples, further supporting the hypothesis that this segment does not consistently contain discriminative impulsive information.

Overall, the classifier demonstrates high accuracy (93.4%) and strong specificity (93.7%), meaning that standard movements are generally identified correctly despite the large number of false positives in absolute terms. The recall and F1-score reflect the expected challenges in detecting short, irregular, and highly variable abrupt movements, especially under the constraints of short temporal windows and real-time applicability.

Considering the intended application, the performance achieved by the network is adequate for safety-oriented early detection. In a real collaborative scenario, the identification of even a single abrupt window would be sufficient to trigger a stop command or a warning signal, ensuring an immediate protective reaction of the robotic system. From this perspective, the classifier's behavior aligns well with the requirements of safety-critical operation, where detecting potentially hazardous movements, rather than minimizing false alarms, is the primary objective.

Although the conservative behavior of the classifier is appropriate from a safety-oriented perspective, the relatively high number of false positives may reduce the efficiency of the collaborative application if each isolated detection directly triggers a robot slowdown or stop. Future work should therefore focus on

improving the training strategy to reduce false-positive detections while preserving the ability to identify abrupt movements promptly. A first improvement concerns the quality of the training dataset. In particular, the standard class should be further refined by identifying and excluding ambiguous windows that, although formally labelled as standard, contain transient peaks, irregular executions, or motion patterns that cannot be considered fully representative of nominal task behavior. A second improvement consists in exploiting the available dataset more effectively through weighted or cost-sensitive training. Instead of strongly reducing the number of standard windows to balance the dataset, class weights could be introduced to use a larger portion of the standard samples while accounting for the lower number of abrupt windows. This would expose the network to a broader variability of nominal movements and could improve its ability to discriminate between truly abrupt gestures and fast but non-critical standard motions. Moreover, an additional training stage could be specifically focused on false-positive samples, for example by identifying standard windows incorrectly classified as abrupt and reintroducing them as hard negative examples during retraining. Beyond the network itself, additional decision layers could be implemented to mitigate the practical effect of false positives. Finally, the actual impact of the classifier must be evaluated in a real collaborative environment, where operator movements are more natural and less constrained by the experimental setup. Such validation would allow the effective false-positive rate, the induced robot interruptions, and the resulting trade-off between safety and productivity to be assessed under realistic operating conditions.

Building on these results, the next section investigates how to translate the trained model into a real-time implementation. Particular attention is dedicated to defining an optimal window-overlap strategy and assessing the computational latency associated with signal processing and network inference, both of which are essential for achieving prompt and reliable detection in practical applications.

4.4 Network implementation

In the previous sections, the classifier was trained and evaluated using non-overlapping windows of 0.5 s, each providing a snapshot of the kinematic behavior within a fixed temporal interval. While this representation is suitable for offline training and analysis, the deployment of the method in a real-time context requires a sliding-window approach: the window is continuously shifted over the incoming data stream, and a classification is produced at each step. This mechanism allows the system to update its estimate of the current motion state as new samples become available.

In this framework, the overlap between consecutive windows becomes a key design parameter. A larger overlap (i.e., a smaller shift between windows) increases the temporal resolution of the detection: the impulsive portion of an abrupt movement is covered by a higher number of windows, thus raising the probability that at least one of them will be correctly classified as abrupt.

Conversely, a smaller overlap reduces the number of windows but also increases the risk of missing short-lived impulsive events. The overlap therefore defines a trade-off between detection performance and computational cost.

Another important consequence of the real-time perspective is the redefinition of the detection target. Rather than evaluating performance at the level of individual 0.5 s windows, the focus shifts to the correct classification of the entire 3 s movement. In this context, a movement is considered correctly detected as abrupt if at least one window within the relevant temporal interval is classified as abrupt. This criterion is more consistent with the intended safety function: a single positive detection is sufficient to trigger a stop or warning action in the robot controller. To move closer to the conditions of the final application, the overlap analysis and real-time simulations were implemented in Python, which is more suitable than MATLAB for integration with deployment frameworks and embedded systems. The LSTM architecture and hyperparameters were replicated exactly, and no relevant differences in recognition performance were observed between the MATLAB and Python implementations. The following analyses were performed on a machine with the following specifications: AMD Ryzen 5 3500U (6 cores, from 2.10 GHz up to 4.20 GHz), AMD Radeon Vega 8 Graphics.

For the overlap analysis, the fundamental parameters of the segmentation were kept consistent with the training phase. The input signal was sampled at 200 Hz, and each analysis window had a fixed duration of 0.5 s, corresponding to 100 samples. Let L denote the length of the signal segment under analysis (in samples), W the window length (100 samples), and S the step size between consecutive windows. The number N of generated windows can then be expressed as:

$$N = \left[\frac{L - W}{S}\right] + 1 \tag{37}$$

The overlap percentage is defined as:

$$Overlap = 1 - \frac{S}{W} \tag{38}$$

so that a larger overlap corresponds to a smaller step size and, consequently, to a larger number of windows per movement, a graphical illustration is presented in Figure 41.

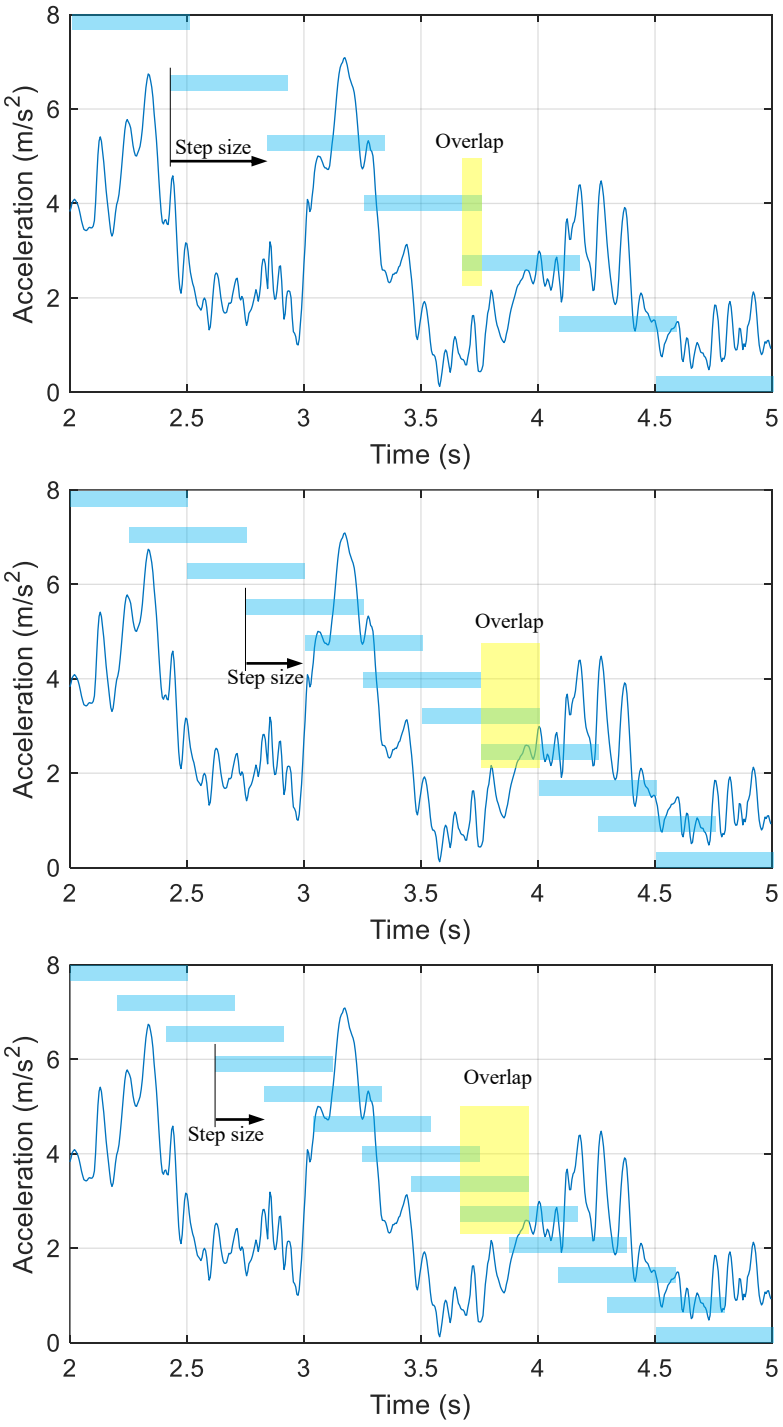

Figure 41 Example of sliding-window segmentation of the acceleration signal for different overlap configurations. Each panel shows 0.5 s windows (blue rectangles) shifted along the same time series with decreasing step size from top to bottom, corresponding to increasing overlap

Several overlap levels were tested, ranging from moderate overlap (e.g. 50%) to very high overlap (up to 99%). For each configuration, the step size S was adjusted accordingly, and the resulting number of windows per 3 s movement and per subject was recorded, the resulting windows for each configuration are reported in Table 12. In the case of abrupt movements, only windows whose centers fell within the interval in which the impulsive response is expected (typically between 0.5 s and 2.0 s from movement onset) were considered candidates for the abrupt class. This ensured consistency with the temporal analysis described in Section 4.3 and avoided artificially inflating the number of abrupt windows by including pre- and post-response segments. A focused example on abrupt segmentation is depicted in Figure 42.

Table 12 Step size, number of abrupt windows per movement, number of windows per trial, total windows per subject, and total abrupt windows per subject for the tested overlap levels, 50%, 75%, 90%, 95%, and 99%.

Overlap	Step size (samples)	Abrupt windows per movement	Windows per trial	Total windows per subject	Total abrupt windows per subject
50%	50	5	359	1077	60
75%	25	9	717	2151	108
90%	10	21	1791	5373	252
95%	5	41	3581	10743	492
99%	1	201	17901	53703	2412

Intuitively, increasing the overlap leads to a denser sampling of the kinematic signal: the impulsive region of an abrupt movement is covered by multiple, strongly overlapping windows, while standard movements generate a larger set of windows with similar content. This increases the probability of detecting at least one abrupt window within an abrupt movement, but also raises the computational load, since both the number of windows to be processed and the frequency of classification calls grow with the overlap. The following sections quantify these effects in terms of recognition performance at movement level and in terms of processing time, to identify an overlap configuration that is compatible with real-time operation.

The first analysis considered the effect of overlap on the recognition performance at window level. For each overlap configuration (50%, 75%, 90%, 95%, and 99%), the network predictions for all sliding windows were compared with the corresponding ground-truth labels, and confusion matrices were constructed. From these, accuracy, balanced accuracy, precision, recall, specificity and F1-score were computed, the overall results are shown in Figure 43.

Here the balanced accuracy is introduced; this metric describes the overall network performance more effectively in the presence of a strongly imbalanced dataset.

$$Balanced\ Accuracy = \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right) \quad (39)$$

Across all overlap percentages, the balanced accuracy remained approximately constant and above 80%, indicating that the network's ability to distinguish between abrupt and standard windows is not strongly affected by the degree of overlap.

However, the analysis of precision, recall and F1-score for the abrupt class shows that window-level metrics can be difficult to interpret in this context. Precision values remain comparatively low and tend to decrease as the overlap increases, reflecting a substantial number of false-positive windows. Recall remains higher. This behavior can be explained by the way abrupt movements are distributed over multiple overlapping windows: within a single impulsive movement, the network may correctly identify the event but assign the abrupt label to a number of windows that does not exactly match the expected count. As a consequence, window-based metrics may penalize the classifier even when the movement is, in practice, correctly detected.

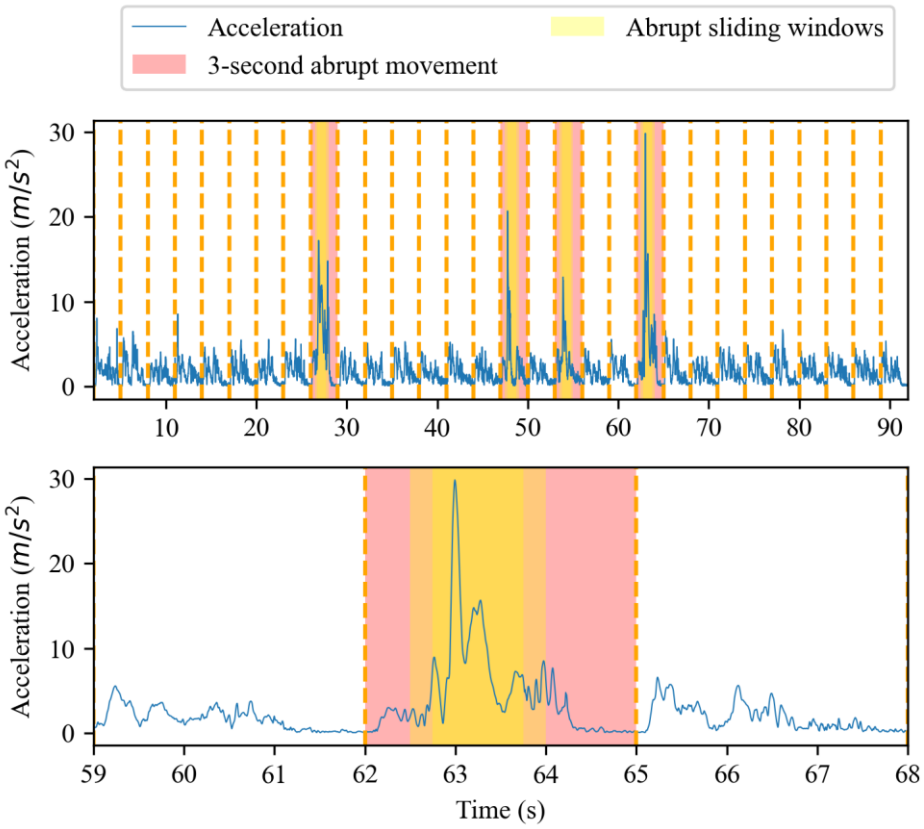

Figure 42 (top) Acceleration signal segmented into 3 s windows (orange dashed vertical lines), with the four abrupt movements (red rectangles) and the abrupt sliding windows with 50% overlap highlighted (yellow rectangles), (bottom) zoomed view of the fourth abrupt movement. The central yellow area appears brighter as a result of the overlap of multiple windows.

Given these considerations, window-level performance is useful to verify that the overlap does not degrade the discriminative capability of the network, but it is not fully representative of the practical behavior of the system. For safety-related

applications, what ultimately matters is whether an abrupt movement is detected at least once during its duration. This motivates the transition to a movement-level analysis.

To align the evaluation with the intended use of the classifier, the detection

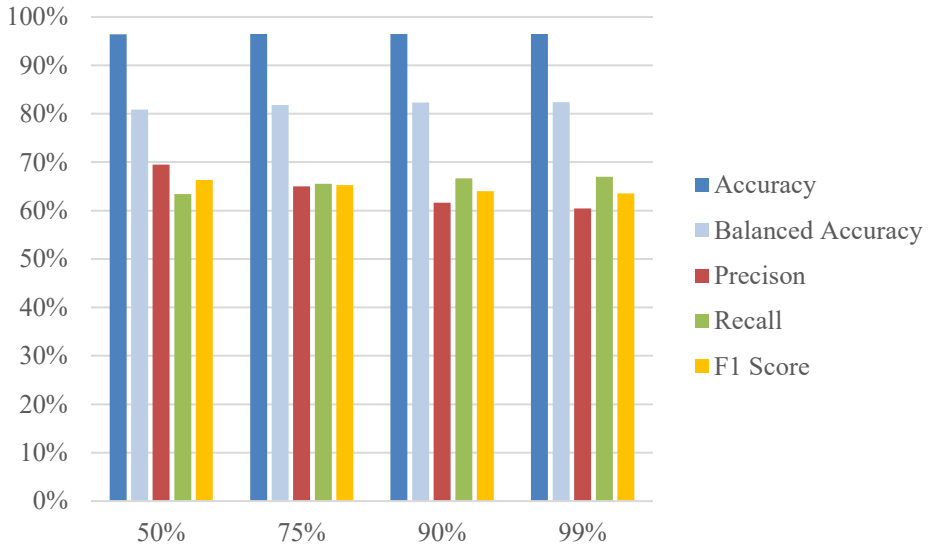

Figure 43 Performance metrics for the different overlap configurations for window accuracy

problem was reformulated at movement level. Each 3 s movement was considered correctly recognized as abrupt if at least one sliding window within the movement was classified as abrupt; otherwise, the movement was considered standard. On this basis, confusion matrices were constructed for each overlap configuration by comparing, for all subjects and trials, the set of movements commanded as abrupt with those identified as abrupt by the network.

The resulting metrics show that the network achieves high balanced accuracy, with values consistently above approximately 83% for all overlap percentages (Figure 44). This indicates a good overall ability to discriminate between abrupt and standard movements, independent of the chosen overlap. The introduction of the movement-level criterion does not modify the network itself, which is not retrained, nor does it alter the output of the classifier at window level. The LSTM still produces the same prediction for each 0.5 s window; what changes is only the interpretation of these predictions. In particular, the redefinition of the abrupt class operates solely at the decision level: windows are grouped into 3 s movements, and a movement is considered abrupt if at least one of its windows is classified as abrupt. This reformulation is introduced to approximate more closely how the system would behave under real operating conditions, where a single positive detection within a movement would be sufficient to trigger a safety reaction. As a consequence, the number of windows counted as abrupt for evaluation is reduced, while the set of standard windows and their associated false positives remains unchanged.

This explains the observed decrease in precision for the abrupt class when switching from window-level to movement-level evaluation. By aggregating

windows into movements, the effective number of abrupt events becomes smaller, whereas the number of false positives generated on standard windows is not affected. The same mechanism also accounts for the further reduction in precision as the overlap increases: a higher overlap does not increase the number of abrupt movements, but it does increase the likelihood that at least one window within a standard movement is misclassified as abrupt, thereby raising the false-positive count.

In contrast, the behavior of the recall remains very favorable. The movement-level recall indicates that the system almost always succeeds in detecting at least one abrupt window within each abrupt movement. Increasing the overlap does not substantially change this result, because the classifier was already able to recognize nearly all abrupt movements even at lower overlap levels. From a safety perspective, this behavior is desirable: the system rarely misses truly abrupt movements (low false-negative rate), while a certain number of false positives, standard movements incorrectly flagged as abrupt, is acceptable, as it only results in conservative slowdowns or stops of the robot. Moreover, the stability of movement-level abrupt identification across different overlap percentages suggests that the overlap can be chosen mainly based on computational considerations and false positive rate, without compromising the reliability of abrupt-movement detection. The next section therefore focuses on the analysis of inference times as a function of overlap, with the aim of identifying a configuration that satisfies real-time constraints while preserving the movement-level performance described above.

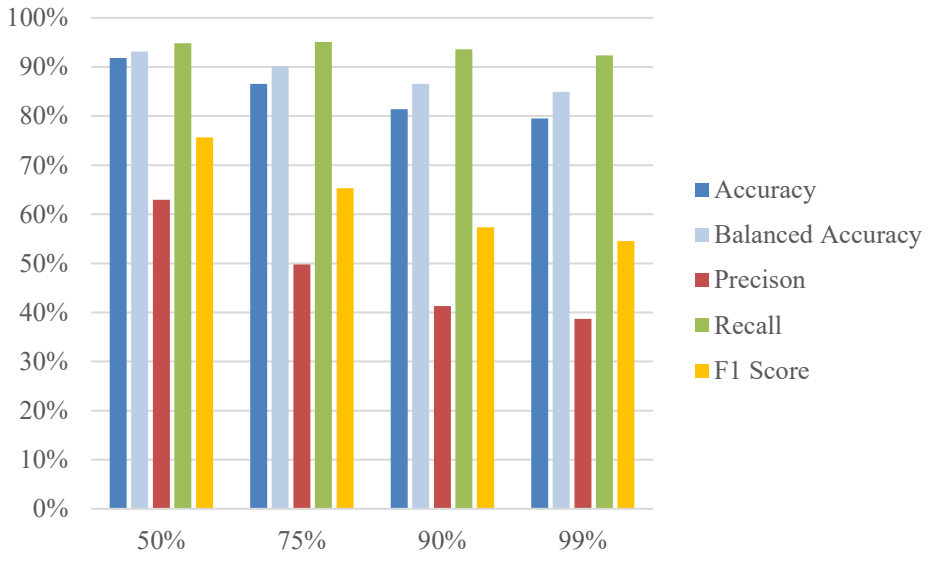

Figure 44 Performance metrics for the different overlap configurations for the redefined abrupt identification rule

In addition to recognition performance, the feasibility of deploying the proposed method in a real-time scenario depends critically on the computational cost associated with the sliding-window classification. For each overlap configuration, the inference time of the LSTM network was therefore quantified

in the same Python environment used for the overlap analysis. The laptop adopted for this analysis is has the following specification : AMD Ryzen 5 3500U (6 cores, from 2.10 GHz up to 4.20 GHz), AMD Radeon Vega 8 Graphics.

A dedicated timing analysis was carried out using the Python function *time.perf_counter()*, which provides a high-resolution wall-clock timer. The timer was started at the beginning of the classification process and stopped at the end of the computation. Two inference-time indicators were computed:

1. **Average inference time.** For each subject, the time required by the network to classify all windows associated with that subject was measured. The average inference time per subject was then obtained by averaging these values over all 60 subjects.
2. **Total inference time.** The total time required to classify the complete dataset (all 60 subjects) was measured by starting the timer at the beginning of the outer for loop, where subjects were processed one by one, and stopping it after the loop had completed.

For each overlap level, the total number of windows, the average inference time per subject, and the total inference time are reported in Table 13. As expected, the computational cost is strongly influenced by the overlap: increasing the overlap reduces the step size between consecutive windows, thereby increasing the total number of windows to be processed and, consequently, the total inference time.

Table 13 Average inference time across all subjects and total inference time (in seconds) for each overlap percentage

Overlap	50%	75%	90%	95%	99%
Average inference time (s)	1.34	2.14	4.65	9.43	63.34
Total inference time (s)	81.95	130.67	283.76	574.95	3985.81

For moderate overlap values (e.g. 50%–90%), the growth in the number of windows and the corresponding inference time remains limited. In these

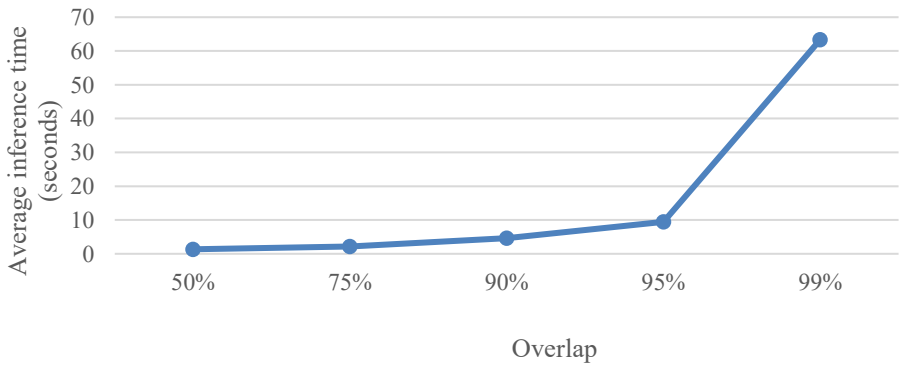

Figure 45 Average inference time trend over overlap percentage

conditions, the average processing time per subject is comfortably below the

actual duration of the recordings, indicating that the classifier can operate in or faster than real time on a standard computer. For very high overlap values (e.g. 95%–99%), the number of windows per movement grows rapidly (Figure 45), leading to a disproportionate increase in inference time while providing only marginal improvements in movement-level detection performance, as shown in the previous section.

A further insight is provided by analyzing the average inference time per window, reported in Table 14. The values remain essentially constant across all overlap configurations, with an average classification time of approximately 1 ms per window. This confirms that the computational cost of a single forward pass through the LSTM network is relatively low and does not depend significantly on the overlap. The substantial differences in total and per-subject inference time observed for higher overlaps are therefore almost entirely attributable to the increased number of windows to be classified, rather than to any change in the per-window processing time.

Table 14 Average time required to analyze a single window (in milliseconds) for each overlap percentage

Overlap	50%	75%	90%	95%	99%
Average inference time per windows (ms)	1.24	0.99	0.87	0.88	1.22

From an implementation perspective, the real-time applicability of the proposed method is ultimately governed by the temporal constraints imposed by the sliding-window scheme. In a purely serial architecture, the processing pipeline can be decomposed into three stages: (i) data acquisition, (ii) signal pre-processing (e.g. computation of norms and buffering of samples into windows), and (iii) LSTM inference. For the system to operate in real time, the total processing time for one step of this pipeline must remain strictly below the step size of the sliding window. In other words, the sum of acquisition, pre-processing, and classification times must be shorter than the temporal shift between consecutive windows; otherwise, a new classification request would be triggered before the previous one has been completed, causing a growing delay that would progressively desynchronies the classifier from the incoming data stream, an example of optimal and non-optimal configuration is shown in Figure 46.

The timing analysis presented above indicates that the per-window inference time is on the order of 1 ms and essentially independent of the overlap configuration. This suggests that the main factor affecting real-time feasibility is not the computational cost of the network itself, but the total number of windows generated by the chosen overlap. In practical terms, this means that the overlap should be selected so as to provide sufficient temporal resolution for abrupt-movement detection while keeping the aggregate processing time per step (including pre-processing and any communication overhead) comfortably below the step size. Additional considerations may include the latency of data transfer between sensors and processing unit, the execution environment (embedded vs.

host PC), and the safety requirements on the robot's reaction time. Taken together, these aspects define the operating envelope within which the proposed classifier can be deployed as a reliable and timely component of a real-time safety supervision module.

Based on the considerations above, an overlap of 90% appears to be a suitable choice for the proposed system. This configuration provides a high temporal density of coverage, ensuring that the impulsive phase of abrupt movements is sampled by multiple windows, while still allowing a processing budget of approximately 10 ms per step, which is compatible with the inference times estimated for the different stages of the pipeline (pre-processing and classification).

It is important to emphasize that the timing results reported here depend on several hardware and software factors, including processor architecture, implementation details, and the number of concurrent processes running on the system. In the offline analysis, the computational resources were largely dedicated to window processing, whereas in the final application the classifier will share the available resources with other tasks (e.g. sensor management, robot control, logging, user interface). As a consequence, a final tuning of the overlap and processing parameters will be required at deployment time, to verify that the real-time constraints are met under the actual operating conditions and to adjust the configuration if necessary.

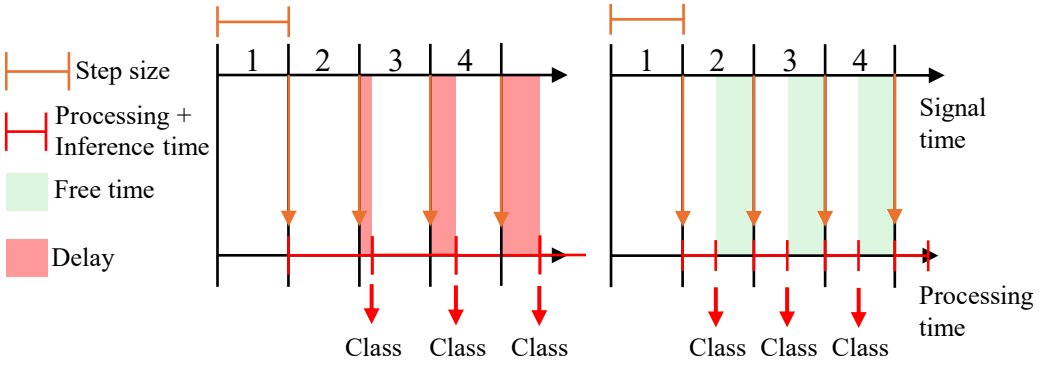

Figure 46 Schematic representation of the temporal behavior of the sliding-window pipeline. (right) optimal configuration; (left) non-optimal configuration. In the optimal case, the processing time for acquisition, pre-processing, and classification is shorter than the step size, so each classification is completed before the next window is triggered. In the non-optimal case, the processing time exceeds the step size, causing overlapping processing blocks and an accumulating delay that progressively desynchronizes the classifier from the data stream.

4.5 Conclusion

This chapter presented a complete framework for the prompt recognition of abrupt upper-limb movements based on wearable inertial sensing and an LSTM neural architecture, with the goal of enhancing safety in human–robot collaboration. Starting from the requirements imposed by the updated regulatory framework, particularly the need to distinguish voluntary from involuntary interactions, the work focused on movements that deviate from the standardized, rhythmic patterns typical of industrial tasks and that therefore undermine the assumptions of predictability on which many collaborative control strategies rely.

A dedicated experimental protocol was designed to reproduce representative standard pick-and-place actions and to elicit abrupt reactions through controlled visual and auditory stimuli. The resulting dataset, collected from 60 participants and instrumented with wrist-worn MIMUs, was analyzed to characterize the kinematic signatures of abrupt movements and to verify their statistical separability from standard gestures. This analysis showed that discriminative information is concentrated in a limited temporal interval following the stimulus and led to the definition of a 0.5 s windowing strategy focused on the impulsive phase of the motion.

On this basis, an LSTM network was designed and trained on two features, the norms of linear acceleration and angular velocity, using a balanced dataset and 5-fold cross-validation. The model achieved a validation accuracy of 84.7% and, when evaluated on an independent test set, showed high accuracy and specificity, together with a recall that confirmed its capability to detect the majority of abrupt events. The relatively low precision on the test set was interpreted in light of the strong class imbalance and the conservative choice of favoring missed-standard over missed-abrupt classifications, which is consistent with the safety-oriented objective of the system.

To bridge the gap between offline training and deployment, the detection problem was reformulated at movement level and a sliding-window scheme with different overlap percentages was investigated. The movement-level analysis demonstrated that the network maintains high balanced accuracy and recall across overlaps, while precision is mainly affected by the rate of false positives at standard movements. A detailed timing study in Python showed that the per-window inference time is on the order of 1 ms and essentially independent of overlap, and that the computational cost grows primarily with the number of windows to be processed.

Combining recognition performance and computational considerations, an overlap of 90% was identified as a suitable compromise: it provides dense temporal coverage of the impulsive phase and leaves a processing budget of approximately 10 ms per step, which is compatible with real-time constraints on standard hardware. At the same time, it ensures that an abrupt movement is very likely to be detected at least once during its duration, enabling timely triggering of protective actions in the robot controller.

Future work will focus on the integration of the proposed framework into a complete collaborative application, including real-time data acquisition from wearable sensors, embedded implementation of pre-processing and inference, and coupling with the robot safety controller. Additional efforts will be devoted to refining the decision logic (e.g. adaptive thresholds, fusion with contextual information), extending the approach to multi-sensor configurations or additional kinematic features, and validating the system in realistic industrial scenarios under the constraints of the current safety standards.

Chapter 5

Path planning with RL

5.1 Introduction

The advent of collaborative robotics has progressively removed the physical separation between humans and robots, enabling shared workspaces in which close interaction is required. As a consequence, robot motion can no longer rely on fixed, offline pre-programmed trajectories, but must instead adapt online to the presence and behavior of human operators. In such contexts, robot trajectories are defined dynamically as a function of the current workspace conditions, including the operator's position, the presence of tools, and the geometry of the manipulated objects.

These requirements introduce additional constraints on motion planning, as the generated trajectories must simultaneously ensure task execution, internal self-collision avoidance, and safe interaction with external entities in a highly variable and partially unpredictable environment. Classical motion planning algorithms have been extensively developed to address these challenges, including sampling-based and optimization-based approaches such as Covariant Hamiltonian Optimization for Motion Planning (CHOMP), Rapidly-exploring Random Tree (RRT), and Probabilistic Roadmap (PRM). These methods are capable of generating collision-free trajectories in complex workspaces and allow optimization with respect to specific criteria, such as execution time or trajectory smoothness.

However, despite their effectiveness, classical planners exhibit several limitations when applied to collaborative and highly dynamic scenarios. In particular, their computational cost tends to increase significantly with the dimensionality of the robot and the complexity of the environment, making real-time re-planning challenging. Moreover, these algorithms typically require explicit modelling of constraints and cost functions, which must be carefully tuned to balance competing objectives such as safety, smoothness, and task efficiency. In environments characterized by frequent changes, such as human

motion or tool manipulation, classical planners often rely on repeated re-planning cycles, which may result in discontinuous trajectories or delayed reactions to external events.

To overcome these limitations, artificial intelligence techniques, and in particular reinforcement learning (RL), have gained increasing attention in the field of robot motion planning. Reinforcement learning is a model-free machine learning paradigm based on trial-and-error interactions with an environment, enabling an agent to learn control policies for tasks of varying complexity. RL-based approaches offer the potential to generate adaptive, continuous, and constraint-aware motions without relying on explicit analytical planners.

Reinforcement learning has already demonstrated remarkable performance in robotic control problems, especially in locomotion and whole-body control of legged robots. Its extension to motion planning for collaborative manipulators opens new perspectives for learning proprioception-like policies that implicitly encode kinematic constraints, collision avoidance, and task objectives.

The problem addressed in this chapter concerns the development of a reinforcement learning–based agent capable of planning and/or controlling a robotic manipulator such that, given an initial and a target pose, a safe and collision-free motion trajectory is generated. The objective is to ensure that the resulting motion satisfies task requirements while avoiding both internal self-collisions and collisions with static elements of the environment.

To this end, the problem is investigated through two different approaches, both relying on reinforcement learning but characterized by different levels of complexity and abstraction. The first, more challenging approach, aims to teach an agent that directly controls the manipulator joints online, generating joint-level commands at each time step without relying on an explicit trajectory planner. In this case, motion planning and control are implicitly embedded within the learned policy. The second approach, more closely aligned with classical motion planning methodologies, focuses on obtaining an agent capable of generating a feasible cartesian trajectory connecting the initial and final poses. The resulting trajectory can then be executed by a lower-level controller, preserving a clearer separation between planning and control.

Despite these differences, both approaches share a common objective: enabling the agent to internalize the kinematic and geometric constraints of the robot. In particular, the learning process is designed to promote an implicit awareness of the manipulator's spatial occupancy, leading to a proprioception-like behavior. This allows the agent to generate motions that are intrinsically collision-free with respect to both internal robot structures and fixed elements of the environment, without the need for explicit collision avoidance strategies during execution.

The proposed reinforcement learning framework is implemented in MATLAB, exploiting a simulation-based environment in which the agent interacts with the robot model and the surrounding workspace. The manipulator considered in this work is a UR3 collaborative robot.

In this chapter, the more complex joint-level control approach is first presented, followed by the simplified path-planning formulation, allowing a direct comparison between the two strategies in terms of learning complexity, performance, and robustness.

Reinforcement learning: background and terminology

Reinforcement learning is a learning paradigm within machine learning that addresses the problem of sequential decision making under uncertainty. Its defining characteristic is the ability of an autonomous agent to learn an optimal behavior through direct interaction with an environment, without relying on explicit supervision or labelled datasets. Instead, learning is driven by evaluative feedback in the form of scalar rewards, which quantify the desirability of the agent's actions over time as illustrated in Figure 47. This formulation makes reinforcement learning particularly suitable for control, robotics, and decision-making problems, where actions influence not only immediate outcomes but also future states of the system.

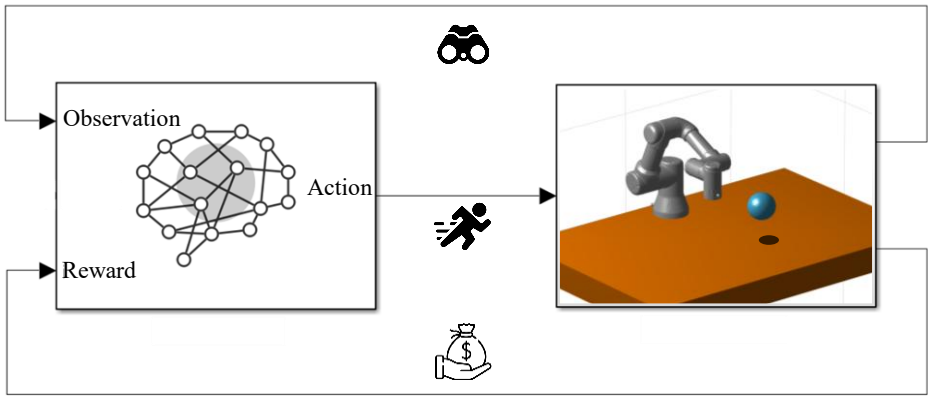

Figure 47 Reinforcement Learning paradigm.

Unlike supervised learning, where the goal is to approximate a static input–output mapping, reinforcement learning focuses on learning a policy, i.e. a strategy that defines how actions are selected as a function of the current state of the environment. The policy is optimized to maximize a long-term objective, typically expressed as the cumulative reward obtained during the interaction. This long-term perspective distinguishes reinforcement learning from greedy or myopic optimization approaches and allows the agent to reason about delayed effects of its actions [101].

The reinforcement learning problem is commonly formalized using the framework of Markov Decision Processes (MDPs). An MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{S}$ represents the set of possible states of the environment, $\mathcal{A}$ the set of available actions, $\mathcal{P}$ the state transition probability distribution, r the reward function, and $\gamma \in [0,1]$ the discount factor that determines the relative importance of future rewards. At each discrete time step, the agent observes the current state, selects an action according to its policy, receives a reward, and

transitions to a new state. Through repeated interactions, the agent progressively improves its policy in order to maximize the expected cumulative discounted return.

A fundamental concept in reinforcement learning is the notion of value functions. The state-value function quantifies the expected return achievable from a given state when following a specific policy, while the action-value function extends this concept to state–action pairs. These functions provide a measure of long-term utility and play a central role in many reinforcement learning algorithms, either as explicit learning targets or as internal signals guiding policy improvement. From an optimization perspective, reinforcement learning can be interpreted as the problem of estimating value functions and improving the policy based on these estimates.

Reinforcement learning methods can be broadly categorized into model-based and model-free approaches. Model-based methods explicitly exploit a model of the environment dynamics to plan or simulate future outcomes, whereas model-free methods learn optimal behavior directly from experience, without requiring prior knowledge of the transition dynamics. In practical applications, and particularly in robotics, model-free reinforcement learning is often preferred due to its ability to handle complex, nonlinear systems and partially unknown environments [102].

When reinforcement learning is applied to problems characterized by continuous state and action spaces, as is the case for robotic manipulators, function approximation becomes necessary. In modern reinforcement learning, neural networks are commonly employed to approximate policies and value functions, giving rise to the field of deep reinforcement learning. Among the various algorithmic families, actor–critic methods have emerged as a prominent solution for continuous control problems. These methods combine an explicit policy representation (the actor) with a value function estimator (the critic), enabling stable learning and efficient optimization in high-dimensional spaces, Figure 48.

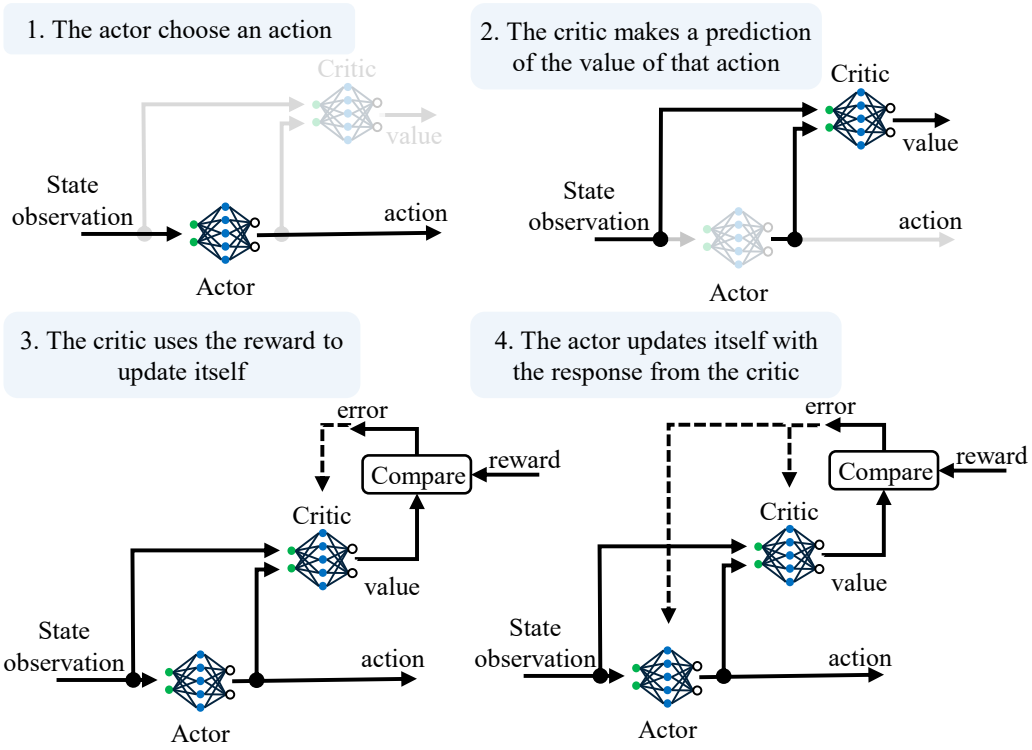

Figure 48 Actor-Critic algorithms working principle [97]

Due to its generality and flexibility, reinforcement learning has been successfully applied to a wide range of domains, including game playing, autonomous navigation, and robotic control. In robotics, RL is particularly attractive because it allows control policies to be learned directly from interaction, potentially integrating motion generation, constraint handling, and task execution within a unified framework.

5.3 Robot and set up descriptions

The robotic manipulator employed in this study is the UR3, a six-degree-of-freedom (6-DoF) collaborative robot developed by Universal Robots. The UR3 is a serial manipulator specifically designed for safe operation in shared human–robot workspaces, making it a suitable platform for investigating learning-based motion planning and control strategies in collaborative scenarios, a picture of the robot is reported in Figure 49.

The kinematic chain of the UR3 consists of six revolute joints, which can be conceptually divided into three arm joints and three wrist joints. The first three joints primarily contribute to positioning the end-effector within the workspace, while the last three joints form a non-monocentric wrist. As a result, wrist joint motions affect both the position and the orientation of the end-effector.

The robot features a maximum reach of approximately 500 mm and a nominal payload capacity of 3 kg. All joints are subject to rotational limits of ±360 degrees, with the exception of the terminal joint, which allows continuous rotation. The maximum joint velocities are 180 deg/s for the first three joints and 360 deg/s for the wrist joints. External control commands can be issued at a

maximum frequency of 125 Hz, which defines the upper bound for the control loop rate in the reinforcement learning environment.

For the purposes of this study, the UR3 is modelled using the rigid-body representation provided by the Robotics System Toolbox in MATLAB. This model includes the full kinematic structure of the manipulator, joint limits, and coordinate frame definitions. The robot model is shown in Figure 49 (right), where the robot is depicted in its zero-axis configuration. In particular, the *base_link* frame and the *tool0* frame, located respectively at the base and at the end-effector of the manipulator, play a central role in the subsequent analysis. These two frames are used to express the robot pose, task objectives, and Cartesian quantities within the reinforcement learning framework. For collision detection purposes, the robot geometry is approximated using simple bounding shapes that envelope the real body of the manipulator. This simplified representation enables efficient distance and collision computations while retaining a conservative description of the robot's spatial occupancy, as shown in Figure 50.

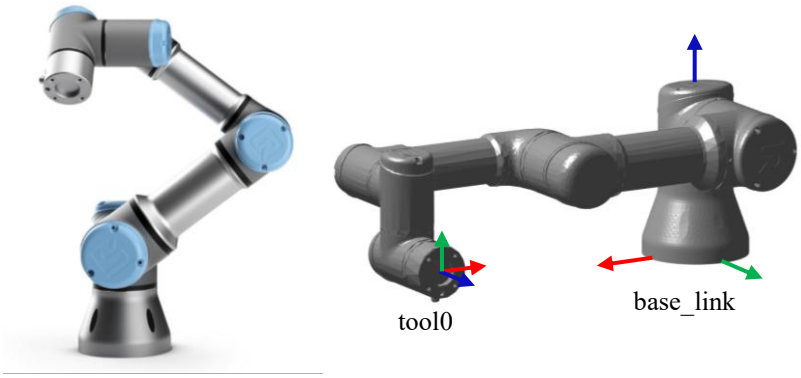

Figure 49 (Left) Physical UR3 collaborative robot. (Right) Corresponding UR3 model depicted in the zero joint configuration, highlighting the base_link and tool0 reference frames.

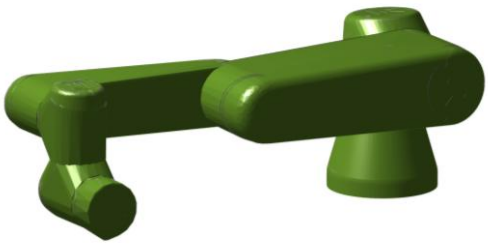

Figure 50 Simplified collision model of the UR3 manipulator. The robot links are approximated by basic geometric primitives that conservatively envelope the real structure

Common Set up

Both reinforcement learning formulations investigated in this chapter rely on a common experimental setup, which is shared by the two approaches. The setup consists of a UR3 collaborative robotic manipulator, the fixed bench on which the robot is mounted, and a single obstacle representing the presence of a human operator. A three-dimensional representation of the complete system and workspace is shown in Figure 51.

The robot workspace is limited to the physical volume defined by the bench area. Within this constrained workspace, the robot is required to move from an initial pose to a target pose while avoiding collisions with the supporting bench, the obstacle, and its own structure. The reinforcement learning agent is therefore responsible for determining how the robot transitions from the initial configuration to the final one while satisfying all safety and feasibility constraints.

Both the initial pose, the target pose, and the obstacle position are allowed to vary within the workspace boundaries enabling the evaluation of the agent's ability to generalize across different task configurations while operating under a consistent physical and geometric setup. A further assumption concerns the temporal behaviour of the obstacle. In this work, the obstacle is treated as static during the execution of each generated trajectory. For each planning query, the initial point, final point, and obstacle position are sampled, and the obstacle remains fixed until the trajectory is completed.

All Cartesian quantities defining the task, including poses and obstacle position, are expressed in the *base_link* reference frame, which is adopted throughout the chapter as the global frame of reference.

The common setup described in this section provides a common context for the two reinforcement learning approaches presented in the following sections.

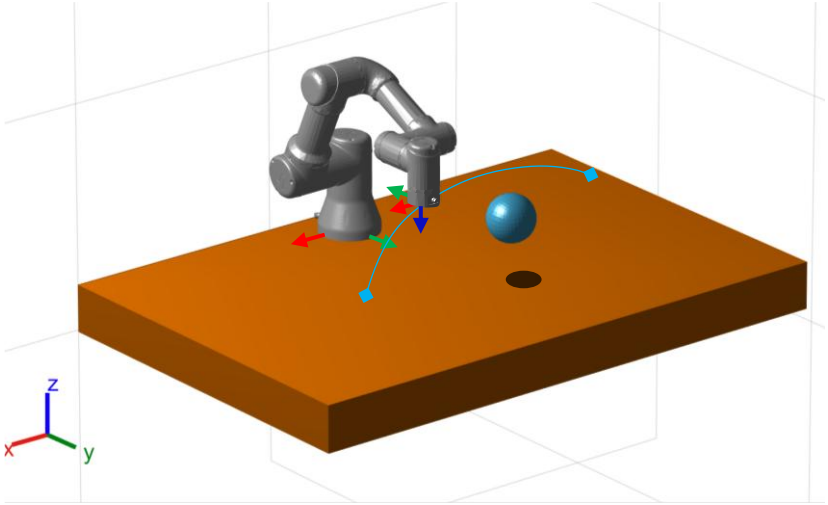

Figure 51 Tridimensional representation of the environment, the base_link references frame and the tool0 reference frame are reported

5.3 Joint-control environment

In the joint-control reinforcement learning environment, the agent is tasked with directly controlling the UR3 manipulator at joint level in order to move the robot from an initial pose to a target pose within the common experimental setup described in the previous section. In this formulation, no explicit trajectory planner or inverse kinematics solver is employed online. Instead, the agent generates joint-space velocity commands at each control step, while task objectives and safety constraints are defined in Cartesian space.

The overall structure of the joint-control environment is illustrated in Figure 52, which reports the Simulink implementation adopted for training. The environment encapsulates the kinematic model of the robot (blue boxes), collision checking modules (red box), reward computation (green box), and the interaction loop with the reinforcement learning agent (yellow box). At each time step, the agent receives observations describing the current state of the system and outputs joint velocity commands, which are applied to the robot model to update its configuration. The blocks *PtO*, *d0*, *traj*, and *PtF* represent, respectively, the obstacle position, the initial distance between the robot and the obstacle, a predefined trajectory, and the target point.

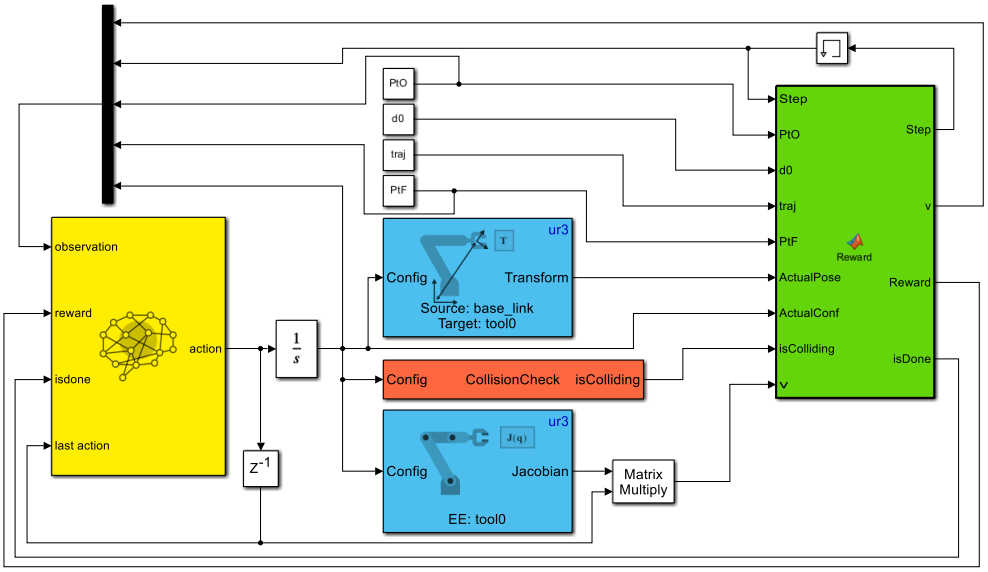

Figure 52 Simulink framework utilized to perform the reinforcement learning training

The observations include the current joint configuration of the manipulator, the Cartesian position of the target, the Cartesian position of the obstacle representing the human operator, the current Cartesian velocity of the tool center point (TCP), and a step counter used as a proxy for time progression within the episode. Together, these quantities allow the agent to estimate the robot state, evaluate task progress, and reason about safety-related aspects of the motion.

The definition of the observation and action spaces determines what the agent can perceive and how it can influence the environment. However, these elements alone are not sufficient to fully specify the reinforcement learning problem. A critical component is the definition of the training objective, which describes the desired behavior that the agent is expected to acquire through interaction with the environment.

This objective is encoded through the reward function, which provides a scalar evaluation of the agent's behavior at each time step. The reward function plays a central role in shaping the learning process, as it represents the only feedback signal through which the agent can assess the quality of its actions. In the joint-control formulation, the long-term objective is to generate robot motions that drive the manipulator from an initial configuration to a target pose while satisfying multiple, potentially competing requirements.

This formulation represents the most challenging learning scenario investigated in this chapter. By acting directly in joint space, the agent is required to implicitly learn the mapping between joint-level actions and Cartesian task outcomes, effectively internalizing the inverse kinematics of the manipulator. At the same time, the agent must satisfy task execution requirements while avoiding internal self-collisions and ensuring safe interaction with the surrounding environment.

Unlike classical motion planning approaches, where planning and control are handled by separate algorithmic modules, this joint-control formulation delegates both functions to a single learned policy. As a consequence, the agent must simultaneously address planning, control, and safety within a unified decision-making framework, resulting in a learning problem characterized by high dimensionality and significant complexity.

For this reason, a progressive training strategy is adopted, in which task requirements and constraints are gradually introduced to guide the learning process. The details of this strategy, together with the corresponding reward formulation and training results, are discussed in the following sections.

Observation space, action space, and environment dynamics

In this section, the observation space, action space, and environment dynamics adopted for the joint-level reinforcement learning control of the UR3 manipulator are described.

The agent receives a continuous observation vector composed of a total of 19 scalar quantities, designed to provide sufficient information to characterize the current state of the system and the task context. The observations include:

- **Target position (3 values):** The three-dimensional Cartesian coordinates of the target point to be reached by the end-effector. These coordinates are expressed in the *base_link* reference frame.
- **Obstacle (operator) position (3 values):** The three-dimensional Cartesian coordinates of the center representing the position of the obstacle. In the considered setup, this obstacle is static during each episode and is also expressed in the *base_link* reference frame, the obstacle is represented as a sphere of 10 cm of diameter.
- **Step counter (1 value):** An integer counter incremented at each control step, providing the agent with information about the temporal progress of the episode.
- **Current joint configuration (6 values):** The angular positions of the six revolute joints of the UR3 manipulator.
- **TCP Cartesian velocity (6 values):** The linear $v_{TCP} = [v_x, v_y, v_z]$ and angular $\omega_{TCP} = [\omega_x, \omega_y, \omega_z]$ velocity components of the tool center point (TCP), expressed in Cartesian space.

The action space is continuous and consists of six scalar commands, each representing a joint angular velocity command for the UR3 manipulator. The actions directly control the robot at joint level and are subject to saturation consistent with the physical limits of the real system. Specifically, the commanded joint velocities are bounded to ±180 deg/s for the first three joints and ±360 deg/s for the wrist joints. This constraint ensures that all actions applied within the simulation remain physically feasible and transferable to the real robot.

The environment dynamics are modelled at a kinematic level. At each control step, the joint velocity command provided by the agent is numerically integrated to update the joint configuration of the manipulator. The updated joint positions are then used to compute the forward kinematics, yielding the current TCP position. The manipulator Jacobian is evaluated at the current configuration and is used to map joint velocities to TCP Cartesian velocities, which form part of the observation vector.

The updated robot configuration is also employed to evaluate internal self-collisions and collisions with external entities, including the static obstacle representing the human operator. These collision checks, together with task-related quantities such as the distance to the target, are used to compute the reward signal that guides the learning process.

An episode is terminated when one of the following conditions is met:

1. a predefined maximum number of steps is exceeded,
2. a collision (internal or external) occurs,
3. the target position is successfully reached.

Reward function design

The reward function represents the core component of the reinforcement learning framework, as it constitutes the only mechanism through which the

desired behavior is communicated to the agent. At each interaction step, the reward provides a scalar evaluation of the action performed, indicating whether the resulting system evolution is consistent with the expected behavior. Through this evaluative signal alone, the agent progressively learns to discriminate between desirable and undesirable actions and to associate system states with appropriate control decisions.

For this reason, the reward function must encode, in a quantitative form, all the behaviors that the agent is expected to acquire. Moreover, it is not sufficient to simply specify the desired behaviors: the reward must also reflect their relative importance. By appropriately prioritizing different objectives, the agent can learn which requirements are fundamental and which are secondary or preferential. The structure and weighting of the reward terms therefore play a decisive role in shaping the final policy learned by the agent.

The reward function designed in this work aims to translate the following high-level training objectives into a numerical optimization problem:

1. **Task completion:** The agent must be capable of completing the assigned task, namely driving the manipulator from an initial configuration to a specified target pose.
2. **Safety:** The generated motion must be safe both for the manipulator and for the surrounding environment, including the human operator. This entails avoiding internal self-collisions, collisions with external objects such as the supporting bench, and collisions with the obstacle representing the operator, while maintaining appropriate safety distances.

These objectives are inherently competing. For instance, overly prioritizing safety may lead the agent to adopt trivial solutions, such as remaining stationary to avoid any risk of collision, whereas excessive emphasis on task completion may result in aggressive motions characterized by high joint velocities and poor smoothness. Consequently, the reward design must carefully balance these objectives to guide the learning process towards a meaningful solution.

To address the multi-objective nature of the problem, the reward function is constructed as a weighted sum of multiple terms, each associated with a specific aspect of the desired behavior. This decomposition allows individual objectives to be expressed explicitly and facilitates both tuning and interpretability of the learning process.

For clarity, the reward components are grouped into two categories:

- **Task-related terms**, which encourage progress towards and successful completion of the target-reaching task.
- **Safety-related terms**, which penalize collisions and unsafe proximity to internal or external entities.

In the following subsections, each group of reward terms is described in detail, together with the rationale behind its formulation and weighting.

The task-related components of the reward function are designed to explicitly guide the agent towards successful completion of the reaching task, while preserving sufficient flexibility to explore alternative solutions in the presence of obstacles and kinematic constraints.

Task-related reward terms

To limit the exploration space and stabilize the learning process, the task is formulated with respect to a predefined reference trajectory. Instead of allowing unconstrained exploration over the entire workspace, the agent is encouraged to follow a specific path, such that at each time step a corresponding reference point along the trajectory is defined. The reference path lies on a plane parallel to the base plane and has a parabolic shape, while the associated velocity profile is deliberately chosen to be non-trivial and time-varying. This design choice ensures that the agent is required to track a trajectory characterized by both spatial and temporal variability, rather than a simple constant-velocity motion.

Let d_i denote the Euclidean distance between the current TCP position and the reference point on the trajectory that the TCP is expected to reach at the current time step. This distance provides a direct measure of how well the agent is adhering to the desired path and timing.
The task-related reward associated with path tracking is defined as:

$$R = \begin{cases} a \log(b \cdot d_i + 1), & \text{if } d_i > 0.005 \text{ m} \\ +5, & \text{if } d_i < 0.005 \text{ m} \end{cases} \tag{40}$$

where a and b are positive scaling parameters.
This logarithmic formulation was chosen to provide a strong reward gradient in the desired direction, particularly when the TCP is close to the reference trajectory. For small values of d_i, the reward varies rapidly, producing a significant learning signal that clearly indicates suboptimal tracking behaviour and encourages the agent to reduce the distance to the reference point. As d_i increases, the growth of the penalty term progressively attenuates, preventing excessively large reward magnitudes that could destabilize the training process.

In particular, the transition from low to moderate values of d_i (e.g., from 0.1 m to 1 m) results in a reduced rate of increase of the penalty. This design choice avoids overly punitive rewards for large deviations, which could otherwise dominate the optimisation process and hinder the effective integration of additional reward components related to safety and motion quality.

A constant positive reward is assigned when the TCP lies within a small tolerance of the reference point ($d_i \leq 0.005$ m). This term explicitly reinforces correct tracking behaviour and signals to the agent that the desired trajectory and timing have been successfully achieved.

Overall, this task-related reward formulation constrains exploration to the neighborhood of a predefined trajectory, promotes convergence towards a consistent path-following behavior, and provides well-shaped gradients that support stable learning, even when additional reward components are introduced to handle more complex safety-related scenarios.

Safety-related reward terms

Safety-related terms constitute a fundamental component of the reward function, reflecting the central role of safety in collaborative robotic applications.

These terms are designed to discourage robot configurations and motions that could result in damage to the manipulator or unsafe interactions with the surrounding environment, particularly with the human operator.

A first safety-related component penalizes internal self-collisions of the manipulator. This term enforces awareness of the robot's own spatial extent and prevents configurations in which different links intersect. When a self-collision is detected, a large negative reward is assigned and the episode is immediately terminated. This mechanism ensures that self-collision avoidance becomes an intrinsic property of the learned policy, rather than an external constraint enforced only at execution time. The same termination logic and penalty structure are applied in the event of collisions between the manipulator and the supporting bench to which it is mounted.

Beyond binary collision events, an additional safety-related reward component is introduced to regulate the robot behavior in situations where the presence of the human body influences the motion trajectory. The primary objective of this term is to prevent robot motions that could be harmful to the operator, even in the absence of physical contact. To achieve this, robot safety is enforced through velocity modulation as a function of the distance between the end-effector and the operator.

A safety zone is defined around the operator, within which the robot is required to respect a reference safety velocity v_{sic}, chosen to be sufficiently low to avoid hazardous interactions. Figure 53 provides a visual representation of the safety-related reward landscape, illustrating how the reward varies as a function of the TCP position in the vicinity of the operator. The figure highlights the non-traversable zone close to the operator, the surrounding safety region, and the smooth spatial decay of the reward outside these areas.

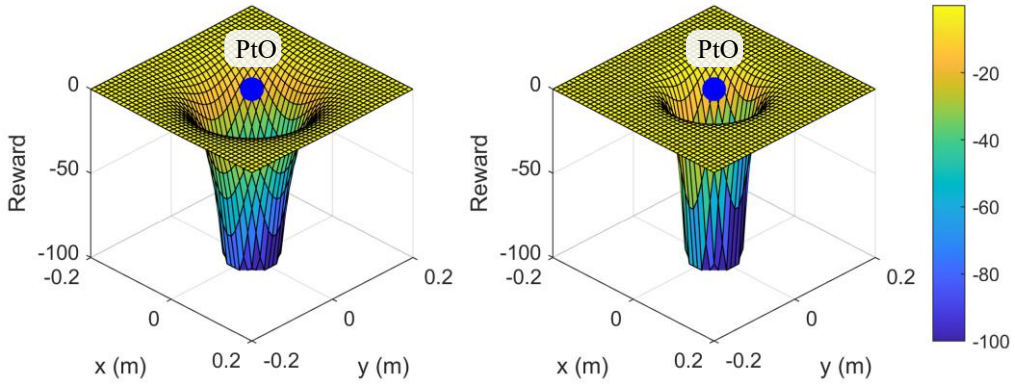

Figure 53 Safety-related reward surface R_{safety} around the obstacle for (left) TCP velocity $v_{\text{TCP}} = 2\ \text{m/s}$ and (right) $v_{\text{TCP}} = 0.3\ \text{m/s}$. The reward exhibits a strong negative region in the proximity of the obstacle position PtO, corresponding to the non-traversable zone, and a progressively decreasing penalty with increasing distance.

The reward component associated with this mechanism depends on two key quantities:

- d_o: the instantaneous Cartesian distance between the TCP and the obstacle;

- g: a velocity-related factor defined as the absolute difference between the robot TCP velocity (v_{TCP}) and the reference safety velocity (v_{sic}),

$$g = \|v_{\mathrm{sic}} - v_{\mathrm{TCP}}\| \tag{41}$$

When the manipulator operates outside the safety zone, this reward component has negligible influence, allowing the robot to move at higher velocities without penalization. As the robot approaches the operator, the penalty becomes increasingly significant, enforcing a progressive reduction of the TCP velocity. This design avoids abrupt changes in behavior while ensuring a smooth transition towards safe operation.

The radius of the safety zone is not fixed but dynamically adapts according to the velocity factor g. Specifically, a variable safety distance $\mathrm{r_{safe}}$ is defined as:

$$\mathrm{r_{safe}} = 0.1 + (g - 1) \cdot 0.02 \tag{42}$$

which enlarges the safety zone at higher velocities and reduces it when the robot is already moving close to the safety velocity. This adaptive mechanism ensures that velocity regulation begins earlier when the robot is fast, preventing sudden decelerations, while allowing more compact operation at low speeds.

The reward term R_{safety} is defined as:

$$R_{safety} = \begin{cases} -k \cdot (p \cdot e^{q \cdot d_o}), & \text{if } d_o > 0.05 \text{ m}, \\ -100, & \text{if } d_o \leq 0.05 \text{ m}, \end{cases} \tag{43}$$

where k is a weighting factor controlling the severity of the penalty. The threshold $d_o = 0.05$ m defines a non-traversable zone, introduced to strictly prevent robot motion in dangerously close proximity to the operator.

The coefficients q and p are defined as:

$$q = \begin{cases} \dfrac{\log(10)}{(0.05 - \mathrm{r_{safe}})}, & g > 0.05 \ \dfrac{\mathrm{m}}{\mathrm{s}}, \\ 0, & g \leq 0.05 \ \dfrac{\mathrm{m}}{\mathrm{s}}, \end{cases}$$
$$p = \begin{cases} \dfrac{1}{e^{\,q\,\mathrm{r_{safe}}}}, & g > 0.05 \ \dfrac{\mathrm{m}}{\mathrm{s}}, \\ -w, & g \leq 0.05 \ \dfrac{\mathrm{m}}{\mathrm{s}}, \end{cases} \tag{44}$$

where w is a positive reward value (in this work $w = 0.5$) assigned when the robot operates within the safety zone while respecting the reference safety velocity. The parameters a and b are chosen such that the reward function exhibits a consistent exponential profile between the boundary of the non-traversable zone and the variable safety radius $\mathrm{r_{safe}}$. This formulation guarantees that the penalty smoothly increases as the robot approaches the operator while moving too fast, and that it vanishes, or becomes positive, when the robot behaves

safely. If the manipulator operates inside the safety zone while maintaining a TCP velocity within the tolerance of v_{sic}, the penalty is annulled and a positive reward is provided, explicitly reinforcing correct and safe behaviors. This positive reinforcement remains valid as long as the robot stays outside the non-traversable zone, beyond which a constant high penalty is enforced to strongly discourage unsafe configurations.

Overall, this safety-related reward formulation enables the agent to learn distance-aware and velocity-aware behaviors, allowing close human–robot interaction while maintaining safety through smooth and adaptive motion regulation.

It is important to emphasize once again that the reward terms are defined in Cartesian space, whereas the actions generated by the agent are expressed in joint space. This design choice intentionally delegates the learning of inverse kinematic relationships to the agent, allowing the reward to be formulated at a higher level using physically meaningful and easily interpretable quantities. As a result, the agent is encouraged to implicitly learn the mapping between joint-level commands and Cartesian task outcomes, rather than relying on explicit analytical inverse kinematics solutions.

Reset function

The reset function is a fundamental component of the reinforcement learning environment, as it defines how new training episodes are initialised and therefore strongly influences the exploration process and the stability of learning. Its primary role is to reinitialise the environment to a valid and well-defined state whenever an episode terminates, either due to task completion, collision, or timeout.

From a conceptual perspective, the reset function determines the initial state distribution from which the agent starts interacting with the environment. By appropriately designing this distribution, it is possible to expose the agent to a wide variety of initial conditions, thereby improving robustness and generalisation, or, conversely, to restrict the initial states to a limited region of the state space in order to stabilise early training.

In the considered framework, the reset function is responsible for reinitialising all the relevant elements of the environment, including the robot configuration and the task-related variables. In particular, at the beginning of each episode:

- the joint configuration of the UR3 manipulator is set to a valid initial configuration, consistent with joint limits and free of internal or external collisions;
- the target position is initialised according to the task definition, either fixed or sampled within a predefined region of the workspace;
- the position of the obstacle representing the human operator is reset to a predefined location;
- the episode step counter is reset to zero.

The reset function also ensures that the initial state satisfies all safety and feasibility constraints, preventing the agent from starting an episode in an invalid or already penalised configuration. This aspect is particularly important in safety-critical applications, where learning from infeasible initial conditions could bias the policy or slow down convergence.

Beyond simple reinitialization, the reset mechanism plays a key role in controlling the difficulty of the learning problem.

Overall, the reset function defines the boundary conditions of the reinforcement learning problem and directly affects exploration, convergence speed, and policy generalisation. For these reasons, its design must be carefully aligned with the objectives of the training process and the characteristics of the robotic task under consideration.

Training Procedure

The training phase represents the process through which the agent learns to solve the assigned task by interacting with the environment and receiving reward signals associated with the actions it performs. Through repeated trial-and-error interactions, the agent progressively refines its policy in order to maximize the cumulative reward defined by the reward function.

To improve training stability and facilitate convergence, a progressive training strategy is adopted, as illustrated in Figure 54. Rather than imposing all task constraints from the beginning, the agent is guided towards the final solution by learning one behavior at a time. This approach is motivated by the complexity of the task and by the presence of multiple, potentially competing objectives, such as task completion, safety, and motion regularity. Introducing all constraints simultaneously may lead to unstable learning dynamics or convergence to poor local optima.

In the progressive training framework, task complexity is gradually increased by:

- incrementally activating the terms of the reward function, and
- progressively enlarging the variability of the initial conditions defined by the reset function.

In this work, the training process is structured into the following stages:

- **Training stage 1 – Path execution:**

  The agent is trained to execute a predefined trajectory between fixed initial and final positions. The initial and final Cartesian points are kept constant, while the end-effector orientation is unconstrained. At this stage, collision checking is disabled. The objective is to allow the agent to learn basic path-following and velocity profiling behaviors in a simplified setting.

- **Training stage 2 – Orientation constraint:**

A specific target orientation is imposed on the end-effector. This step increases task complexity by coupling position and orientation control, while still operating in the absence of collision constraints.

- **Training stage 3 – Internal and external collision avoidance:**

  Collision checking is activated for both internal self-collisions of the manipulator and collisions with the supporting bench. Corresponding safety-related reward terms are introduced, forcing the agent to account for its own spatial occupancy and workspace constraints.

- **Training stage 4 – Human-aware behavior:**

  An obstacle representing the human operator is introduced into the environment. In addition to collision avoidance, velocity regulation within the safety zone is enforced through the dedicated safety-related reward components. This final stage corresponds to the complete training scenario described in the previous sections.

For each of these training stages, multiple training runs are performed by varying key training hyperparameters, including the maximum number of training episodes, the episode reward stop criteria, and the magnitude of the Gaussian exploration noise. These parameters govern the balance between exploration and exploitation during training, determining how aggressively the agent explores new behaviors versus refining actions that already yield high rewards.

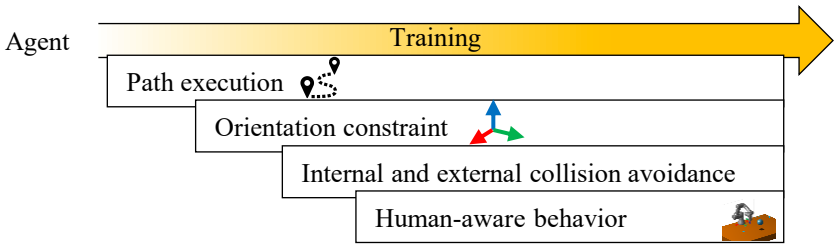

Figure 54 Progressive training strategy adopted for the joint-control reinforcement learning environment. Task complexity is gradually increased by sequentially introducing additional constraints, starting from basic path execution and extending to orientation control, internal and external collision avoidance, and human-aware behavior, while previously learned behaviors are preserved throughout the training process.

The reinforcement learning agent is selected among the algorithms available in the MATLAB Reinforcement Learning Toolbox. In this work, a *Twin Delayed Deep Deterministic Policy Gradient* (TD3) agent is adopted. TD3 is an off-policy actor–critic algorithm specifically designed for continuous action spaces and is known for its robustness to overestimation bias and its improved training stability compared to earlier deterministic policy gradient methods. These characteristics make TD3 particularly well suited for joint-level control of robotic manipulators and for learning in environments characterized by complex reward structures and safety constraints.

Results

This section presents the results obtained in the joint-control reinforcement learning environment, following the progressive training strategy described in the previous sections. Each subsection corresponds to a specific training stage, reflecting the gradual increase in task complexity and constraints. Each training session is terminated either when the maximum number of episodes is reached or when the average episode reward exceeds a predefined threshold, indicating that the agent has successfully learned the behavior specified by the corresponding reward function.

Path execution

The first training stage aims at obtaining an agent capable of accurately following a predefined Cartesian trajectory without imposing any constraint on the end-effector orientation. A total of eight training sessions were required to achieve satisfactory performance. The initial sessions were primarily devoted to exploration of the state–action space, whereas the later sessions progressively refined the policy through reduced exploration noise and parameter tuning. The training parameters adopted in each session are reported in Table 15. As shown in the table, in the final training sessions the agent converges before reaching the maximum number of episodes, providing evidence that the behaviors encoded in the reward function have been successfully learned.

Table 15 Training hyperparameters used during the progressive learning of the joint-control agent. The table reports the maximum number of episodes, Gaussian exploration noise standard deviation and bounds, and the target policy smoothing parameters adopted at each training stage.

Max Episode	Gaussian Noise std	Gaussian Upper/Lower Limit	Target Smooth policy. std	Target Upper/Lower Limit
10000/10000	0.8	±4	0.7	±0.9
4000/4000	0.8	±3	0.7	±0.9
2000/2000	0.7	±2	0.7	±0.9
1000/1000	0.6	±2	0.7	±0.9
896/1000	0.5	±1	0.7	±0.9
156/500	0.3	±0.5	0.7	±0.9
41/500	0.2	±0.5	0.7	±0.9

Quantitative results are summarized in Table 16, which reports the position and velocity tracking errors evaluated after the 3rd, 6th, and final training sessions. The results show a clear and consistent reduction of both error metrics as training progresses. At convergence, the agent achieves position tracking errors on the order of 1 mm and velocity errors of approximately 0.04 m/s.

Table 16 Position and velocity tracking errors at different stages of training in the joint-control environment, illustrating the progressive improvement of the agent performance from early to final training stages.

Training stage	Position error (m)	Velocity error (m/s)
Early stage	0.051	0.328
Middle stage	0.0021	0.094
Final stage	0.0012	0.040

Figure 55 and Figure 57 compare the position and velocity profiles generated by the agent with the corresponding reference profiles. The position trajectories closely overlap with the target path, indicating accurate spatial tracking, as further confirmed by the position error plot reported in Figure 56. The velocity profiles exhibit small oscillations around the desired values; however, these oscillations remain bounded and within an acceptable range for the intended application. It can be observed that, at the end of the velocity profile, the agent does not reach zero velocity but retains a residual speed. This behavior is a consequence of the episode termination condition, which is triggered upon reaching the target position, leaving no additional time for the agent to stabilize the velocity to zero.

Overall, the obtained performance is considered satisfactory, and the agent is deemed capable of reliably following the desired path under the given conditions.

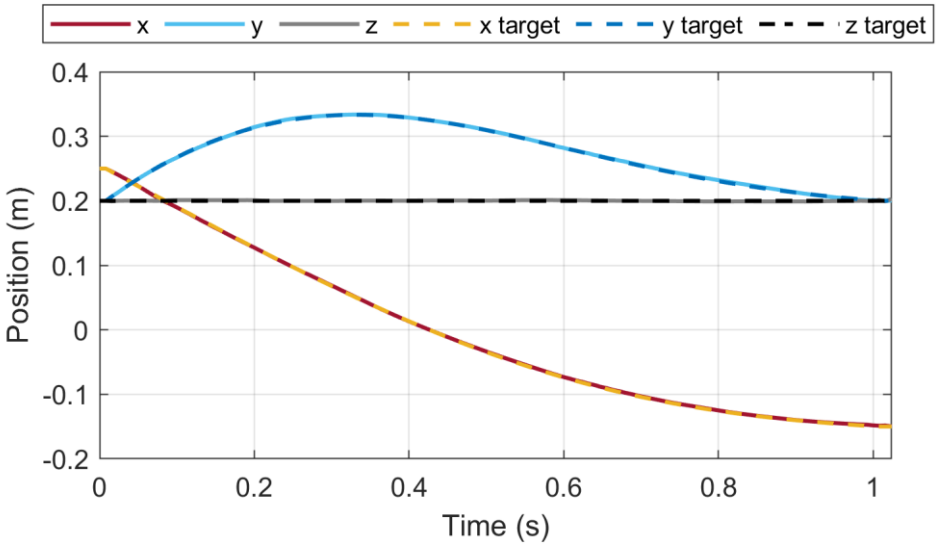

Figure 55 Comparison between the Cartesian trajectory components generated by the agent and the target trajectory over time.

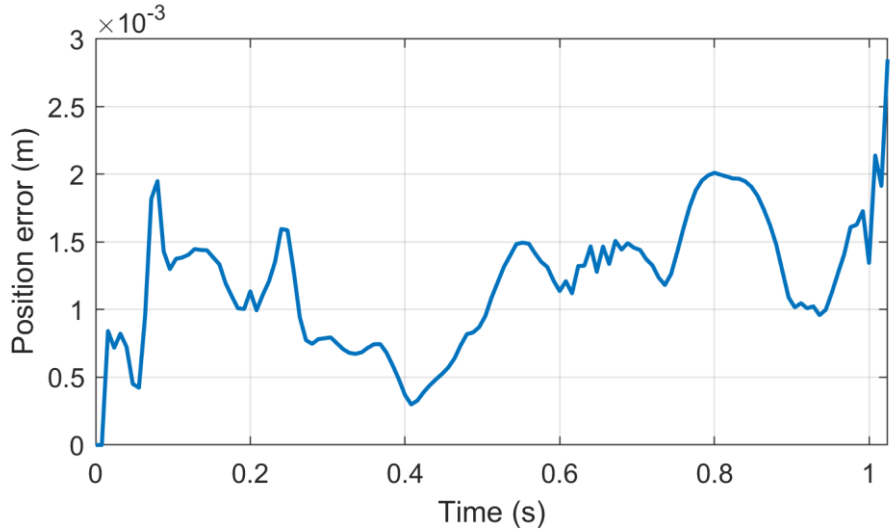

Figure 56 Instantaneous position error between the target trajectory and the trajectory executed by the agent

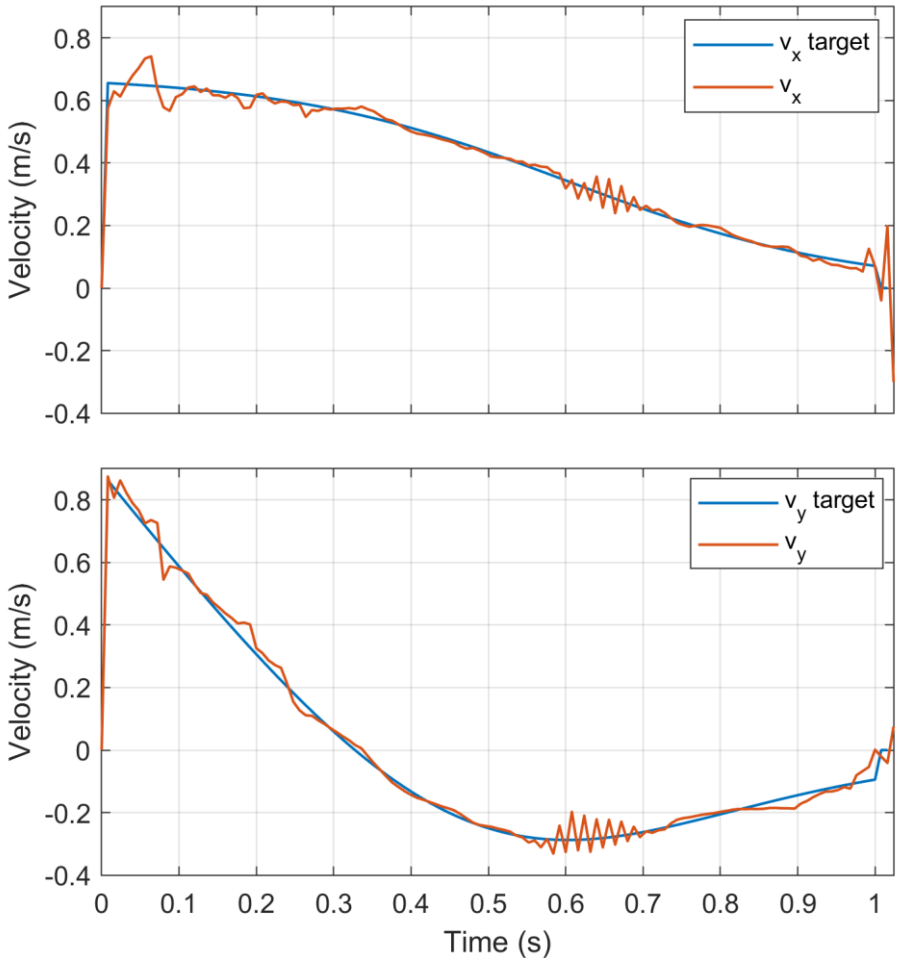

Figure 57 Comparison between the Cartesian velocity components generated by the agent and the corresponding target velocity profiles over time.

Orientation Constraint

After introducing orientation constraints, the objective was to train an agent capable of maintaining a constant end-effector orientation along the entire trajectory. Figure 58 top illustrates the variation of the *tool0* reference frame orientation after the initial path-following training, highlighting the absence of explicit orientation control.

To address this limitation, six additional training sessions were performed, as reported in Table 17. At the end of this tuning phase, the agent exhibits a significantly more stable orientation behavior, as shown in Figure 58 bottom. Moreover, a slight improvement in both position and velocity tracking accuracy is observed. This refinement can be attributed to the additional training iterations, which further regularized the policy and improved trajectory execution consistency. The final tracking errors are reported in Table 18.

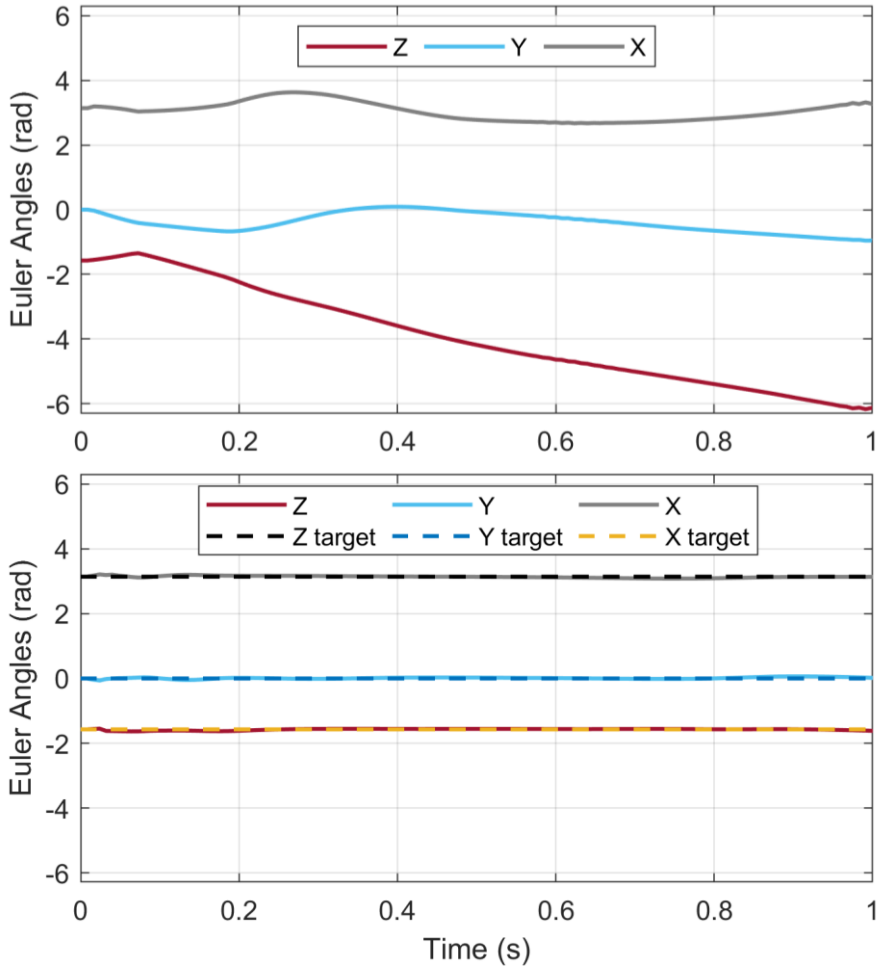

Figure 58 TCP orientation expressed in Euler angles over time, ZYX sequence. (Top) Orientation evolution before orientation-constrained training, showing significant variations along the trajectory. (Bottom) Orientation behavior after training, demonstrating stable tracking of the target orientation throughout the motion

Table 17 Training hyperparameters. The table reports the maximum number of episodes, Gaussian exploration noise standard deviation and bounds, and the target policy smoothing parameters adopted at each training stage.

Max Episode	Gaussian Noise std	Gaussian Upper/Lower Limit	Target Smooth policy. std	Target Upper/Lower Limit
5000/5000	0.7	±3	0.7	±0.9
2000/2000	0.6	±2	0.7	±0.9
1000/1000	0.6	±2	0.7	±0.9
1000/1000	0.5	±1.5	0.7	±0.9
500/500	0.4	±1	0.7	±0.9
452/500	0.3	±1	0.7	±0.9

Table 18 Position, velocity and orientation tracking errors at different training stage.

Training stage	Position error (m)	Velocity error (m/s)	Orientation error (rad)
Middel stage	0.001	0.039	0.074
Final stage	0.0009	0.030	0.047

Collision checking

In the subsequent training stage, collision checking is enabled for both internal self-collisions and collisions with the supporting bench. Under the specific test conditions considered in this work, the introduction of collision constraints does not significantly alter the learned policy. This behavior is expected, as the agent operates in close proximity to the predefined reference trajectory, which is collision-free by construction.

Despite the limited impact observed at this stage, collision checking represents a necessary prerequisite for more complex scenarios in which the robot is required to deviate from the nominal trajectory, such as in obstacle avoidance tasks. In these cases, collision constraints play a crucial role in shaping the learned behavior.

Human-aware behavior

At this stage of the training process, the agent has successfully learned to execute the predefined trajectory while maintaining the desired end-effector orientation. The final training stage aims at extending this capability by introducing the presence of an obstacle representing a human operator, requiring the agent to modify the previously learned motion in order to ensure safe interaction.

To explicitly enforce this behavior, the obstacle is positioned directly along the reference trajectory that the agent has already learned to follow. In this way, the agent is forced to deviate from the nominal path in order to avoid the obstacle, rather than simply executing the previously acquired trajectory. During the training sessions of this stage, the obstacle position is kept fixed, allowing the agent to focus on learning the fundamental avoidance behavior before any further generalization is introduced.

The introduction of the obstacle requires the agent to simultaneously satisfy path-following, orientation maintenance, collision avoidance, and velocity regulation constraints. Four training sessions were performed; however, these did not result in successful convergence of the policy.

Specifically, the agent is able to follow the nominal trajectory up to the neighborhood of the obstacle, but fails to re-converge towards the reference path after performing an avoidance maneuver, as illustrated in Figure 59. The figure shows the trajectory executed by the agent after four training sessions, including the starting point (*PtI*), the target point (*PtF*), the reference path corresponding to the trajectory previously learned by the agent, and the obstacle to be avoided. Unlike the previous training stages, the learning process does not exhibit clear signs of improvement even during the early phases of training.

This behavior suggests that the combined task complexity exceeds the learning capability of the joint-control formulation under the considered observation and reward design. The simultaneous requirements of Cartesian path tracking, orientation control, joint-level action generation, and safety-aware behavior significantly increase the dimensionality and nonlinearity of the learning problem. Moreover, the limited generalization observed even under highly constrained initial and target configurations indicates intrinsic limitations of this approach in practical scenarios.

For these reasons, further training was not pursued. Instead, the insights gained from these results motivated the development of an alternative reinforcement learning formulation, characterized by a reduced problem complexity and improved generalization properties. This alternative approach, based on Cartesian path generation, is presented in the following section.

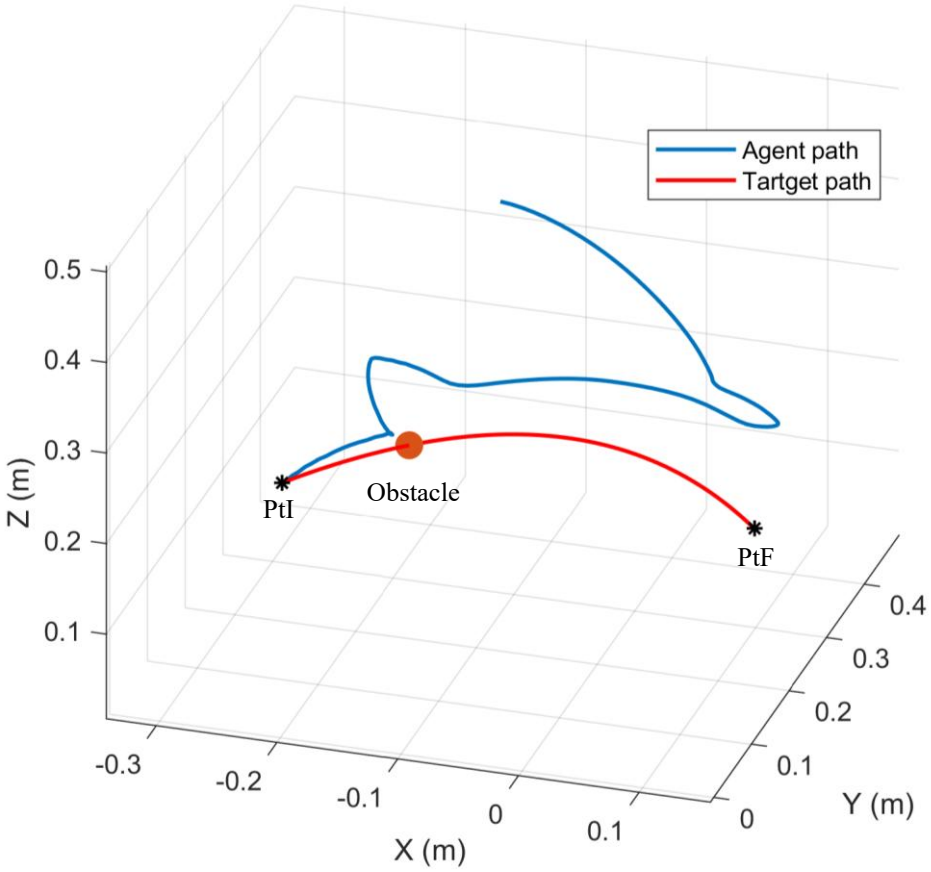

Figure 59 Three-dimensional comparison between the trajectory generated by the agent and the target path in the presence of an obstacle. While the agent successfully deviates from the nominal path to avoid the obstacle, it fails to re-converge towards the target trajectory after the avoidance maneuver.

5.4 Path planner environment

In this second reinforcement learning formulation, a different approach is adopted in order to simplify the constraints imposed on the agent while preserving the core objective of generating safe, collision-free robot motions. Although the overall physical setup remains identical to the joint-level control environment, comprising the UR3 manipulator, the supporting bench, and an obstacle representing the human operator, the role assigned to the agent is substantially modified.

In this formulation, the agent is no longer required to control the robot at each time step. Instead, it is tasked with generating a safe Cartesian trajectory connecting a given initial point to a target point. Once the trajectory is generated, classical robotic methodologies are employed to compute the inverse kinematics and to associate a temporal parametrization to the path. As a consequence, the explicit coupling between Cartesian space and joint space, previously delegated to the agent, is removed. Moreover, task completion is implicitly guaranteed, since the generated path always connects the initial and final positions by construction.

This simplification allows the agent to focus exclusively on safety-related aspects, such as obstacle avoidance and collision-free path generation, without being burdened by low-level control or inverse kinematics learning. As will be shown in the following sections, this reduced problem complexity leads to improved training performance and enhanced generalization capabilities, particularly with respect to variations in the initial and target positions and in the location of the obstacle.

Observation space and action spaces

The observation space provided to the agent in this environment is continuous and consists of nine scalar quantities, defined as follows:

Initial point position (3 values): Cartesian coordinates of the starting point of the path.

Final point position (3 values): Cartesian coordinates of the target point.

Obstacle position (3 values): Cartesian coordinates of the obstacle representing the human operator.

All quantities are expressed in the same reference frame and fully define the geometric configuration of the task.

The action space is continuous and composed of four scalar parameters, which are used to parametrically generate the path. Unlike the joint-control environment, these actions do not directly command the robot, but instead define the shape of the Cartesian path.

Path parametrization

The path is generated using a parametric formulation defined over a normalized scalar variable $x_{\text{norm}} \in [0,1]$. Three action parameters are used to shape the path along the normalized axis through a polynomial expression:

$$f_{\text{path}}(x_{\text{norm}}) = x_{\text{norm}}(1 - x_{\text{norm}})(a + b\, x_{\text{norm}} + c\, x_{\text{norm}}^2) \tag{45}$$

where a, b, and c are the first three components of the action vector. This formulation guarantees zero deviation at the beginning and at the end of the trajectory, while allowing flexible shaping in between.

The resulting path is initially defined in a local reference frame and then mapped into the global Cartesian space by:

1. scaling the path according to the Euclidean distance between the initial and final points,
2. rotating it to align with the direction connecting the two points,
3. translating it to the initial point position.

In addition to path shaping, the fourth action parameter sets the plane where the path belongs by a rotation angle around its main direction vector. The roll angle is bounded through a hyperbolic tangent function to ensure smoothness and numerical stability:

$$\theta = \theta_{\max} \tanh(d) \tag{46}$$

where d is the fourth action component and $\theta_{\max}$ is the maximum allowed roll angle.

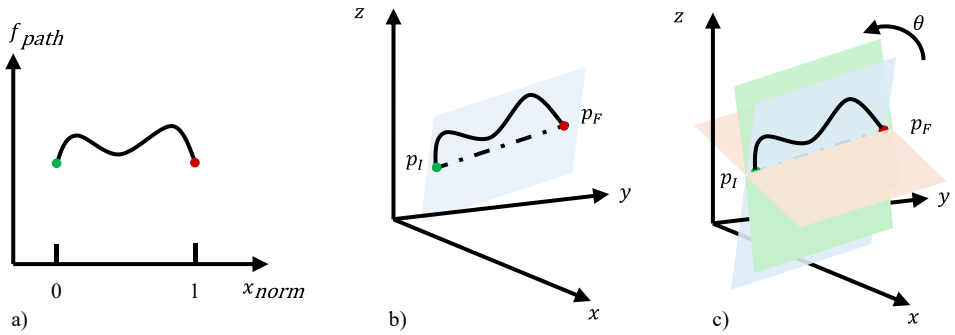

Figure 60 Path construction process. a) The path is generated in a normalized space using the shaping parameters produced by the reinforcement learning agent. b) The path is geometrically mapped through scaling and rotation to align it with the direction connecting the initial and final points $p_I \rightarrow p_F$. c) The plane containing the path is subsequently rotated about the main trajectory axis according to the agent-controlled parameter θ

This parametrization enables the agent to generate a wide variety of smooth three-dimensional trajectories using a compact action representation. An illustration of the path constraction is reported in Figure 60.

Environment dynamics and collision checking

Once the Cartesian path is generated, the environment evaluates its feasibility by performing collision checks along the entire path. Both internal self-collisions of the manipulator and collisions with external entities (bench and obstacle) are considered. The orientation of the robot is constrained to be constant and with the z-axes perpendicular to the bench surfaces.

If a collision is detected at any point along the trajectory, the corresponding safety-related penalties are applied and the episode is terminated. Conversely, collision-free trajectories are rewarded according to the safety and motion-quality criteria defined in the reward function.

Through this formulation, the environment acts as a validator of the generated paths, providing the agent with feedback exclusively based on global path properties, rather than on instantaneous control actions.

Reward function

In the second reinforcement learning environment, the reward formulation is intentionally simplified in order to reflect the reduced complexity of the learning problem. Since the agent is tasked exclusively with generating a collision-free Cartesian path between a given initial and final point, the reward function is composed of only two components: a collision-related penalty and a successful termination reward.

As already discussed in the previous formulation, three different types of collisions are considered and must be avoided: internal self-collisions between the robot links, collisions between the manipulator and the supporting bench, and collisions with the obstacle representing the human operator. Depending on the path generated by the agent, any of these collision events may occur during trajectory execution. In this simplified reward structure, no distinction is made between the different collision types, which are all treated uniformly in order to reduce the dimensionality of the learning problem and allow the agent to focus exclusively on the global safety of the generated trajectory.

Internal self-collisions and collisions with the bench are independent of the operator position and can occur at any stage of the trajectory. For this reason, the collision-related reward is designed to explicitly account for the point along the path at which the collision occurs. Unlike the joint-control environment, where collision penalties are constant and differentiated by collision type, in this formulation the penalty is defined as a function of the remaining portion of the path at the moment of collision. Collisions that occur during the early stages of the trajectory are penalized more heavily than those occurring closer to the end of the path, as illustrated in Figure 61. The progressive penalty structure provides an implicit measure of task advancement, encoding how much of the trajectory has been completed without collisions and guiding the agent towards the generation of increasingly safer paths.

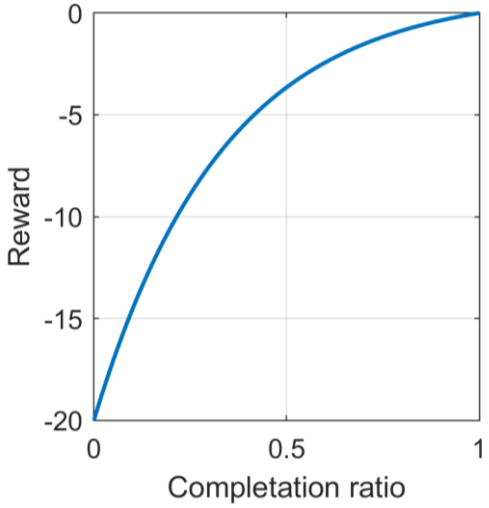

Figure 61 Reward function associated with collision events. Collisions occurring in the early portion of the path generate a more severe negative reward than collisions occurring in the final stage of the trajectory.

The successful termination reward is instead defined as a constant positive value, assigned when the generated trajectory is entirely collision-free from the initial point to the target. Since task completion is guaranteed by construction in this environment, this terminal reward serves solely to reinforce the generation of safe paths and to clearly distinguish successful episodes from failed ones.

Reset function

In the second reinforcement learning environment, the reset function plays a central role in promoting the generalization capabilities of the agent and in enabling the learning of a robust policy across different system configurations. Within the common experimental setup described in the previous section, this translates into training the agent to generate collision-free paths for multiple combinations of initial positions, target positions, and obstacle locations.

To achieve this objective, the workspace is explicitly divided into distinct regions, as illustrated in Figure 62, which shows the three-dimensional representation of the shared setup with the corresponding zones highlighted. The outer regions, depicted in green, define the volumes within which both the initial and final points of the trajectory can be sampled. The central region corresponds instead to the transit area, within which the obstacle representing the human operator is placed. The spatial boundaries of these regions are reported in Table 19, where the corresponding extents are defined with respect to the *base_link* reference frame.

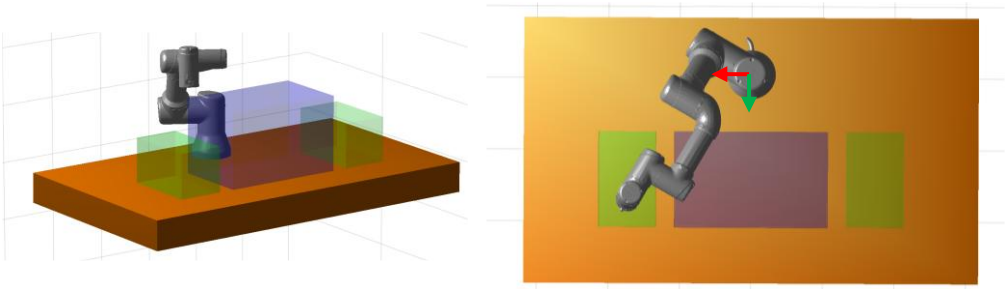

Figure 62 Schematic partitioning of the workspace used by the reset function in the path-generation environment. The central region represents the transit area where the obstacle (human operator) is sampled, while the lateral regions correspond to loading and unloading zones from which the initial and final points of the trajectory are generated.

During the reset phase, the initial and final points are independently sampled within the loading and unloading regions, while the obstacle position is sampled within the central transit region. All regions are defined as three-dimensional volumes, allowing variability along all spatial directions and resulting in a wide range of possible task configurations. This sampling strategy ensures a high degree of variability in both the geometric arrangement of the task and the relative position of the obstacle. Unlike the joint-control formulation, where start and target positions were fixed to stabilize learning, the path-generation environment explicitly exploits the reset function to expose the agent to a wide range of spatial configurations.

Table 19 Spatial extent of the left, central, and right workspace regions used in the reset function. The limits of each region are expressed in meters with respect to the base_link reference frame.

	Left area limits	Central area limits	Right area limits
x (m)	-0.4 to -0.25	-0.2 to 0.2	0.25 to 0.4
y (m)	0.15 to 0.40	0.15 to 0.40	0.15 to 0.40
z (m)	0.005 to 0.15	0.005 to 0.25	0.005 to 0.15

The adopted workspace partitioning reflects a realistic industrial scenario, in which distinct areas can be identified for material loading and unloading, while a central region is shared by the robot and the human operator during collaborative or transitional operations. As a result, the agent is trained under conditions that closely resemble practical application contexts, rather than under artificially constrained or purely random configurations.

In addition to sampling task-related variables, the reset function verifies that the initial manipulator configuration is free from internal and external collisions. This check ensures that each episode starts from a valid and physically feasible state, preventing the agent from being penalized for invalid initial states that are unrelated to the path-generation task.

Overall, the reset function defines the distribution of training scenarios and constitutes a key element in enabling the agent to learn general, collision-free path-generation strategies that are applicable across a wide range of workspace configurations.

Training procedure

In the path-generation environment, a progressive training strategy is no longer required, as the learning problem is characterized by a single, well-defined objective: the generation of collision-free path. By construction, task completion is guaranteed, and the agent is not required to learn low-level control or inverse kinematics. As a result, the training process can focus exclusively on safety-related aspects.

Two distinct agents are trained in this environment. The first agent considers the manipulator without an end-effector tool, while the second agent includes a gripper mounted on the robot flange. The presence of the gripper introduces an additional collision body, increasing the likelihood of collisions and therefore the complexity of the safety constraints. Despite this increased difficulty, the gripper configuration is essential to obtain an agent that can be deployed in realistic application scenarios, where tool attachments are unavoidable.

Although a progressive increase in task complexity is not employed, multiple training sessions are still performed for each agent configuration. Early training runs are designed to promote exploration by using higher levels of stochasticity in the action selection, whereas later training runs focus on fine-tuning the learned policy by reducing exploration noise and refining the agent's behavior. This two-phase strategy enables the agent to first discover feasible regions of the solution space and subsequently converge towards more robust and reliable path-generation policies.

As in the joint-control environment, the reinforcement learning agent is implemented using the TD3 algorithm.

Results

As in the joint-control environment, multiple training sessions were performed to train the agent in the path-generation environment. The training hyperparameters adopted in these experiments are reported in Table 20. Unlike the joint-control formulation, the performance of the agent in this environment cannot be meaningfully quantified in terms of position or velocity tracking errors, since the agent does not directly control the robot motion over time. Instead, performance is evaluated based on the agent's ability to generate collision-free path. For this reason, the primary evaluation metric adopted is the success rate of the path planning process. An episode is considered successful if the agent generates a trajectory that moves the robot from the initial condition to the final one without any collisions, including internal self-collisions between robot links, collisions with the obstacle representing the human operator, or collisions with the supporting bench. The success rate therefore represents the ratio between the number of successful, collision-free trajectories and the total number of evaluated trials.

Table 20 Training hyperparameters. The table reports the maximum number of episodes, Gaussian exploration noise standard deviation and bounds, and the target policy smoothing parameters adopted at each training stage for both the environment, with and without the gripper.

Max Episode	Gaussian Noise std	Gaussian Upper/Lower Limit	Target Smooth policy. std	Target Upper/Lower Limit
2000/2000	0.6	(+/-) Inf	0.5	(+/-) 0.5
2000/2000	0.6	(+/-) Inf	0.4	(+/-) 0.5
1000/1000	0.5	(+/-) Inf	0.3	(+/-) 0.5
1000/1000	0.4	(+/-) Inf	0.2	(+/-) 0.5
500/500	0.3	(+/-) Inf	0.15	(+/-) 0.5
500/500	0.3	(+/-) Inf	0.1	(+/-) 0.5

Table 21 reports the success rates obtained at different training stages for both the agent trained without a gripper and the agent trained with an attached gripper. The results show high and stable success rate, confirming the agent's ability to learn robust collision-free planning strategies. As expected, the presence of the gripper slightly reduces the success rate due to the increased collision geometry and tighter safety constraints. Nevertheless, the agent trained with the gripper maintains a high level of performance, demonstrating the robustness of the learned policy and its suitability for realistic application scenarios. In addition to quantitative evaluation, Figure 63 illustrates a representative example of a trajectory generated by the trained agent. The figure highlights the ability of the planner to produce smooth Cartesian paths that successfully avoid the obstacle while connecting the randomly sampled initial and final points.

Table 21 Success rates of collision-free path generation at different training stages for the path-generation environment, comparing agent performance with and without the gripper attached.

Training stage	No gripper Successful rate	With gripper Successful rate
Early stage	87%	82%
Middle stage	92%	90%
Final stage	96%	94%

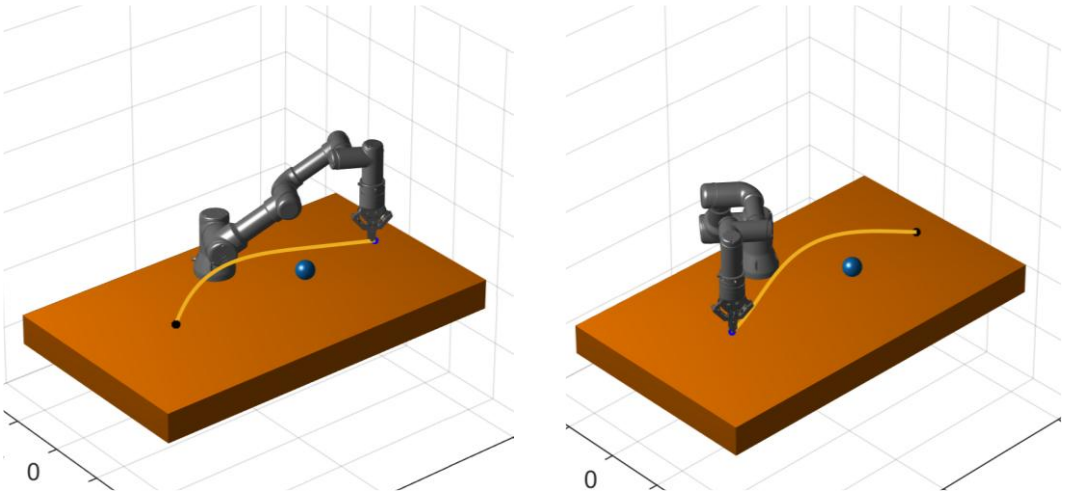

Figure 63 Examples of a collision-free Cartesian path generated by the reinforcement learning agent. The gold curve represents the path planned by the RL policy, the blue sphere denotes the obstacle representing the human operator, and the robot is shown in its initial configuration.

Comparison with traditional motion planning approaches

To better contextualize the proposed reinforcement learning-based path planner, it is necessary to compare it with established motion planning methods reported in the literature, in order to clarify its distinctive characteristics, strengths, and limitations. For this purpose, this section briefly reviews the main families of traditional motion planning techniques, emphasizing their underlying principles and relevant performance indicators. In particular, traditional motion planning methods are broadly classified into sampling-based planners, optimization-based planners, artificial potential field methods, and model-predictive approaches.

Sampling-based planners are among the most widely adopted methods for robotic manipulators, particularly in high-dimensional configuration spaces. These algorithms explore the robot configuration space by generating random or pseudo-random samples and connecting them through feasible local motions. Representative examples include the PRM, RRT, RRT-Connect, RRT*, KPIECE, BKPIECE and related variants [103–105]. Their main advantage is their ability to solve complex planning problems without requiring an explicit analytical representation of the free space. Instead, they rely on a robot kinematic model and a collision checker to determine whether sampled configurations and connecting

motions are feasible. This makes them suitable for articulated manipulators operating in cluttered environments.

Among these methods, RRT and RRT-Connect are commonly used for single-query planning because they rapidly expand trees toward unexplored regions of the configuration space. RRT-Connect further improves the exploration process by growing two trees, one from the initial configuration and one from the goal configuration, and attempting to connect them. RRT* extends the basic RRT formulation by introducing an asymptotic optimality property, although this improvement is generally obtained at the cost of increased computational time. Other planners, such as KPIECE and BKPIECE, exploit projections of the configuration space to guide exploration and improve performance in constrained or high-dimensional problems.

The main strength of sampling-based planners is their relatively high success rate in finding collision-free paths, especially when sufficient planning time is available. However, the paths generated by these algorithms are often geometrically irregular, non-smooth, and not directly suitable for execution without post-processing. For this reason, smoothing or time-parameterization procedures are commonly applied after path generation. Moreover, sampling-based planners are generally designed for planning in a static representation of the environment. When obstacles move, the planner must be repeatedly invoked, which may increase computational cost and limit responsiveness in highly dynamic human–robot interaction scenarios [104].

A second important class of methods is represented by optimization-based motion planners. Instead of constructing a path through random sampling, these approaches formulate motion planning as the minimization of a cost function defined over a complete trajectory [106]. The cost function typically includes terms related to obstacle avoidance, trajectory smoothness, path length, joint limits, velocity or acceleration constraints, and, in some cases, dynamic feasibility. Representative examples include CHOMP, STOMP, TrajOpt, and Gaussian-process-based motion planning methods such as GPMP and GPMP2.

CHOMP formulates trajectory generation as a covariant gradient-based optimization problem [107]. It starts from an initial trajectory and iteratively deforms it to reduce obstacle-related costs while preserving smoothness. Its main advantage is the ability to generate smooth trajectories by explicitly optimizing a functional defined over the path. However, because it is a local optimization method, its performance depends strongly on the quality of the initial trajectory and it may converge to local minima, especially in cluttered environments or narrow passages.

STOMP addresses some of these limitations by using a stochastic trajectory optimization strategy [108]. Instead of relying on analytical gradients, it samples noisy trajectory variations and updates the nominal trajectory according to their associated costs. This allows STOMP to optimize non-differentiible or complex cost functions, including obstacle avoidance, smoothness, torque, or energy-related terms. Nevertheless, the sampling process may increase computational

time, and the method still requires an initial trajectory and iterative optimization for each planning query.

TrajOpt is another optimization-based planner that formulates motion planning as a sequential convex optimization problem [109]. It approximates non-convex collision-avoidance constraints through convex formulations and iteratively solves a sequence of optimization problems. One of its relevant features is the possibility of performing continuous-time collision checking, reducing the risk of collisions occurring between discrete waypoints. This is particularly important for robotic manipulation, where a trajectory that is collision-free at sampled configurations may still collide with obstacles during interpolation. TrajOpt can generate high-quality trajectories efficiently, but, similarly to other local optimization methods, it depends on initialization and does not provide global optimality guarantees.

Gaussian-process-based motion planners, such as GPMP and GPMP2, formulate motion planning as probabilistic inference. The trajectory is represented as a continuous-time function with a Gaussian-process prior, and obstacle avoidance, smoothness, and goal constraints are represented as factors in a graph. This formulation enables efficient trajectory optimization and allows the computation of smooth continuous-time trajectories. Incremental variants, such as iGPMP2, are particularly relevant for replanning, since they can update a previously computed solution when the environment changes. These methods are therefore well suited for discussing online adaptation, although they still require an explicit model, collision information, and iterative numerical optimization [110].

Artificial Potential Field methods represent a simpler and more reactive family of motion planning strategies. In these approaches, the robot is treated as a point or articulated structure subject to virtual forces: attractive forces pull the robot toward the goal, while repulsive forces push it away from obstacles. The resulting motion is obtained by following the negative gradient of the artificial potential function [111]. The main advantage of these methods is their low computational cost, which makes them attractive for real-time obstacle avoidance and local reactive control. However, artificial potential field methods suffer of some limitations. The most critical issue is the possible convergence to local minima, where the attractive and repulsive forces cancel each other before the robot reaches the goal. In addition, the generated paths may exhibit oscillations near obstacles or in narrow passages, and the method does not inherently guarantee global feasibility. For manipulators, the design of potential functions in joint space or Cartesian space can also become non-trivial, especially when multiple links, self-collisions, and human-aware safety distances must be considered. Despite these limitations, potential-field-based approaches remain relevant as local reactive layers or as components of hybrid motion planning architectures [112].

Model predictive control (MPC) and receding-horizon approaches provide another class of model-based strategies, particularly relevant when dynamic constraints and moving obstacles must be considered. In these methods, the

planner repeatedly solves a finite-horizon optimization problem using the current state of the robot and the environment. Only the first part of the optimized control sequence is applied, and the optimization is then repeated at the next time step using updated sensory information. This receding-horizon structure enables online adaptation to changes in the workspace and is therefore well suited for dynamic environments [113].

In robotic manipulation, model predictive approaches can explicitly incorporate kinematic and dynamic constraints, joint limits, velocity limits, acceleration limits, collision-avoidance constraints, and task objectives. When combined with control barrier functions, they can also provide a formal mechanism to enforce safety constraints, such as maintaining a minimum distance from obstacles or human operators. Their main drawback is the computational burden associated with solving constrained optimization problems online. The feasibility of these methods therefore depends on the complexity of the robot model, the prediction horizon, the number of constraints, and the required control frequency.

From the perspective of collaborative robotics, these traditional approaches offer complementary strengths. Sampling-based methods provide robust path feasibility in complex configuration spaces; optimization-based methods improve trajectory quality and smoothness; potential-field methods offer fast local reactivity; and MPC-based approaches enable online adaptation under explicit kinematic or dynamic constraints. At the same time, each family presents limitations when applied to human–robot collaboration. Sampling-based methods may require repeated replanning in dynamic environments; optimization-based methods may be sensitive to initialization and computationally demanding; potential-field methods may suffer from local minima; and MPC-based methods require accurate models and efficient online solvers.

These characteristics provide the basis for positioning the reinforcement learning path-generation method proposed in this work. The objective of the comparison is not to establish a direct benchmark against all traditional planners, since reported results in the literature are obtained on different robots, workspaces, obstacle configurations, and hardware platforms. Rather, the aim is to contextualize the proposed method with respect to established motion planning strategies using commonly adopted quantitative indicators, primarily success rate and trajectory generation time. These metrics allow the proposed planner to be discussed in relation to classical model-based approaches, highlighting both its potential advantages in terms of fast inference after training and its limitations regarding formal guarantees and online replanning.

To contextualize the proposed method, a literature-based comparison was carried out by identifying studies in which robotic manipulator motion planners were evaluated under conditions comparable to those considered in this work. Among the analyzed contributions, the review by Liu et al. [112] is particularly relevant, as it includes a benchmarking study in which different motion planning algorithms are implemented and tested in a manipulation scenario involving pick-and-place operations and an obstacle within the robot workspace. The study

compares planners belonging to different methodological families, including sampling-based, artificial-potential-field-based, optimization-based, and learning-based approaches, and reports quantitative performance indicators such as success rate and planning time.

The results reported in that study are used here as a reference for positioning the proposed reinforcement learning-based path-generation method with respect to established motion planning strategies. It should be noted, however, that these results are not directly superimposable on those obtained in this thesis. The robotic platform, workspace geometry, obstacle configuration, planner implementation, hardware, and task constraints differ from those adopted in the present simulations. Therefore, the comparison should not be interpreted as a direct benchmark, but rather as a literature-based contextual analysis aimed at highlighting the relative advantages and limitations of the proposed approach.

In the proposed framework, path generation and trajectory generation are treated as two distinct stages. First, the learned planner generates a collision-free Cartesian path. Then, a subsequent post-processing stage maps this path into the robot joint space and assigns a temporal law to obtain an executable trajectory. This distinction is important when comparing the proposed method with literature results, since the reported computation time includes the complete implemented pipeline and not only the inference of the learned policy.

To evaluate the proposed algorithm, 1000 pick-and-place planning queries were generated. The initial point, final point, and obstacle position were randomized using the same procedure adopted during the training phase. Two main performance indicators were computed: the success rate and the trajectory generation time. A planning query was considered successful when the algorithm generated a trajectory satisfying two conditions: absence of collision between the robot and the obstacle, and absence of self-collision.

All computation times reported for the proposed method were measured on a laptop equipped with a 12th Gen Intel(R) Core(TM) i7-12650H processor at 2.30 GHz, 16 GB RAM, and an NVIDIA GeForce RTX 3050 Laptop GPU with 6 GB memory. The GPU was not used during the computation. The tests were performed in MATLAB 2025b under Windows 11.

In this analysis, the reported trajectory generation time refers to the complete planning pipeline implemented in this work. It includes the generation of the Cartesian path, its mapping into the robot joint space, and the assignment of a specific temporal law to obtain an executable robot trajectory. The proposed planner achieved an average trajectory generation time of 12 ms and a success rate of 95%, consistent with the performance observed during the final stages of training. The inference time of the neural network was also measured separately, resulting in an average value of 1.2 ms. This result indicates that most of the computational burden is associated with the post-processing stage, particularly with the conversion of the Cartesian path into a joint-space trajectory, rather than with the execution of the learned policy itself.

Table 22 Comparison between the proposed reinforcement learning-based path-generation method and motion planning approaches reported in [112]

Method	Type	Time (ms)	Success rate (%)
A∗	Graph search	403	100
RRT	Sampling-based algorithms	45.3	99.8
Bias-goal RRT		3.83	98.2
RRT-connect		1.02	100
RRT ∗		593	98.3
PRM		1410	100
BIT ∗		377	99.8
APF	APF	0.667	100
TrajOpt	Optimization-based algorithms	371	98.8
CHOMP		316	99.7
STOMP		4030	83
DDPG	RL-based algorithms	14.6	99.90
TD3		14.7	100.00
SAC		19.8	99.90
Proposed		12.6	95.00

The results reported in Table 22 show that the proposed method achieves a favourable trade-off between trajectory generation time and planning success. The average generation time is 0.0126 s, which is considerably lower than the times reported for graph-search, most sampling-based planners, and optimization-based approaches such as TrajOpt, CHOMP, and STOMP. In particular, the proposed method is approximately one order of magnitude faster than CHOMP and TrajOpt, and more than two orders of magnitude faster than STOMP. This difference is mainly related to the fact that optimization-based planners require the solution of an iterative optimization problem for each planning query, whereas the proposed method exploits a policy learned offline and therefore shifts most of the computational burden to the training phase.

It should also be noted that some classical planners, such as APF, RRT-Connect, and bias-goal RRT, report lower planning times in the benchmark study. However, these methods differ substantially in terms of trajectory representation, post-processing requirements, and execution quality. Moreover, the time reported for the proposed method includes the complete conversion of the generated Cartesian path into an executable joint-space trajectory, whereas the meaning of

planning time may vary depending on the planner and implementation considered in the reference study.

Compared with the reinforcement-learning-based planners reported by Liu et al., the proposed method shows a trajectory generation time of the same order of magnitude and slightly lower than those reported for DDPG, TD3, and SAC. It is important to note that the time reported for the proposed method does not refer only to the neural network inference time, but rather to the complete implemented planning pipeline, including Cartesian path generation, mapping into the robot joint space, and temporal parametrization of the motion.

The success rate of the proposed method is 95%, which confirms the ability of the planner to generate feasible collision-free trajectories in most of the tested pick-and-place queries. This value is lower than those reported for several planners in the benchmark study, including some sampling-based, optimization-based, and reinforcement-learning-based methods. However, this difference must be interpreted with caution, since the results were obtained using different robotic platforms, workspace geometries, obstacle configurations, implementation frameworks, and feasibility constraints. In the present work, the planner was evaluated in a UR3-based collaborative scenario in which the obstacle represents the human operator, and the generated Cartesian path was required to be converted into an executable joint-space trajectory.

Path roughness and smoothness were not included in the quantitative comparison, although they could be computed for the trajectories generated by the proposed method. This choice is motivated by the fact that these indicators strongly depend on the trajectory representation, interpolation strategy, sampling frequency, derivative approximation, and temporal parametrization adopted by each planner. Since the benchmark study by Liu et al. does not provide all the implementation details required to reproduce these metrics under identical conditions, a direct numerical comparison would be only partially meaningful.

Nevertheless, some qualitative considerations can be made. The proposed method generates a continuous Cartesian path by construction, and the temporal law is assigned in a subsequent post-processing stage. This decoupling between geometric path generation and time parametrization provides additional flexibility: once a collision-free path has been obtained, different temporal laws can be selected to satisfy execution constraints and improve the regularity of the resulting motion. In particular, smooth time-scaling functions may be adopted to limit velocity, acceleration, or jerk profiles, thereby reducing trajectory roughness without modifying the geometric path generated by the learned planner. Therefore, while roughness and smoothness are not directly compared with the values reported in the literature, the proposed formulation is compatible with post-processing strategies aimed at improving trajectory regularity.

Overall, the proposed planner can be positioned as a fast learning-based path-generation strategy suitable for applications in which low trajectory generation time is required. Its main advantage lies in the rapid execution of the learned policy after training and in the possibility of producing an executable trajectory within a few milliseconds. However, some limitation has to be addressed. First,

the obstacle position is considered at the time of path generation, but dynamic variations of the obstacle during trajectory execution are not explicitly handled by the proposed planner. Therefore, if the human operator changes position after the path has been generated, this variation must be managed by an additional monitoring, replanning, or safety-supervision layer. In this respect, other model-based approaches, such as receding-horizon, model-predictive, or incremental replanning methods, are intrinsically more suitable for dynamic execution, since they can repeatedly update the planned motion according to the current state of the robot and the environment.

Moreover, although the reward function used during training encodes collision avoidance and feasibility requirements, the learned policy does not provide a formal guarantee that these constraints will always be satisfied, especially outside the training distribution. This differs from some classical model-based or optimization-based methods, where constraints can be explicitly imposed within the planning problem and verified during solution generation.

5.6 Conclusion

This chapter has investigated the application of reinforcement learning techniques to the generation of collision-free motions for a robotic manipulator operating in a collaborative environment. The primary objective was to develop learning-based policies capable of internalizing both kinematic and safety constraints, such that collision avoidance emerges as an intrinsic property of the learned behavior, resembling a form of proprioceptive awareness.

To achieve this objective, two distinct reinforcement learning formulations were explored. The first approach represents the most challenging formulation, as the agent is required to directly control the manipulator at joint level. In this case, the policy must implicitly learn the relationship between Cartesian task objectives and joint-space actuation, effectively internalizing the inverse kinematics of the system while simultaneously satisfying safety constraints. A progressive training strategy was adopted, in which task complexity and reward terms were gradually introduced to guide the learning process.

The results obtained with this formulation show that the agent is capable of learning accurate path-following behaviors and maintaining a desired end-effector orientation when operating in highly constrained scenarios. However, when additional requirements such as obstacle avoidance and dynamic trajectory modification are introduced, the learning process fails to converge towards a satisfactory solution. In particular, the agent is able to reliably deviate from the nominal trajectory to avoid collisions but unable to subsequently re-converge to the desired path. These results indicate that the combined complexity of joint-level control, Cartesian task execution, and safety enforcement exceeds the practical learning capability of the adopted formulation under the considered observation and reward design.

These limitations motivated the development of a second reinforcement learning environment, in which the learning problem is deliberately simplified. In

this formulation, the agent is tasked exclusively with generating a collision-free Cartesian path between an initial and a target point, while classical inverse kinematics and time parametrization techniques are used to execute the resulting trajectory. By decoupling trajectory generation from low-level control, the agent can focus entirely on safety-related aspects, such as obstacle avoidance and collision-free planning.

The results obtained with this second approach demonstrate significantly improved performance and robustness. The learned policies achieve high success rates across a wide range of previously unseen initial and target configurations, both in the presence and absence of an end-effector tool. This confirms that the agent is able to internalize the geometric and safety constraints of the system and to generalize effectively across different workspace configurations. From a practical perspective, this formulation proves to be more suitable for real-world deployment, as it reduces computational complexity and avoids the need for high-frequency policy inference at the robot control rate.

Despite its advantages, the path-generation approach also presents inherent limitations. In particular, the generated path is static with respect to the initial conditions and does not adapt instantaneously to changes in the environment. As a result, dynamic obstacle avoidance and rapid reaction to unforeseen events remain outside the scope of the current formulation.

Future work may address these limitations by combining the strengths of the two approaches investigated in this chapter. One possible direction consists in adopting a multi-agent or hierarchical architecture, in which different agents are responsible for complementary tasks. For instance, a first agent could generate a globally collision-free path based on the current environment configuration, a second agent could control the manipulator to track the planned trajectory, and a third agent could provide real-time corrective actions to locally modify the trajectory in response to dynamic obstacles. Such a decomposition of responsibilities would reduce the complexity faced by each individual agent and could lead to more robust and scalable learning-based control architectures.

Finally, computational considerations must also be taken into account. The joint-control formulation requires policy inference at the robot control frequency (125 Hz in the considered setup), which would necessitate dedicated hardware to ensure real-time feasibility. In contrast, the path-generation approach allows planning to be performed at a lower rate, making it more compatible with existing robotic control infrastructures.

In conclusion, this chapter highlights both the potential and the limitations of reinforcement learning for safe motion generation in collaborative robotics. While fully end-to-end joint-level learning remains a challenging problem, learning-based path generation emerges as a practical and effective solution. The next chapter presents the integration of the proposed path-generation method within a collaborative robotic framework, in which the reinforcement learning agent is tasked with generating collision-free trajectories in the presence of human operators.

Chapter 6

Collaboration Framework

6.1 Overview

This chapter describes the integration of the methods and subsystems presented in the previous chapters into a unified framework capable of real-time operation in a human–robot collaborative scenario. While earlier chapters focused on individual components, namely human motion estimation, abrupt movement detection, and reinforcement learning–based motion planning, the objective of this final chapter is to demonstrate how these components can be combined into a coherent software and hardware system suitable for real deployment. In this sense, the chapter represents the system-level validation of the thesis, showing how perception, safety assessment, and decision making interact to enable effective and safe human–machine collaboration. Figure 64 provides a system-level overview of the proposed framework, highlighting the end-to-end data flow from raw sensory inputs to safety-aware robot execution. The figure explicitly reflects the layered organization adopted throughout the thesis and serves as a reference for the detailed architectural description presented in the following sections.

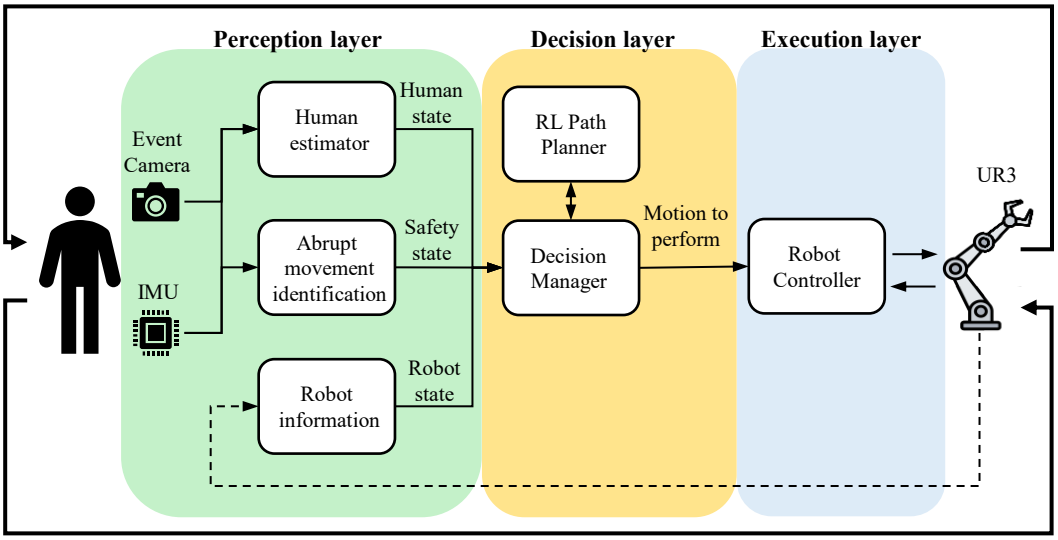

Figure 64 Overview of the integrated framework for real-time human–robot collaboration. The system combines perception, safety assessment, and decision-making layers to generate safety-aware robot motions based on human state estimation, abrupt movement detection, and reinforcement learning–based path planning.

The proposed framework is designed around a collaborative assembly task in which a human operator and a UR3 manipulator share part of the workspace and execute a predefined sequence of actions. The task is structured into discrete phases, some involving independent operation of the human or the robot, and others requiring simultaneous activity within the shared workspace. This phase-based organization reflects typical industrial assembly processes and provides a controlled context in which to evaluate the behavior of the integrated system.

From a hardware perspective, the system comprises three main physical components: the collaborative manipulator, the sensing apparatus, and a computing unit responsible for executing the control and decision logic. The sensing subsystem consists of wearable inertial measurement units and an event-based camera, which together provide complementary information about the operator's motion. The robot provides state feedback and executes motion commands generated by the high-level control framework.

From a software and logical perspective, the system is organized into three conceptual layers: a perception layer, a decision layer, and an execution layer. The inputs to the framework consist of raw inertial measurements, event-camera data, and the robot state. These data streams are processed in the perception layer to estimate the operator's position within the workspace and to assess the safety of the ongoing human motion. The operator position estimate is obtained through the sensor fusion approach described in Chapter 3, combining information from inertial sensors and event-based vision. In parallel, the safety state of the operator motion is evaluated by the abrupt movement detection method presented in Chapter 4.

The outputs of the perception layer, namely the estimated human state and the motion safety assessment, are integrated within the decision layer. This layer is responsible for selecting the appropriate robot behavior given the current phase of the task and the instantaneous human–robot interaction conditions. In

collaborative phases, robot motion is generated using the reinforcement learning–based planner introduced in the previous chapter, which produces collision-aware paths conditioned on the estimated human state. Safety information is continuously monitored and used to modulate robot behavior, enabling actions such as speed reduction, trajectory adaptation, or immediate stop in the presence of elevated risk.

The execution layer translates the selected high-level behavior into robot commands and supervises their execution on the physical manipulator. During execution, the system maintains continuous feedback from both the robot and the safety monitoring modules, ensuring that deviations or hazardous human motions can be detected and handled in real time.

The entire system logic is implemented in ROS 2, which provides the middleware infrastructure required for modularity, synchronization, and real-time data exchange between heterogeneous components. Each logical layer is encoded as one or more ROS 2 nodes, with clearly defined interfaces that reflect the flow of information from sensing to decision making and finally to actuation. This modular organization allows the full pipeline of the thesis, from raw sensor data to collaborative robot motion, to be reconstructed and analyzed at the system level.

The remainder of this chapter details the ROS 2 software architecture, the interaction between the main system nodes, and the real experimental setup used to validate the proposed framework in a collaborative assembly task.

6.2 ROS2 Architecture

This section describes the software architecture adopted to implement the proposed human–robot collaboration framework in ROS 2. The objective is to provide a clear description of how the perception, safety, decision-making, and execution layers introduced in the previous sections are developed as interacting software components. Particular emphasis is placed on the modular organization of the system, the definition of node responsibilities, and the flow of information that enables real-time operation and continuous safety monitoring.

The architecture has been designed to integrate heterogeneous sensing modalities, learning-based planning, and robot control within a unified middleware, while preserving separation of concerns and enabling independent development and testing of individual subsystems.

By detailing the ROS 2 node graph, the main communication interfaces, and the role of each component, this section provides the necessary elements to reconstruct the full system pipeline, from raw sensor acquisition to safety-aware robot motion execution. This architectural description also serves as the foundation for the collaborative assembly task presented in the following section, where the proposed framework is developed and validated.

Design constraints and Architectural choice

The design of the proposed framework is driven by a set of practical constraints that arise from the requirements of close-proximity human–robot collaboration. In such scenarios, the system must operate reliably in real time while integrating heterogeneous sensing and decision-making components, each characterized by different computational and temporal properties.

A first fundamental constraint concerns processing time and latency. To support collaborative operation, the system must process sensory information and generate robot actions within time bounds compatible with human motion. This requirement is particularly critical for safety-related operations, such as abrupt movement detection, where delayed responses may compromise the effectiveness of protective behaviors. Consequently, the architecture must allow low-latency data propagation and avoid unnecessary coupling between computationally intensive modules and time-critical control loops.

A second constraint is related to the asynchronous nature of the sensing modalities. The wearable inertial sensors and the event-based camera operate at different acquisition rates and produce data streams with distinct temporal characteristics. In addition, robot state feedback is provided at yet another frequency. The system must therefore be able to handle asynchronous inputs without enforcing rigid synchronization, while still allowing consistent integration of multi-modal information at the decision level.

Further constraints concern flexibility and scalability. The framework is intended to support iterative development, enabling modifications to individual subsystems, such as replacing a sensing module or updating the planning strategy, without requiring extensive changes to the rest of the system. Moreover, the architecture should allow scalability to additional sensors, alternative planners, or different robotic platforms, preserving the generality of the proposed approach beyond the specific experimental setup considered in this chapter.

These constraints motivate the adoption of a distributed and multimodal architecture, in which the main functional blocks of the system are implemented as independent components that communicate through well-defined interfaces. Such an organization allows computational load to be distributed across processes, supports asynchronous execution at different update rates, and facilitates the isolation of safety-related functionalities from task-level decision making.

Within this context, ROS 2 provides a suitable middleware for implementing the proposed architecture. Its node-based execution model naturally supports distributed systems, while its communication mechanisms and Quality of Service policies allow data exchange to be tailored to the latency and reliability requirements of different information flows. Additionally, the availability of standardized tools for configuration, monitoring, and logging simplifies development and experimental validation.

The ROS 2 distribution adopted in this work is ROS 2 Humble Hawksbill. Although this is not the most recent release, it was selected as the latest long-term

support (LTS) version available at the time of development. This choice was motivated by the need to ensure system stability and reliability during integration and experimental validation. Using an LTS release reduces the likelihood of encountering unresolved issues and minimizes compatibility problems among different software components, which is particularly important for complex, multi-node systems intended for real-time operation.

The simplified ROS 2 node graph shown in Figure 65 provides an overview of the main components of the implemented system and their interactions. The

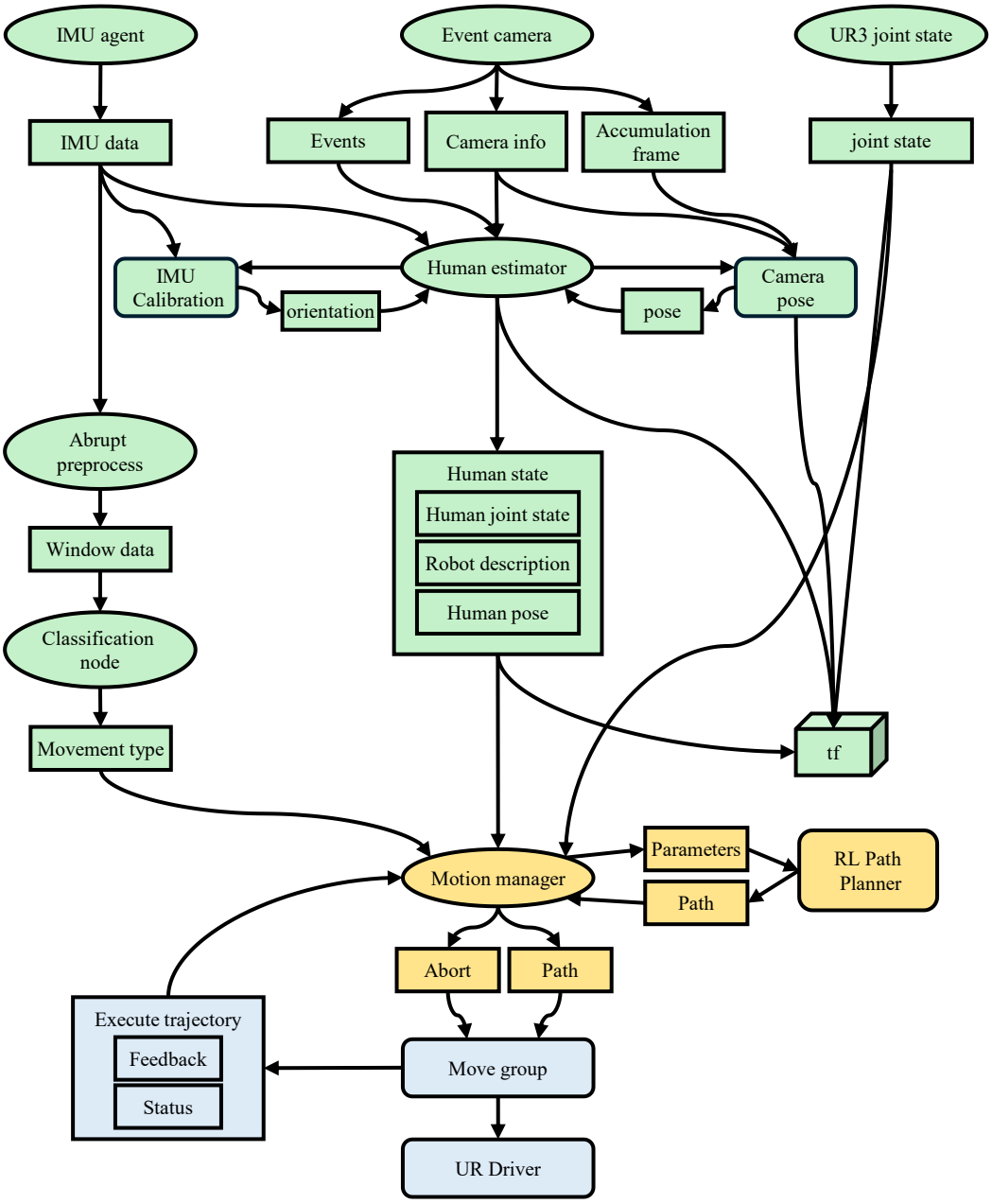

Figure 65 Simplified ROS 2 node graph of the implemented collaboration framework. The diagram illustrates the main ROS 2 nodes and data flows involved in the integration of inertial sensing, event-based vision, safety monitoring, and robot motion execution. Sensor interface nodes provide inertial data, event streams, and robot joint states, which are processed to estimate the human state and to classify the operator's motion safety. These outputs are integrated by the motion manager, which supervises task execution and interacts with the reinforcement learning–based path planner. The selected motion commands are executed through the robot control stack, while execution feedback and robot state are continuously monitored.

diagram is intended to be read from top to bottom, following the direction of the information flow through the architecture. The upper portion of the figure corresponds to the input layer, which is composed of three independent acquisition nodes responsible for collecting sensory data from the environment and the robot.

The first input component is the IMU agent, which acquires data from the wearable inertial sensors and publishes them to the ROS 2 ecosystem. This node provides raw inertial measurements consisting of linear acceleration, angular velocity, and orientation expressed as quaternions. Measurements from two inertial sensors are transmitted together with a timestamp, resulting in a total of 21 values per message. Unlike the other nodes in the system, the IMU agent is not executed on the main computing unit but is implemented directly in the firmware of the microcontroller managing the sensors. This design choice reduces communication overhead and allows the data to be transmitted at the maximum stable acquisition rate supported by the hardware, corresponding to 200 Hz. To satisfy the low-latency requirements of safety-related processing, the associated ROS 2 topic is configured using the predefined *sensor_data* Quality of Service (QoS) profile, which prioritizes fast delivery and minimizes buffering at the expense of communication robustness.

The second input component is the event camera node, which manages communication with the event-based camera and handles the acquisition of visual information. The node is implemented in C++ to ensure efficient interaction with the hardware and publishes two distinct data streams. The first is the *events* topic, containing asynchronous events accumulated over a temporal window of 10 ms, yielding an effective update rate of 100 Hz, consistent with the configuration described in Chapter 3. This topic also adopts the *sensor_data* QoS profile to limit transmission latency. The second stream is the *accumulation frame* topic, which provides an image-like representation generated from the event stream and published at 20 Hz. These frames are primarily used for estimating the operator's position relative to the camera. Given their lower real-time criticality, the default QoS profile is used. In addition, a static node publishes the intrinsic parameters of the camera through the *camera_info* topic, which is required by downstream processing modules.

The third input source is the UR3 joint state publisher, which provides the current state of the robot manipulator. This node is part of the *universal_robots_ros2_driver* package and publishes joint state information at 125 Hz. Together with the inertial and vision-based data, this completes the set of input signals required to represent both the human operator state and the robot configuration.

Once the input data streams are available, the system branches into two main processing pipelines corresponding to the functionalities introduced in the previous chapters: motion safety recognition and operator tracking. These pipelines operate in parallel and provide complementary information to the decision layer. In the following, the motion safety recognition branch is described first.

The motion safety recognition pipeline is implemented through two main nodes: a preprocessing node and a classification node, both operating according to a callback-based execution model. Each time new inertial data are received, the preprocessing node is triggered. At this stage, gravity compensation is applied to the raw acceleration signals as described in Chapter 4, and an internal buffer storing the temporal window to be classified is updated. A new data window is published whenever the number of newly acquired samples matches the step size determined by the selected window overlap. In the experimental configuration adopted in this work, a window length of 0.5 s with a 90% overlap is used, resulting in an effective publication rate of 20 Hz.

The classification node receives the preprocessed data windows and performs inference to determine the current movement type. To prevent the inference time from interfering with other time-critical processes, this functionality is implemented as a dedicated node whose sole responsibility is classification. This design choice improves temporal predictability and ensures that safety monitoring remains responsive even under variable computational load.

The Human State Estimator node implements the operator tracking functionality by fusing inertial and event-based visual information, as described in Chapter 3. Within the overall architecture shown in Figure 65, this node represents the core perception component responsible for providing a real-time estimate of the human operator state to the decision layer.

From a software perspective, the node is structured around multiple callback functions and service clients, reflecting the asynchronous nature of the available sensor data and the need to decouple runtime estimation from calibration procedures. In particular, the node subscribes to two input topics: inertial measurements from the wearable IMUs and event packets from the event camera.

The callback associated with the inertial data stream is executed at the same rate as the IMU acquisition, namely 200 Hz. Its role is deliberately kept lightweight in order to ensure predictable execution at high frequency. For each incoming message, the callback averages the most recent measurements from the two inertial sensors and stores the resulting values in an internal data structure. No state estimation is performed at this stage; instead, the inertial data are continuously updated and made available to the sensor fusion pipeline executed in the event-driven callback.

The event-based callback defines the effective update rate of the operator tracking pipeline. Each time a new packet of events is received, corresponding to an acquisition rate of approximately 100 Hz, the full sensor fusion procedure is executed. In the experimental conditions considered in this work, the number of events contained in each packet is consistently above the minimum threshold required to trigger an update. As a result, the estimation process is effectively executed at every event callback, leading to a stable and regular update rate driven by the event stream.

At each update, the node computes an estimate of the operator’s kinematic state by combining inertial and visual information according to the methodology described in Chapter 3. The resulting estimate includes both a joint-level

representation of the upper limb and the pose of selected reference frames relevant to the collaborative task, with particular emphasis on the terminal point of the forearm. These quantities are published as ROS 2 messages and made available to downstream modules at the same rate as the event acquisition. In addition, the corresponding transformations are broadcast through the TF mechanism to ensure spatial consistency with the robot model and the shared workspace. To ensure kinematic consistency of the estimated human state, the node relies on a kinematic model of the upper limb defined through a URDF description, aligned with the described model in Chapter 3.2. At initialization, the URDF is retrieved from the *human/robot_state_publisher* and parsed to extract the joint hierarchy, joint limits, and fixed transformations required for forward kinematics.

Calibration procedures required by the tracking pipeline are handled separately from the runtime estimation logic. The node includes two service clients that are activated only on demand. One service estimates the static transformation between the event camera and the operator reference frame using a marker-based approach, while the second service computes the orientation offsets of the two inertial sensors under static conditions. Both calibration routines are described in Chapter 3.6 and are executed independently from the main estimation loop, ensuring that calibration does not interfere with real-time operation.

The Motion Manager node represents the central supervisory component of the proposed framework and is responsible for coordinating task execution based on the estimated human state, safety conditions, and available motion generation strategies. Within the system architecture illustrated in Figure 64, this node operates at the decision layer and acts as the integration point where perception outputs and safety assessments are translated into executable robot behaviors.

Unlike perception or planning modules, the Motion Manager does not directly estimate states or generate trajectories. Instead, its primary role is to orchestrate the execution of the collaborative task by selecting the appropriate motion strategy, triggering planning and execution modules, and supervising their outcomes. This design choice enables a clear separation of concerns, allowing perception, safety monitoring, planning, and execution to evolve independently while remaining coherently integrated at the system level.

The node receives three main categories of input information. First, it subscribes to the output of the abrupt movement classification module, which provides a discrete safety state indicating whether the current human motion is considered safe. Second, it continuously retrieves the estimated position of the operator's forearm endpoint through the TF tree, ensuring spatial consistency between the human model and the robot reference frame. Third, it monitors the current robot configuration through joint state feedback. Based on these inputs, the Motion Manager issues commands to planning services, action servers, and low-level robot interfaces, thereby determining the robot's behavior at runtime.

Task execution is governed by a phase-based finite state machine that encodes the structure of the collaborative assembly process. Each state corresponds to a well-defined phase of the task, such as waiting for operator interaction, executing a pick or release motion, performing a transfer in shared space, or completing the

final collaborative approach. Transitions between states are triggered by a combination of conditions, including the completion of robot motions, the operator's presence within predefined regions of interest, and the current safety state.

Safety is treated as a global supervisory constraint within the Motion Manager. The node continuously monitors the safety classification output and enforces conservative behavior whenever unsafe conditions are detected. If a transition from a safe to an unsafe state occurs while a robot motion is being executed, the Motion Manager immediately cancels the active trajectory execution and forces the system into a dedicated safe waiting state. From this state, task execution is suspended until safe conditions are restored. This mechanism ensures that safety-related information can preempt nominal task execution at any time, independently of the planning strategy currently in use.

The Motion Manager supports two distinct categories of robot motion, selected according to the task phase and the level of interaction with the human operator. Deterministic motions are used for well-defined actions such as pick, release, and approach maneuvers. These motions rely on predefined Cartesian waypoints and are planned using classical motion planning services provided by *MoveIt*. Such motions are repeatable, predictable, and confined to regions where human interaction is either absent or tightly controlled.

In contrast, adaptive motions are employed during transfer phases in which the robot operates in close proximity to the human operator. In these phases, the Motion Manager invokes the reinforcement learning–based planner through a dedicated inference service. The planner generates a collision-aware Cartesian path conditioned on the current robot position, the target location, and the estimated position of the operator. By restricting the use of learning-based planning to these specific phases, the framework leverages the adaptability of reinforcement learning only where it provides a clear benefit, while preserving deterministic behavior elsewhere.

All planning and execution requests issued by the Motion Manager are handled asynchronously. Service calls and action requests are non-blocking, and their completion is monitored through callbacks. An internal execution flag is used to prevent overlapping commands and to ensure that only one motion request is active at any given time. Upon completion of a motion, the result is evaluated and the state machine is updated accordingly. Planning failures, execution errors, or invalid responses cause a transition to an error state, from which recovery must be handled externally. This asynchronous execution model improves responsiveness and prevents blocking of the supervisory loop.

The behavior of the Motion Manager is extensively parameterized. Workspace regions, target positions, velocity and acceleration scaling factors, safety thresholds, and the number of task repetitions are all defined through ROS 2 parameters. This design allows the same supervisory logic to be reused across different experimental setups and task configurations with minimal modification. As a result, the Motion Manager provides a flexible and extensible mechanism for

coordinating human–robot collaboration while maintaining a clear and structured decision-making process.

In summary, the Motion Manager node embodies the system-level integration of perception, safety monitoring, and motion generation. By encoding task execution as a phase-based state machine and by explicitly arbitrating between deterministic and adaptive motion strategies under safety constraints, the node closes the loop between sensing and action and enables reliable execution of the collaborative assembly task described in the following section.

The execution layer of the proposed framework is largely delegated to existing and well-established software components. In particular, low-level robot control and state feedback are handled by the *universal_robots_ros2_driver*, while motion planning and trajectory execution rely on the *MoveIt* packages already integrated within this driver.

6.3 System Integration

Hardware Updates

Compared to the systems described in the previous chapters, various hardware modifications were introduced to support the integration of all components into a single real-time collaborative framework. These updates were primarily driven by latency, computational performance, and integration requirements arising from the simultaneous execution of perception, safety monitoring, and motion planning modules.

In the final system, the wearable inertial sensing is performed using ICM-20948 IMUs (TDK InvenSense), managed by an ESP32-S3 (Espressif Systems) microcontroller running firmware based on *micro-ROS*. The event-based visual sensing is provided by a DVXplorer (Inivation) event camera, while all software components are executed on a Dell workstation (Intel Core i7-11850H, 8 core, up to 4,80 GHz, NVIDIA T600, 4 GB, GDDR6) running Ubuntu 22.04 with a real-time kernel.

The ICM-20948 sensors were selected due to their improved performance compared to the MPU6050 devices previously used in the tracking system, as well as their better suitability for real-time operation when compared to the Opal APDM units adopted in the abrupt movement dataset acquisition. In particular, the latency introduced by the latter was not compatible with online safety monitoring.

The ESP32-S3 microcontroller replaces the Arduino Due used in earlier prototypes. This choice was motivated by its more compact form factor and by the availability of integrated wireless connectivity, which enables potential future developments without requiring major architectural changes.

The DVXplorer event camera represents an upgrade with respect to the device employed in Chapter 3. It offers nearly three times higher spatial resolution and communication latency below 1 ms, which is critical for real-time perception. The lack of standard frame output is addressed by generating image-like

representations through event accumulation, as described in the perception pipeline.

The central computing unit was selected to provide sufficient computational resources for the concurrent execution of multiple ROS 2 nodes. The adoption of a real-time Linux kernel is a key element in ensuring predictable scheduling and reliable execution of time-critical processes, particularly those related to robot control and safety supervision.

Experimental set-up

The experimental setup is shown in Figure 66 and represents a practical deployment of the proposed framework in a shared human–robot workspace. The UR3 robot is mounted on a workbench measuring 120 × 70 cm and positioned close to the back edge to maximize the available workspace for collaboration. The event camera is placed to the right of the robot, near the corner of the bench, and mounted on a dedicated support to provide a stable and well-defined view of the workspace.

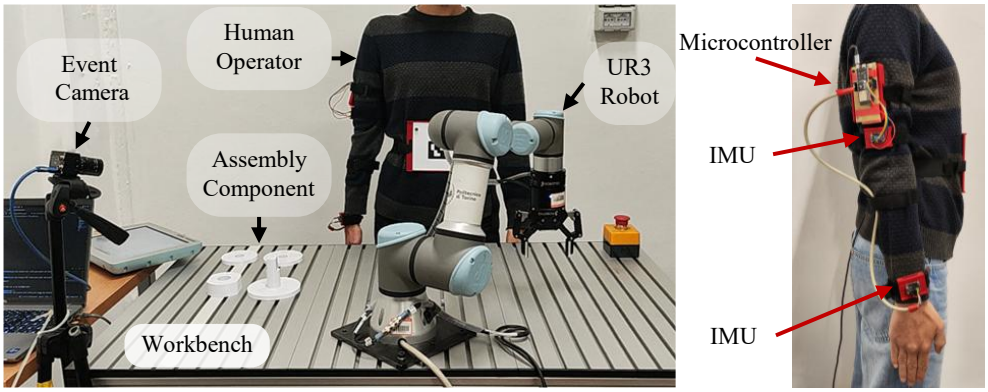

Figure 66 Experimental setup and wearable sensing system. On the left, the collaborative workstation is shown, including the UR3 robot, the event camera, the workbench, and the assembly components together with the human operator. On the right, a detailed view of the wearable IMU-based sensing system mounted on the operator's arm is reported.

Two IMUs are mounted on the operator's right arm using custom 3D-printed supports and elastic bands. An ArUco marker is attached to the operator to enable the initial calibration of the operator's position. Compared to the setup described in Chapter 3, the marker size is reduced to 5 cm per side, reflecting the improved resolution and sensitivity of the event camera.

Communication between the IMUs and the microcontroller is implemented using the SPI protocol. The microcontroller is connected to the central computer via USB, while the event camera communicates through a USB 3.0 interface to guarantee high data throughput and low latency. The robot controller is connected to the computer via Ethernet using a TCP/IP protocol.

Activation procedure

The activation of all system components described in the previous sections is a mandatory step for the correct operation of the framework. Calibration routines are implemented as ROS 2 services and are executed only when required, in order to limit unnecessary computational load during runtime.

First, the operator wears the sensing equipment, and all hardware components are connected to the central computer. The robot is then powered on and initialized. Subsequently, a dedicated ROS 2 launch file is executed, starting all nodes required for system operation. The launch file enforces a predefined temporal sequence, ensuring that each subsystem is fully initialized before dependent components are started. At this stage, sensor data streaming becomes active, and the system logs prompt the user to perform the required calibration procedures.

The operator positions himself in front of the workspace and assumes the calibration posture described in Chapter 3. The IMU calibration service is then triggered. Calibration is performed over a 1 s interval, during which it is verified that the IMU orientations do not vary by more than 1 degree. Next, the operator position is estimated using the ArUco marker detected from accumulated event frames. Multiple accumulated frames are used to improve robustness, and the calibration is accepted only if the estimated position varies by less than 2 cm across consecutive frames.

Once calibration is completed, the system enters its operational state. Operator tracking is active, and a dedicated service is callled to notify the Motion Manager to start task execution. At this point, the system is fully functional and capable of executing the collaborative task.

Several visualization tools are available to support system verification and debugging and are shown in Figure 67. These tools allow the visualization of raw IMU data, accumulated event frames, segmented event clouds associated with the operator’s arm, and a three-dimensional scene including the robot, the human model, the event camera reference frame, and the trajectories generated by the reinforcement learning planner. However, these visualization tools are optional, are not required for standard system operation, and introduce additional computational load.

The spatial relationship between the event camera and the robot is calibrated offline using a procedure similar to that employed for the operator, in which an ArUco marker is rigidly attached to the robot flange.

Data can be acquired and stored using the ROS 2 bag tool, which allows recording the data published on selected topics.

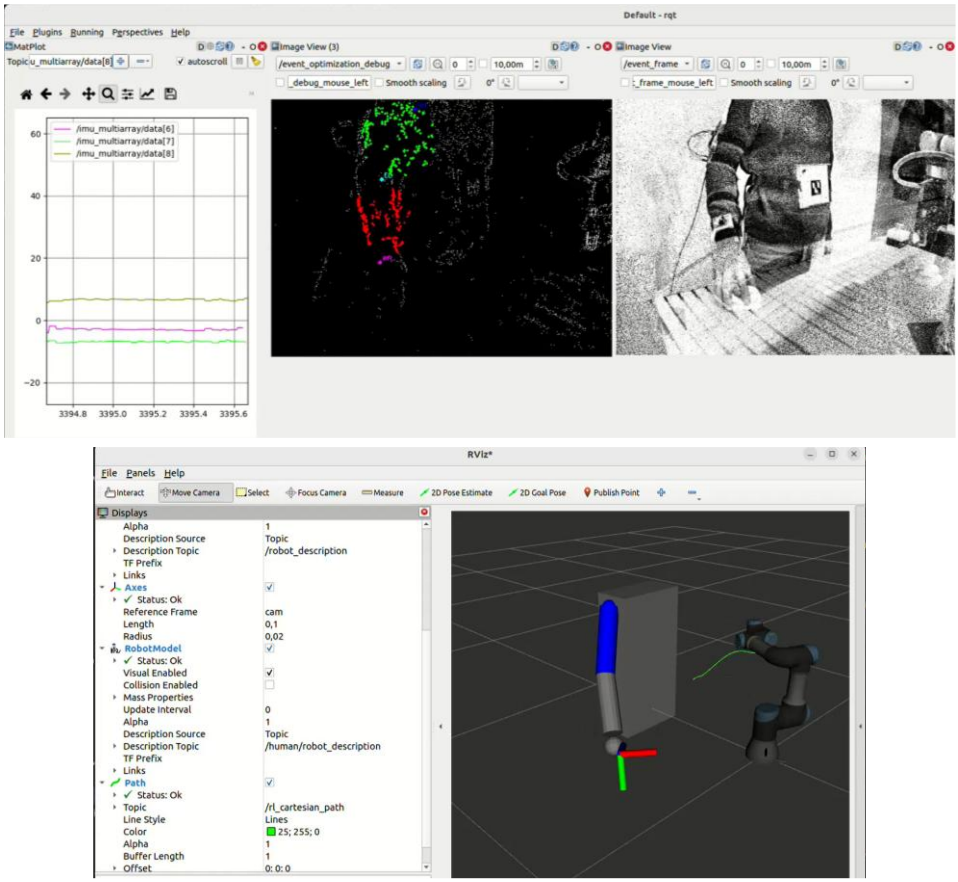

Figure 67 Visualization tools used for system monitoring and debugging. The upper row shows real-time inertial measurements of the operator's forearm, the event-based segmentation of upper arm and forearm, and the image obtained through event accumulation. The lower view presents a three-dimensional visualization of the scene, including the robot, the kinematic model of the operator, the camera reference frame, and the planned trajectories.

Collaboration Task

To demonstrate the practical operation of the proposed framework and its effective integration in a real scenario, a collaborative assembly task was designed and implemented. The task involves the assembly of a component composed of four parts, following the sequence illustrated in Figure 68. The assembly process is intentionally structured to include both independent and collaborative phases, allowing the system to exploit parallelism between the robot and the human operator while maintaining synchronization during actions.

During the task, the robot and the operator work on two separate subassemblies in the same workspace. In particular, the robot is responsible for assembling subassembly ***A***, while the operator independently composes subassembly ***B***. In the final phase, the robot brings subassembly ***A*** into a position favorable to the operator, enabling the manual completion of the final assembly.

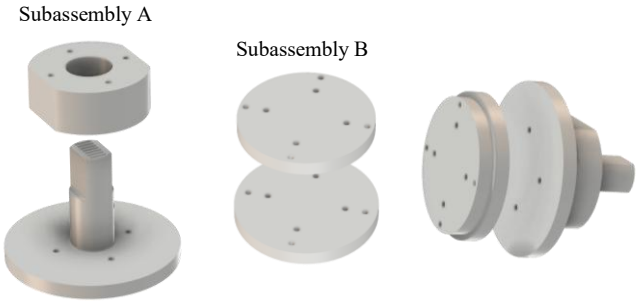

Figure 68 Assembly sequence of the target component.

At a high level, the task execution can be summarized as follows. Initially, all individual components are placed in predefined pick areas within the shared workspace. The robot and the operator sequentially pick the first component and place it in a designated intermediate area. Both agents then return to the pick area to retrieve the second component and proceed with the formation of their respective subassemblies. Once subassembly A is completed, the robot transports it toward the operator to allow the final joining with subassembly B. This description captures the overall logic of the task, while the actual execution is governed by a reactive and event-driven control strategy.

The execution of the collaborative assembly task relies directly on the perception and decision-making components introduced in the previous sections of this chapter. In particular, the activation of task phases is driven by the operator tracking pipeline, which provides a real-time estimate of the operator arm position. This information is used to detect the operator's interaction with predefined spatial regions of interest. The resulting events are then handled by the Motion Manager described in Section 6.2, which supervises the phase-based task execution and selects the appropriate motion strategy for each phase.

This design ensures synchronization between the human and the robot and prevents the two agents from acting as independent, unsynchronized processes. Specifically, two spatial regions of interest are defined in the workspace, corresponding to the pick and place areas. When the operator's hand enters one of

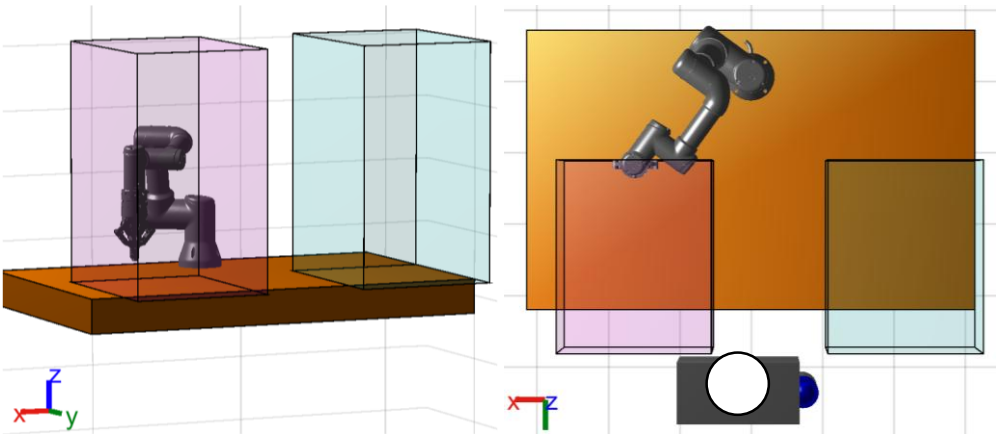

Figure 69 Spatial transition volumes used for task phase activation. (left) General three-dimensional view of the workspace volumes; (right) top view including the operator. The light blue volume corresponds to the pick area, while the purple volume defines the place area. Entry of the operator's hand into these regions triggers the corresponding task phase transitions.

these regions, the associated task phase is activated. The volumes corresponding to these regions are shown in Figure 69.

Figure 70 reports the temporal evolution of the operator's and robot's positions along the three Cartesian axes. Vertical dashed lines indicate the instants at which the operator's motion triggers a transition between task phases. This representation highlights the tight coupling between human motion and robot behavior.

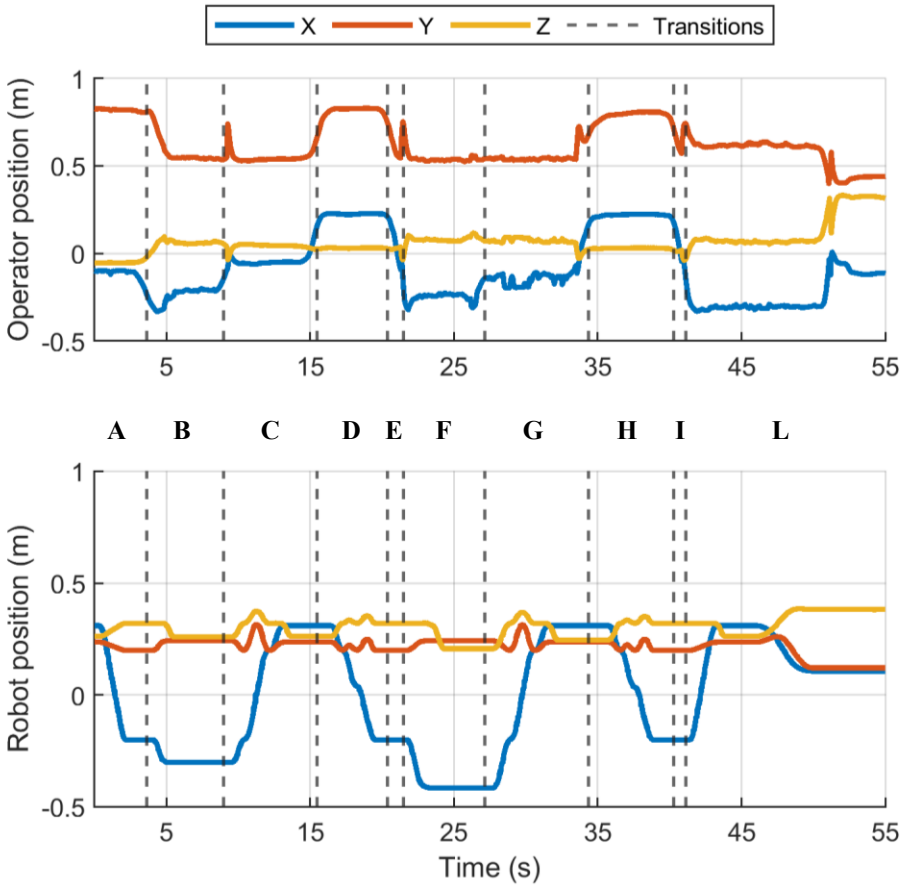

Figure 70 Temporal evolution of operator and robot Cartesian positions during the collaborative assembly task, with the phase sequences.

Figure 71 instead provides a three-dimensional visualization of the robot trajectories executed during the different phases, together with the spatial regions that must be reached by the operator to enable the subsequent transitions and the operator path. The robot is shown in the configuration associated with the starting point of each phase.

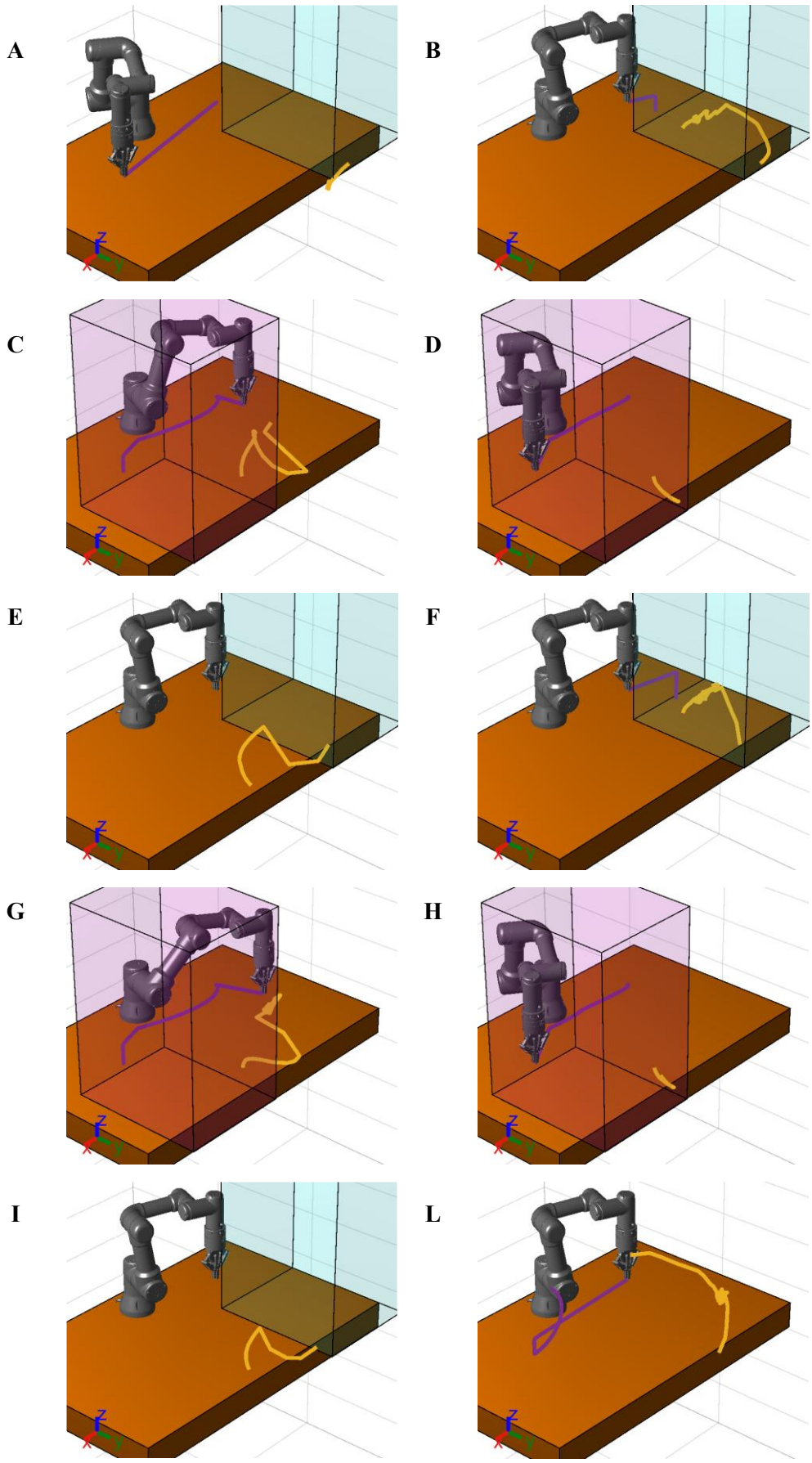

Figure 71 Three-dimensional visualization of the collaborative task execution across different phases. The robot trajectories (purple) and the operator hand motion (yellow) are shown. Colored volumes indicate the spatial zones whose activation triggers transitions between task phases.

The actual execution of the task proceeds as follows. At the beginning, the robot is placed in a generic initial configuration, while the operator assumes a calibration posture required for system initialization. The only constraints imposed on the operator during this phase are remaining within the field of view of the event camera and maintaining a static trunk posture once calibration is completed. After calibration, the robot moves to a waiting position at the boundary of the pick area and remains idle until the operator enters this region (Phase **A**). Once the operator is detected inside the pick area, the robot performs the pick operation and then waits for the operator to leave the region (Phase **B**).

When the operator exits the pick area, the robot plans and executes a transfer motion toward the place area. In this phase, the path is generated using the reinforcement learning–based planner, after which a deterministic place motion is executed. Upon completion, the robot waits until the operator also reaches the place area (Phase **C**). The robot then executes a return motion, again generated through the RL-based planner (Phase **D**), and waits for the operator to return to the pick area (Phase **E**).

The sequence is repeated for the second component. When the operator enters the pick area, the robot performs the second pick operation and waits (Phase **F**). It then moves to the place area and completes the assembly of subassembly A (Phase **G**). When the operator re-enters the place area, the robot leaves the shared workspace and returns to the pick area, while the operator completes subassembly B independently (Phase **H**). Finally, the operator brings subassembly B into the pick area, triggering the last phase of the task (Phase **I**). In this final phase, the robot picks up subassembly A and transports it to a position favorable to the operator, allowing the manual completion of the final assembly (Phase **L**). Figure 72 shows representative frames corresponding to the different task phases, captured at the initial instant of each phase, together with the final frame of the collaborative assembly task (Figure 73), illustrating the real execution context of the scenario.

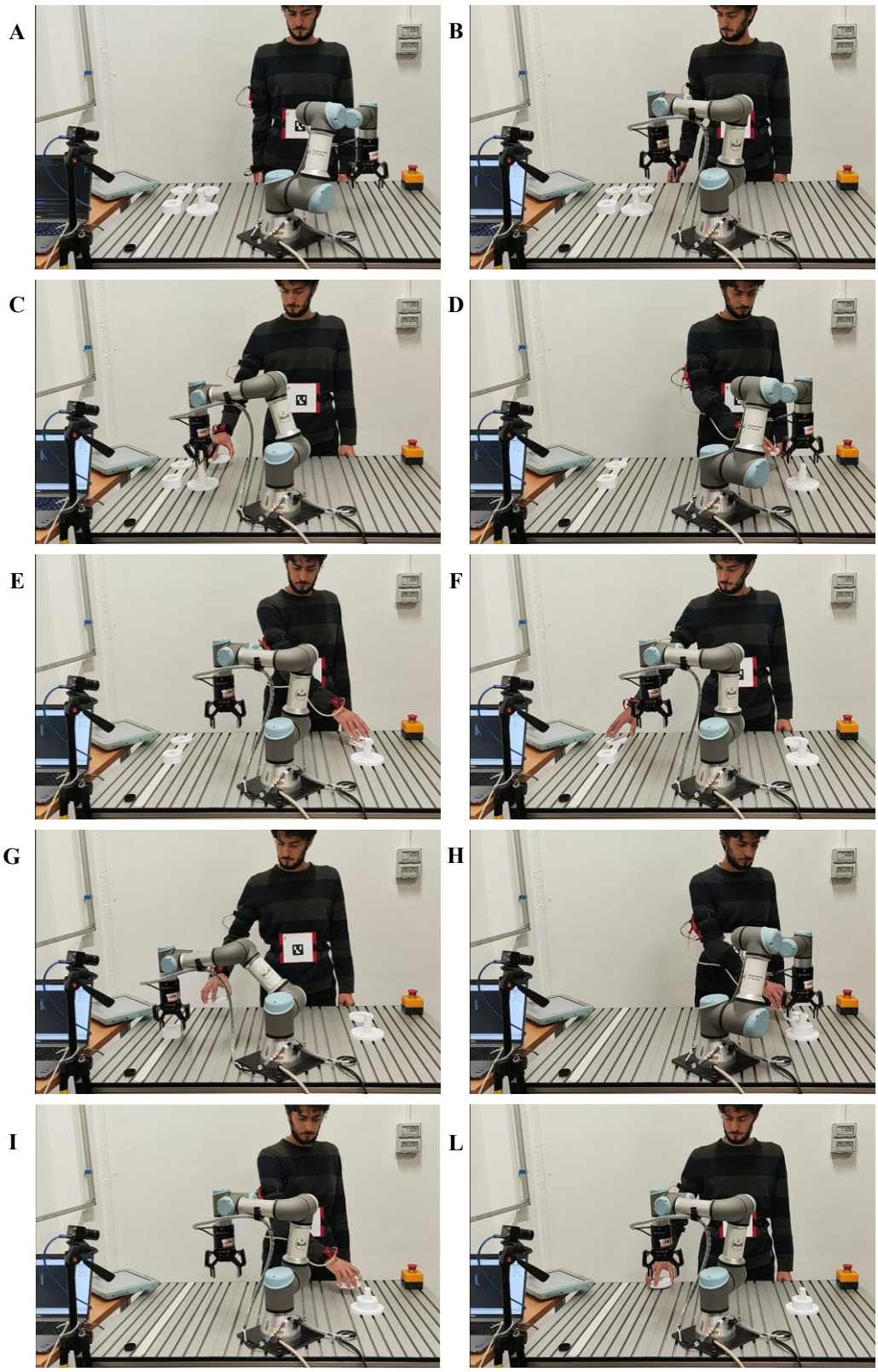

Figure 72 Representative frames of the collaborative assembly task execution. The sequence shows the initial instant of each task phase, including independent manipulation by the robot and the operator, coordinated pick-and-place actions.

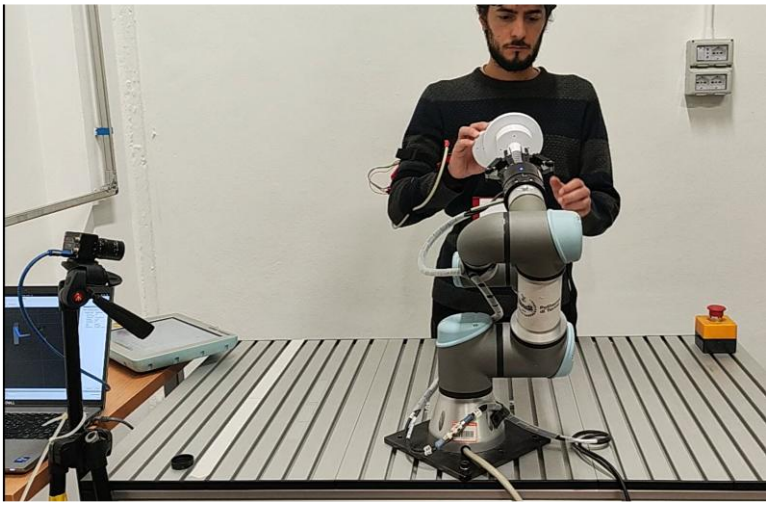

Figure 73 Final stage of the task, in which the robot presents the subassembly to the operator to enable completion of the final operation.

Safety Response to Abrupt Human Motion

A second experimental trial was conducted to evaluate the behavior of the proposed safety system in the presence of abrupt human motion. In this experiment, the same collaborative assembly task described in the previous section was executed; however, during task execution, the operator deliberately performed a sudden and impulsive movement to trigger the safety mechanism. The objective of this trial was to assess the responsiveness of the abrupt movement detection module and to analyze the resulting robot behavior under safety-critical conditions.

The safety response observed in this experiment is the result of a sequential processing chain that spans multiple system components. Inertial data acquired from the wearable sensors are first processed by the abrupt movement detection module, whose output is continuously monitored by the Motion Manager. Upon detection of an unsafe condition, the Motion Manager overrides the nominal task execution and issues a stop command to the execution layer, which is ultimately handled by the robot controller. This chain reflects the tight integration between perception, decision-making, and execution layers within the proposed framework.

Figure 74 shows the temporal evolution of the inertial measurements acquired from the operator's forearm, including linear acceleration and angular velocity components, together with the joint positions of the robot manipulator. The vertical dashed line indicates the instant at which the abrupt movement is detected by the classification module. As visible from the acceleration signals, the impulsive motion produces a clear peak, followed by a rapid variation in the angular velocity components.

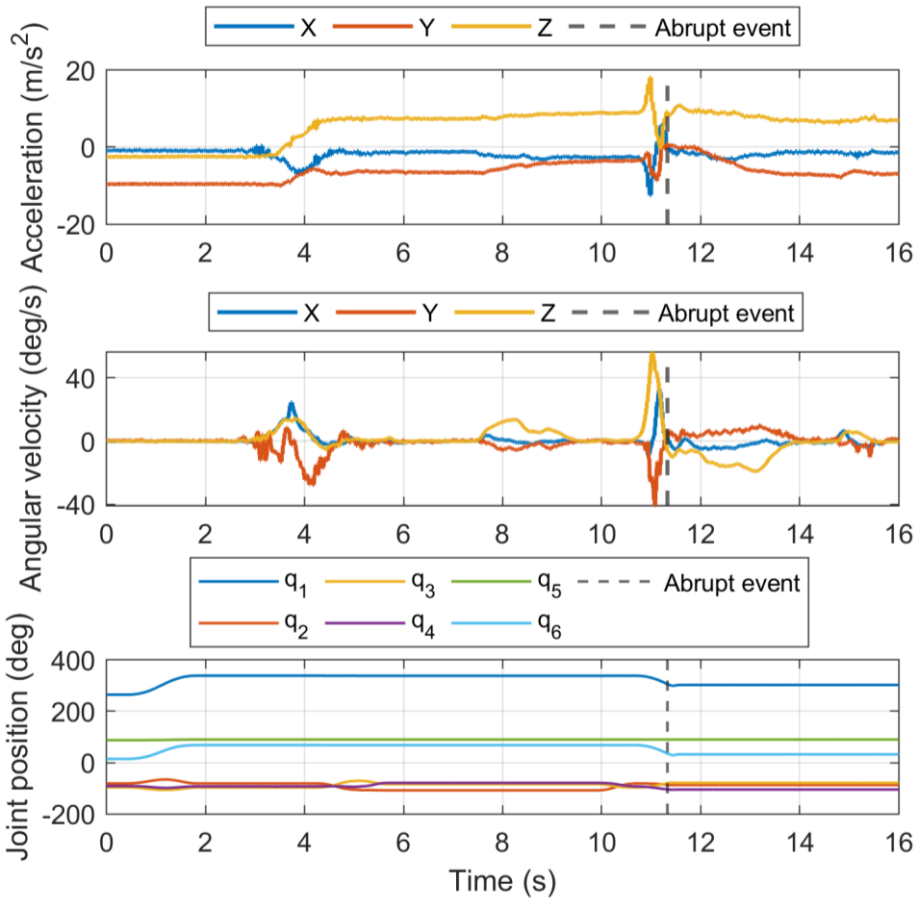

Figure 74 System response to an abrupt human motion during task execution. Forearm inertial measurements (linear acceleration and angular velocity) together with the robot joint positions. The vertical dashed line indicates the instant at which the abrupt movement is detected, triggering trajectory interruption and a safety stop of the robot.

From the recorded data, it can be observed that the abrupt movement is identified approximately 0.3 s after the occurrence of the acceleration peak. Following this detection event, the robot motion is interrupted and the manipulator comes to a complete stop after approximately 0.1 s. It is important to distinguish between the latency associated with abrupt movement detection and the subsequent physical stopping time of the robot. The former is primarily determined by the window-based processing and classification of inertial data, while the latter depends on the propagation of the stop command through the supervisory and execution layers, as well as on the dynamics of the robot controller. The measured delays therefore reflect the combined effect of sensing, computation, communication, and actuation, rather than the performance of a single module.

Figure 75 provides a zoomed view around the abrupt movement event, showing the temporal evolution of all robot joint positions. Within this zoomed interval, particular attention is given to joint 1 to illustrate the system response in detail. Once the abrupt movement is identified, the Motion Manager immediately issues a trajectory abort command. This is implemented by commanding the robot to maintain its current configuration, effectively overriding the ongoing trajectory execution. The plot highlights a short transient phase between the detection of the

abrupt movement and the effective enforcement of the stop condition. During this interval, the robot continues to move slightly along the direction of the originally planned trajectory while the stop command is being processed.

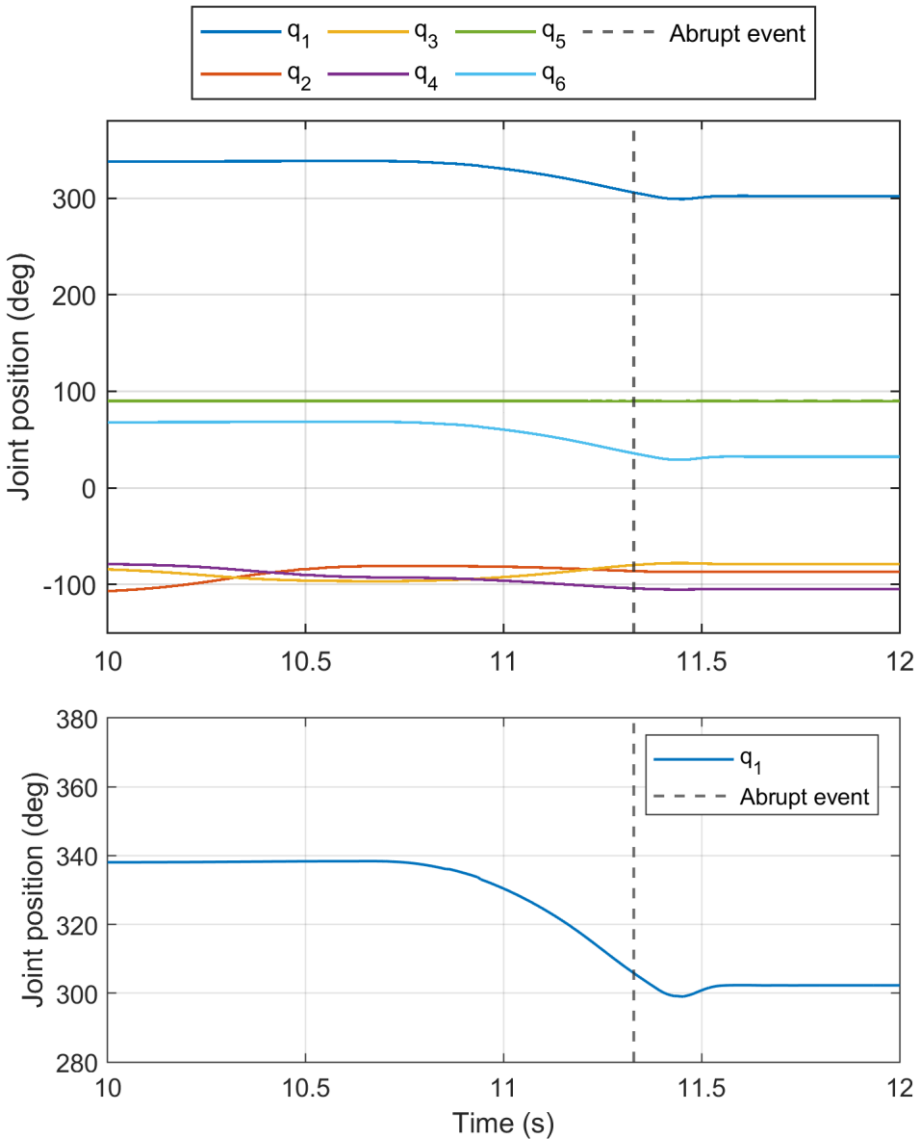

Figure 75 (Top) Zoomed view of robot joints response to abrupt movement detection. (Bottom) Position of joint q_1 around the detection instant (dashed line). After the abrupt event is identified, the ongoing trajectory is aborted, and the controller drives the joint toward the commanded stop configuration.

After this transient, the controller receives the updated command and sets a new reference position corresponding to the robot configuration at the instant of abrupt movement detection. This instant approximately coincides with the minimum value observed in the joint 1 position trace. From this point onward, the low-level controller drives the robot toward the commanded stop configuration and maintains it. The system remains in this halted state until an explicit restart command is issued by the central control unit, ensuring that task execution does not resume automatically after a safety event.

In Figure 76 shows the linear acceleration of the operator's forearm plotted in a time window centered around the abrupt movement event. Selected video

frames are associated with the acceleration signal to provide qualitative correspondence between the measured inertial data and the observed system behavior. The frames reported above the plot correspond, respectively, to the operator at the start of the motion, to the instant close to the acceleration peak, an instant shortly before the identification of the abrupt movement, and to the moment in which the robot starts to reverse its motion. The frame reported below the plot shows the configuration in which the robot is completely stopped.

The same frames are reported in Figure 77, where the first four frames are superimposed to highlight the robot motion occurring during the detection and reaction interval, while the lower frame again shows the configuration in which the system has reached a full stop.

Overall, this experiment demonstrates that the proposed framework is capable of enforcing effective safety behavior in dynamic collaborative scenarios by tightly coupling perception, supervisory control, and execution.

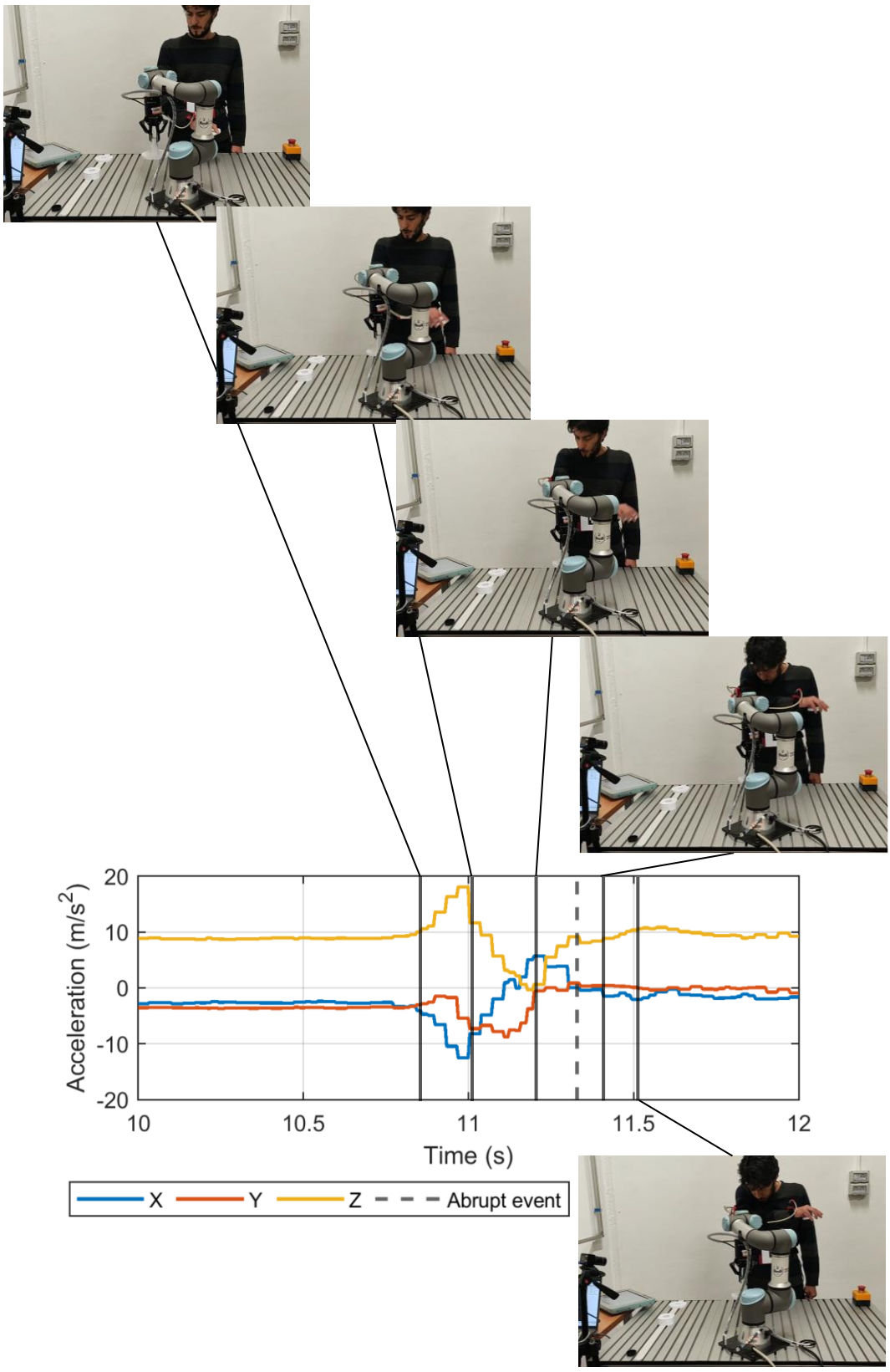

Figure 76 Forearm acceleration around the abrupt movement event with associated video frames. The linear acceleration components are plotted over a time window centered on the abrupt movement. Selected frames are linked to the signal to illustrate the operator motion before the identification of the abrupt event, during the transient response, and after the robot has come to a complete stop.

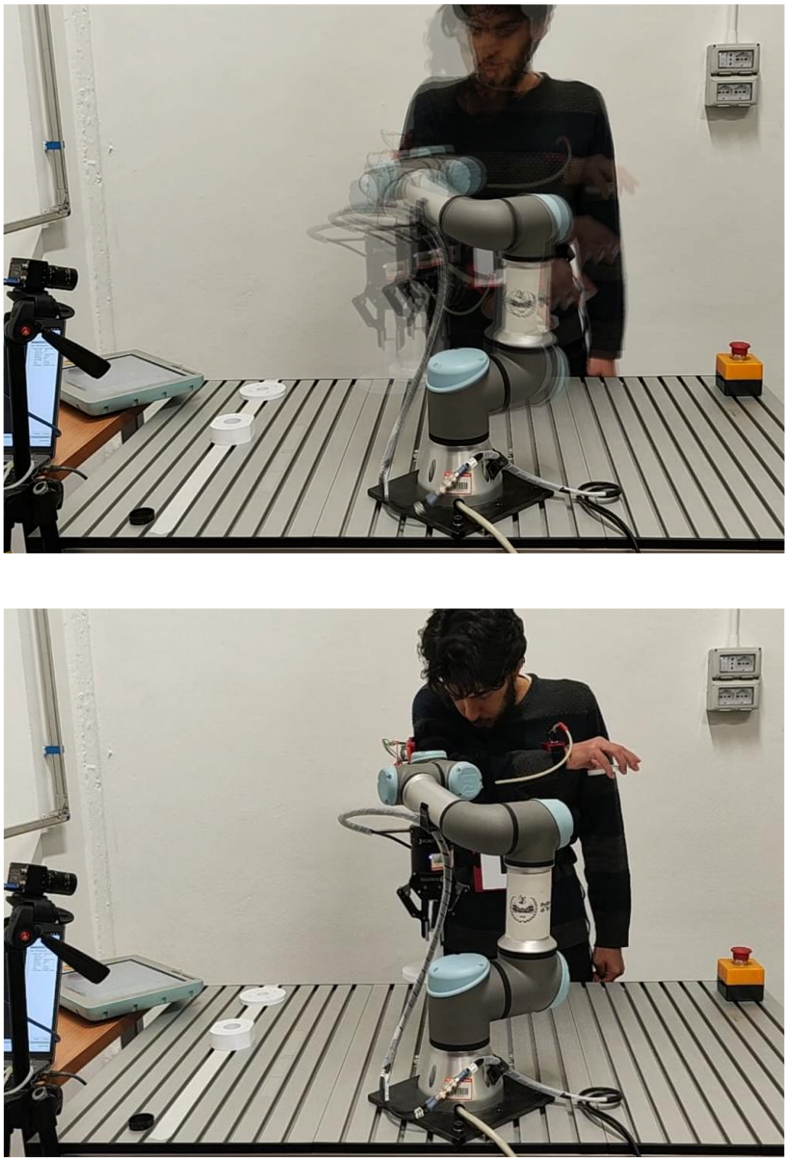

Figure 77 Robot behavior in the vicinity of an abrupt human movement. The upper image shows four superimposed frames captured during the abrupt movement, illustrating the robot displacement while the stop command is being processed. The lower image shows the robot already stopped.

6.4 Conclusion

This chapter presented the complete integration of the methods developed throughout the thesis into a single operational framework for human–robot collaboration. The ROS 2 software architecture, the experimental setup, and the system activation procedure were described to show how perception, safety monitoring, and motion planning are combined in a real-time system.

The collaborative assembly task demonstrated how the proposed framework enables synchronized human–robot interaction without relying on predefined timing, but instead on continuous perception of the operator and supervisory decision-making. The additional safety experiment showed that abrupt human

motions are detected within bounded latency and that robot motion can be reliably interrupted through the Motion Manager.

Overall, this chapter demonstrated that the proposed approaches can be effectively deployed in a real collaborative scenario, achieving real-time operation, safe interaction, and modular integration of heterogeneous sensing and planning components.

Chapter 7

Conclusion

This thesis addressed the problem of effective and safe human–robot collaboration by focusing on the perception and cognition layers of the interaction loop. Starting from the limitations of current collaborative robotic applications, which are often constrained to conservative safety mechanisms and limited adaptability, the work investigated how multimodal perception and learning-based decision-making can enhance robot awareness of human behavior and enable more context-aware actions.

The research was framed within a perception–decision–action paradigm, with the explicit objective of improving the robot's understanding of the human operator and exploiting this information to generate safer and more adaptive motions. To this end, the thesis developed and experimentally validated a human-centered framework integrating multimodal motion tracking, safety-related motion assessment, and human-aware path planning.

A first major contribution of this work is the development of a multimodal human arm tracking system based on the fusion of inertial sensing and event-based vision. The proposed approach combines two wearable IMUs with a single event camera to estimate the configuration of the human upper limb through a kinematic model and a Kalman filter–based fusion framework. By exploiting the complementary characteristics of the two sensing modalities the system achieves accurate and stable estimation of elbow and wrist positions while maintaining reduced computational load. Experimental validation demonstrated that the fused solution improves robustness and accuracy with respect to unimodal approaches, while remaining suitable for real-time operation in collaborative scenarios.

The second contribution concerns the assessment of motion safety through the detection of abrupt movements. Using the same inertial sensing infrastructure, a data-driven method was developed to distinguish standard task-related motions from abrupt, potentially hazardous behaviors. A dedicated experimental protocol was designed to collect representative motion data during a pick-and-place task, including both standard and stimulus-induced abrupt movements. Machine

learning–based classifiers were trained and evaluated using a sliding-window approach, and their performance was analyzed in terms of accuracy, temporal resolution, and inference latency. The results showed that abrupt movements can be reliably detected within time windows compatible with real-time safety monitoring, providing a meaningful abstraction of the operator's dynamic safety state.

A further contribution of the thesis lies in the development of learning-based motion planning strategies for human-aware robot behavior. Reinforcement learning techniques were applied to both joint-level control and Cartesian path generation, with safety constraints explicitly embedded in the reward design. The proposed planners were trained to generate collision-free motions in the presence of obstacles representing the human operator, adapting the robot trajectory while respecting kinematic and dynamic constraints. The experimental results highlighted the ability of the learned policies to deviate from nominal paths to ensure safety, demonstrating the feasibility of integrating learning-based planning within collaborative robotic applications.

Finally, these components were integrated into a unified collaboration framework implemented using a ROS 2 architecture. The framework combines multimodal perception, motion safety assessment, and learning-based planning into a coherent system capable of operating in real time. The modular design allows the perception and decision-making layers to interact continuously, enabling the robot to adapt its behavior based on the estimated human state and the assessed level of motion safety.

The results presented in this thesis confirm that multimodal perception plays a key role in advancing human–robot collaboration beyond reactive and conservative safety strategies. The integration of event-based vision and inertial sensing proved particularly effective in capturing human motion with low latency and robustness to environmental conditions, addressing some of the limitations of conventional frame-based vision systems. At the same time, the use of data-driven methods for motion safety assessment and reinforcement learning for path planning illustrates how cognitive layers can exploit perceptual information to support adaptive robot behavior.

From a broader perspective, the proposed framework is aligned with the principles underlying current and emerging international safety standards for collaborative robotics. In particular, the emphasis on continuous perception, dynamic safety assessment, and adaptive behavior is consistent with the evolution of ISO 10218 toward application-level collaboration and context-aware safety management.

Future Improvements

The system has been validated under controlled experimental conditions and with a specific collaborative task. While this choice enabled a repeatable evaluation of the proposed methods, it limits the generalizability of the conclusions to more complex tasks and less structured environments. Extending

experimental validation to a broader set of scenarios represents a necessary step toward assessing robustness and scalability in realistic industrial applications.

Human motion tracking is based on a simplified kinematic model of the upper limb. This ensures computational efficiency and stability but reduces anatomical fidelity, particularly in the presence of complex or highly articulated movements. Moreover, the adopted sensor fusion strategy operates at the level of independent pose estimates obtained from inertial sensing and event-based vision. As a consequence, the fused estimate remains strongly dependent on the quality of the individual modalities and cannot significantly diverge from their respective errors. A natural evolution of the framework would involve the adoption of a lower-level sensor fusion approach, directly integrating raw or minimally processed measurements. Such a strategy would allow a tighter coupling between modalities, improving robustness and reducing sensitivity to errors affecting single sensor streams.

In the current implementation, the trunk is treated as a fixed reference frame, which simplifies the estimation problem but reduces tracking accuracy when torso motion occurs and restricts the operator's effective workspace. This limitation could be addressed through two complementary approaches. A first solution consists in periodically updating the torso pose by low-rate sampling of the ArUco marker already used for reference frame alignment, thereby preserving the existing sensing architecture while limiting additional computational cost. Alternatively, the introduction of an additional inertial sensor on the trunk would enable direct measurement of torso motion, supporting a fully wearable configuration and allowing more natural whole-body movements.

The abrupt movement detection module currently provides a binary classification of motion safety. While effective for identifying potentially hazardous behaviors, this formulation limits the range of possible robot responses. A tighter integration between abrupt movement detection and the planning layer would enable graded or adaptive safety strategies. For instance, the robot could avoid unnecessary stops when the operator remains at a sufficiently safe distance, or activate more conservative behaviors only in genuinely critical situations. In more dangerous scenarios, abrupt motion detection could trigger active avoidance maneuvers rather than simple motion interruption, improving both safety and task continuity.

With respect to motion planning, the reinforcement learning agents were trained in simulation and validated under specific conditions. Although the results demonstrate effective behavior within the trained scenarios, significant deviations from the training distribution may lead to performance degradation. Extending the training process to encompass a wider range of tasks, constraints, and dynamic human behaviors would improve generalization and support more flexible collaboration.

Finally, the current framework lacks explicit task awareness, as it does not include mechanisms for identifying or localizing the components involved in the collaborative process. Furthermore, task programming currently requires the development of dedicated code, limiting accessibility in real industrial

environments. Future extensions could integrate additional sensing modalities to enhance task understanding while also enabling more intuitive interaction channels for the operator, thereby increasing both usability and adaptability.

Overall, these limitations delineate the current boundaries of the implemented system while outlining a clear path for its evolution. Addressing them would allow the framework to progress toward more general, adaptive, and user-centered human–robot collaboration.

In conclusion, this thesis demonstrated how multimodal perception and learning-based methods can be combined to enhance safety and adaptability in human–robot collaboration. By integrating inertial sensing, event-based vision, motion safety assessment, and reinforcement learning–based planning within a unified framework, the work contributes toward more human-aware robotic systems. The proposed methods provide a concrete step beyond static safety mechanisms, supporting collaborative applications in which robots continuously perceive, interpret, and respond to human behavior in shared workspaces.

References

1. (2024) World Robotics 2024
2. Liu D, Son S (2024) Exploring the impact mechanism of collaborative robot on manufacturing firm performance: A dynamic capability perspective. Sustainable Futures 8:. https://doi.org/10.1016/j.sftr.2024.100262
3. Gambao Ernesto (2023) Analysis exploring risks and opportunities linked to the use of collaborative industrial robots in Europe. [European Parliament]
4. (2024) Collaborative Robots HOW ROBOTS WORK ALONGSIDE HUMANS. Frankfurt
5. Bonci A, Cheng PDC, Indri M, et al (2021) Human-robot perception in industrial environments: A survey. Sensors 21:1–29
6. Wang T, Zheng P, Li S, Wang L (2024) Multimodal Human–Robot Interaction for Human-Centric Smart Manufacturing: A Survey. Advanced Intelligent Systems 6
7. Yang J, Liu Y, Morgan PL (2024) Sensor and data: key elements of human-machine interaction for human-centric smart manufacturing. In: Procedia Computer Science. Elsevier B.V., pp 191–200
8. Duncan JA, Alambeigi F, Pryor MW (2024) A Survey of Multimodal Perception Methods for Human-Robot Interaction in Social Environments. ACM Trans Hum Robot Interact 13:. https://doi.org/10.1145/3657030
9. Zhao W, Gangaraju K, Yuan F (2025) Multimodal perception-driven decision-making for human-robot interaction: a survey. Front. Robot. AI 12
10. Robinson N, Tidd B, Campbell D, et al (2023) Robotic Vision for Human-Robot Interaction and Collaboration: A Survey and Systematic Review. ACM Trans Hum Robot Interact 12:. https://doi.org/10.1145/3570731
11. Arvanitis G, Piperigkos N, Anagnostopoulos C, et al (2023) Real time enhancement of operator's ergonomics in physical human - robot collaboration scenarios using a multi-stereo camera system. In: 2023 IEEE International Conference on Industrial Technology (ICIT). IEEE, pp 1–6
12. Slovák J, Melicher M, Šimovec M, Vachálek J (2021) Vision and RTLS Safety Implementation in an Experimental Human—Robot Collaboration Scenario. Sensors 21:2419. https://doi.org/10.3390/s21072419
13. Gao Z, Wang Z, Saint-Bauzel L, Ben Amar F (2023) 2D LiDAR-Based Human Pose Tracking for a Mobile Robot. In: Proceedings of the 20th International Conference on Informatics in Control, Automation and

Robotics. SCITEPRESS - Science and Technology Publications, pp 511–519
14. Lee S, Lee D-W, Jun K, et al (2022) Markerless 3D Skeleton Tracking Algorithm by Merging Multiple Inaccurate Skeleton Data from Multiple RGB-D Sensors. Sensors 22:3155. https://doi.org/10.3390/s22093155
15. Hartmann Y, Paul RE, Klöckner J, et al (2025) Gait parameter estimation from a single depth sensor. Journal of Smart Cities and Society 4:35–61. https://doi.org/10.1177/27723577251320237
16. Pascual-Hernández D, Oyaga de Frutos N, Mora-Jiménez I, Cañas-Plaza JM (2022) Efficient 3D human pose estimation from RGBD sensors. Displays 74:102225. https://doi.org/10.1016/j.displa.2022.102225
17. Tölgyessy M, Dekan M, Chovanec Ľ, Hubinský P (2021) Evaluation of the Azure Kinect and Its Comparison to Kinect V1 and Kinect V2. Sensors 21:413. https://doi.org/10.3390/s21020413
18. Zimmermann C, Welschehold T, Dornhege C, et al (2018) 3D Human Pose Estimation in RGBD Images for Robotic Task Learning. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp 1986–1992
19. Gallego G, Delbruck T, Orchard G, et al (2022) Event-Based Vision: A Survey. IEEE Trans Pattern Anal Mach Intell 44:154–180. https://doi.org/10.1109/TPAMI.2020.3008413
20. Chakravarthi B, Verma AA, Daniilidis K, et al (2025) Recent Event Camera Innovations: A Survey. pp 342–376
21. iniVation AG Event Accumulation. https://dv-processing.inivation.com/master/accumulators.html. Accessed 13 Jan 2026
22. Rebecq H, Ranftl R, Koltun V, Scaramuzza D (2021) High Speed and High Dynamic Range Video with an Event Camera. IEEE Trans Pattern Anal Mach Intell 43:1964–1980. https://doi.org/10.1109/TPAMI.2019.2963386
23. Duarte L, Safeea M, Neto P (2021) Event-based tracking of human hands. Sensor Review 41:382–389. https://doi.org/10.1108/SR-03-2021-0095
24. Viola P, Jones MJ (2004) Robust Real-Time Face Detection
25. Zhang L, Vaughan R (2016) Optimal robot selection by gaze direction in multi-human multi-robot interaction. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp 5077–5083
26. Condés I, Cañas JM (2018) Person Following Robot Behavior Using Deep Learning. Springer, Cham, pp 147–161
27. Ke X, Zhu Y, Yang Y, et al (2016) Vision system of facial robot SHFR- III for human-robot interaction. In: ICINCO 2016 - Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics. SciTePress, pp 472–478
28. Lam MC, Prabuwono AS, Arshad H, Chan CS (2011) LNCS 7066 - A Real-Time Vision-Based Framework for Human-Robot Interaction

29. Li THS, Kuo PH, Tsai TN, Luan PC (2019) CNN and LSTM Based Facial Expression Analysis Model for a Humanoid Robot. IEEE Access 7:93998–94011. https://doi.org/10.1109/ACCESS.2019.2928364
30. Ganesh Choudhary B, Chethan Ram B V (2014) Real time robotic arm control using hand gestures. In: 2014 International Conference on High Performance Computing and Applications (ICHPCA). IEEE, pp 1–3
31. Foster ME, Gaschler A, Giuliani M, et al (2012) Two people walk into a bar. In: Proceedings of the 14th ACM international conference on Multimodal interaction. ACM, New York, NY, USA, pp 3–10
32. Lambrecht J, Kruger J (2012) Spatial programming for industrial robots based on gestures and Augmented Reality. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, pp 466–472
33. Mazhar O, Navarro B, Ramdani S, et al (2019) A real-time human-robot interaction framework with robust background invariant hand gesture detection. Robot Comput Integr Manuf 60:34–48. https://doi.org/10.1016/j.rcim.2019.05.008
34. Xu J, Li J, Zhang S, et al (2020) Skeleton guided conflict-free hand gesture recognition for robot control. In: 2020 11th International Conference on Awareness Science and Technology, iCAST 2020. Institute of Electrical and Electronics Engineers Inc.
35. Barattini P, Morand C, Robertson NM (2012) A proposed gesture set for the control of industrial collaborative robots. In: 2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication. IEEE, pp 132–137
36. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society, pp 779–788
37. Vasquez A, Kollmitz M, Eitel A, Burgard W Deep Detection of People and their Mobility Aids for a Hospital Robot
38. Weber T, Triputen S, Danner M, et al (2018) Follow Me: Real-Time in the Wild Person Tracking Application for Autonomous Robotics. In: Akiyama H, Obst O, Sammut C, Tonidandel F (eds). Springer International Publishing, Cham, pp 156–167
39. Wei SE, Ramakrishna V, Kanade T, Sheikh Y (2016) Convolutional pose machines. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society, pp 4724–4732
40. Cao Z, Hidalgo G, Simon T, et al (2021) OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. IEEE Trans Pattern Anal Mach Intell 43:172–186. https://doi.org/10.1109/TPAMI.2019.2929257
41. Luo X, Zhang D, Jin X (2019) A Real-time Moving Target Following Mobile Robot System with Depth Camera. In: IOP Conference Series: Materials Science and Engineering. Institute of Physics Publishing

42. Mollaret C, Mekonnen AA, Pinquier J, et al (2016) A multi-modal perception based architecture for a non-intrusive domestic assistant robot. In: 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI). IEEE, pp 481–482
43. Wu Y, Yang Q, Zhou X (2019) An improved method of optical flow using human body-following wheeled robot. International Journal of Modeling, Simulation, and Scientific Computing 10:1950003. https://doi.org/10.1142/S179396231950003X
44. Bellarbi A, Kahlouche S, Achour N, Ouadah N (2016) A social planning and navigation for tour-guide robot in human environment. In: 2016 8th International Conference on Modelling, Identification and Control (ICMIC). IEEE, pp 622–627
45. Landi CT, Cheng Y, Ferraguti F, et al (2019) Prediction of Human Arm Target for Robot Reaching Movements. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp 5950–5957
46. Zardykhan D, Svarny P, Hoffmann M, et al (2019) Collision Preventing Phase-Progress Control for Velocity Adaptation in Human-Robot Collaboration. In: 2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids). IEEE, pp 266–273
47. Garcia-De-Villa S, Casillas-Perez D, Jimenez-Martin A, Garcia-Dominguez JJ (2023) Inertial Sensors for Human Motion Analysis: A Comprehensive Review. IEEE Trans Instrum Meas 72:. https://doi.org/10.1109/TIM.2023.3276528
48. Gill WA, Howard I, Mazhar I, McKee K (2022) A Review of MEMS Vibrating Gyroscopes and Their Reliability Issues in Harsh Environments. Sensors 22:7405. https://doi.org/10.3390/s22197405
49. Ricci L, Taffoni F, Formica D (2016) On the Orientation Error of IMU: Investigating Static and Dynamic Accuracy Targeting Human Motion. PLoS One 11:e0161940. https://doi.org/10.1371/journal.pone.0161940
50. Digo E, Pastorelli S, Gastaldi L (2022) A Narrative Review on Wearable Inertial Sensors for Human Motion Tracking in Industrial Scenarios. Robotics 11
51. De Marchi M, Turetta C, Pravadelli G, Bombieri N (2024) Real-Time Multi-Person Identification and Tracking via HPE and IMU Data Fusion
52. Amorim A, Guimares D, Mendona T, et al (2021) Robust human position estimation in cooperative robotic cells. Robot Comput Integr Manuf 67:. https://doi.org/10.1016/j.rcim.2020.102035
53. Škulj G, Vrabič R, Podržaj P (2021) A wearable imu system for flexible teleoperation of a collaborative industrial robot. Sensors 21:. https://doi.org/10.3390/s21175871
54. Antonelli M, Digo E, Pastorelli S, Gastaldi L (2021) Wearable MIMUs for the identification of upper limbs motion in an industrial context of human-robot interaction. In: Proceedings of the 18th International Conference on

Informatics in Control, Automation and Robotics, ICINCO 2021. SciTePress, pp 403–409
55. Kim J, Lee J, Kim W (2024) Transferable Convolutional Neural Networks for IMU-based Motion Gesture Recognition in Human-Machine Interaction. In: 2024 24th International Conference on Control, Automation and Systems (ICCAS). IEEE, pp 61–66
56. Sopidis G, Haslgrübler M, Azadi B, et al (2022) Micro-activity recognition in industrial assembly process with IMU data and deep learning. In: ACM International Conference Proceeding Series. Association for Computing Machinery, pp 103–112
57. Kahanowich ND, Sintov A (2024) Learning Human-Arm Reaching Motion Using a Wearable Device in Human-Robot Collaboration. IEEE Access 12:24855–24865. https://doi.org/10.1109/ACCESS.2024.3365661
58. Islam A, Sundaraj K, Ahmad B, et al (2012) Mechanomyography Sensors for Muscle Assessment: a Brief Review. J Phys Ther Sci 24:1359–1365. https://doi.org/10.1589/jpts.24.1359
59. Abioye AO, Prior SD, Saddington P, Ramchurn SD (2022) The performance and cognitive workload analysis of a multimodal speech and visual gesture (mSVG) UAV control interface. Rob Auton Syst 147:103915. https://doi.org/10.1016/j.robot.2021.103915
60. Abioye AO, Prior SD, Thomas GT, et al (2018) The Multimodal Speech and Visual Gesture (mSVG) Control Model for a Practical Patrol, Search, and Rescue Aerobot. pp 423–437
61. Liu H, Fang T, Zhou T, Wang L (2018) Towards Robust Human-Robot Collaborative Manufacturing: Multimodal Fusion. IEEE Access 6:74762–74771. https://doi.org/10.1109/ACCESS.2018.2884793
62. Trick S, Herbert F, Rothkopf CA, Koert D (2022) Interactive Reinforcement Learning With Bayesian Fusion of Multimodal Advice. IEEE Robot Autom Lett 7:7558–7565. https://doi.org/10.1109/LRA.2022.3182100
63. Portugal D, Alvito P, Christodoulou E, et al (2019) A Study on the Deployment of a Service Robot in an Elderly Care Center. Int J Soc Robot 11:317–341. https://doi.org/10.1007/s12369-018-0492-5
64. Foster ME, Gaschler A, Giuliani M (2017) Automatically Classifying User Engagement for Dynamic Multi-party Human–Robot Interaction. Int J Soc Robot 9:659–674. https://doi.org/10.1007/s12369-017-0414-y
65. Rascon C, Meza I (2017) Localization of sound sources in robotics: A review. Rob Auton Syst 96:184–210. https://doi.org/10.1016/j.robot.2017.07.011
66. Bayer IS (2022) MEMS-Based Tactile Sensors: Materials, Processes and Applications in Robotics. Micromachines (Basel). 13
67. Jin J, Wang S, Zhang Z, et al (2023) Progress on flexible tactile sensors in robotic applications on objects properties recognition, manipulation and human-machine interactions. Soft Science 3

68. Xu J, Pan J, Cui T, et al (2023) Recent Progress of Tactile and Force Sensors for Human–Machine Interaction. Sensors 23
69. Esposito D, Centracchio J, Andreozzi E, et al (2021) Biosignal-based human–machine interfaces for assistance and rehabilitation: A survey. Sensors 21
70. Welihinda DVDS, Gunarathne LKP, Herath HMKKMB, et al (2024) EEG and EMG-based human-machine interface for navigation of mobility-related assistive wheelchair (MRA-W). Heliyon 10:e27777. https://doi.org/10.1016/j.heliyon.2024.e27777
71. Alickovic E, Kevric J, Subasi A (2018) Performance evaluation of empirical mode decomposition, discrete wavelet transform, and wavelet packed decomposition for automated epileptic seizure detection and prediction. Biomed Signal Process Control 39:94–102. https://doi.org/10.1016/j.bspc.2017.07.022
72. Fahmi M, Noorani MRS, Zakeri M (2024) Machine Learning for Human-Robot Interaction using EMG Signals to Control the Grip Force. In: ICRoM 2024 - 12th RSI International Conference on Robotics and Mechatronics. Institute of Electrical and Electronics Engineers Inc., pp 437–442
73. Cao T, Sun J, Liu D, et al (2022) Wireless Collaboration Technology Based on Electroencephalograph and Electromyography. In: 2022 IEEE 4th International Conference on Power, Intelligent Computing and Systems, ICPICS 2022. Institute of Electrical and Electronics Engineers Inc., pp 921–925
74. Meng L, Yang L, Zheng E (2024) Hierarchical Human Motion Intention Prediction for Increasing Efficacy of Human-Robot Collaboration. IEEE Robot Autom Lett 9:7637–7644. https://doi.org/10.1109/LRA.2024.3430131
75. Bimbraw K, Zheng M (2023) Towards The Development of a Low-Latency, Biosignal-Controlled Human-Machine Interaction System. In: 2023 IEEE/SICE International Symposium on System Integration, SII 2023. Institute of Electrical and Electronics Engineers Inc.
76. Wang M, Yan Z, Wang T, et al (2020) Gesture recognition using a bioinspired learning architecture that integrates visual data with somatosensory data from stretchable sensors. Nat Electron 3:563–570. https://doi.org/10.1038/s41928-020-0422-z
77. Lee Y, Do W, Yoon H, et al (2021) Visual-inertial hand motion tracking with robustness against occlusion, interference, and contact. Sci Robot 6:. https://doi.org/10.1126/scirobotics.abe1315
78. Song Q, Sun B, Li S (2023) Multimodal Sparse Transformer Network for Audio-Visual Speech Recognition. IEEE Trans Neural Netw Learn Syst 34:10028–10038. https://doi.org/10.1109/TNNLS.2022.3163771
79. Ngai WK, Xie H, Zou D, Chou K-L (2022) Emotion recognition based on convolutional neural networks and heterogeneous bio-signal data sources.

Information Fusion 77:107–117. https://doi.org/10.1016/j.inffus.2021.07.007
80. Lyu J, Ruppel P, Hendrich N, et al (2023) Efficient and Collision-Free Human–Robot Collaboration Based on Intention and Trajectory Prediction. IEEE Trans Cogn Dev Syst 15:1853–1863. https://doi.org/10.1109/TCDS.2022.3215093
81. Kothari A, Tohme T, Zhang X, Youcef-Toumi K (2025) Enhanced Human-Robot Collaboration using Constrained Probabilistic Human-Motion Prediction. In: 2025 IEEE 21st International Conference on Automation Science and Engineering (CASE). IEEE, pp 541–547
82. Martini E, Parekh H, Peng S, et al (2024) A Robust Filter for Marker-less Multi-person Tracking in Human-Robot Interaction Scenarios. In: 2024 33rd IEEE International Conference on Robot and Human Interactive Communication (ROMAN). IEEE, pp 424–429
83. Liu H, Brandli C, Li C, et al (2015) Design of a spatiotemporal correlation filter for event-based sensors. In: 2015 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, pp 722–725
84. Markley FL, Cheng Y, Crassidis JL, Oshman Y (2007) Averaging Quaternions. Journal of Guidance, Control, and Dynamics 30:1193–1197. https://doi.org/10.2514/1.28949
85. Shintemirov A, Taunyazov T, Omarali B, et al (2020) An Open-Source 7-DOF Wireless Human Arm Motion-Tracking System for Use in Robotics Research. Sensors 20:3082. https://doi.org/10.3390/s20113082
86. Wei W, Kurita K, Kuang J, Gao A (2021) Real-Time 3D Arm Motion Tracking Using the 6-axis IMU Sensor of a Smartwatch. In: 2021 IEEE 17th International Conference on Wearable and Implantable Body Sensor Networks (BSN). IEEE, pp 1–4
87. Zheng T, Cai C, Chen Z, Luo J (2022) Sound of Motion: Real-time Wrist Tracking with A Smart Watch-Phone Pair. In: IEEE INFOCOM 2022 - IEEE Conference on Computer Communications. IEEE, pp 110–119
88. Amorim A, Guimares D, Mendona T, et al (2021) Robust human position estimation in cooperative robotic cells. Robot Comput Integr Manuf 67:102035. https://doi.org/10.1016/j.rcim.2020.102035
89. Wei W, Kurita K, Kuang J, Gao A (2021) Real-Time Limb Motion Tracking with a Single IMU Sensor for Physical Therapy Exercises. In: 2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC). IEEE, pp 7152–7157
90. Castellote JM, Van den Berg MEL, Valls-Solé J (2013) The StartReact Effect on Self-Initiated Movements. Biomed Res Int 2013:1–9. https://doi.org/10.1155/2013/471792
91. Lasota PA, Fong T, Shah JA (2017) A Survey of Methods for Safe Human-Robot Interaction. Foundations and Trends® in Robotics 5:261–349. https://doi.org/10.1561/2300000052
92. Rosso V, Gastaldi L, Pastorelli S (2022) Detecting Impulsive Movements to Increase Operators' Safety in Manufacturing. pp 174–181

93. Castellote JM, Valls-Solé J (2015) The StartReact effect in tasks requiring end-point accuracy. Clinical Neurophysiology 126:1879–1885. https://doi.org/10.1016/j.clinph.2015.01.028
94. Kirschner RJ, Burr L, Porzenheim M, et al (2021) Involuntary Motion in Human-Robot Interaction: Effect of Interactive User Training on the Occurrence of Human Startle-Surprise Motion. In: 2021 IEEE International Conference on Intelligence and Safety for Robotics (ISR). IEEE, pp 28–32
95. Polito M, Digo E, Pastorelli S, Gastaldi L (2023) Abrupt Movements Assessment of Human Arms Based on Recurrent Neural Networks for Interaction with Machines. pp 143–151
96. Polito M, Digo E, Pastorelli S, Gastaldi L (2023) Deep Learning Technique to Identify Abrupt Movements in Human-Robot Collaboration. pp 73–80
97. Di Vincenzo G, Polito M, Digo E, et al (2025) Improvement of safety in collaborative robotics through prompt detection of abrupt movements
98. Digo E, Polito M, Caselli E, et al (2025) A Dataset of Standard and Abrupt Industrial Gestures Recorded Through MIMUs. Robotics 14:176. https://doi.org/10.3390/robotics14120176
99. Digo E, Polito M, Pastorelli S, Gastaldi L (2024) Detection of upper limb abrupt gestures for human–machine interaction using deep learning techniques. Journal of the Brazilian Society of Mechanical Sciences and Engineering 46:227. https://doi.org/10.1007/s40430-024-04746-9
100. Digo E, Polito M, Caselli E, et al (2025) DASIG. https://zenodo.org/records/17660014. Accessed 30 Nov 2025
101. Sutton RS., Barto AG. (2020) Reinforcement learning : an introduction. The MIT Press
102. MathWorks How Reinforcement Learning Works. https://www.mathworks.com/discovery/reinforcement-learning.html. Accessed 26 Dec 2025
103. Kavraki LE, Svestka P, Latombe J-C, Overmars MH (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation 12:566–580. https://doi.org/10.1109/70.508439
104. Xu T (2024) Recent advances in Rapidly-exploring random tree: A review. Heliyon 10:e32451. https://doi.org/10.1016/j.heliyon.2024.e32451
105. Sucan IA, Kavraki LE (2012) A Sampling-Based Tree Planner for Systems With Complex Dynamics. IEEE Transactions on Robotics 28:116–131. https://doi.org/10.1109/TRO.2011.2160466
106. Weingartshofer T, Bischof B, Meiringer M, et al (2023) Optimization-based path planning framework for industrial manufacturing processes with complex continuous paths. Robot Comput Integr Manuf 82:102516. https://doi.org/10.1016/j.rcim.2022.102516
107. Ratliff N, Zucker M, Bagnell JA, Srinivasa S (2009) CHOMP: Gradient optimization techniques for efficient motion planning. In: 2009 IEEE International Conference on Robotics and Automation. IEEE, pp 489–494

108. Kalakrishnan M, Chitta S, Theodorou E, et al (2011) STOMP: Stochastic trajectory optimization for motion planning. In: 2011 IEEE International Conference on Robotics and Automation. IEEE, pp 4569–4574
109. Schulman J, Ho J, Lee A, et al (2013) Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization. In: Robotics: Science and Systems IX. Robotics: Science and Systems Foundation
110. Khatib O Real-time obstacle avoidance for manipulators and mobile robots. In: Proceedings. 1985 IEEE International Conference on Robotics and Automation. Institute of Electrical and Electronics Engineers, pp 500–505
111. Khatib O (1986) Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. Int J Rob Res 5:90–98. https://doi.org/10.1177/027836498600500106
112. Liu J, Yap HJ, Khairuddin ASM (2024) Review on Motion Planning of Robotic Manipulator in Dynamic Environments. J. Sens. 2024
113. Momynkulov Z, Tursynova A, Olzhayev O, et al (2025) Three-Dimensional Trajectory Planning for Robotic Manipulators Using Model Predictive Control and Point Cloud Optimization. Computer Modeling in Engineering & Sciences 145:891–918. https://doi.org/10.32604/cmes.2025.068615

Appendix A