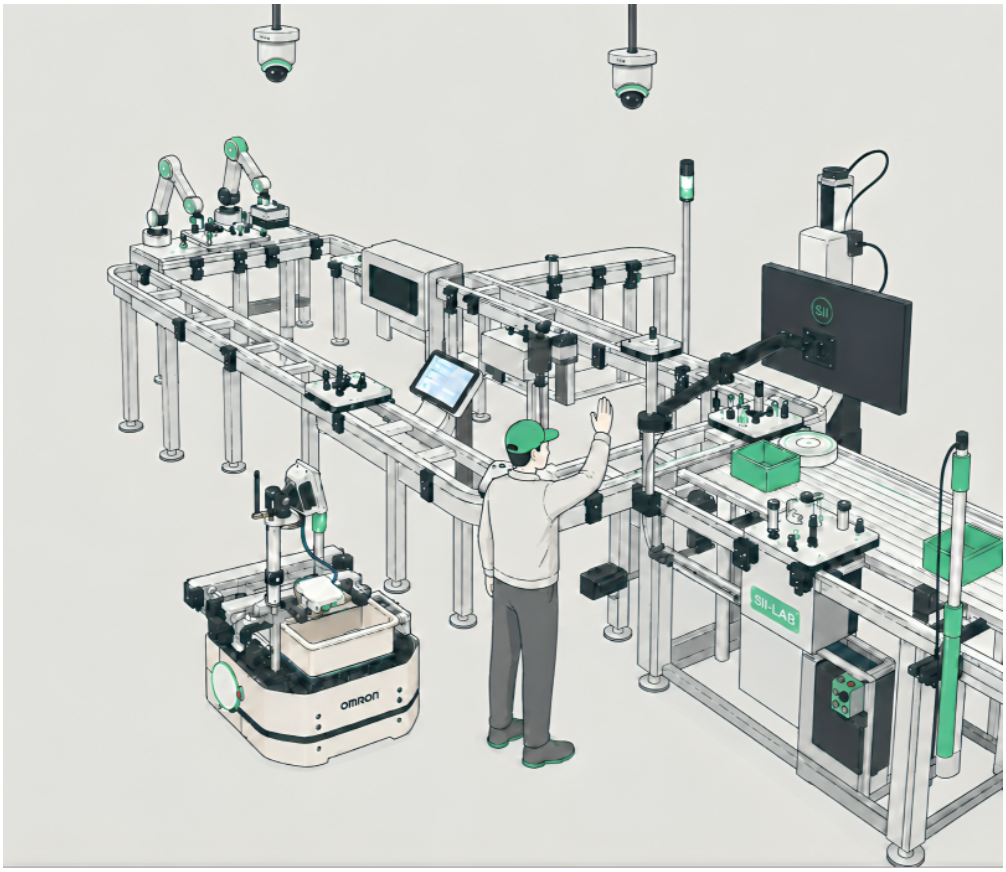
CHALMERS
UNIVERSITY OF TECHNOLOGY



Gesture-Based Control for Human-Centric Manufacturing in Industry 5.0

A Human-Centric Approach to Controlling Industrial Assets Using Real-Time 3D Pose Data

Master's thesis in Production Engineering

AEMAN ABDULLAH
MOHI EDDIN BILAL

DEPARTMENT OF MECHANICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Gesture-Based Control for Human-Centric Manufacturing in Industry 5.0

A Human-Centric Approach to Controlling Industrial Assets Using Real-Time 3D Pose Data

AEMAN ABDULLAH
MOHI EDDIN BILAL

CHALMERS
UNIVERSITY OF TECHNOLOGY

DEPARTMENT OF MECHANICAL ENGINEERING
Division of Production Systems
Chalmers University of Technology
Gothenburg, Sweden 2026

Gesture-Based Control for Human-Centric Manufacturing in Industry 5.0
A Human-Centric Approach to Controlling Industrial Assets Using Real-Time 3D Pose Data
AEMAN ABDULLAH
MOHI EDDIN BILAL

Supervisor: Huizhong Cao, Industrial and Materials Science
Supervisor: Pietro Lungaro, Occurrence Technologies
Examiner: Björn Johansson, Industrial and Materials Science

Master's Thesis 2026
Department of Mechanical Engineering
Division of Production Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Overview of the gesture-based control architecture connecting camera-based pose estimation, industrial command routing, AMR and conveyor execution, and digital-shadow-supported observation.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Gesture-Based Control for Human-Centric Manufacturing in Industry 5.0
A Human-Centric Approach to Controlling Industrial Assets Using Real-Time 3D Pose Data
AEMAN ABDULLAH & MOHI EDDIN BILAL
Department of Mechanical Engineering
Chalmers University of Technology

Abstract

Industry 5.0 emphasizes human-centric manufacturing, where digital technologies should support operators in industrial work. In this context, interaction methods are needed that allow operators to communicate with automated assets in a simple and practical way. This thesis investigates a gesture-based control layer for connected manufacturing systems, where camera-based 3D pose estimates are translated into discrete command actions and routed to industrial assets.

The aim of the work was to design, implement, and evaluate an integration blueprint for gesture-based human-system interaction in a laboratory manufacturing environment. The system uses a small vocabulary of static upper-body gestures with hold-time confirmation to reduce accidental triggering. Recognized gestures are processed by a Python-based gesture module and routed either through ThingWorx and Kepware or, for selected AMR actions, through a direct ARCL command path. The implementation was evaluated in a use case involving an OMRON LD-250 Autonomous Mobile Robot (AMR), a conveyor, and an Emulate3D-based digital shadow.

The final implementation included AMR mission commands, direct AMR stop handling, an optional robot-facing activation condition, and a stop-and-replace workflow that clears current and queued AMR mission requests after a stop gesture. For the conveyor, the implemented gesture commands included toggle, speed increase, speed decrease, and quick-stop actions. The digital shadow was used for monitoring and repeatable observation of AMR behavior, but it was not treated as a full digital twin.

The system was evaluated through controlled practical trials and a preliminary user survey. Under the tested laboratory conditions, the gesture vocabulary was recognized as intended, accepted gestures were mapped to the correct command actions, and the connected assets responded according to the implemented command logic. The main limitations were related to the simplified virtual AMR model in Emulate3D, the two-camera setup, pose-estimation jitter, occasional unexplained response-time delays, privacy and data-governance considerations, and the limited number of user participants. Overall, the thesis shows that gesture-based control can be used as a complementary interaction layer for selected manufacturing commands, provided that the sensing conditions, command logic, and safety limitations are clearly defined.

Keywords: Industry 5.0, gesture control, human-system interaction, AMR, digital shadow.

Acknowledgements

We would like to express our sincere gratitude to our academic supervisor, Huizhong Cao, for her guidance, feedback, and support throughout this thesis work. We would also like to thank our industrial supervisor, Pietro Lungaro, and his team for their technical support and insightful feedback.
We are grateful for the company of Sven Ekered during our long working hours and for all the little chit-chats that made the work more enjoyable.
A big thank you to the amazing Per Lönnehed for always stepping in when senior expertise was needed.
We would also like to thank Simon Heyes and Abbe Ahmed for the guidance, tips, and tricks they kindly shared with us.
We are grateful to Björn Johansson for examining the thesis and providing academic direction. We would also like to thank the staff and researchers connected to SII-Lab and Occurrence Technologies for their support during the laboratory work, demonstrations, and system integration.
Finally, a huge thank you to our friends, who shared with us the happiness of passing exams and the frustration of things not going as hoped. They have seen us at our best and at our worst, stood by us through every challenge, and made these long study years lighter, brighter, and far more enjoyable.
Last but not least, we would like to thank the reason we came this far, our families, who have done everything in their power to help us reach this point. We can never truly repay the people who taught us how to eat, drink, speak, crawl, walk, and keep pushing no matter the circumstances. We could only hope that we could make them proud and show them that their love, patience, and sacrifices were never in vain.

Aeman Abdullah and Mohi Eddin Bilal, Gothenburg, June 2026

List of Acronyms

Below is the list of acronyms used throughout this thesis, listed in alphabetical order:

ACAP	AXIS Camera Application Platform
AMR	Autonomous Mobile Robot
ARCL	Advanced Robotics Command Language
CPS	Cyber-Physical System
DLPU	Deep Learning Processing Unit
DPIA	Data Protection Impact Assessment
GDPR	General Data Protection Regulation
HRC	Human-Robot Collaboration
HRI	Human-Robot Interaction
IANA	Internet Assigned Numbers Authority
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
MQTT	Message Queuing Telemetry Transport
OPC UA	Open Platform Communications Unified Architecture
REST	Representational State Transfer
TCP	Transmission Control Protocol

Contents

List of Figures

List of Tables

1
Introduction

Industrial automation has, over the last decade, been strongly shaped by the goals of Industry 4.0: connectivity, data-driven decision making, and increased automation through cyber-physical systems. More recently, the discussion has expanded toward Industry 5.0, which places stronger emphasis on human-centric production, resilience, and sustainability. In this perspective, the operator is not treated as a source of manual labor to be replaced, but as an expert whose capabilities should be supported through safer collaboration, better situational awareness, and more intuitive interaction with automated systems.

In industrial environments where operators work alongside robots and programmable equipment, interaction methods play a central role. Conventional interfaces such as push buttons, touch screens, and software dashboards remain common, but they are not always convenient when operators move between stations, perform manual tasks, or need to interact with a system without using handheld devices. A practical Industry 5.0-oriented goal is therefore to enable interaction that is simple, reliable, and compatible with natural operator behavior.

Gesture-based interaction is one possible approach. If a system can interpret a small set of clearly defined body postures, operators can issue commands or confirmations without carrying a device or navigating menus. For industrial use, the main requirement is not an expressive gesture language, but a compact command vocabulary. Gestures should be easy to perform, visually distinct, and held for a specific duration to reduce unintentional gesture detection during ordinary work.

This thesis investigates a vision-based gesture control approach built on camera-based human pose estimation. The system uses real-time body-joint data to detect a limited vocabulary of static gestures and translate them into discrete command signals. These signals are then integrated into an industrial automation workflow involving an *Autonomous Mobile Robot* (AMR), a conveyor, middleware-based command routing, and a synchronized virtual representation of the physical system. In the implemented setup, gesture commands are processed by a Python-based gesture module and forwarded through the selected command path. Ordinary AMR mission requests and conveyor commands are routed through ThingWorx and Kepware, while selected AMR actions, such as stop, are sent through a direct ARCL command path. Open Platform Communications Unified Architecture (OPC UA) supports industrial data exchange and digital-shadow synchronization.

The thesis therefore focuses not only on gesture recognition, but on the complete gesture-to-action pipeline. This includes gesture definition, hold-time confirmation, command encoding, industrial communication, asset-specific execution logic, and digital-shadow-supported observation. The work is positioned as a feasibility-

oriented study of how gesture-based input can be integrated into a connected manufacturing system.

1.1 Background and problem description

In industrial automation, introducing new interaction methods requires balancing usability, reliability, and integration complexity. Gesture control can reduce interaction friction for common actions such as starting a predefined routine, confirming an operation, adjusting an actuator, or requesting a stop. However, the interaction must remain predictable. A gesture recognition method must therefore handle natural variation in how people move, differences in body proportions, and changes in operator orientation. It must also reduce false detections that could trigger unintended system behavior.

For mobile robots, an additional challenge is command-state management. If an operator stops an AMR while several missions are queued, the system must define what should happen to the interrupted mission and to the remaining queue. A stop command may otherwise be followed by older queued missions that no longer represent the operator's intention. This creates a need for clear behavior after stopping, including whether the system should resume previous missions or clear them and wait for a new command.

Vision-based approaches are attractive because they do not require wearable devices and can provide body-pose information from a monitored work area. At the same time, camera-based systems can be sensitive to occlusion, camera placement, viewing angle, and pose-estimation jitter. These factors can affect the quality of the estimated body-joint data and, in turn, the reliability of gesture classification. In practical deployments, such limitations can be mitigated through system design choices, such as multi-camera layouts, careful placement, conservative gesture definitions, and hold-time confirmation.

A second challenge is system integration. For gesture control to be useful beyond a standalone demonstration, recognized gestures must be represented as clear system signals and connected to the automation stack. Different industrial assets may also require different execution semantics. For example, an AMR mission may need to be queued and executed sequentially, while conveyor commands may need to act immediately as toggle, speed-adjustment, or quick-stop requests. This creates a need for a structured control architecture that separates gesture recognition from asset-specific execution logic.

A third challenge concerns validation and observation. A gesture-based interface should not only be evaluated as an isolated recognition function, but also as part of an end-to-end system. In this thesis, a synchronized virtual representation is used as a digital shadow to support monitoring and repeatable observation of gesture-triggered AMR behavior. The digital shadow is not treated as a full digital twin, but as a system-level observation layer connected to live data from the physical setup.

1.2 Aim and objectives

The aim of this thesis is to develop and evaluate a gesture-based control approach for a connected manufacturing scenario, and to demonstrate how camera-based pose data can be transformed into industrially meaningful command signals for physical assets and digital-shadow-supported observation.

The objectives are to:

- Define a compact vocabulary of static gestures suitable for industrial command input, focusing on clarity, separability, and low risk of accidental activation.
- Implement real-time gesture detection based on 3D body-joint data, using hold-time confirmation to improve robustness.
- Implement an optional robot-facing activation condition for AMR commands to reduce unintended command activation.
- Represent recognized gestures as discrete command meanings that can be transferred to an industrial integration layer or direct AMR command path.
- Integrate the gesture-command pipeline with connected manufacturing assets, including an AMR and a conveyor with different execution semantics.
- Implement AMR command-state handling through a stop-and-replace workflow that clears current and queued mission requests before accepting a new mission command.
- Extend conveyor interaction to include toggle, speed-increase, speed-decrease, and quick-stop commands.
- Use a synchronized virtual representation as a digital shadow for monitoring and repeatable observation of gesture-triggered system behavior.
- Evaluate the implemented workflow with respect to gesture recognition, command delivery, asset execution, physical-virtual observation, and exploratory user feedback.

1.3 Research questions

This thesis is guided by the following research questions:

RQ1: How can a compact gesture vocabulary be designed for industrial command input while reducing ambiguity and the risk of false triggering?

RQ2: How can camera-based 3D body-joint data be used to detect static command gestures in real time under controlled laboratory conditions?

RQ3: How can recognized gestures be encoded and communicated through an industrial integration layer to support asset-specific command execution?

RQ4: How can a digital shadow support monitoring and repeatable observation of gesture-triggered behavior in a connected manufacturing setup?

RQ5: What initial usability impressions do participants report after interacting with the gesture-based control system?

1.4 Scope and delimitations

This work focuses on discrete command gestures rather than continuous motion tracking, dynamic gesture recognition, or sign-language-like interaction. The gesture vocabulary is intentionally limited to reduce cognitive load, simplify interpretation, and improve reliability under controlled conditions.
The thesis emphasizes feasibility and system integration in an industrial laboratory context. It does not aim to provide a universal benchmark across all possible factory layouts, camera configurations, or operator populations. Instead, it studies transferable design principles, including compact gesture selection, hold-time confirmation, command encoding, middleware-based communication, asset-specific execution logic, and digital-shadow-supported observation.
The implemented system is not intended as a safety-rated control system. The AMR stop function, the robot-facing activation condition, and the conveyor quick-stop function are treated as application-level control and robustness mechanisms rather than certified safety functions. Similarly, the user feedback collected in this work is used as exploratory input and does not support definitive conclusions about user experience, user acceptance, mental workload, or long-term usability.
These delimitations are intentional because the main contribution of the thesis is a realistic feasibility demonstration and integration blueprint for gesture-based control in a connected manufacturing setting.

1.5 Thesis Outline

Chapter 2 presents the theoretical background relevant to the thesis, including Industry 5.0, human-centric manufacturing, gesture-based human-robot interaction, pose estimation, industrial communication, and digital shadows.
Chapter 3 describes the system design and methodology. This includes the physical and virtual setup, the gesture vocabulary, the pose-based gesture detection logic, the robot-facing activation condition, the AMR stop-and-replace workflow, conveyor speed adjustment, and the command-routing paths used for the connected assets.
Chapter 4 presents the evaluation results. The chapter covers gesture recognition and command delivery, AMR and conveyor execution, physical and virtual observation through the digital shadow, exploratory user feedback, and a summary of the main findings.
Chapter 5 discusses the results in relation to feasibility, privacy and GDPR, diverse operator needs, vulnerable user groups, broader technological impact, scalability, and future development directions.
Finally, Chapter 6 concludes the thesis by summarizing the main findings and contributions.

2

Theory and Background

The purpose of this chapter is to provide a technical and conceptual foundation for the work presented in this thesis. It introduces key concepts from industrial automation, human–robot interaction, vision-based perception, and system integration that are necessary to understand the design and implementation choices made in later chapters. The chapter avoids detailed discussion of specific tools and platforms; instead, it focuses on general principles and established frameworks that underlie the methods used in this work. Each section is grounded in research literature so that theoretical claims are supported by peer-reviewed evidence where appropriate.

2.1 Industry 4.0 and the Shift Toward Industry 5.0

The term Industry 4.0 is commonly used to describe the integration of digital technologies with industrial production. A core idea is that physical systems such as machines, robots, and production lines are connected to digital systems that can monitor states, exchange data, and support decision-making in near real time. This is often discussed in terms of *cyber-physical systems* (CPS), where computation and communication are tightly linked to physical processes.

Industry 4.0 is also described through practical principles that guide implementation. Hermann et al. identify six design principles: interoperability, virtualization, decentralization, real-time capability, service orientation, and modularity [1]. Together, these principles describe production environments where connectivity and data exchange enable higher levels of automation, flexibility, and reconfigurability.

More recently, the discussion has broadened toward Industry 5.0. In the European Commission's framing, Industry 5.0 does not replace Industry 4.0; it complements it by emphasizing societal priorities alongside technological progress. The Commission highlights three themes: human-centricity, sustainability, and resilience [2]. In this view, technology should not only increase efficiency, but also support people in their work, improve working conditions, and contribute to robust and responsible industrial systems.

For this thesis, the Industry 5.0 perspective is relevant because it motivates interaction methods that are practical for operators in real work settings. Gesture-based command input is an example of a human-centric interface, since it can enable hands-free and direct interaction with automated systems. At the same time, Industry 4.0 concepts remain important as the technical basis for integration, data exchange, and simulation-supported testing. Thus, the work is positioned as an Industry 5.0 interaction layer built on Industry 4.0 connectivity and integration principles.

2.2 Human-Robot Interaction and Collaboration in Industrial Environments

Human-robot interaction (HRI) concerns how humans and robots exchange information and actions in order to complete tasks. In industrial settings, this interaction is shaped by production requirements, safety constraints, and the need for predictable system behavior. Interfaces are therefore not only a usability topic; they influence how easily a robot can be instructed, how clearly the robot communicates its state, and how reliably humans and robots can work in the same environment [3].

A related concept is *human-robot collaboration* (HRC). Collaboration is typically used when humans and robots share a workspace or jointly contribute to a task, rather than operating in fully separated areas. Compared to traditional industrial robotics, where safety is often achieved by strict separation through fences or guarding, collaborative applications aim to combine the strengths of both parties: the robot can provide precision and repeatability, while the human contributes flexibility and problem solving [4].

For collaborative operation, two requirements are repeatedly emphasized in the literature: safety and intuitive interaction. Villani et al. describe safety as a fundamental prerequisite and highlight the need for interaction methods that reduce the effort required to instruct and adapt robot behavior in industrial tasks [4]. Similarly, Berg and Lu review industrial HRI interfaces and note that research increasingly considers more natural interaction modalities, including speech and gestures, often as part of multimodal interfaces [3].

Safety for industrial robot systems is addressed by international standards. The ISO 10218 series specifies safety requirements for industrial robots and for robot applications and robot cells, covering both robot design aspects and integration into industrial environments [5], [6]. For collaborative applications specifically, ISO/TS 15066 provides additional safety requirements and guidance for collaborative industrial robot systems and their work environment, supplementing ISO 10218 [7]. In practice, this standards framework supports a structured approach where hazards are identified and risk reduction measures are selected before humans and robots are allowed to share space.

Within this context, gesture-based interfaces can be viewed as one candidate interaction channel for industrial HRI and HRC. The goal is typically not to provide a rich command language, but to enable a small set of clear, low-effort commands and confirmations that fit industrial work. Although the present work is not limited to direct human-robot collaboration, HRI and HRC provide useful foundations for understanding gesture-based command input in shared industrial environments. This thesis builds on that motivation by investigating pose-based gesture detection as an operator input method, and by studying how such input can be integrated into a broader human-system automation workflow.

2.3 Gesture-Based Interaction for Human-Robot Interaction

Gestures are a form of non-verbal communication where meaning is conveyed through body motion, often using the hands and arms. In human-robot interaction (HRI), gestures are commonly used as an input channel for discrete commands, confirmations, or requests, especially when hands-free interaction is beneficial. A key advantage of gesture-based control is that it can be designed to match natural operator behavior while still producing clear, machine-readable signals [8].

Gesture interfaces are often described using two practical distinctions. First, gestures can be static, where a posture is held for a short time, or dynamic, where meaning is conveyed through a motion pattern over time. Second, a system can target different parts of the body, such as hand gestures, arm gestures, or full-body gestures, depending on the task and the sensing setup. In industrial environments, the aim is often not to recognize a large and expressive gesture language, but to detect a small command vocabulary with low risk of confusion [9]. In this thesis, the term gesture refers primarily to upper-body postures rather than fine hand or finger gestures.

From a recognition perspective, gesture detection is typically handled as a processing pipeline. Qi et al. describe vision-based gesture recognition as a sequence of steps that includes data acquisition, gesture detection and segmentation, feature extraction, and classification [10]. A common high-level distinction is between sensor-based approaches, such as gloves or wearable sensors, and vision-based approaches using cameras [10]. Vision-based methods are attractive in many settings because they can operate without wearable devices and allow non-contact interaction. At the same time, they depend on the quality of visual perception and are influenced by factors such as occlusion and camera placement. These properties motivate conservative design choices, including robust decision logic and careful gesture selection.

A practical design issue is the choice of gesture vocabulary. Wachs et al. emphasize that gesture systems should be designed around the intended application and that recognition performance depends not only on the algorithm, but also on how gestures are chosen and represented [9]. For industrial command input, this supports using gestures that are easy to perform, clearly separated, and unlikely to occur unintentionally during normal work.

Gesture-based interaction has also been explored for industrial robot programming and assistance. For example, Tsarouchi et al. demonstrated a gesture vocabulary for directing robot motion during programming tasks using visual sensing [11]. Such work supports the relevance of gesture-based input as a practical interface concept in manufacturing contexts, provided that the gesture set and recognition logic are designed for reliability.

2.4 Pose Estimation and Skeleton Representations

Human pose estimation is the task of locating anatomical keypoints, referred to as joints in this thesis, on the human body in images or video. The output is commonly

represented as a skeleton, where each joint has an estimated position and often a confidence score. This representation is useful because it reduces a visual scene to a compact set of coordinates that can be processed as structured signals.
Pose representations depend on the chosen keypoint layout. Many computer vision datasets define a fixed set of joints to enable consistent evaluation and comparison between methods. A widely used example is the person keypoint annotation scheme in the Microsoft COCO dataset, which provides a standard set of body keypoints [12]. Using a fixed layout is important for gesture recognition, because it keeps the joint indices consistent across frames and subjects.
A second aspect is the dimensionality of the pose. In *2D pose estimation*, each joint is represented by image coordinates (x, y). In *3D pose estimation*, each joint is represented in three dimensions (x, y, z) in a chosen coordinate frame. 3D pose can be obtained through several approaches, including multi-view camera setups, depth sensing, or inference methods that estimate 3D structure from 2D observations. Human3.6M is a widely used benchmark dataset for 3D human sensing and provides large-scale paired image and 3D pose data for evaluation [13].
In multi-person scenes, pose estimation also requires separating keypoints that belong to different individuals. This is commonly handled either by detecting each person first and then estimating joints, or by detecting all joints and then associating them. OpenPose is a well-known example of a multi-person 2D pose method that estimates body keypoints and their associations in a bottom-up manner [14]. While the specific algorithms are not central to this thesis, these works illustrate how pose estimation outputs can be treated as structured signals for downstream tasks.
For gesture recognition, skeleton data can be processed frame by frame, while temporal information can be used to improve stability. For example, a posture may be accepted only if it is held for a minimum duration. Since operators vary in body size and stance, gesture logic often benefits from relative joint relations rather than absolute positions. Examples include distances and angles between joints, or joint coordinates normalized by a reference body length. This reduces sensitivity to camera placement and individual body proportions, and supports more consistent gesture definitions across users. For broader context on 3D pose estimation methods and datasets, a recent review is provided by Wang et al. [15]. In this thesis, pose estimation is treated as an input modality that enables rule-based recognition of static arm postures, where a gesture is detected when predefined joint relations are satisfied and held for a minimum duration.

2.5 From Skeleton Signals to Gesture Decisions

Once a human pose has been estimated, gesture recognition can be viewed as a decision problem: given skeleton joints, determine whether a gesture is present and, if so, which one. In the literature, gestures are often divided into static gestures, where the meaning is represented by a posture, and dynamic gestures, where the meaning is represented by a motion pattern over time [8]. This distinction matters because it influences both the representation and the decision logic.
A common first step is to transform raw joint coordinates into features that are more

stable across users and viewpoints. Rather than using absolute positions, many approaches rely on relative relations between joints, such as distances, angles, or normalized coordinates with respect to a reference body length [10]. These relative measures reduce sensitivity to body size differences and support clearer and more transferable gesture definitions.
For static command gestures, recognition can be formulated as verifying that a posture satisfies predefined conditions. In vision-based gesture recognition pipelines, this corresponds to detection and segmentation followed by a classification or decision stage [10]. A practical way to improve reliability is to use *temporal confirmation*, where a posture is accepted as a gesture only if the relevant conditions remain satisfied for a minimum duration. This reduces false detections caused by short-lived pose fluctuations and supports conservative command triggering.
An alternative to rule-based posture checks is a learning-based approach, where gesture classes are inferred from data using classifiers trained on examples. Learning-based methods are often used for dynamic gestures and full-body actions, since they can model temporal patterns and variations in how people perform movements [16]. However, these methods typically require representative labelled datasets and careful validation in the intended application environment.
In this thesis, gestures are treated as static arm postures. Temporal information is used primarily to confirm that a posture is held for a sufficient duration, rather than to model a motion pattern. The specific gesture definitions and decision rules are presented in the Method chapter.

2.6 Digital Twins and Simulation-Supported Development

A *digital twin* is commonly described as a digital representation of a physical entity that is kept aligned with the physical entity through data connections. Standardization efforts reflect this view by providing common terminology and concepts for digital twins across domains [17].
In manufacturing, digital twins are discussed as a way to connect production assets and processes to digital models that support monitoring, analysis, and improvement. Lu reviews digital twin-driven smart manufacturing and presents a reference model that emphasizes the link between the physical system, its digital representation, and the data and communication mechanisms that connect them [18]. This framing is useful in industrial contexts because it clarifies that a digital twin is not only a simulation model, but also depends on systematic data acquisition, information modeling, and updates between the physical and digital sides.
A common source of confusion is the difference between an offline simulation model and a digital twin. Kritzinger et al. propose a practical classification based on data integration level: a *digital model* has no automated data exchange with the physical system; a *digital shadow* has an automated one-way link from the physical system to the digital representation; and a digital twin has automated data exchange in both directions [19]. This distinction helps describe the intended capabilities of a system and the degree of synchronization that can realistically be achieved.

Digital twins are closely related to *cyber-physical systems* (CPS), since both concepts rely on tight coupling between physical processes and cyber components. Tao et al. compare CPS and digital twins and describe them as overlapping but not identical concepts, with digital twins providing a practical path toward cyber-physical integration in smart manufacturing [20]. In this context, a digital twin can be seen as a structured way to represent and synchronize a manufacturing element so that it can be monitored, analyzed, and, when relevant, influenced through feedback.
A central reason for using digital representations in manufacturing is to support development and evaluation through simulation. When a digital representation is linked to a real system, it becomes possible to test changes more safely, run repeatable experiments, and evaluate system behavior under controlled conditions before deployment. In this thesis, the simulation-supported perspective is used through a digital shadow that supports system-level testing of gesture-driven commands and their effects, without treating the digital representation itself as the primary contribution.

2.7 Communication Concepts for Industrial Integration

Industrial automation systems often combine equipment and software from different vendors. To exchange data reliably, a system needs agreed ways to represent values and agreed ways to transmit them. Two common communication patterns are *client-server* data access, where a client requests values from a server, and *publish-subscribe* messaging, where senders publish messages and receivers subscribe to them.
A practical concept used in many industrial systems is *tag-based* data exchange. A tag represents a named process value, such as a sensor reading, a state flag, or a setpoint. Tag-based exchange is useful because it makes integration explicit: each value has a defined meaning, data type, and update behavior. Depending on the interface, clients may read and write tags on demand, or they may subscribe to changes so that updates are delivered automatically when values change.
OPC UA is a widely used industrial standard for structured data exchange. It defines an information model where data and functions are represented in an address space consisting of typed nodes, such as variables and methods. This supports not only reading and writing values, but also describing the meaning and structure of the exposed data. OPC UA also defines client-server services and mechanisms for subscriptions, which allow clients to receive updates when monitored values change [21]. OPC UA is standardized in the IEC 62541 series, where Part 1 provides the overall concepts and overview [22].
Security is a central requirement in industrial communication. OPC UA specifies a security architecture and objectives such as authentication, authorization, confidentiality, integrity, and auditability [23]. In the IEC 62541 series, Part 2 describes the OPC UA security model and the security threats and features considered by the standard [24]. In practice, these concepts support secure deployment through secure channels, certificates, and access control policies, while keeping the communication model consistent across systems.

Message Queuing Telemetry Transport (MQTT) is a lightweight publish-subscribe messaging protocol standardized by the *Organization for the Advancement of Structured Information Standards* (OASIS). MQTT is broker-based: clients publish messages to named *topics*, and other clients subscribe to topics to receive relevant messages [25], [26]. The protocol is designed to be simple and efficient, and it is commonly used for event-style messaging and command distribution. Unlike OPC UA, MQTT does not define a domain information model; the application defines the message payload and its meaning.

In many industrial architectures, these approaches are complementary. Client-server data access and structured information modeling are useful for exposing process states and parameters, while publish-subscribe messaging is useful for distributing events and commands. This thesis relies on these general concepts to integrate gesture-derived commands as discrete system signals and to support system-level testing using simulation.

3
Method

3.1 Method overview and workflow

This thesis follows an engineering-oriented approach in which a functional gesture-based control system is designed, implemented, integrated into a laboratory-scale automation environment, and evaluated through practical tests. The work results in an artifact consisting of a gesture detection module based on camera-based pose estimation, a communication path that forwards recognized gesture commands to industrial control interfaces, and a digital-shadow setup used to support system-level observation and evaluation. The method is reported to make the development process reproducible and to motivate the main design choices.

The work was carried out in three main parts:

1. **Gesture detection:** A Python-based module was developed to recognize a compact vocabulary of static body postures. The gesture vocabulary includes single-arm gestures that can be interpreted either as arm-independent commands or as right- and left-arm variants when additional command functionality is required. Gestures are accepted only when the posture conditions remain valid for a predefined hold time. This hold-time requirement reduces false triggers caused by short-lived pose fluctuations or incidental movements.
2. **Command forwarding and control integration:** When a gesture is recognized, it is mapped to a command action according to the selected control mode. Conveyor commands and ordinary AMR mission requests are routed through ThingWorx and Kepware. Selected AMR actions, such as stop, are sent through a direct ARCL command path. This allows the same gesture layer to support different command semantics for different assets.
3. **Digital-shadow-supported observation:** A virtual representation of the OMRON LD-250 AMR was implemented in Emulate3D. In the final implementation, the AMR motion was represented using a kinematic planar joint approach, where live position and heading values from the physical AMR were mapped to planar translation and rotation in the virtual environment. The virtual setup is therefore treated as a digital shadow rather than a full digital twin, since it supports monitoring and repeatable observation but does not provide behaviorally equivalent simulation of the AMR.

The overall workflow used in the project can be summarized as follows:

1. Define the gesture vocabulary and command meanings.
2. Implement pose-based posture checks and temporal hold-time confirmation.
3. Map accepted gestures to command actions according to the selected control

mode.
4. Route ordinary AMR mission requests and conveyor commands through the industrial integration layer.
5. Send selected AMR actions, such as stop, through the direct ARCL command path.
6. Execute AMR and conveyor commands according to their asset-specific execution semantics.
7. Use the digital shadow to observe gesture-triggered AMR behavior.
8. Validate gesture-level performance, command delivery, asset execution, and physical and virtual observation through practical experiments.

The following sections describe the system architecture, hardware setup, gesture detection implementation, command-routing logic, digital-shadow-supported observation, and evaluation procedure in more detail.

3.2 System architecture

This thesis uses a layered architecture that connects human gesture input to industrial command execution and digital-shadow-supported observation. The purpose of the architecture is to convert camera-based body-joint data into discrete command signals and route them to connected manufacturing assets in a controlled and traceable way.

Figure 3.1 summarizes the overall information flow. The architecture can be understood through three main parts. The first part is the interaction and perception layer, where the operator is monitored through a camera-based pose-estimation system and a Python gesture module evaluates static posture rules. The second part is the industrial integration and execution layer, where recognized gestures are mapped to command actions and forwarded through the selected command path. Ordinary AMR mission requests and conveyor commands are routed through ThingWorx, Kepware, and OPC UA, while selected AMR actions, such as stop, are sent through a direct ARCL command path. The third part is the digital-shadow layer, where selected AMR state values are mirrored in Emulate3D for monitoring and repeatable observation.

The AMR and conveyor use different execution semantics. Ordinary AMR commands are treated as mission requests and are executed sequentially to avoid overlapping missions. Conveyor commands are handled as immediate actuator requests, including toggle, speed increase, speed decrease, and quick-stop functions. In parallel, selected AMR state values are used to update a virtual AMR representation in Emulate3D through planar translation and heading updates, creating a digital shadow for monitoring and repeatable observation.

In the final implementation, the AMR control path was divided into two command types. Ordinary mission commands were routed through the industrial integration layer as mission requests. Immediate AMR actions, such as stopping the robot, were sent through a direct *Advanced Robotics Command Language* (ARCL) command path. This separation was introduced because some AMR actions need to interrupt the current behavior, while ordinary missions can be handled as sequential task requests.

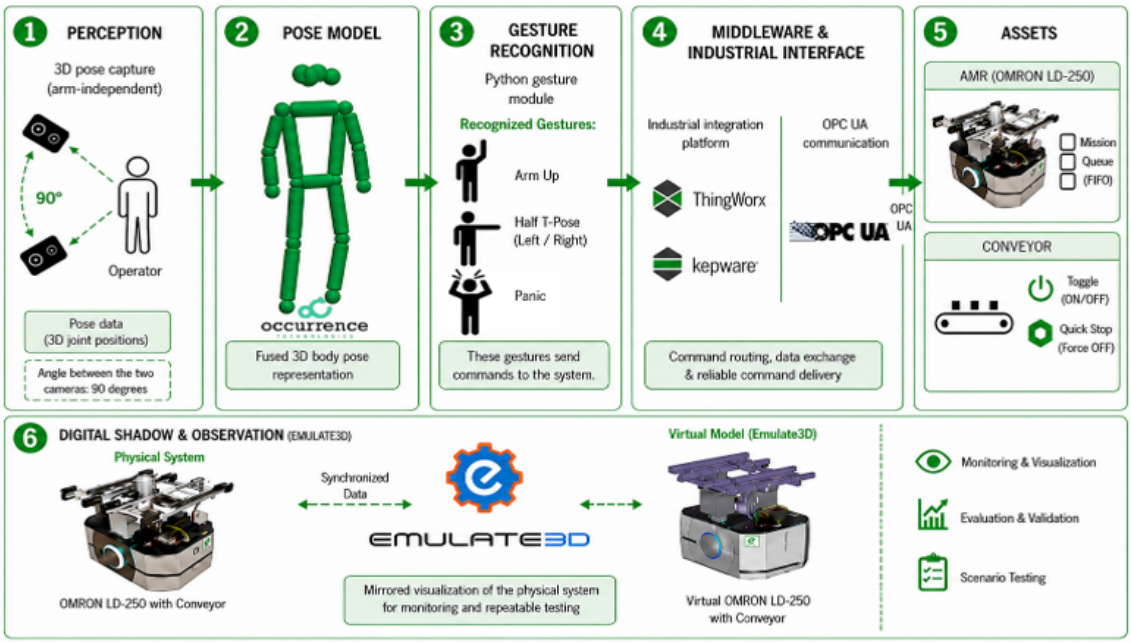


Figure 3.1: Overall architecture and information flow of the gesture-based control system.

The architecture also included an optional robot-facing activation condition for AMR control. When this function was enabled, an AMR-related gesture command was accepted only if the operator was estimated to be facing the robot. This was implemented as a software-level robustness measure to reduce unintended command activation. It should not be interpreted as a certified safety function.
Overall, the architecture separates gesture interpretation from asset-specific execution logic. This allows the operator-facing interaction to remain simple, while the underlying system handles command routing, communication, execution, and observation according to the requirements of each connected asset.
The subsequent sections describe the hardware setup and the gesture detection implementation in more detail.

3.3 Hardware setup

This section describes the physical and virtual equipment used in the laboratory setup and the parts of the environment that are relevant for data collection, system integration, and digital-shadow-supported observation. The goal is to provide enough detail to understand the measurement conditions and the interfaces available to the software components.

3.3.1 Camera configuration

The gesture detection system relies on a camera-based pose estimation setup that provides 3D joint positions for a single person inside a defined working area. The laboratory setup used two Axis M3085-V cameras mounted in fixed positions around

the operator area, with an angular separation of approximately 90 degrees. The Axis M3085-V is a fixed 2 MP mini dome camera with support for deep-learning analytics on the edge [27]. The cameras remained stationary throughout all experiments.
The pose estimation system outputs a skeleton representation per time step, where each joint is represented by (x, y, z) coordinates in a consistent coordinate frame. The following parameters describe the camera setup used in the experimental setup:

- Camera model: Axis M3085-V.
- Capture rate: 20 Hz.
- Output joints used for gesture detection: all joint arrays are ordered as Nose, LeftEye, RightEye, LeftEar, RightEar, LeftShoulder, RightShoulder, LeftElbow, RightElbow, LeftHand, RightHand, LeftHip, RightHip, LeftKnee, RightKnee, LeftFoot, and RightFoot.

The camera model is relevant not only as a sensing component, but also from a deployment and privacy perspective. The M3085-V includes a deep learning processing unit that supports analytics on the edge [27]. In future deployments, such edge-processing capability could reduce the need to transmit raw video through the system, supporting a metadata-oriented architecture. Axis also provides privacy-oriented functions such as AXIS Live Privacy Shield, which can apply dynamic privacy masking to live and recorded video [28].
This thesis treats pose estimation as an input modality and focuses on the downstream posture checks, temporal hold-time logic, and system integration.

3.3.2 OMRON LD-250 AMR platform

Robot missions were executed on an OMRON LD-250 autonomous mobile robot (AMR). The LD-250 is designed for indoor material transport and supports integration through standard interfaces and safety features. Relevant specifications for this thesis include the maximum payload (250 kg), maximum speed (1200 mm/s), and typical runtime (approximately 13 h without payload and 10 h with full payload) [29]. The robot includes a safety scanning laser (Class 1) with a wide field of view and additional safety and sensing elements intended for operation in shared environments [29].
In this work, the robot is treated as the physical execution platform for gesture-triggered commands. Mission definitions and mission execution logic were provided as part of the existing system setup. The thesis focuses on generating and delivering command signals rather than modifying the internal mission logic.

3.3.3 Conveyor system interface

A conveyor was included as a secondary controlled asset. Unlike AMR missions, conveyor actions do not require sequential mission scheduling. Conveyor commands were therefore handled as immediate actuator requests. In the implemented system, the conveyor supported four command actions: toggle, speed increase, speed decrease, and quick-stop. The toggle command changed the conveyor state, the speed commands adjusted the conveyor speed in predefined steps, and the quick-stop command forced the conveyor into the OFF state.

The conveyor was controlled through the existing laboratory integration layer. From the perspective of the gesture module, conveyor commands were represented as software requests sent to ThingWorx and then forwarded through the industrial communication setup. The implementation did not modify the conveyor controller or any safety-rated hardware.
The quick-stop function was implemented as an application-level control action within the experimental setup. It should not be interpreted as a certified emergency stop or as a replacement for safety-rated hardware.

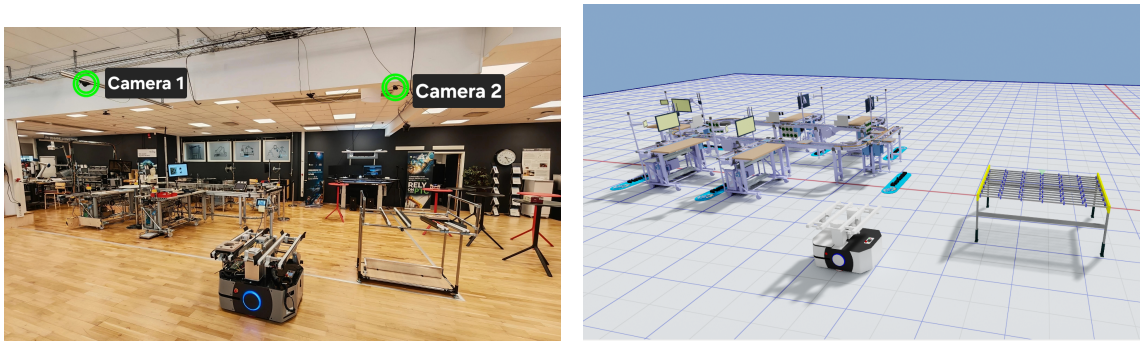
3.3.4 Network infrastructure

The system components were connected through the laboratory network. The gesture detection computer received pose data through MQTT and sent accepted gesture commands through the selected command path. Conveyor commands and ordinary AMR mission requests were routed through ThingWorx and Kepware, while selected AMR actions, such as stop, were sent through a direct ARCL command path.
For OPC UA communication, the endpoint is typically provided through the `opc.tcp` transport mapping, and the well-known service name `opcua-tcp` is registered by the Internet Assigned Numbers Authority (IANA) on TCP port 4840 [30]. In the laboratory setup, communication was configured within the local network and restricted to the devices required for the experiments.
Specific IP addresses, credentials, and deployment-specific endpoint values are not reported in the thesis, since they depend on the local laboratory configuration.
Table 3.1 summarizes the key hardware components that form the experimental platform.

Table 3.1: Summary of the hardware components used in the experimental setup.

Component	Role in this thesis
Camera-based pose system	Provides 3D joint positions for the operator in the monitored interaction area.
Gesture processing computer	Runs the Python gesture detection module, command mapping logic, and selected communication clients.
OMRON LD-250 AMR	Executes gesture-triggered robot missions and selected direct ARCL commands [29].
Conveyor system	Executes toggle, speed increase, speed decrease, and quick-stop commands.
Network equipment	Provides local connectivity between the pose system, gesture processing computer, industrial integration layer, AMR, conveyor, and digital-shadow setup.

Figure 3.2 shows the physical laboratory setup and the corresponding Emulate3D setup used for digital-shadow-supported observation. The virtual AMR was connected to live AMR state values and represented using a kinematic planar joint structure to reproduce planar translation and heading changes.



(a) Physical laboratory setup, including the AMR, drone-factory workstation, conveyor, operator area, and camera arrangement.

(b) Virtual setup in Emulate3D used for digital-shadow-supported observation.

Figure 3.2: Physical and virtual setups used in the gesture-based control system.

3.4 Gesture detection implementation

This section describes how pose data is converted into discrete gesture commands. The implementation processes a live pose stream and applies rule-based posture checks with a minimum hold time. Gestures are treated as *static postures*; temporal information is used only to confirm that a posture is held long enough to be accepted.

3.4.1 Gesture vocabulary design

The implemented gesture vocabulary was intentionally kept small to reduce cognitive load and lower the risk of false triggering. The vocabulary consisted of three static posture types: an arm-raised posture, a half T-pose, and a two-hand quick-stop posture. The single-arm postures could be interpreted either as arm-independent commands or as right- and left-arm variants, depending on the command function.

The gesture set was selected according to four main criteria. First, gestures should be easy for operators to perform. Second, gestures should be visually distinct in the body-joint data. Third, gestures should be unlikely to occur unintentionally during ordinary work. Fourth, the vocabulary should support both simple command input and limited command expansion through right- and left-arm variants.

Table 3.2: Gesture vocabulary used in this thesis and the corresponding command meanings in AMR and conveyor mode.

Label	Gesture	Hold time	AMR mode	Conveyor mode
G1R	Right arm raised vertically upward.	3 s	Send dock command to the AMR.	Toggle the conveyor ON or OFF.
G1L	Left arm raised vertically upward.	3 s	Send undock command to the AMR.	Toggle the conveyor ON or OFF.
G2R	Half T-pose with the right arm extended sideways.	3 s	Send Mission 1 command to the AMR.	Increase the conveyor speed.
G2L	Half T-pose with the left arm extended sideways.	3 s	Send Mission 3 command to the AMR.	Decrease the conveyor speed.
G3	Two-hand stop posture, where both hands are held close to the head.	1 s	Send stop command to the AMR.	Send quick-stop command to the conveyor.

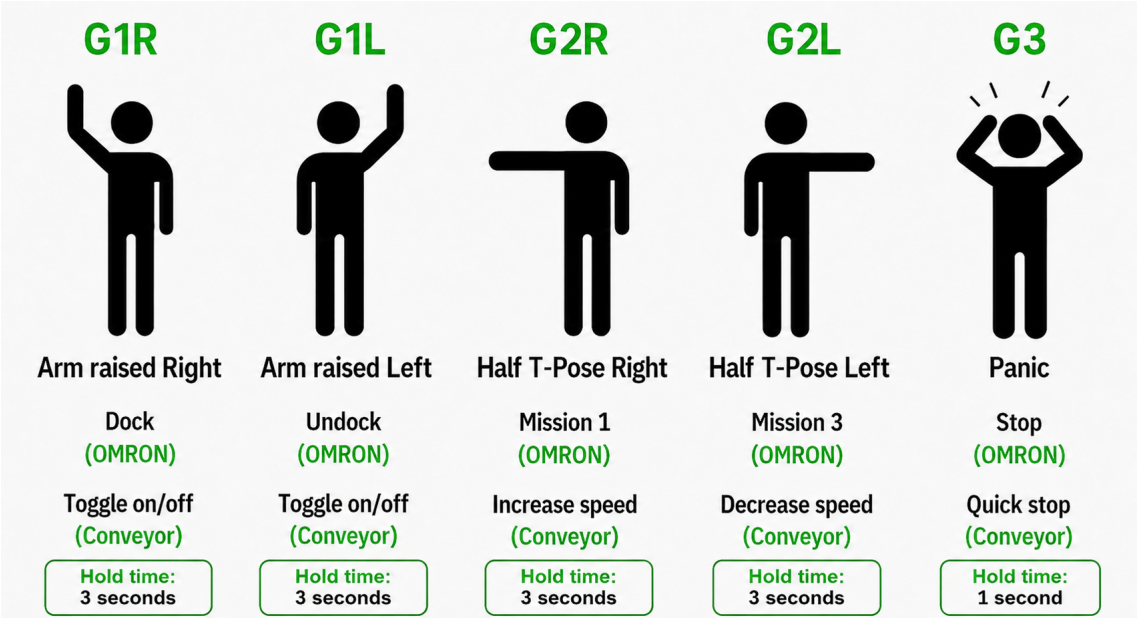


Figure 3.3: Implemented gesture vocabulary and associated command actions.

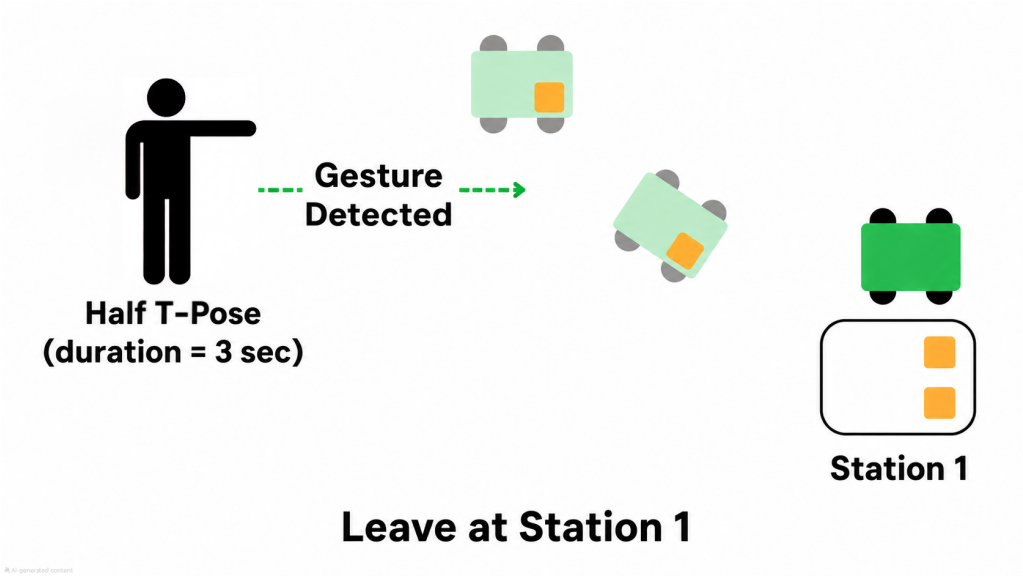
3.4.2 Gesture encoding strategy

Each accepted gesture was mapped to a command action. The command action depended on the selected control mode. In AMR mode, gestures were interpreted as robot mission commands or direct AMR control commands. In conveyor mode, the same gesture vocabulary was interpreted as conveyor actuator commands. This allowed the gesture recognition layer to remain unchanged while the command interpretation changed depending on the controlled asset.
The formal mapping between gestures, hold times, and command actions is summarized in Table 3.2. Figure 3.3 is included as a visual aid for the posture vocabulary.

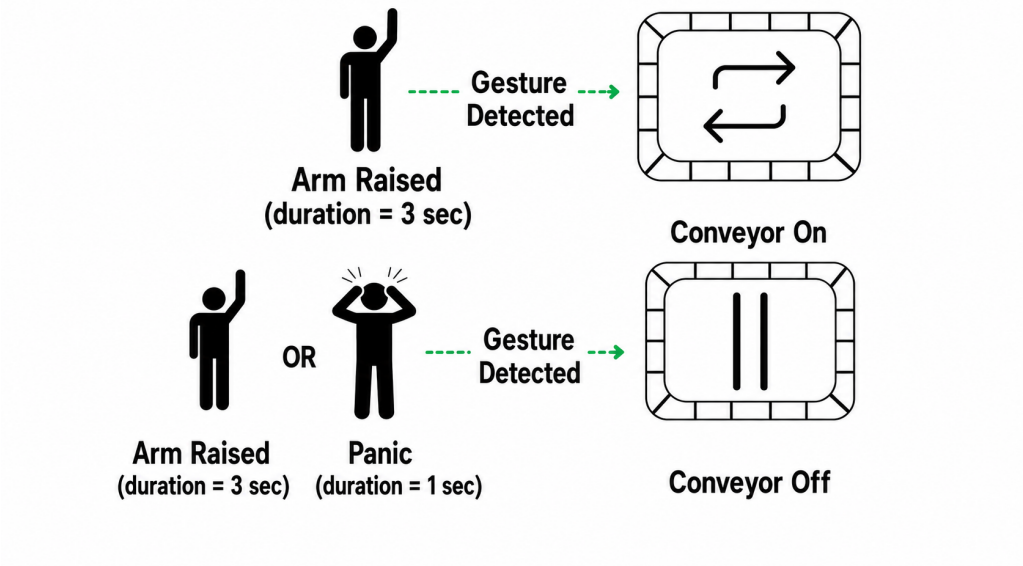
Commands were emitted using a send-on-trigger principle. A command was sent once when a gesture was accepted, rather than continuously while the posture was held. This reduced repeated command transmission and made the command events easier to log and interpret.

3.4.3 AMR stop-and-replace workflow

The AMR stop function was implemented as a stop-and-replace workflow rather than as a resume function. When the stop gesture was accepted in AMR mode, the system sent a direct ARCL stop command to the AMR and cleared the mission queue through the integration layer. This removed both the current mission request and pending mission requests from the command sequence.
After the stop command, the system waited for a new valid AMR mission gesture. The next accepted mission gesture was treated as a replacement command, not as a continuation of the previous queue. This behavior was introduced to avoid the situation where the AMR is stopped, receives a new mission, and then continues executing older queued missions that may no longer match the operator's intention. This workflow changes the meaning of the stop gesture from "pause and later resume" to "stop, clear, and wait for a new command". The function is therefore a command-management mechanism for the experimental setup. It is not a replacement for safety-rated stop functions or certified emergency-stop hardware.
Examples of the AMR and conveyor command logic are shown in Figure 3.4. The AMR example illustrates gesture-triggered mission execution, while the conveyor example illustrates toggle and quick-stop behavior.



(a) Example of gesture-triggered AMR mission execution.



(b) Example of gesture-triggered conveyor toggle and quick-stop behavior.

Figure 3.4: Examples of gesture-triggered command execution.

3.4.4 Pose data input and preprocessing

The gesture detection module processes a live pose stream containing 3D body-joint data. In the implemented setup, the pose stream was received by the Python module through MQTT, using the Eclipse Paho MQTT client. This MQTT link was used for pose-data input only. After a gesture was detected, the resulting command action was forwarded through the selected command path, as described in Section 3.2.
Each message can contain one or more detected bodies. The script supports both of the following formats:

- a dictionary with a `bodies` field, or
- a list of body entries directly.

For each body entry, the script iterates through `fused_positions`. For each fused position, the pose is read from `pose_3d`. Frames with fewer than 17 keypoints are ignored. A person identifier is extracted from `id_1` when available and used to maintain per-person gesture timing [31].
Before rule evaluation, the following checks are applied:

- **Minimum keypoints:** frames with fewer than 17 keypoints are skipped.
- **Required joints:** if joints required by a gesture rule are missing, that gesture evaluation is skipped for the current frame.

3.4.5 Feature computation and normalization

Gesture rules are based on relative joint relations rather than absolute positions. This supports more consistent behavior across different users and working distances. In the implementation, posture checks rely on distances and relative positions derived from the 3D keypoints (for example wrist position relative to shoulder or head-related joints), combined with normalization where needed. The detailed thresholds and helper functions are defined in the gesture detection module [31].

3.4.6 Robot-facing activation condition

An optional robot-facing activation condition was implemented for AMR control. When this function was enabled, an AMR-related gesture command was accepted only if the operator was estimated to be facing the robot. The purpose was to reduce accidental command activation when the operator performed similar body postures while facing away from the AMR or interacting with another part of the workstation.
The activation condition used the operator's nose and ear positions together with the AMR position. Since the human pose data and the robot position were originally expressed in different coordinate frames, the robot position was first transformed into the pose-estimation coordinate frame. The comparison was then performed in the horizontal x-z plane.
Let N denote the nose position, E_L and E_R the left and right ear positions, and R the transformed robot position. The operator was treated as facing the robot when the nose was closer to the robot than both ears:

$$d_{xz}(N, R) < d_{xz}(E_L, R) \quad \text{and} \quad d_{xz}(N, R) < d_{xz}(E_R, R). \tag{3.1}$$

where $d_{xz}(\cdot,\cdot)$ is the Euclidean distance in the x-z plane. If this condition was not satisfied, the detected gesture was ignored for AMR control. The condition was implemented as an optional software-level activation gate and should not be interpreted as a certified safety function.

3.4.7 Rule-based gesture definitions

The gesture detector uses deterministic posture rules based on the 3D skeleton joints. The vertical axis is denoted by y, while horizontal distances are computed in the x-z plane. Let S_y denote the average shoulder height, computed from the left and right shoulder joints when both are available. If only one shoulder is available, the available shoulder height is used. A gesture rule is evaluated only when the joints required for that rule are available in the current frame.
Table 3.3 summarizes the implemented posture conditions. The rules were intentionally conservative, since the goal was to recognize a compact command vocabulary rather than a large set of natural gestures.

Table 3.3: Rule-based posture conditions used for gesture detection.

Label	Gesture	Main posture conditions
G1R	Right arm raised	Right wrist and elbow above shoulder height; wrist higher than elbow; wrist at least 0.08 m above elbow; wrist not close to the right ear.
G1L	Left arm raised	Same as G1R, but applied to the left wrist, left elbow, and left ear.
G2R	Right half T-pose	Right wrist and elbow near shoulder height, within ± 0.15 m; right wrist horizontally extended at least 0.60 shoulder widths, minimum 0.20 m; right elbow extended at least 0.30 shoulder widths, minimum 0.10 m.
G2L	Left half T-pose	Same as G2R, but applied to the left wrist, left elbow, and left shoulder.
G3	Quick-stop posture	Both wrists close to the corresponding ears; right wrist within 0.03 m of the right ear and left wrist within 0.03 m of the left ear.

3.4.8 Temporal filtering mechanism

To reduce false triggers from short-lived pose fluctuations, each gesture had to be held for a minimum duration before it was accepted. The hold time was gesture-dependent:

$$T_{\min}(g) = \begin{cases} 3 \text{ s}, & g \in \{G1R, G1L, G2R, G2L\}, \\ 1 \text{ s}, & g = G3. \end{cases} \tag{3.2}$$

A gesture was accepted only when the corresponding detector returned a positive trigger after the required hold time. This mechanism prevented transient posture matches from being interpreted as valid commands.
In the implementation, timing is handled inside the gesture detector functions, which receive an elapsed-time value and return both the detection result and the measured duration for the posture [31]. A gesture is considered accepted only when the detector returns `True` (trigger event).

3.4.9 Robustness measures

The implementation included several measures intended to improve stability during practical operation:

- **Hold-time confirmation:** Gestures were accepted only after the posture had been maintained for the required duration.
- **Input validation:** Frames with insufficient or missing keypoints were discarded before gesture evaluation.
- **Relative joint checks:** Gesture rules were based on relative body-joint relations rather than absolute image positions, reducing sensitivity to operator position and body size.
- **Arm-side handling:** Single-arm gestures could be detected for both arms. The detected side was used either for traceability or as an additional command parameter.
- **Robot-facing activation:** For AMR control, an optional activation condition required the operator to be facing the robot before a gesture command was accepted. This reduced the likelihood of unintended AMR commands caused by similar body postures during other tasks.
- **Send-on-trigger command emission:** Commands were sent once when a gesture was accepted, rather than repeatedly while the gesture remained active.
- **Stop-and-replace handling:** After an AMR stop gesture, current and queued AMR mission requests were cleared before a new mission command could be issued.
- **Quick-stop prioritization:** The two-hand quick-stop posture used a shorter hold time to allow faster conveyor shutdown through software control.

3.4.10 Software implementation

The live processing was implemented as an event-driven Python loop. The pose-stream client received each payload, parsed the JSON message, and passed the extracted body-joint data to the gesture-processing function. For each detected person, the implementation checked whether the required joints were available and then evaluated the gesture detectors.
Gesture evaluation was performed by calling dedicated detector functions for the arm-raised posture, half T-pose variants, and two-hand quick-stop posture. Each detector returned whether the gesture had been accepted, the measured hold duration, and the detected arm side when applicable.

When a trigger occurred, the software first checked the selected control mode. In conveyor mode, the command was forwarded as a conveyor request through ThingWorx. In AMR mode, ordinary mission commands were forwarded as mission requests, while immediate commands such as stop were sent through the direct ARCL command path. If the robot-facing activation condition was enabled, the AMR command was accepted only when the operator was estimated to be facing the robot.
Each event was logged with timestamp, detected gesture, command action, arm side when applicable, hold duration, and command result. These logs were used to support documentation and later analysis of the experiments.
A simplified pseudocode representation of the runtime loop is shown below:

```
on pose message:
    parse JSON payload
    for each detected person:
        keypoints = extract 3D body joints
        if required joints are missing:
            continue

        for each gesture detector:
            detected, duration, arm_side = detect_gesture(keypoints,
                person_id)

            if detected:
                command = gesture_command_mapping[gesture_name,
                    control_mode]

                if control_mode == "AMR":
                    if robot_facing_required:
                        if not operator_is_facing_robot(keypoints,
                            robot_position):
                            log ignored gesture
                            continue

                    if command == "stop":
                        send_direct_arcl_stop()
                        clear_current_and_queued_missions()
                    else:
                        send_amr_mission(command)

                elif control_mode == "CONVEYOR":
                    send_conveyor_command(command)

                log event(time, gesture_name, command, arm_side, duration)
```

4

Results

This chapter presents the results from the implemented gesture-based control system. The evaluation focuses on the complete gesture-to-action pipeline rather than on gesture recognition in isolation. The main evaluation dimensions are gesture recognition and command delivery, asset execution, physical and virtual observation through the digital shadow, and exploratory user feedback.

The results should be interpreted as a feasibility evaluation under controlled laboratory conditions. The evaluation does not aim to provide a statistically generalizable usability study or a universal benchmark for all industrial environments.

4.1 Evaluation procedure

The system was evaluated through controlled trials in the laboratory setup described in Section 3.3. Each trial started with the operator performing one of the implemented gestures inside the monitored interaction area. A trial was counted as valid when the gesture was held for the required confirmation time, mapped to the intended command action, transferred through the selected command path, and followed by the expected system response.

The evaluation covered the implemented gesture-command mappings for the AMR and conveyor. AMR-related commands were evaluated as ordinary mission requests, direct stop commands, and stop-and-replace behavior after a stop gesture. Conveyor-related commands were evaluated as immediate actuator commands, including toggle, speed increase, speed decrease, and quick-stop. In addition, the digital shadow was observed to assess whether the virtual representation reflected the physical AMR behavior at the scenario level during gesture-triggered execution.

4.2 Gesture recognition and command delivery

The first part of the evaluation focused on whether the gesture detection module recognized the implemented gesture vocabulary and whether accepted gestures were translated into the intended command actions. Table 4.1 summarizes the overall gesture and command-delivery results.

Table 4.1: Overall gesture recognition and command-delivery results.

Metric	Value	Total trials	Notes
Gesture recognition success rate	100%	50	–
False-trigger count	0	50	–
Correct command mapping rate	100%	50	–
Command delivery success rate	100%	50	–

The implemented gesture vocabulary was recognized in the controlled trials. No false triggers were observed, and all accepted gestures were mapped to the intended command actions. This suggests that the combination of static posture rules and hold-time confirmation was suitable for the evaluated laboratory setup.

The result should not be interpreted as evidence that the recognition logic is robust in all industrial environments. The tests were performed under controlled laboratory conditions, with known camera placement and a predefined interaction area. However, the result supports the use of a compact gesture vocabulary as a command input method when the sensing setup and interaction context are controlled.

4.3 Asset execution results

The second part of the evaluation focused on the response of the connected physical assets. Since the AMR and conveyor use different execution semantics, the evaluation considered whether each asset executed the intended action after receiving a valid gesture-derived command.

The AMR received both ordinary mission requests and direct stop commands. Ordinary AMR missions were handled through sequential execution. The stop gesture was handled through the stop-and-replace workflow, where the AMR was stopped and the current and queued mission requests were cleared before a new mission command was accepted. The conveyor received immediate actuator commands, including toggle, speed increase, speed decrease, and quick-stop commands. Table 4.2 summarizes the observed asset execution results.

Table 4.2: Asset execution results for the AMR and conveyor.

Metric	Value	Total trials	Notes
AMR mission execution success rate	100%	50	–
AMR response time	~ 1 s	50	Consistent; mainly dominated by the configured middleware safety delay.
Conveyor command execution success rate	100%	50	Includes toggle, speed increase, speed decrease, and quick-stop commands.
Conveyor response time	~ 1 s	50	Typical observed response time. Occasional larger delays were observed, but the cause has not yet been isolated.

The AMR response time remained close to 1 s across the evaluated trials. This delay was mainly associated with the configured middleware safety delay and was therefore expected. The AMR mission execution behavior was consistent, and all valid gesture-derived AMR mission commands resulted in the expected mission execution.

The stop-and-replace workflow also behaved as intended during implementation testing. After the stop gesture was accepted, the AMR stop command was issued and the mission queue was cleared. A later mission gesture was then treated as a new command rather than as a resume command. This made the AMR behavior more predictable after a stop, since earlier queued missions were not executed after the replacement command.

The conveyor executed all valid commands, including toggle, speed increase, speed decrease, and quick-stop. In the updated tests, the conveyor response time was typically around 1 s, which is comparable to the AMR response time observed in the same setup. However, occasional larger delays were still observed. The cause of these delayed responses has not yet been isolated. For this reason, conveyor timing should be interpreted as generally responsive in the tested setup, but not fully characterized.

Overall, the asset execution results indicate that the gesture-to-command pipeline worked in the tested laboratory setup. The results also show why asset-specific execution logic is important. AMR commands are naturally handled as sequential mission requests or direct stop commands, whereas conveyor commands are better represented as immediate actuator requests.

4.4 Physical and virtual observation results

The third part of the evaluation focused on the digital shadow. The purpose of the digital shadow was not to control the physical system, but to support monitoring and repeatable observation of the AMR behavior during gesture-triggered execution. In the final implementation, the virtual AMR was represented in Emulate3D using a kinematic planar joint approach. Live position and heading values from the physical AMR were mapped to planar translation and rotation in the virtual environment. The physical AMR position values were converted from millimetres to metres before being applied to the virtual model, while the heading value was used to update the AMR orientation. The virtual motion was defined relative to the initial AMR position in Emulate3D, which was set to $(0, 0, 0)$. Therefore, the virtual AMR had to start from this reference position before each observation sequence so that the relative movement matched the physical AMR trajectory and the robot could reach the corresponding workstation position accurately inside the Emulate3D environment. Table 4.3 summarizes the observed agreement between the physical AMR behavior and the virtual representation.

Table 4.3: Physical and virtual observation results for the digital shadow.

Metric	Value	Total trials	Notes
Scenario agreement rate	100%	50	–
Scenario-level mismatch count	0	50	No mismatch in the observed scenario outcome.
Observed digital-shadow delay	~ 1 s	50	Consistent with the live update cycle and synchronization offset.
Virtual-motion consistency	Successful	50	The planar joint representation supported consistent observation of AMR translation and heading when initialized from the defined reference position.

The digital shadow consistently reflected the physical AMR behavior at the scenario level. The observed delay was approximately 1 s, which is consistent with the implemented update cycle and synchronization offset. This value is therefore reported as an observed system characteristic rather than as a precise synchronization-lag measurement.

The kinematic planar joint approach provided a structured way to represent AMR motion in Emulate3D. Since the physical AMR operates mainly on a planar floor surface, representing the virtual AMR through planar translation and heading was suitable for observation of gesture-triggered missions. The accuracy of the virtual

motion depended on initializing the virtual AMR from the defined reference pose, currently $(0, 0, 0)$, because the applied movement was relative to this starting position. When this initialization was maintained, the virtual model followed the physical AMR behavior sufficiently for monitoring and repeatable observation during the evaluated scenarios.
Although the digital shadow matched the physical AMR behavior at the scenario level, it should still not be interpreted as a behaviorally equivalent digital twin. The virtual representation mirrored selected live AMR state values and supported observation of gesture-triggered behavior, but it did not provide predictive simulation, dynamic validation, or full physical equivalence with the real AMR.

4.5 Exploratory user feedback

To complement the technical evaluation, a short post-test user survey was conducted after participants interacted with the system. The survey was intended to provide initial feedback on perceived usability, intuitiveness, reliability, safety, and usefulness. Because the number of participants was limited, the survey is treated as an exploratory complement to the technical validation rather than as a basis for broad conclusions about user experience, user acceptance, or mental workload.
Overall, the feedback indicated that participants were able to understand and perform the gesture-based interaction after limited instruction. The system was generally perceived as useful for simple command input. At the same time, the feedback also pointed to practical concerns that would need further investigation in future work, including occasional response delays, clearer system feedback, and the risk of false detections in more complex environments.
The user feedback therefore provides an initial indication that the interaction concept was understandable under controlled conditions. However, stronger claims about usability, workload, accessibility, or long-term operator acceptance would require a larger and more diverse participant group, as well as dedicated evaluation methods.

4.6 Summary of results

The results show that the implemented system connected gesture recognition, command mapping, industrial communication, asset-specific execution, and digital-shadow-supported observation in one end-to-end workflow. Across the controlled trials, the system recognized the tested gestures, mapped them to the intended command actions, delivered the commands through the selected command path, and triggered the expected asset behavior.
The main findings are summarized as follows:

- The implemented gesture vocabulary was recognized under controlled laboratory conditions.
- No false triggers were observed during the controlled trials.
- Accepted gestures were mapped to their intended command actions and delivered through the selected command path.

- AMR mission commands were executed as sequential mission requests.
- The AMR stop-and-replace workflow cleared current and queued mission requests before a new mission was issued.
- Conveyor commands were executed in the tested cases, including toggle, speed increase, speed decrease, and quick-stop.
- Conveyor response timing was typically around 1 s, although occasional larger delays were still observed for reasons that have not yet been isolated.
- The digital shadow supported scenario-level observation of AMR using kinematic planar joint representation, with an observed delay of approximately 1 s.
- Exploratory user feedback indicated that the interaction concept was understandable, but broader conclusions about usability or acceptance require further user studies.

Taken together, the results indicate that the proposed gesture-based control approach worked in the laboratory-scale connected manufacturing setup. The findings also highlight important limitations, especially regarding occasional unexplained delays, sensing conditions, and the need for broader user validation.

5

Discussion and Future Work

This chapter discusses the results of the thesis in relation to the broader aims of human-centric manufacturing. It also reflects on privacy, GDPR, accessibility, vulnerable user groups, scalability, and future development directions. The discussion should be understood in relation to the controlled laboratory scope of the work. The implemented system was developed as a feasibility-oriented prototype and integration blueprint, not as a production-ready or safety-rated industrial control system.

5.1 Interpretation of the results

The results indicate that the implemented gesture-based control approach worked under the tested laboratory conditions. The system connected pose-based gesture recognition, command interpretation, industrial communication, physical asset execution, and digital-shadow-supported observation in one workflow. The main contribution is therefore not only the recognition of gestures, but the integration of gesture input into an industrial command structure.

The reported success rates should be interpreted carefully. The trials were carried out in a known laboratory environment with fixed camera positions, a limited gesture vocabulary, and a predefined interaction area. These conditions are useful for evaluating the concept, but they do not represent the full range of variation that could occur in an industrial workplace. Lighting conditions, occlusions, operator position, body proportions, and work tasks could affect the pose data and therefore the gesture detection result.

The optional robot-facing activation condition added context to the command decision. Instead of accepting an AMR command based only on body posture, the system also checked whether the operator appeared to be oriented toward the AMR. This reduces the likelihood of unintended AMR commands when similar postures occur during other activities. However, the function depends on pose-estimation quality, coordinate transformation between the robot and camera frames, and stable detection of the head joints. It should therefore be treated as a software-level robustness measure, not as a certified safety mechanism.

The stop-and-replace workflow made AMR behavior easier to interpret after a stop command. Without this function, queued missions could remain active after the robot was stopped, which could make the later behavior unclear to the operator. By clearing the current and queued mission requests, the next gesture command is treated as a new operator command. This is a practical improvement for the experimental setup, but it also shows that gesture-based control requires careful

command-state management. The meaning of a gesture depends not only on the posture itself, but also on the current state of the controlled asset.
The conveyor results show that the same gesture layer can support both binary commands and simple adjustment commands. Toggle and quick-stop commands change the conveyor state, while the speed increase and speed decrease commands modify the conveyor behavior in steps. In the updated tests, the conveyor response time was typically around 1 s. This indicates that the conveyor command path can be responsive under the tested laboratory conditions. However, occasional larger delays were still observed, and their cause has not yet been isolated. This shows that the full system behavior is affected not only by the gesture detector, but also by the controlled asset, the communication path, and the state of the integration layer.
The digital shadow supported observation of gesture-triggered AMR behavior at the scenario level. In the final implementation, the virtual AMR was represented using a kinematic planar joint approach in Emulate3D, where selected live position and heading values from the physical AMR were mapped to planar translation and rotation. This was suitable for the evaluated laboratory scenario because the AMR operated on a flat floor and the main observation requirement was to follow its scenario-level motion. However, the setup should still not be described as a full digital twin. The virtual representation mirrored selected live data from the physical AMR, but it did not provide behaviorally equivalent simulation or predictive validation.

5.2 Privacy, GDPR, and data governance

Because the system uses camera-based pose estimation, privacy and data protection are central deployment considerations. Even if the implemented system focuses on body-joint metadata rather than storing raw video, camera-based sensing can still involve personal data when individuals are identifiable or when the data can be linked to a person. For deployment in a European industrial context, the system would therefore need to be assessed in relation to the General Data Protection Regulation (GDPR), including principles such as lawfulness, fairness, transparency, purpose limitation, data minimization, storage limitation, integrity, and confidentiality [32]. The architecture used in this thesis is aligned with a data-minimization direction because the gesture module operates on pose-related metadata rather than relying on continuous human interpretation of raw video. However, this does not remove the need for GDPR analysis. A practical deployment would still need to define the legal basis for processing, the purpose of the processing, what data is stored, how long it is retained, who can access it, and how operators are informed about the system. If the system is used for monitoring workers or influencing workplace decisions, additional assessment and safeguards would be required.
For deployments involving systematic camera-based monitoring of workers, a Data Protection Impact Assessment (DPIA) may also be required before the system is introduced.
The camera hardware is relevant for this discussion. The Axis M3085-V is a fixed 2 MP mini dome camera with support for deep-learning analytics on the edge [27].

Its deep learning processing unit can support analytics directly on the device, which can reduce the need to transmit raw video through the system. The camera ecosystem also supports privacy-related functions such as AXIS Live Privacy Shield, which provides dynamic privacy masking for live and recorded video [28]. Such functionality is relevant for future deployment because it supports the idea of processing only the data needed for the interaction task.

For future industrial use, data protection should be considered from the design stage rather than added after implementation. This is consistent with the GDPR principle of data protection by design and by default [33]. In practice, this would mean limiting raw video access, processing data locally where possible, avoiding unnecessary storage, anonymizing or pseudonymizing data where appropriate, documenting the processing purpose, and ensuring that workers receive clear information about what the system does and does not record.

5.3 Diverse needs and vulnerable user groups

A human-centric interaction method should not assume that all operators have the same physical abilities, working conditions, or comfort with camera-based systems. The implemented gesture vocabulary was intentionally compact and based on upper-body postures, but it still assumes that the operator can raise or extend the arms and hold a posture for a defined duration. This may not be suitable for all users.

Vulnerable groups may include operators with reduced mobility, fatigue, temporary injuries, chronic pain, neurodivergent interaction preferences, or discomfort with being observed by camera systems. The system may also affect operators differently depending on height, body proportions, clothing, protective equipment, cultural norms, or the physical layout of the workstation. These differences matter because an interaction method that is efficient for one group may be difficult, tiring, or exclusionary for another.

Future versions should therefore support alternative interaction paths. Examples include configurable gestures, shorter or adjustable hold times, voice input, button-based fallback commands, wearable confirmation devices, or workstation-specific interaction profiles. A gesture interface should be treated as one possible input channel rather than the only way to control an asset. This would make the system more inclusive and reduce the risk that operators are forced into an interaction style that does not fit their needs.

The exploratory user feedback in this thesis gives only an initial indication of understandability. It does not provide enough evidence to draw conclusions about accessibility, workload, long-term comfort, or acceptance across diverse operator groups. A broader study should therefore include participants with different physical characteristics, levels of experience, and accessibility needs.

5.4 Broader impact of gesture-based industrial control

Gesture-based control can have a significant impact on how operators interact with automated systems. In a positive scenario, it can reduce interaction friction, make commands easier to issue while moving between stations, and support more natural human-system interaction. It can also make some industrial assets more accessible by reducing dependence on dashboards or handheld devices.
However, the same technology also introduces risks. Camera-based interaction systems can be perceived as surveillance if their purpose, data handling, and limitations are not clearly communicated. If gesture data or camera data were used beyond the intended control task, for example for productivity monitoring or individual performance evaluation, this could affect trust and workplace acceptance. The social and organizational effects of the technology are therefore as important as the technical implementation.
The technology also changes the relationship between human intention and machine action. A physical gesture can become an industrial command, which means that ambiguity, false triggering, delayed feedback, or unclear system state can have practical consequences. This makes feedback design, command confirmation, target selection, and stop handling important parts of responsible deployment.
The implementation in this thesis should therefore be understood as a controlled feasibility study. It shows that gesture input can be connected to industrial command execution, but it also shows that responsible deployment requires more than recognition accuracy. It requires privacy safeguards, inclusive design, clear operator feedback, documented limitations, and integration with existing safety and supervisory procedures.

5.5 Scalability and deployment potential

The architecture has potential for scalability because it separates gesture recognition from asset-specific execution logic. The same gesture detection layer can be mapped to different command meanings depending on the selected control mode, while the underlying industrial integration layer handles communication with specific assets. This separation is useful because a gesture interface should not need to be redesigned completely for every machine.
At the same time, scaling the system to a larger manufacturing environment would introduce new challenges. A larger setup may contain several robots, conveyors, workstations, and operators. The system would then need reliable target selection, operator identification or authorization, conflict handling between simultaneous commands, and clear feedback about which asset is being controlled. Without these mechanisms, a gesture could be ambiguous or could affect the wrong asset.
Scalability also depends on sensing coverage. The current two-camera laboratory setup was suitable for a limited interaction area, but a larger environment would require careful placement of cameras, calibration, occlusion handling, and network configuration. Edge processing, such as analytics on the camera or local process-

ing close to the workstation, could reduce bandwidth requirements and support privacy-preserving deployment. However, this would need to be balanced against maintainability, cost, cybersecurity, and system validation requirements.
A scalable implementation would also require standardized configuration methods. Gesture mappings, controlled assets, safety restrictions, operator permissions, and feedback mechanisms should be configurable rather than hard-coded. This would make it easier to adapt the system to new workstations while keeping the interaction logic consistent and traceable.

5.6 Future work

Future work should first evaluate the proposed blueprint in broader and more realistic manufacturing contexts. The present study was conducted in a laboratory-scale setup with a limited number of controlled trials and five participants. Further studies should include larger and more diverse user groups, longer interaction sessions, and more varied workstation conditions. This would make it possible to evaluate usability, cognitive load, accessibility, operator acceptance, and long-term reliability more systematically.
A second direction is to improve the sensing setup. The current two-camera arrangement was sufficient for the implemented gesture vocabulary under controlled conditions, but it constrained the operator's positioning and rotation within the interaction area. Additional or better-positioned cameras could reduce occlusion, improve pose stability, and allow more natural operator movement. Workstation-level camera placement could also support a broader gesture vocabulary, including finer hand or finger-based gestures if supported by the pose estimation system.
A third direction is to extend the gesture recognition approach. The current implementation uses rule-based detection of static upper-body postures. This was suitable for a compact and interpretable command vocabulary, but it limits the system to predefined posture conditions. Future systems could investigate data-driven or hybrid recognition methods to support dynamic gestures, more natural movement patterns, and greater variation in how different users perform gestures. Such methods would require representative training and validation data from the intended industrial context.
Future work should also address personalization and inclusive design. The current vocabulary assumes that operators can perform a limited set of upper-body gestures and hold them for a specified duration. Future versions could allow operators to personalize gestures, reduce hold-time requirements, or use alternative input modalities such as voice, head-based gestures, or foot-based commands. This would improve accessibility for operators with different physical abilities, movement patterns, or working conditions.
Another important direction is target selection, authorization, and safety integration. In more complex manufacturing environments, an operator may need to choose between several nearby machines or issue commands only when authorized to do so. Future work should therefore investigate mechanisms for selecting the intended target asset, verifying operator authorization, and integrating gesture-triggered commands with existing safety procedures, access control, and supervisory logic. The

robot-facing activation condition used in this thesis is a first software-level step toward context-aware command acceptance, but it is not sufficient for deployment in safety-critical environments. This is especially important because the current system is designed for command interaction and is not a safety-rated control system.
Finally, the digital-shadow component could be further developed. In this thesis, the Emulate3D setup supported monitoring and repeatable observation using a kinematic planar joint representation of the AMR motion, but it did not function as a behaviorally equivalent digital twin. Future work could improve the fidelity of the virtual AMR model, reduce synchronization delay, include more detailed AMR motion and navigation behavior, and extend the virtual environment to support predictive validation, scenario testing, and more advanced physical and virtual analysis.

6
Conclusion

This thesis developed and evaluated a gesture-based control approach for a connected manufacturing scenario. The work focused on transforming camera-based 3D body-joint data into discrete command signals that could be routed through an industrial integration layer and executed by connected assets. The implemented use case included an OMRON LD-250 Autonomous Mobile Robot (AMR), a conveyor, middleware-based command routing through ThingWorx and Kepware, and a synchronized virtual representation in Emulate3D used as a digital shadow.

The final implementation also included two command-management functions for AMR control. First, an optional robot-facing activation condition required the operator to be estimated as facing the AMR before AMR-related gesture commands were accepted. Second, the stop-and-replace workflow cleared the current and queued AMR mission requests after a stop gesture, so that the next mission gesture represented a new operator command rather than a resume of the previous queue. These functions improved the clarity of the interaction logic in the laboratory setup, but they should be understood as software-level control mechanisms rather than certified safety functions.

The results indicate that the proposed gesture-to-action pipeline worked under controlled laboratory conditions. The implemented gesture vocabulary was recognized in the controlled trials, accepted gestures were mapped to the correct command meanings, and command delivery through the selected command path worked as intended. The results also showed that the same interaction layer could support different asset-specific execution semantics. AMR commands were handled as sequential mission requests or direct stop commands, while conveyor commands were handled as immediate actuator requests, including toggle, speed increase, speed decrease, and quick-stop functions. This indicates that gesture-based input can be separated from asset-specific execution logic and integrated into a broader automation workflow.

The digital shadow supported monitoring and repeatable observation of gesture-triggered AMR behavior. The virtual representation reflected the physical AMR behavior at the scenario level, with an observed delay of approximately 1 s. However, the virtual setup should be understood as a digital shadow rather than a full digital twin, since it mirrored selected live data from the physical system but did not provide behaviorally equivalent simulation or predictive validation.

The exploratory user feedback provided an initial indication that the interaction concept was understandable. Participants were able to perform the gesture-based interaction after limited instruction, and the system was generally perceived as useful for simple command input. At the same time, the small number of participants

means that the survey results should be interpreted as preliminary indications rather than as broad conclusions about usability, workload, accessibility, or long-term acceptance.
Overall, the thesis shows that a compact vocabulary of static gestures, combined with hold-time confirmation, command mapping, middleware-based integration, direct AMR command handling, and digital-shadow-supported observation, can provide a practical control layer for selected manufacturing commands. The main contribution is the integration blueprint connecting human gesture input, industrial communication, asset-specific command execution, AMR command-state handling, conveyor speed adjustment, and physical and virtual observation. Within the scope of the laboratory implementation, the work indicates that gesture-based control can be used as a complementary human-centric interaction layer for connected manufacturing systems.

Bibliography

[1] M. Hermann, T. Pentek, and B. Otto, "Design principles for Industrie 4.0 scenarios," in *Proceedings of the 49th Hawaii International Conference on System Sciences (HICSS)*, IEEE, 2016, pp. 3928–3937. DOI: 10.1109/HICSS.2016.488 (cit. on p. 5).

[2] M. Breque, L. De Nul, and A. Petridis, "Industry 5.0: Towards a sustainable, human-centric and resilient European industry," European Commission, Directorate-General for Research and Innovation, Luxembourg, Tech. Rep. KI-BD-20-021-EN-N, 2021. DOI: 10.2777/308407 (cit. on p. 5).

[3] J. Berg and S. Lu, "Review of interfaces for industrial human-robot interaction," *Current Robotics Reports*, vol. 1, no. 2, pp. 27–34, 2020. DOI: 10.1007/s43154-020-00005-6 (cit. on p. 6).

[4] V. Villani et al., "Survey on human-robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, pp. 248–266, 2018. DOI: 10.1016/j.mechatronics.2018.02.009 (cit. on p. 6).

[5] *ISO 10218-1:2025 robotics — safety requirements — part 1: Industrial robots*, Accessed: 2026-02-25, International Organization for Standardization (ISO), 2025. [Online]. Available: https://www.iso.org/standard/73933.html (cit. on p. 6).

[6] *ISO 10218-2:2025 robotics — safety requirements — part 2: Industrial robot applications and robot cells*, Accessed: 2026-02-25, International Organization for Standardization (ISO), 2025. [Online]. Available: https://www.iso.org/standard/73934.html (cit. on p. 6).

[7] *ISO/TS 15066:2016 robots and robotic devices — collaborative robots*, Accessed: 2026-02-25, International Organization for Standardization (ISO), 2016. [Online]. Available: https://www.iso.org/standard/62996.html (cit. on p. 6).

[8] S. Mitra and T. Acharya, "Gesture recognition: A survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 37, no. 3, pp. 311–324, 2007. DOI: 10.1109/TSMCC.2007.893280 (cit. on pp. 7, 8).

[9] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan, "Vision-based hand-gesture applications," *Communications of the ACM*, vol. 54, no. 2, pp. 60–71, 2011. DOI: 10.1145/1897816.1897838 (cit. on p. 7).

[10] J. Qi et al., "Computer vision-based hand gesture recognition for human-robot interaction: A review," *Complex & Intelligent Systems*, vol. 10, pp. 1581–1606, 2024. DOI: 10.1007/s40747-023-01173-6 (cit. on pp. 7, 9).

[11] P. Tsarouchi et al., "High level robot programming using body and hand gestures," *Procedia CIRP*, vol. 55, pp. 1–5, 2016. DOI: 10.1016/j.procir.2016.09.020 (cit. on p. 7).

[12] T. Lin et al., "Microsoft COCO: Common objects in context," in *Computer Vision – ECCV 2014*, ser. Lecture Notes in Computer Science, vol. 8693, Springer, 2014, pp. 740–755. DOI: 10.1007/978-3-319-10602-1_48 (cit. on p. 8).

[13] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1325–1339, 2014. DOI: 10.1109/TPAMI.2013.248 (cit. on p. 8).

[14] Z. Cao, T. Simon, S. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2017, pp. 1302–1310. DOI: 10.1109/CVPR.2017.143 (cit. on p. 8).

[15] J. Wang et al., "Deep 3D human pose estimation: A review," *Computer Vision and Image Understanding*, vol. 210, p. 103 225, 2021. DOI: 10.1016/j.cviu.2021.103225 (cit. on p. 8).

[16] B. Ren, M. Liu, R. Ding, and H. Liu, "A survey on 3D skeleton-based action recognition using learning method," *Cyborg and Bionic Systems*, vol. 5, p. 0100, 2024. DOI: 10.34133/cbsystems.0100 (cit. on p. 9).

[17] *ISO/IEC 30173:2023 digital twin — concepts and terminology*, Accessed: 2026-02-25, International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), Nov. 2023. [Online]. Available: https://www.iso.org/standard/81442.html (cit. on p. 9).

[18] Y. Lu, C. Liu, K. I. Wang, H. Huang, and X. Xu, "Digital twin-driven smart manufacturing: Connotation, reference model, applications and research issues," *Robotics and Computer-Integrated Manufacturing*, vol. 61, p. 101 837, 2020. DOI: 10.1016/j.rcim.2019.101837 (cit. on p. 9).

[19] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital twin in manufacturing: A categorical literature review and classification," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, 2018. DOI: 10.1016/j.ifacol.2018.08.474 (cit. on p. 9).

[20] F. Tao, Q. Qi, L. Wang, and A. Y. C. Nee, "Digital twins and cyber-physical systems toward smart manufacturing and Industry 4.0: Correlation and comparison," *Engineering*, vol. 5, no. 4, pp. 653–661, 2019. DOI: 10.1016/j.eng.2019.01.014 (cit. on p. 10).

[21] OPC Foundation, *OPC UA part 1: Overview and concepts (OPC 10000-1), online reference*, Accessed: 2026-02-25, 2025. [Online]. Available: https://reference.opcfoundation.org/Core/Part1/v105/docs/6 (cit. on p. 10).

[22] *IEC 62541-1:2025 OPC UA — part 1: Overview and concepts*, Accessed: 2026-02-25, International Electrotechnical Commission (IEC), 2025. [Online]. Available: https://webstore.iec.ch/en/publication/81513 (cit. on p. 10).

[23] OPC Foundation, *OPC UA part 2: Security (OPC 10000-2), online reference*, Accessed: 2026-02-25, 2025. [Online]. Available: `https://reference.opcfoundation.org/Core/Part2/v105/docs/` (cit. on p. 10).

[24] *IEC 62541-2:2026 OPC UA — part 2: Security model*, Accessed: 2026-02-25, International Electrotechnical Commission (IEC), 2026. [Online]. Available: `https://webstore.iec.ch/en/publication/81514` (cit. on p. 10).

[25] *MQTT version 5.0*, Accessed: 2026-02-25, OASIS, 2019. [Online]. Available: `https://www.oasis-open.org/standard/mqtt-v5-0-os/` (cit. on p. 11).

[26] *MQTT version 5.0 specification*, Accessed: 2026-02-25, OASIS, 2019. [Online]. Available: `https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html` (cit. on p. 11).

[27] Axis Communications, *AXIS M3085-V Dome Camera*, Accessed: 2026-05-28, 2026. [Online]. Available: `https://www.axis.com/products/axis-m3085-v` (cit. on pp. 16, 34).

[28] Axis Communications, *AXIS Live Privacy Shield*, Accessed: 2026-05-28, 2026. [Online]. Available: `https://www.axis.com/products/axis-live-privacy-shield` (cit. on pp. 16, 35).

[29] *LD-Series mobile robot datasheet*, Accessed: 2026-02-27, OMRON, n.d. [Online]. Available: `https://files.omron.eu/downloads/latest/datasheet/en/i828_ld-series_mobile_robot_datasheet_en.pdf` (cit. on pp. 16, 17).

[30] *Service name and transport protocol port number registry (opcua-tcp, 4840/tcp)*, Accessed: 2026-02-27, Internet Assigned Numbers Authority (IANA), 2026. [Online]. Available: `https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml?page=33&search=4` (cit. on p. 17).

[31] A. Abdullah and M. E. Bilal, ***gesture_detector_livestream.py*** *(project source code)*, Included as project artifact in thesis repository / appendix, 2026 (cit. on pp. 22, 24).

[32] European Parliament and Council of the European Union, *Regulation (EU) 2016/679: General Data Protection Regulation*, Accessed: 2026-05-28, 2016. [Online]. Available: `https://eur-lex.europa.eu/eli/reg/2016/679/oj/eng` (cit. on p. 34).

[33] European Data Protection Board, *Guidelines 4/2019 on Article 25 Data Protection by Design and by Default*, Accessed: 2026-05-28, 2020. [Online]. Available: `https://www.edpb.europa.eu/sites/default/files/files/file1/edpb_guidelines_201904_dataprotection_by_design_and_by_default_v2.0_en.pdf` (cit. on p. 35).

A

Project source code

The implementation developed for this thesis is organized as a project repository. The most relevant files for the final implementation are listed below.

- `gesture_detector_livestream.py`: main live gesture detection and command-routing script.
- `gestures.py`: rule-based gesture detection functions and geometric helper functions.
- `thingworx.py`: REST-based communication functions for AMR mission handling, queue clearing, and conveyor control.
- `opcua_arcl_gateway.py`: ARCL and OPC UA gateway used for digital-shadow-related integration.
- `emulate3d_planar_joint_logic`: QuickLogic implementation used to map live AMR position and heading values to the kinematic planar joint structure in Emulate3D.
- `print_locations.py`: debugging utility for comparing operator and robot position data.

The source code was used as an implementation artifact in the thesis. Credentials, IP addresses, and local deployment settings should be managed outside the report and should not be included in the printed appendix.

DEPARTMENT OF MECHANICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY