

Percipiote: Towards an AI-based safety system for human-robot collaboration

Daniel Bermuth

Angaben zur Veröffentlichung / Publication details:

Bermuth, Daniel. 2026. "Percipiote: Towards an AI-based safety system for human-robot collaboration." Augsburg: Universität Augsburg.

Nutzungsbedingungen / Terms of use:

CC BY-SA 4.0



Percipiote

Towards an AI-based Safety System for Human-Robot Collaboration

Daniel Bermuth

DISSERTATION
zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)



Percipote: Towards an AI-based Safety System for Human-Robot Collaboration

Advisor: Prof. Dr. Wolfgang Reif

Submission Date: 4 May 2026

Reviewers: Prof. Dr. Wolfgang Reif
Prof. Dr. Bernhard Bauer
Prof. Dr. Alexander Schiendorfer

Defense Date: 7 July 2026

Committee: Prof. Dr. Wolfgang Reif
Prof. Dr. Bernhard Bauer
Prof. Dr. Alexander Schiendorfer
Prof. Dr. Lars Mikelsons

Abstract

Human-robot collaboration (HRC) offers the potential to combine the strengths of both humans and robots in industrial manufacturing. However, ensuring worker safety while maintaining flexible and seamless collaboration remains a central challenge. Existing safety systems are generally limited in this regard, as they rely on strict physical separation, enforce large safety distances, or only react after a collision has already occurred. This fundamentally limits the proximity and productivity of the collaboration.

To address these limitations, this thesis presents the development of an AI-based safety system designed to enable closer and more dynamic human-robot interactions. Based on an external, camera-based sensor approach combined with multi-view 3D human pose estimation, the proposed system operates independently of specific robot hardware. It provides robust, real-time detection and localization of human workers and can explicitly distinguish between individual body parts. By combining tightly fitting dynamic safety zones with designated interaction zones, the system can selectively permit contacts — effectively providing the capability to merge the concepts of speed and separation monitoring with power and force limiting.

The system was developed and evaluated in an iterative process, demonstrating latencies comparable to established, certified safety systems while providing notably higher spatial resolution and contextual information. Although formal industrial certification requires further steps, the thesis outlines concrete suggestions towards achieving this goal and confirms that AI-based pose estimation is a highly promising technology for safety-critical applications. Furthermore, the developed methods can be used to gain a deeper understanding of human behavior, thereby enabling more efficient collaboration, and in combination with a dedicated speech interface, they can also enable more intuitive robot interactions.

Overall, this thesis demonstrates that such AI-driven approaches not only offer practical advantages over traditional distance-based safeguarding, but also provide broadly applicable methods that can benefit a wide range of other domains.

Contents

1	Introduction	1
1.1	Motivation	2
1.2	Existing Safety Systems	3
1.2.1	Stop to Entry	4
1.2.2	Hand Guiding	5
1.2.3	Speed and Separation Monitoring	5
1.2.4	Power and Force Limiting	8
1.3	Goal of this Work	11
1.4	Thesis Overview	13
2	Safety Foundations 1: Robot Systems	15
2.1	Overview	16
2.2	Risk Analysis	18
2.2.1	Limits of machinery	18
2.2.2	Hazard identification	18
2.2.3	Risk estimation	19
2.2.4	Risk evaluation	19
2.2.5	Required Performance Level	21
2.3	System Assessment	23
2.3.1	Specification of Safety Functions	23
2.3.2	Mean Time to Dangerous Failure	24
2.3.3	Diagnostic Coverage	24
2.3.4	Common Cause Failures	24
2.3.5	Systematic Failures	25
2.3.6	Category	25
2.3.7	Performance Level Determination	27
2.3.8	Software Requirements	27
2.3.9	Validation	28
3	KoARob - AI for Safety	29
3.1	Scenario Description	30
3.2	Risk Analysis	33
3.3	Safety Concept	35
3.3.1	Safe Robot Position	35
3.3.2	Safe Human Position	35
3.3.3	Distance Monitoring	36
3.3.4	System architecture	37
3.4	Human Pose Estimation	39
3.4.1	Pose Estimation Foundations	39
3.4.2	skelda	45
3.4.3	Existing Approaches	45
3.4.4	SimpleDepthPose	48

3.4.5	VoxelKeypointFusion	52
3.4.6	PlaneSweepPosePlus	55
3.4.7	Comparison	57
3.5	Scenario Evaluation	64
3.5.1	Pose Estimation	64
3.5.2	System Assessment	66
3.6	Related Work	68
3.7	Conclusion	72
4	Safety Foundations 2: Collaboration	73
4.1	Methods of Collaboration	74
4.2	Speed and Separation Monitoring	74
4.3	Power and Force Limiting	76
5	Tutabo-1: Reacting in Real-time	77
5.1	Revisiting KoARob	78
5.2	Improved Implementation	80
5.2.1	Smart Cameras	80
5.2.2	Optimized Pose Estimation	84
5.2.3	Optimized Safety Calculation	85
5.2.4	System architecture	86
5.3	RapidPoseTriangulation	87
5.3.1	Algorithm	87
5.3.2	Dataset Generalization	91
5.3.3	Whole-body Estimation	98
5.3.4	Conclusion	100
5.4	Evaluation	101
5.4.1	Pose Estimation	101
5.4.2	Robot Stop Time	104
5.4.3	Collaboration Distances & Delays	105
5.5	Related Work	108
5.6	Conclusion	109
6	Safety Foundations 3: Using AI	111
6.1	Legislation	112
6.2	ISO/IEC-TR-5469	112
6.2.1	Classification of AI systems	113
6.2.2	Desirable Properties	113
6.2.3	Validation	113
6.2.4	Proposed Architectures	114
7	Tutabo-2 - Next iteration	115
7.1	Hardware upgrades	116
7.1.1	Smart Cameras	116
7.1.2	Robot	117
7.2	Stop- and Interaction-Zones	120
7.2.1	Stop-Zones	120

7.2.2	Interaction-Zones	122
7.3	Using the proposals from <i>RPT</i>	124
7.4	Evaluation	126
7.4.1	AI standards	126
7.4.2	Collaboration Distances & Delays	127
7.5	Related Work	129
7.6	Outlook	132
7.6.1	Further steps toward certification	132
7.6.2	Force limiting	137
7.6.3	Economic Considerations	138
7.6.4	Ergonomics	140
7.7	Conclusion	141
8	Understanding Human Movements	143
8.1	Skarimva: Human Action Recognition	145
8.1.1	Related Work	145
8.1.2	Improving Pose Estimates	146
8.1.3	Results	147
8.1.4	Conclusion	151
8.2	Scriboora: Human Pose Forecasting	152
8.2.1	Related Work	152
8.2.2	Additional real-time Metrics	153
8.2.3	Evaluation under Unified Protocol	154
8.2.4	Scriboora	155
8.2.5	Noisy Coordinates	157
8.2.6	Conclusion	159
9	Improving Human-Robot Interactions	161
9.1	RoboJack: Guidelines for improving Interactions	163
9.2	Jaco-Assistant: A privacy-aware Voice Interface	166
9.2.1	Jaco-Assistant	166
9.2.2	finstreder	169
9.2.3	slungt	171
9.2.4	Benchmarks	172
9.2.5	Conclusion	174
9.3	Crazyflie Swarms: Beyond Industrial Robots	176
10	Conclusion	177
	Bibliography	181
	List of Figures	199
	List of Tables	201
	Own Publications	203

Summary. After a brief introduction to the topic, commercially available safety systems for human-robot collaboration and their limitations are presented, which motivate the need for this work and lead to the main goals of this thesis.

1

Introduction

1.1	Motivation	2
1.2	Existing Safety Systems	3
1.2.1	Stop to Entry	4
1.2.2	Hand Guiding	5
1.2.3	Speed and Separation Monitoring	5
1.2.4	Power and Force Limiting	8
1.3	Goal of this Work	11
1.4	Thesis Overview	13

Throughout the cycle of industrial revolutions, the automation of production has been a key driver for increasing efficiency and reducing costs. With the third industrial revolution, the use of robots in manufacturing has been widely adopted, leading to significant improvements in productivity. In the ongoing fourth industrial revolution, the focus is on the integration of advanced, connected, and in some cases AI-based technologies into the production process, enabling smarter and more flexible manufacturing. Building on these developments, the concept of a fifth industrial revolution is increasingly discussed, with a focus on sustainability and human-centric approaches [95].

1.1 Motivation

In the broader context of the aforementioned industrial revolutions to increase efficiency, human-robot collaboration (HRC) has emerged as a promising approach to combine the strengths of both humans and robots in manufacturing. While robots can perform repetitive and physically demanding tasks with high precision and speed, humans can provide flexibility, creativity, and problem-solving skills. However, ensuring worker safety is a key challenge in HRC, as the close proximity of humans and robots can lead to potential hazards. Accurate and reliable knowledge of the human worker's position within the collaborative workspace is therefore crucial to ensure safe interactions, while also enabling efficient collaboration by allowing the robot to adapt its behavior accordingly. However, existing approaches for human safety in collaborative environments still face limitations in terms of flexibility or robustness, particularly in dynamic and close human-robot interaction scenarios. Therefore, these limitations will be a central focus of this thesis.

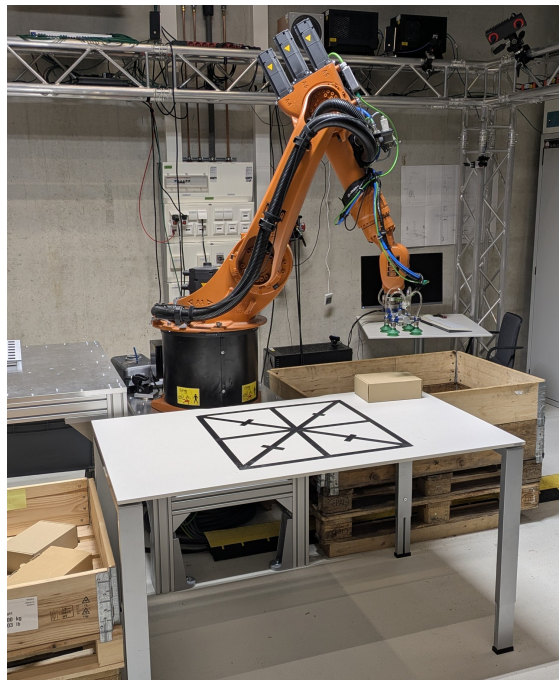


Figure 1.1. A setup in which the robot could support human workers with their tasks.

The term *Percipite* (from Latin, meaning “*I see and understand you*”), which gives this thesis its title, denotes the overarching objective of this work: to develop a system enabling not only co-existence, but effective human-centric collaboration. As a core element towards this objective, this thesis proposes an AI-based safety system for human-robot collaboration that is designed to enable robust, real-time detection and localization of human workers, thereby contributing to closer, more efficient, and ultimately more advanced human-robot interaction, while also aiming to facilitate safe operation.

1.2 Existing Safety Systems

Achieving this objective first requires examining the safety mechanisms that currently enable humans and robots to share workspaces. Industrial robots are widely used in manufacturing, but their high speeds, large payloads, and ability to exert substantial forces can pose serious risks to nearby workers. Therefore, appropriate safety measures are essential to prevent injuries. This section provides an overview of currently available, certified safety systems for human-robot collaboration that provide such protection. The very high reliability requirements for these systems, however, typically result in structural limitations, which are also discussed throughout the section. These limitations then finally motivate the main goals of this thesis.

The easiest way to ensure the safety of human workers is to have a physical separation between them and the robot, so there is no possibility of contact. In most applications, this is achieved by using fences (Figure 1.2). Doors can allow access for maintenance, but must be closed for the robot to operate. In general, this type of setup does not allow any collaboration.



Figure 1.2. Example of a robot system with physical separation by a fence.

According to *EN ISO TS 15066* [112], there are four different methods by which collaboration between robots and humans can be achieved. The first option is an open workspace where the robot has to stop moving before a human can enter the collaborative area. The second is hand-guiding the robot. The third option is speed and separation monitoring, and the fourth is power and force limiting. In the following, examples of systems for those types are given and their limitations are discussed.

1.2.1 Stop to Entry

In the most basic mode, the workspace is openly accessible, but the robot has to cease its motion before a human is allowed to enter the collaborative area (Figure 1.3). One of the most common safety systems in this regard are light curtains, which consist of multiple light beams that are emitted by a transmitter and received by a sensor [213]. When a beam is interrupted, the system detects it and sends a stop signal to the robot. Another simple option is using special safety floor mats, which detect when a person steps on them [8]. Such concepts have the benefit of quick access, but do not allow seamless collaboration.

Instead of large light curtains, it is also possible to equip the end-effector of the robot with a sensor that only protects the area below it [217]. In this case, however, another safety system is required to protect the rest of the robot, so it is not a standalone solution.



Figure 1.3. On the left, a robot system with a virtual separation by a light curtain. On the right-top, an end-of-arm-safeguard system, which protects the area around the robot end-effector. On the right-bottom, a safety floor mat, which detects when a human steps on it. Images from [204], [217], and [8].

1.2.2 Hand Guiding

The second type of collaboration is *hand-guiding*, in which the operator uses a hand-operated device to transmit motion commands to the robot system. In this case, the human has the full responsibility for the safety, and the robot merely executes the commands. The user can either use a hand-held device, which is connected to the robot and allows the user to control it by keys or a joystick, or with some special collaborative robots, the user can directly drag the robot to the desired position (Figure 1.4). While this would allow a very intuitive collaboration, the human operator normally has to control the robot with both hands, and therefore cannot perform any other task at the same time, which limits productivity. Usually this approach is used for programming/teaching the robot, but only rarely for the actual production process. For teaching, the robot is moved to a desired position and the system then records that position. This is repeated for multiple positions, and at the end, the robot controller can connect all positions into a trajectory and execute it automatically.

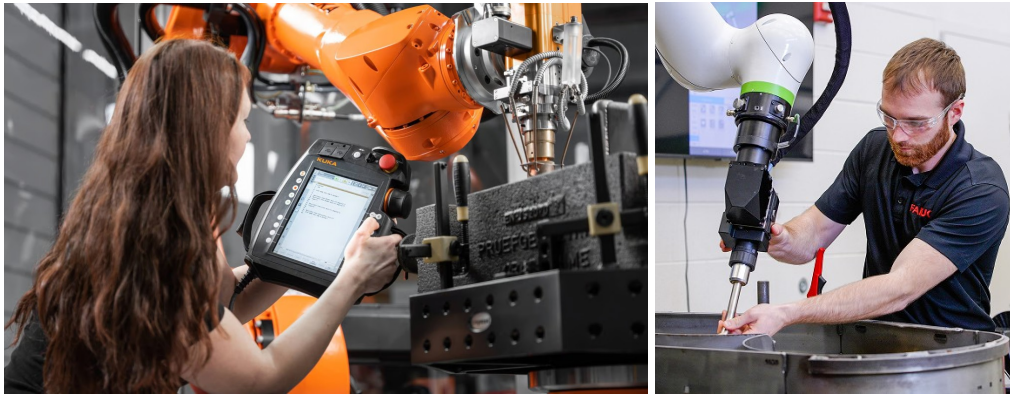


Figure 1.4. Examples of hand-guiding robot movements. On the left, a hand-held device is used to position the robot end-effector with a key-based control, and on the right, the user can directly drag the robot to the desired position. Images from [7] and [6].

1.2.3 Speed and Separation Monitoring

In *Speed and Separation Monitoring*, the robot system must always maintain a minimal protective separation distance to the human operator. If the distance decreases below this limit, the robot must stop. If it increases again, the robot can automatically resume. Since the protective distance also depends on the speed of the robot, it is further possible to dynamically reduce its speed to avoid violating the minimum separation. Therefore, the safety system must be able to reliably detect the distance to a human worker.

Several types of safety sensors can be used for this purpose (Figure 1.5). *Laser scanners* [214] emit multiple laser beams from a rotating head and measure the time it takes for them to be reflected back to calculate distances to objects. *Time-of-Flight (ToF) cameras* [216] work similarly, but instead of single rotating beams, they follow a camera-based sensor principle, where they illuminate the whole area with infrared light and measure the time it takes for the light to be reflected back for each pixel.

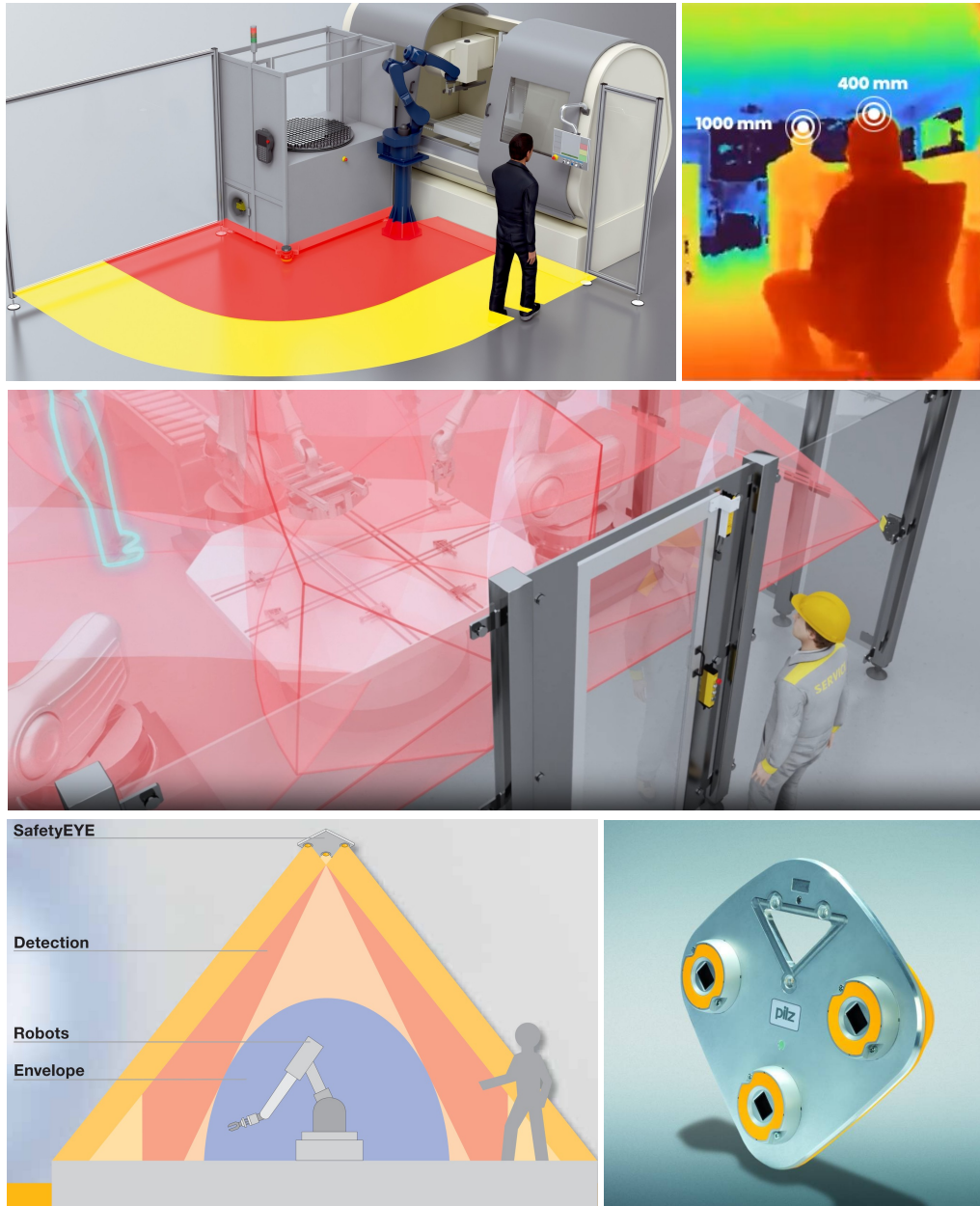


Figure 1.5. Examples of speed and separation monitoring systems. On the top-left, a laser scanner that detects persons walking towards the robot, and depending on intruded zones reduces the speed of the robot (yellow) or stops it completely (red). On the top-right, an example of a time-of-flight camera image in which the color encodes the distance to the objects. In the center, an example of multiple radar sensors which detect the breathing motion of a person in their sensor cone, and in this case, prevent the robots from starting even though the safety door was closed by the second worker. On the bottom, the *Pilz SafetyEye*, which uses three redundant cameras to monitor the area around the robot and detects changes in the environment and intrusions into configurable zones. Images in described order from [10], [12], [9], [185], [37].

Radar sensors [215] emit radio waves in a conical area and measure the time it takes for them to be reflected back, allowing them to detect objects and especially also slight variations in their speed. With this principle, they can distinguish between static objects and non-static humans. The *Pilz SafetyEYE* [185] uses three redundant cameras to monitor the area and detects intrusions into configurable safety zones by comparing to reference images. It was available only for a short time and is now discontinued (reasons unknown to the author).

In general, the main limitation of such systems is that they use relatively large zones and therefore do not allow close collaboration. A more recent, but also more complex, system that allows tighter zones is *Veo Freemove* [25]. It uses multiple *ToF*-cameras, each equipped with two redundant sensors, to monitor the area around the robot and detect unknown objects intruding into the workspace. With the depth-measurements of the cameras, it calculates a point-cloud of the area, and then ensures a minimal distance between the points of the robot and those of the human (Figure 1.6). It is not available as a standalone safety solution anymore (the company behind it was acquired).

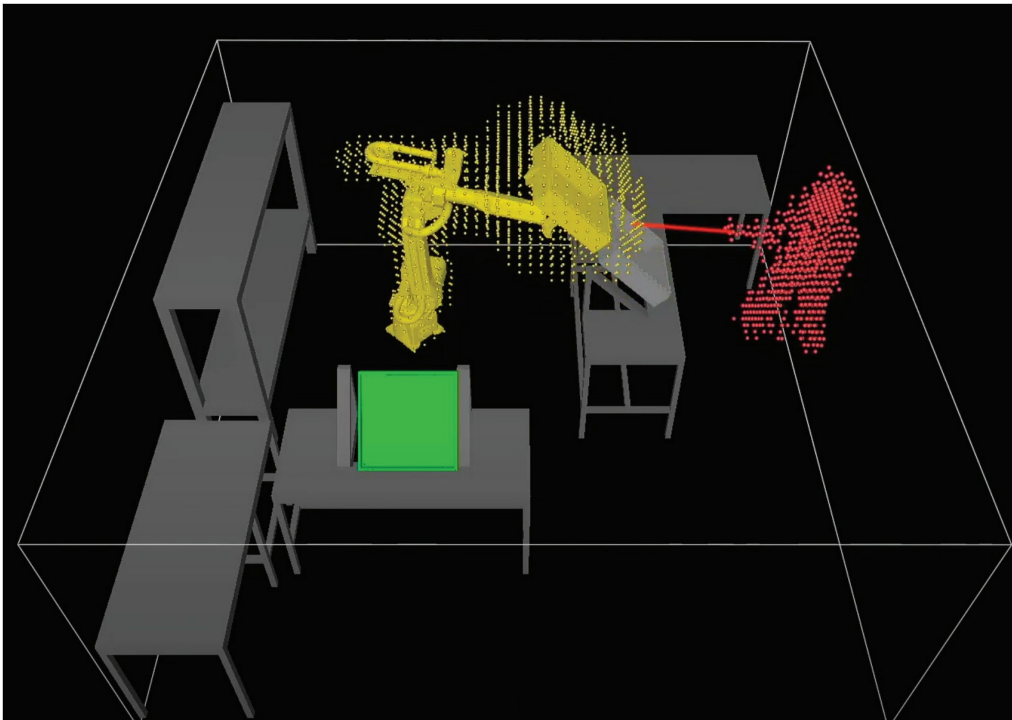


Figure 1.6. Example of a tighter safety zone based on point-clouds around the human using the *Veo Freemove* system. Image from [25].

Another option is the use of *capacitive* sensors, which detect the presence of a human by measuring changes in the electric field around the robot (Figure 1.7). For this purpose, the robot is equipped with an additional covering that includes those sensors. In contrast to the previous systems, they are integrated into the robot structure, and work from an inside-out perspective, which reduces the possibility of occlusions that could occur with

external sensors. Such occlusions can either be caused by objects in the workspace, or by the human itself who could stand between the robot and the sensor, blocking the view on his hands with his back. One of the main limitations of such systems, though, is the short range of the sensors. Commercial systems like the *Bosch APAS* [5] can only detect objects within 5 cm, which limits the maximum speed of the robot (it needs to brake within this range). Research systems like [105] can reach up to 35 cm, but they are not safety-certified and their reliability has not been evaluated in depth. Another limitation of such systems is their sensitivity to environmental influences, for example changing temperatures, electric fields from the robot motors themselves, or static structures in the robot's workspace [105]. Changing those conditions normally requires a recalibration of the system, leading to additional effort. Furthermore, the capacitive sensors have to be attached to the outside of the safeguarded object, which might be possible with some effort for the robot itself, but is difficult for tools and especially for the items the robot handles.



Figure 1.7. Examples of capacitive sensor robots. On the left, the *Bosch APAS* system, on the right a capacitive sensor from a research setup. Images from [5] and [105].

Even if these systems can create tighter zones around humans or robots, they still must maintain a certain distance to ensure safety, which limits collaboration options. They do not allow any contact between robot and human while the robot is moving, for example during a direct handover of objects, but instead always require the robot to stop completely before any contact can occur, which can significantly decrease productivity.

1.2.4 Power and Force Limiting

The last type of collaboration is *Power and Force Limiting*, where the robot is allowed to have physical contact with the human operator, but the contact forces must not exceed certain limits to avoid injuries. This can be achieved by using special collaborative robots (Figure 1.8) which are equipped with multiple sensors to detect collisions and stop the robot if required.

The main limitation of such systems is that they move relatively slowly to limit contact forces during collisions, which can decrease productivity. They also have a lower payload than traditional industrial robots to reduce moving mass and therefore impact energy,

which limits their possible applications. Furthermore, because of the required sensors and more complex construction requirements, they are also more expensive.



Figure 1.8. Example of collaborative robots. On the top image, a *Universal Robots UR*, and on the bottom, a *KUKA LBR iiWA* robot. Images from [195] and [196].

There exist some methods to add additional safety cushions to traditional industrial robots, such as the *AIRSKIN* system [4], which uses special pads that can detect contacts and also provide a soft impact protection (Figure 1.9). However, such systems support only a limited selection of robots.

A general limitation of these force-limited systems is that they cannot prevent unintentional contacts, because they can only react after the contact has already occurred. They also cannot avoid collisions by changing their trajectory, because they do not have any information about the human's position. While these safety systems prevent



Figure 1.9. Example of additional safety cushions using the *AIRSKIN* system. The pads detect contact forces and also provide a soft impact protection. Images from [4].

injuries, they can still cause discomfort and reduce the worker's subjective safety feeling. Another important limitation is that these systems protect against contacts with the robot, but do not necessarily protect against risks from tools attached to the robot, which can have sharp or pointed parts (as in Figure 1.8). While it is possible to secure the tools as well, for example by using special covers (as in Figure 1.9), this normally does not work for items the robot is handling, for example sharp metal sheets in a bending process, especially if they are large or complexly structured.

1.3 Goal of this Work

Taken together, these limitations highlight the need for safety systems that allow closer, more flexible, and thereby more efficient human-robot interaction while maintaining reliable and safe operation. The main goal of this work is therefore to enable seamless collaboration, including close interaction and physical contact during robot motion. Following the limitations of current methods, the main requirements for such a new safety system are:

- **Robot agnosticism:** The system should be applicable to a wide range of robots, especially large traditional industrial robots.
- **Tool and workpiece safety:** The system should not only safeguard the robot itself, but also the tools attached to it, and the items it is handling, even if they are large or complexly structured.
- **Dynamic safety zones:** The system should be able to create dynamic safety zones tightly around the human worker.
- **Distance measurements:** The system should be able to reliably determine the location of the worker and thereby measure the distance between human and robot, to detect potential contacts, to stop the robot to prevent contacts, or to allow dynamic adjustments of the robot's trajectory to avoid collisions.
- **Body-part specific reactions:** The system should be able to distinguish between different body parts of the human worker, to allow contacts with the hands, while keeping a safe distance to the rest of the body and especially the head.
- **Safety certification:** The system should be safety-certifiable according to relevant standards in the long term, to allow for a wide adoption in industry.

While there exists some research in this direction, the different approaches only target some of those requirements, which is the reason why this work will focus on developing a new safety system that can meet all of those requirements at the same time.

The first two requirements suggest that the system should be based on an external sensor system, which can be added to existing robots without requiring any modifications to the robot itself, the tools, or the items it is handling. The next two requirements indicate the need for a sensor system that can detect the human worker anywhere in the robot's workspace with high spatial resolution, rather than only in a few discrete zones. Especially the fifth requirement, of being able to distinguish between different body parts, adds a significant challenge. The best and least intrusive way to address this, as will be further discussed in Section 3.6, is using AI-based methods that can analyze the sensor data and detect the exact pose of the human worker. In conclusion, combining a camera-based sensor system with AI-based pose estimation methods is the most promising approach to meet all of these requirements at the same time. Therefore, this work will develop such a system, and also evaluate its performance and reliability with the target of the last requirement, of being safety-certifiable in the long term.

Figure 1.10 shows an example scenario in which all of those requirements are relevant. The robot is supposed to support the human worker in his task of assembling a bicycle,

by holding the bike while the worker places the cables into the frame. In the optimal case, the robot should be able to move the frame to support the worker with the cable placement. However, to avoid any risk of injury, the system then must ensure that the robot can only move the frame when the worker's hands are on the bike, while ensuring that the worker's head is not stuck in the frame, and the movement also does not lead to new contacts with other body parts.

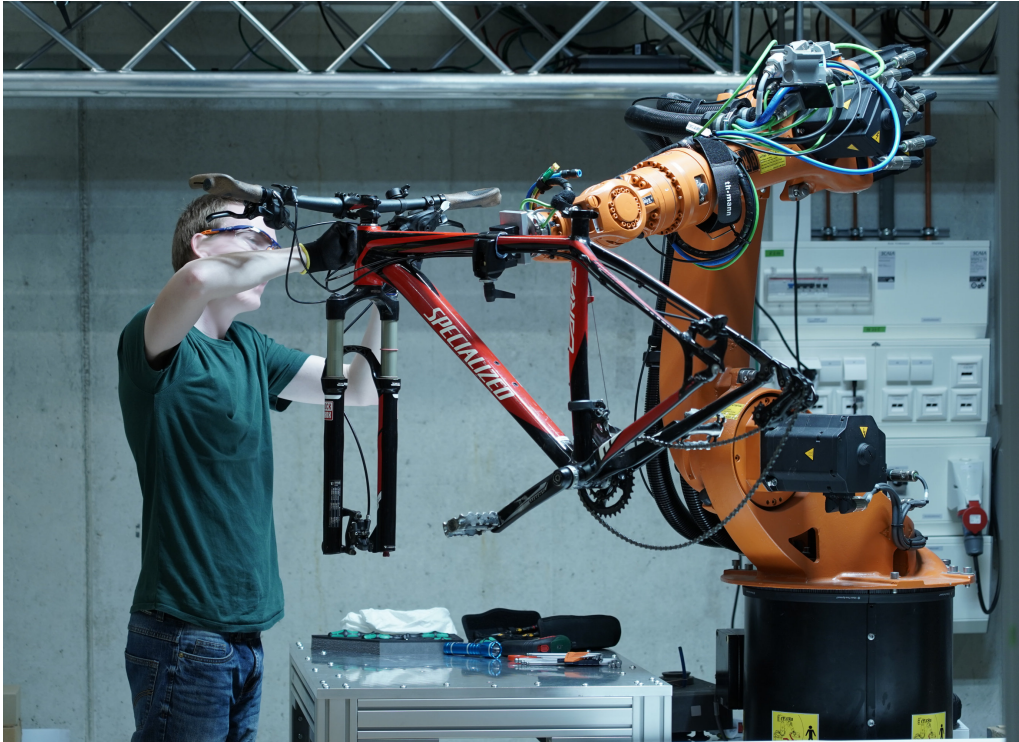


Figure 1.10. An example scenario which would benefit from the proposed safety system.

Beyond the safety aspect, such an approach that is aware of the human worker's pose and movements can also enable more efficient collaboration by allowing the system to understand human behavior and intentions, and adapt its actions accordingly. This then subsequently can allow the creation of more natural and intuitive system interactions, which can further increase productivity and worker satisfaction.

Regarding the differences between the existing research approaches towards smarter safety systems, their limitations, and how they compare to the system developed in this work, a more detailed comparison will be given in Section 3.6. To better understand the requirements and the challenges in this area, an overview of the safety regulations and the new system has to follow first.

1.4 Thesis Overview

The remainder of this thesis is structured as follows: Chapter 2 provides an overview of safety regulations in the context of robotic applications. Chapter 3 presents the first version of the proposed AI-based safety system, develops different methods to very reliably detect and localize humans, and finally evaluates the system's performance in a real human-robot collaboration scenario. This prototype is then further improved in an iterative process in the following chapters. Chapter 4 and Chapter 6 present further safety regulations that are relevant for the directly subsequent developments. Chapter 5 focuses on improving the system regarding the reaction speed and the closeness of the collaboration, and also presents a further, ultra-fast method for multi-view human pose estimation. Chapter 7 continues to improve different system properties, introduces additional capabilities that are possible with the new AI-based system, and discusses subsequent steps towards safety certification. Chapter 8 then broadens the perspective to advantages that the developed methods bring in the context of understanding human behavior and improving collaboration efficiency. Chapter 9 finally focuses on interactions between humans and a robot from a more human-centric perspective, and introduces a custom voice assistant as an additional communication channel. Chapter 10 concludes the thesis and highlights the main contributions again.

Summary. This chapter provides an overview of the general safety regulations in the context of robotic applications.

2

Safety Foundations 1: Robot Systems

2.1	Overview	16
2.2	Risk Analysis	18
2.2.1	Limits of machinery	18
2.2.2	Hazard identification	18
2.2.3	Risk estimation	19
2.2.4	Risk evaluation	19
2.2.5	Required Performance Level	21
2.3	System Assessment	23
2.3.1	Specification of Safety Functions	23
2.3.2	Mean Time to Dangerous Failure	24
2.3.3	Diagnostic Coverage	24
2.3.4	Common Cause Failures	24
2.3.5	Systematic Failures	25
2.3.6	Category	25
2.3.7	Performance Level Determination	27
2.3.8	Software Requirements	27
2.3.9	Validation	28

The safety of a robotic system is a critical aspect, especially in a scenario where the robot is operating close to a human worker. To ensure the safety of a human worker, each system must follow safety regulations. This chapter will present an introduction to the general safety regulations in the context of robotic applications.

2.1 Overview

According to the European Union's *Directive 2006/42/EC on Machinery* [14], all machines placed on the market must be safe for their users. In Germany, this directive is implemented into law by the *Neunte Verordnung zum Produktsicherheitsgesetz* [13], with similar legislation in the other Member States.

The manufacturer or his authorized representative may only place machinery on the market or put it into service if, when properly installed and maintained and used for its intended purpose or in the event of foreseeable misuse, it does not endanger the health and safety of persons, the safety of domestic animals and property and, where applicable, the environment. (§3 S1 9.ProdSV, translated)

So, to sell a machine on the EU market, or install it in a factory, a manufacturer has to prove that the machine is safe to use. They can do this by ensuring that their machines conform to certain official standards.

If a machine has been manufactured in accordance with a harmonized standard, the reference to which has been published in the Official Journal of the European Union, it shall be presumed to comply with the essential health and safety requirements covered by that harmonized standard. (§3 S5 9.ProdSV, translated)

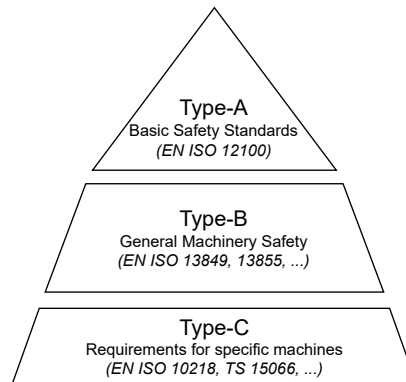


Figure 2.1. Structural organization of safety standards.

There are multiple harmonized standards that describe requirements for different machines. In this chapter, only those related to robots will be taken into account. According to *EN ISO 12100* [111], the machinery standards can be structured into different specificity levels (Figure 2.1): The *Type-A* standards describe basic concepts and principles. *Type-B* standards are applicable for a wide range of machinery and deal with certain safety aspects or safeguarding devices. *Type-C* standards then focus on the detailed requirements for specific machines, like industrial robots for example. From a hierarchy perspective, the more specific standards take precedence over the more generic ones.

In general, for any machine, *ENISO 12100* describes an iterative strategy for machine designers to ensure a user's safety [111, Chap. 4]:

- *determine the limits of the machinery, which include the intended use and any reasonably foreseeable misuse thereof*
- *identify the hazards and associated hazardous situations*
- *estimate the risk for each identified hazard and hazardous situation*
- *evaluate the risk and take decisions about the need for risk reduction*
- *eliminate the hazard or reduce the risk associated with the hazard by means of protective measures*

This process is designed to enable a systematic evaluation of the risks associated with the machine. The standard assumes that hazards will cause harm eventually, and therefore protective measures have to be implemented beforehand. Those measures should preferably be built directly into the machine's design rather than added later by users. The general process aims to achieve the greatest practicable risk reduction while prioritizing safety throughout the machine's life cycle, then functionality, usability, and finally cost considerations.

The next sections will focus on the analysis of the risks, and their reduction by means of a safety system in more detail.

2.2 Risk Analysis

The first steps of the aforementioned iterative strategy include the analysis of potential hazards and the risks to the human operator associated with them. This section will summarize the risk analysis approach described in *EN ISO 12100* [111, Chap. 5].

2.2.1 Limits of machinery

To find possible hazards and the people who could be affected by them, the limits of the machine have to be determined. To name a few examples, this includes the intended use of the machine, such as different operating modes, abilities and experience of users, or foreseeable contacts with other persons. It also includes the machine's range of motion, space requirements for operators in front of the machine, and components that require regular maintenance.

2.2.2 Hazard identification

In the next step, reasonably foreseeable hazards have to be identified. A few examples of possible hazards are shown below (Figure 2.2).

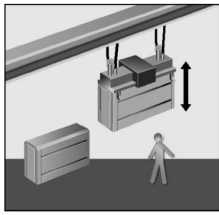
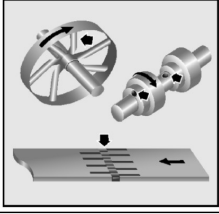
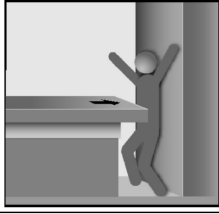
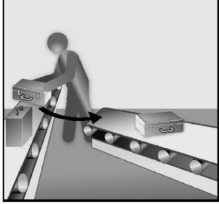
Hazard		Hazard	
	Origin moving elements Potential consequences – crushing – impact – shearing		Origin falling objects Potential consequences – crushing – impact
	Origin rotating or moving elements (three examples) Potential consequences – severing – entanglement		Origin approach of a moving element to a fixed part Potential consequences – crushing – impact
	Origin posture Potential consequences – discomfort – fatigue – musculoskeletal disorder		Origin noisy manufacturing process Potential consequences – fatigue – hearing impairment – loss of awareness – stress

Figure 2.2. Examples of hazards, adapted from *EN ISO 12100* [111].

In general, the whole life cycle of the machine (assembly, commissioning, use, dismantling, ...) and all human interactions during this time (setup, starting, operating, stopping, maintaining, ...) have to be taken into account. Besides the normal machine operation, machine designers must also consider possible malfunctions, for example

caused by failure of components, the workpiece itself, or possible hardware errors caused by electromagnetic interference. They must also identify unintended behavior or reasonably foreseeable misuse by operators, for example due to lack of concentration, reflex actions during malfunctions, or pressure from their supervisors to increase output.

2.2.3 Risk estimation

After identifying the hazards, the next step is to identify the risks associated with them. The risk depends on the severity of the harm caused by the hazard (slight/serious injury, death) and the probability that this harm occurs. The probability depends on how often and how long an operator is exposed to a hazard, and how often a hazardous event occurs. It is also influenced by the possibility of the users avoiding that harm, for example because they are aware of the risks, have experience with the machine, and have time and space to escape the danger.

If protective measures are used, the suitability and reliability of those measures must also be considered. A special case here is the possibility that those protective measures are circumvented by the users. They might be motivated to do so, if they slow production, are difficult to use, or are perceived as unsuitable. They might also forget about their annual hazard trainings, or avoid personal protective equipment because it is inconvenient.

2.2.4 Risk evaluation

After all risks have been identified, it has to be evaluated whether the risks of the application are too high and have to be reduced. In this case appropriate protective measures have to be selected and then applied. In general, the risks can be reduced by either eliminating the hazard, or by reducing the severity or probability of the harm caused by the hazards.

To reduce the risks, the *ENISO 12100* standard specifies a three-step process with a hierarchical order (Figure 2.3).

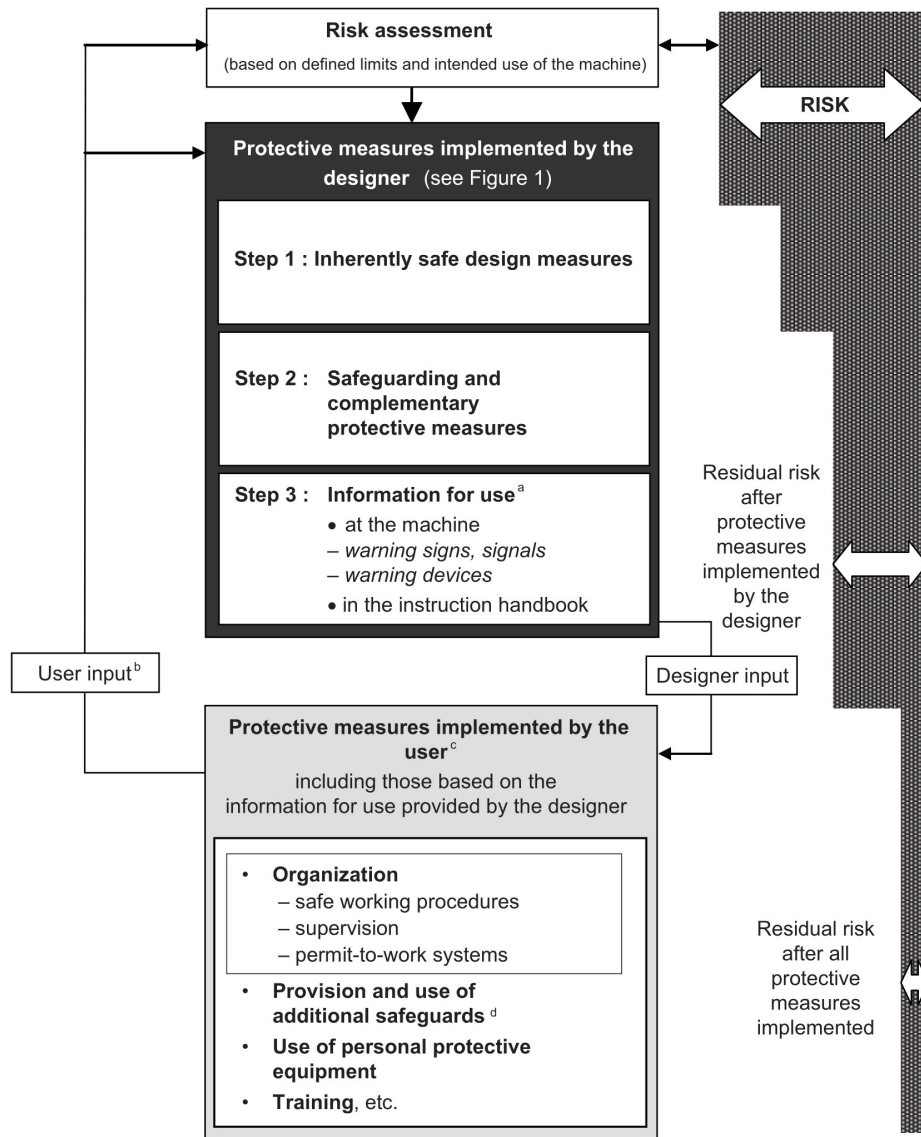
The first step includes implementing inherently safe design measures into the machine. Options for this could be, for example, reducing applied forces or the mass of moving objects, removing sharp edges, or using appropriate components for the machine's tasks. For the operator, the machine designers can increase the visibility of working areas and hazard zones, and take into account ergonomic principles.

In the second step, additional guards or protective devices can be added, if it was not reasonably possible to reduce the risks through the inherently safe machine design. Depending on the application requirements, this could be fences, light curtains or sensitive protection equipment for example. Those measures again have to reliably reduce the risks, which can be evaluated by another standard that is described in more detail in the next section.

In the last step, any remaining risks must be identified in the information for use. These include, for example, operating procedures, expected operator abilities, recommended work practices and personal protective equipment, and warnings of residual risks.

Generally, those steps are applied iteratively per step, so if the currently planned measures are not sufficient to reduce the risks, and it would be possible to implement

additional measures from the same level, those have to be implemented first before moving on to the next step.



^a Providing proper information for use is part of the designer's contribution to risk reduction, but the protective measures concerned are only effective when implemented by the user.

^b The user input is that information received by the designer from either the user community, regarding the intended use of the machine in general, or from a specific user.

^c There is no hierarchy between the various protective measures implemented by the user. These protective measures are outside the scope of this International Standard.

^d These are protective measures required due to a specific process or processes not envisaged in the intended use of the machine or to specific conditions for installation that cannot be controlled by the designer.

Figure 2.3. Risk reduction process for machine design, from *EN ISO 12100* [111].

Finally, the whole process of risk analysis and reduction has to be documented. The documentation has to include, among other things, any relevant assumptions made about the machine, identified hazards and risks, and which protective measures were implemented to reduce them.

2.2.5 Required Performance Level

If the risk evaluation determines that additional guards or protective devices are needed, the *ENISO 13849-1* [114] standard describes the requirements for this case. It defines which protection methods can be used, and how they have to be implemented.

For each risk, a *safety function* has to be implemented and integrated into the control system of the machine that will measure and, if the defined limits are violated, reduce that risk. For example, it could stop the movement of the machine or reduce the force it applies. The standard then defines the term *performance level (PL)* as the *ability of safety-related parts of control systems to perform a safety function under foreseeable conditions*. Depending on the risk, each *safety function (SF)* needs to have at least the *required performance level (PL_r)* to be able to reduce that risk successfully. Given the risk estimation from before, a simple process can be followed to identify the PL_r (Figure 2.4).

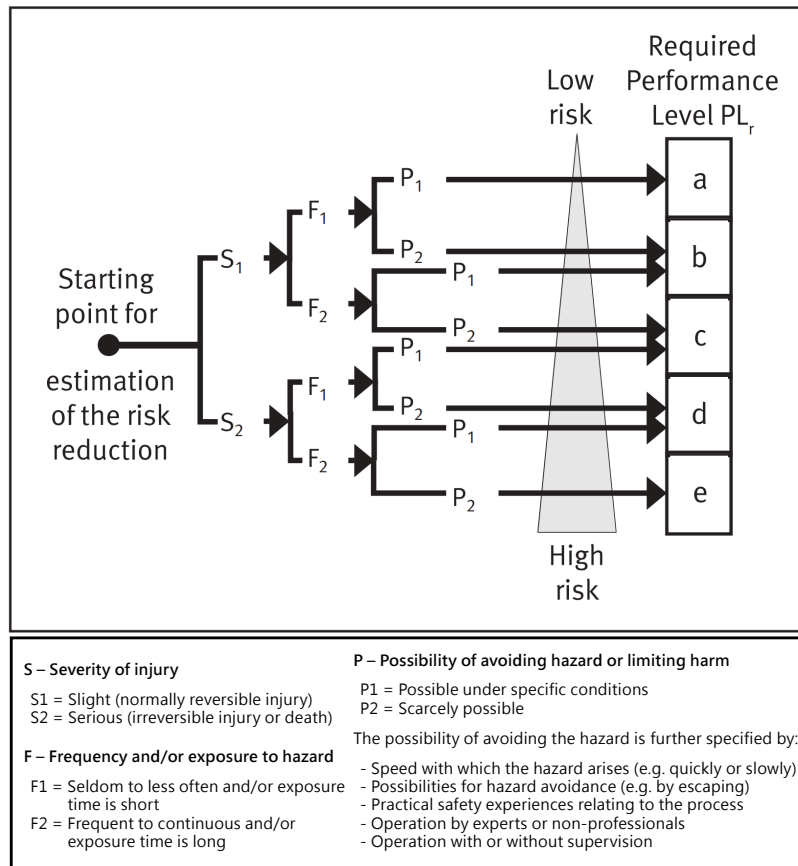


Figure 2.4. Risk assessment according to *ENISO 13849-1* [83].

For example, in a robotic application an operator may need to regularly refill a magazine of workpieces used by the robot. When the robot moves, it could squeeze the worker between the gripper and the magazine, leading to severe injuries (**S2**). If the operator needs to check the magazine every hour and takes two minutes for it, the exposure is rated as not frequent (**F1**). Because the robot operates at high speed, escaping the danger would not be possible in time (**P2**). To reduce that risk, the robot must be set to a safe state (for example stopped) by a *safety function* that requires at least $PL_r=d$. A solution could be to place a safety-certified light curtain in front of the robot's workspace, which reliably stops the robot if someone enters.

If the machine has multiple risks, they can be addressed by the same *safety function*. In that case, this *SF* must have the highest PL_r required by those risks.

For common industrial robots *ENISO 10218-1* requires a minimum performance of the safety functions implemented within the robot of at least $PL_r=d$ [119, Chap. 5.3.3]. In robotic applications, if no comprehensive risk evaluation was executed, *ENISO 10218-2* requires as default at least $PL_r=d$ for the external safety functions as well [120, Chap. 5.5.3] [82, Chap. 4.3.1].

Besides the approach of *ENISO 13849-1* described here, *ENISO 10218-2* also provides alternative procedures to determine the PL_r for robotic applications, which include additional parameters to allow a more detailed classification at decision boundaries [120, Ann. C.2]. For the purposes of this work, the simpler approach of *ENISO 13849-1* will, however, be sufficient.

2.3 System Assessment

Based on *ENISO 12100*, the *ENISO 13849-1* [114] standard describes the design of *safety-related parts of control systems (SRP/CS)*. It presents a generically applicable methodology to develop systems that perform safety functions, and how to determine their achieved performance level (*PL*). The application of *ENISO 13849-1* is described in great detail in [83] by the *Deutsche Gesetzliche Unfallversicherung (DGUV)*. This section will summarize the main steps to assess a safety system according to that standard.

The higher and more harmful the risks are, the more reliable the safety functions that are intended to reduce those risks have to be. Technically this means they must achieve at least the previously determined PL_r . The reliability of a safety function is expressed by its performance level, which is defined as the *average probability of dangerous failure per hour* (PFH_D). *ENISO 13849-1* defines five performance levels depending on their average failure probability (Table 2.1).

Performance Level	PFH_D in $hours^{-1}$
a	$10^{-5} \leq PFH_D < 10^{-4}$
b	$3 \cdot 10^{-6} \leq PFH_D < 10^{-5}$
c	$10^{-6} \leq PFH_D < 3 \cdot 10^{-6}$
d	$10^{-7} \leq PFH_D < 10^{-6}$
e	$PFH_D < 10^{-7}$

Table 2.1. Definition of *Performance Levels* according to *ENISO 13849-1* [114].

To simplify proving the PFH_D of a given safety function, the standard proposes a practical methodology that will be described in the following.

2.3.1 Specification of Safety Functions

The first step is to create a specification of the safety functions offered by the control system. It has to include general information about the project and product, safety-relevant functional information about the machine, and the documentation of the risk assessment. Then, for each safety function, it has to describe the intended function (input, logic, output), triggering events and conditions, behavior of the machine after the function was activated, and some further details which are skipped here for brevity. For the validation, it also has to include, among other things, information about the technology by which the safety function is to be implemented, performance data like cycle and response times, or generic standards that will be applied in the design. It also states the target reliability characteristics (*Category*, *MTTF*, *DC*, ...) that are intended to be achieved, and which will be described in more detail in the following. It additionally needs to describe which principles and standards have been applied to make the safety interfaces ergonomically suitable for the users.

2.3.2 Mean Time to Dangerous Failure

The reliability of the components used to implement the safety system has a significant influence on its overall reliability. It is quantified by the *Mean Time to Dangerous Failure* ($MTTF_D$), and measures the basic reliability of the typical component. Only failures that have an impact on the safety function are considered *dangerous* and are included in the calculation of the value. The standard distinguishes between three different $MTTF_D$ levels (Table 2.2).

For most components, those values are provided by the manufacturers, can be found in special databases, or are provided in the standard itself. If multiple components are combined to solve a task, their individual $MTTF_D$ values have to be combined as well to get the overall value for the combined channel [114, Ann. D].

Classification	$MTTF_D$ per channel
low	$3 \text{ years} \leq MTTF_D < 10 \text{ years}$
medium	$10 \text{ years} \leq MTTF_D < 30 \text{ years}$
high	$30 \text{ years} \leq MTTF_D < 100 \text{ years}$

Table 2.2. Definition of *Mean Time to Dangerous Failure* levels in EN ISO 13849-1 [114].

2.3.3 Diagnostic Coverage

The ability of the safety system to detect *dangerous* failures is expressed by the *Diagnostic Coverage* (DC). It is defined as the ratio between the rate of detected dangerous failures and the total rate of dangerous failures of the system. If such a failure is detected, the safety system then has to transition the machine to a safe state again. Testing the components regularly can compensate a lower reliability of the components to some extent. The standard distinguishes between four different DC levels (Table 2.3).

Classification	DC per channel
none	$DC < 60\%$
low	$60\% \leq DC < 90\%$
medium	$90\% \leq DC < 99\%$
high	$99\% \leq DC$

Table 2.3. Definition of *Diagnostic Coverage* levels in EN ISO 13849-1 [114].

2.3.4 Common Cause Failures

It can happen that two or more separate faults have a common cause. In some cases this can cause the system to miss detecting those faults, for example if multiple channels are affected simultaneously. To reduce the probability of such *Common Cause Failures* (CCF), adequate measures can be implemented in the design of the safety system.

The standard proposes multiple categories of such measures, which are scored depending on their effectiveness (Table 2.4). To achieve adequate measures against *CCF*, a minimum score of 65 points has to be reached.

Measure	Score
Physical separation between signal paths of different channels	15
Diversity in technology, design or physical principles in the channels	20
Protection against possible overloading (voltage, current, temperature, ...)	15
Use of well-tried components	5
Failure mode and effects analysis during development to identify <i>CCF</i>	5
Training of designers in causes and consequences of <i>CCF</i>	5
Protection against contamination of pressure fluids and electromagnetic disturbances	20
Protection against unfavorable environmental conditions	10

Table 2.4. Measures against *Common Cause Failures* in *EN ISO 13849-1* [114].

2.3.5 Systematic Failures

Besides random hardware failures in operation, systematic failures beforehand can also lead to dangerous situations. Those can be caused by wrong specifications of the design, human errors in the design of hardware and software, or by using complex components that originally were not developed for safety applications. Other reasons could be errors while manufacturing the safety system, or in the integration process at the customer's site. Therefore, activities that are necessary to avoid such failures have to be applied in the design process and documented in the safety function specification.

2.3.6 Category

The architecture of the safety control system notably influences its tolerance of faults. Therefore, the standard distinguishes between five different *categories* (*B*, 1, 2, 3, 4) with increasing error detection and fault tolerance capabilities.

The basic category, on which the others then build upon, is *B*. It follows a simple input-logic-output structure (Figure 2.5) and only requires the usage of *basic safety principles*. This includes, for example, the usage of components that are suitable for the intended application and withstand the expected operating stresses. Therefore, it requires at least an $MTTF_D$ classification of *low*. It does not include any fault detection, and failures may lead to a loss of the safety functions.

Category 1 has the same structure, but requires using well-tried components and well-tried safety principles, resulting in an $MTTF_D$ of at least *high*.

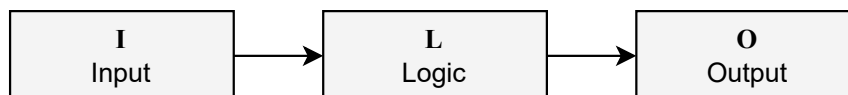


Figure 2.5. Categories *B* and 1 according to *EN ISO 13849-1* [114].

Category 2 builds upon *B*, but requires the implementation of *periodic tests* to detect faults in the system (Figure 2.6). All components of the safety function have to be tested, and if a fault is detected, the system has to be brought into a safe state. The tests must be executed at system start-up, before reaching hazardous situations, periodically during operation, or immediately after a demand on the safety function. If at least $PL_r=d$ is required, the safe state must be maintained until the fault is removed. The category also requires at least an $MTTF_D$ classification of *low* for the components, and a *DC* of at least *low*, including adequate measures against *CCF*.

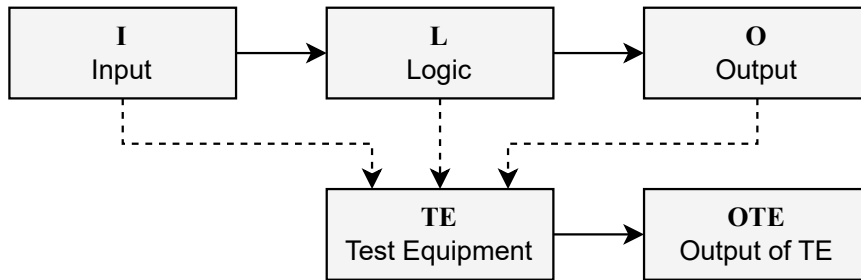


Figure 2.6. Category 2 according to ENISO 13849-1 [114].

Category 3 must be designed such that a single fault in any of its parts does not lead to the loss of the safety function. Usually, this is implemented through a two-channel system (Figure 2.7). Each channel performs the same safety function independently, so if one channel fails, the other can still perform the safety function. Whenever it is reasonably practicable, the fault should be detected at or before the next demand on the safety function. The system also has to test itself continuously for possible faults, and implement a cross-monitoring between the channels. Regarding $MTTF_D$, *DC*, and *CCF*, it has the same minimum requirements as category 2.

Category 4 extends category 3 by requiring that both the $MTTF_D$ per channel and the *DC* are *high*. It additionally requires that an accumulation of undetected faults does not lead to the loss of the safety function.

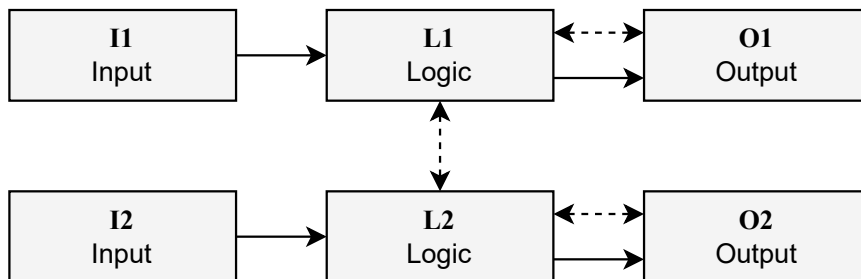


Figure 2.7. Categories 3 and 4 according to ENISO 13849-1 [114]. Possible testing points are marked by the dashed arrows.

2.3.7 Performance Level Determination

After the previous parameters have been determined, the standard provides a simple method to obtain the achieved performance level (*PL*) of the safety function (Figure 2.8). According to the implemented *category* and *DC*, a column is selected, and depending on the $MTTF_D$ classification, the achieved *PL* can be read out on the left axis.

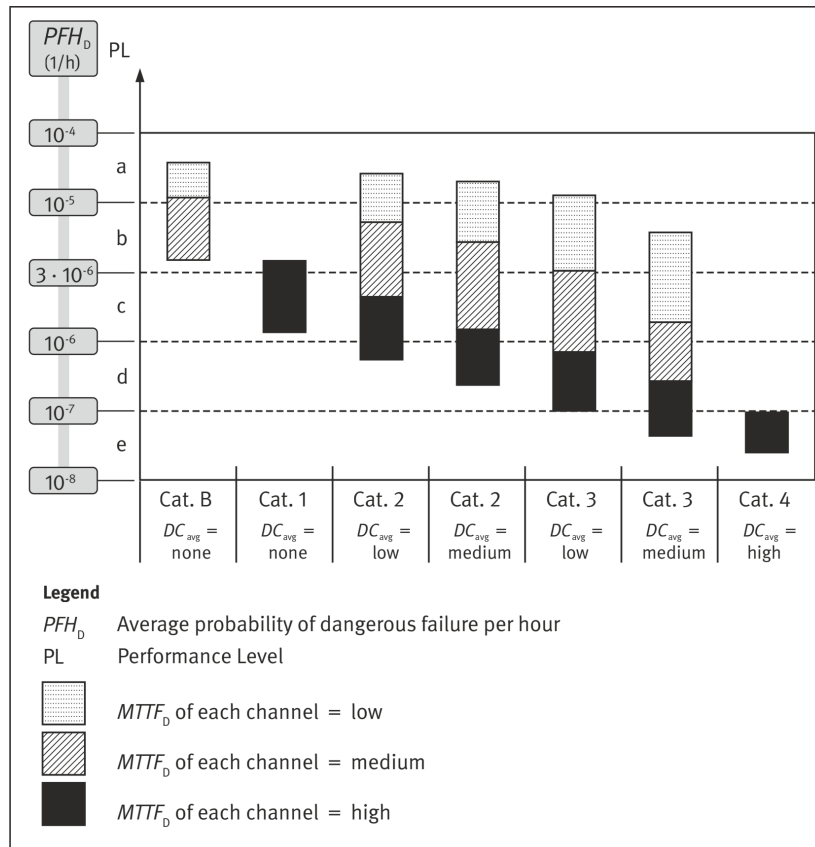


Figure 2.8. System assessment according to EN ISO 13849-1, image from [83].

2.3.8 Software Requirements

Regarding the development of safety-related software, EN ISO 13849-1 requests that the process should primarily consider the avoidance of faults during the complete software lifecycle. The development process should follow a structured V-model approach, with a strong focus on specification and verification activities (Figure 2.9). The software itself has to follow a modular and structured design, and be thoroughly tested and documented. With higher performance levels, the software requirements become stricter as well, for example by requiring further tests and extensive code walkthroughs. The standard also includes additional requirements for programming languages, pre-programmed libraries, and closed-source hardware drivers. In general, it is allowed to run safety and non-safety software on the same hardware, as long as this does not interfere with

the safety functions. If software updates are to be deployed, they have to be verified systematically again according to the initial software development process.

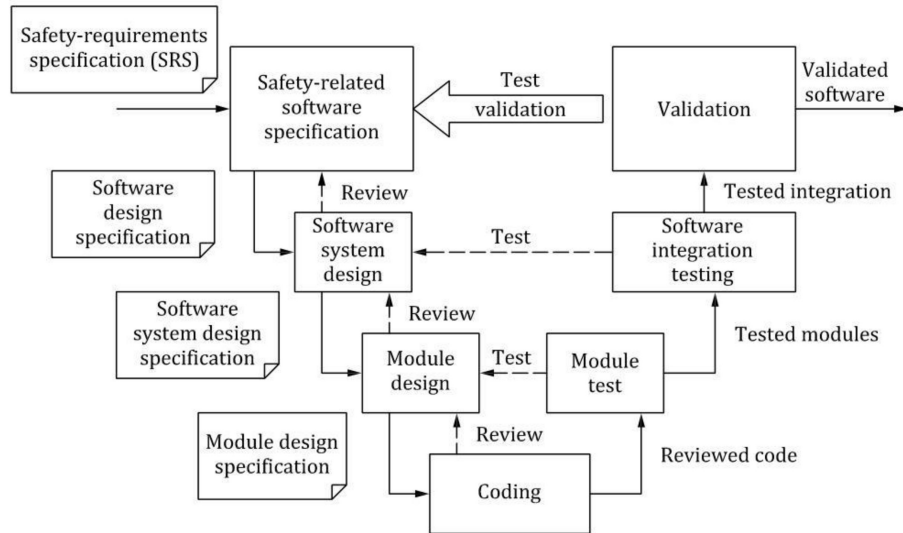


Figure 2.9. Software development process according to EN ISO 13849-1, image from [114].

2.3.9 Validation

After developing the safety system, it has to be validated to ensure that it meets the specified requirements. For this, a validation plan has to be created, which identifies and describes the validation activities to be performed. This includes, among other things, specifications and documentations from the development process, as well as a list of possible faults that could occur. The first validation step is then to check the requirement specification of the safety functions for completeness in their intended use. For the system itself, to check that all target reliability characteristics (*Category*, *MTTF*, *DC*, ...) were achieved, the validation is generally performed by analysis. This includes manually reviewing the specification and its respective implementation, as well as using automatic tools for static and dynamic hardware and software analysis. If the analysis alone is not conclusive, additional tests (using the fault list from before) have to be performed to demonstrate that all requirements are met. Finally, the whole validation process has to be documented in a validation report. To simplify the validation, specialized software tools can be used, for example *SISTEMA* [83, 85] from the *DGUV*.

In summary, using the process described in this chapter, a new safety system can be designed and validated. Building on this foundation, the next chapter continues with a practical implementation of an AI-based safety system in a specific robotic application.

Summary. This chapter presents the *KoARob* project, which investigated the use of AI-based pose estimation for safety monitoring in a human-robot collaboration scenario.

3

KoARob - AI for Safety

3.1	Scenario Description	30
3.2	Risk Analysis	33
3.3	Safety Concept	35
3.3.1	Safe Robot Position	35
3.3.2	Safe Human Position	35
3.3.3	Distance Monitoring	36
3.3.4	System architecture	37
3.4	Human Pose Estimation	39
3.4.1	Pose Estimation Foundations	39
3.4.2	skelda	45
3.4.3	Existing Approaches	45
3.4.4	SimpleDepthPose	48
3.4.5	VoxelKeypointFusion	52
3.4.6	PlaneSweepPosePlus	55
3.4.7	Comparison	57
3.5	Scenario Evaluation	64
3.5.1	Pose Estimation	64
3.5.2	System Assessment	66
3.6	Related Work	68
3.7	Conclusion	72

The research project *KoARob: Combination of Assistance and Sensor Systems for Safe Human-Robot Cooperation* had the goal of integrating the potential of commonly used and cost-effective but non-safety-rated sensors into a safety system for human-robot collaboration. Using AI-based methods, more accurate and also dynamic three-dimensional safety zones were implemented, to allow for a more flexible and efficient collaboration. The project was developed in cooperation with the *MRK-Systeme GmbH* and the *Roboception GmbH*, and funded by the *Bavarian Ministry of Economic Affairs, Regional Development and Energy*.

The following chapter gives an overview of the project, with a deeper focus on my contributions of the AI-based methods, the software architecture and safety zone implementation, and the risk and system evaluation. The chapter is based on my following published work, with selected text passages directly copied from it:

KoARob: Towards AI-based Safety in Human-Robot-Collaborations.

Bermuth, Daniel and Poeppel, Alexander and Reif, Wolfgang. August 2025.

In *2025 IEEE 21st International Conference on Automation Science and Engineering (CASE)*. Pages 837-844. © 2025 IEEE. [48]

Structurally, this chapter first describes the scenario and the system setup, then gives a brief overview of the risk analysis, and presents the concept behind the proposed safety system. This is followed in Section 3.4 by an introduction to the field of human pose estimation, a detailed description of the newly developed AI-based methods, and their comparison on multiple public benchmarks. Finally, the system is evaluated in the context of the described scenario, and the results are compared to relevant related work.

3.1 Scenario Description

For the *KoARob* project, the scenario of checking returned packages in a logistics center was chosen to demonstrate the potential of the developed system (Figure 3.1). This is a common task for many online retailers. The packages are returned by customers and need to be checked for completeness and damage so that their contents can be put back into stock if they are still in good condition. This task is normally done by a human worker, because the quality inspection is hard to automate. The worker needs to open the packages, check the contents, and sort them into different categories.

In the examined example scenario, the packages are delivered on a Euro-pallet. The workers take a package from the pallet and put it on a table in front of them. The package is then opened and its contents are checked. After the check is completed, the worker sorts the package into one of two different boxes, depending on the condition of the contents. This task is very repetitive and can lead to physical strain for the worker.

The robot's task in the *KoARob* example is thus to assist the worker by fetching the packages from the pallet and placing them on the table. This reduces the physical strain on the workers and allows them to focus on the quality inspection. Since robot and worker work closely together, the robot system needs to be able to detect the worker, keep an appropriate distance, and ensure their safety. Also, in case a package is too damaged for the robot to lift it with the suction gripper, the worker should be able to fetch it manually.



Figure 3.1. Close collaboration at handling packages. [48][© 2025 IEEE].

Figure 3.2 shows the setup for the application. The robot is a *KUKA KR16* with a custom suction gripper. It lifts the packages from the pallet on the right side and places them onto the table, alternating between two drop positions. Technically, a specialized collaborative robot like the *KUKA LBR iiwa* could have been integrated as well, but an industrial robot has the advantages of higher payload, longer reach, and lower costs. The worker stands in front of the table, fetches the new package, and opens it to check the contents. After finishing the inspection, the worker sorts the packages into two boxes on the left side (yellow-green for reusable ones, pink-red for waste). In principle, the robot could also place the packages into the correct boxes itself, but in the context of this work, this would not materially contribute to the research questions and would instead make the scenario more complex, so it was decided to leave this part to the worker. In case the robot is unable to lift a package from the delivery pallet, the worker can walk around the table and fetch it manually.

So in general, the human worker needs to freely access the workspace of the robot, and both workspaces partially overlap. To ensure the safety of the workers, the robot system needs to be able to detect the persons, and if they get too close to the robot, the robot's movement must be stopped. However, the workers should be able to work as close to the robot as possible, to allow for an efficient collaboration, and therefore a dynamic safety zone around the person is required. Additionally, to increase the efficiency, the size of the safety zone can be further optimized if it is adapted to the different body parts, so that the worker's hands can get closer to the robot than the rest of the body. With those requirements, an AI-based system would be a good choice to fulfill the task of detecting



Figure 3.2. The robot setup. [48][© 2025 IEEE].

the human worker and their pose. To collect the needed system information, the worker is observed by five *Roboception rc-visard* stereo-cameras (see Figure 3.3) mounted on the trusses, while an additional camera detects the locations of the packages on the pallet.



Figure 3.3. An rc-visard stereo-camera. Image from [23].

3.2 Risk Analysis

Given the described scenario, a risk analysis can be performed to identify the potential hazards and their associated risks, as previously described in Chapter 2. Since a full analysis is not required for the research questions of this chapter, only a brief overview is given here.

Regarding the limits of the machinery, only the main operating mode is considered, and it can be used by experienced researchers or by inexperienced students with oversight by the researchers. At least all hazards from Figure 2.2 (moving elements, falling objects, ...) are present. For the experimental safety, some inherently safe design choices were made, such as using a suction gripper that is somewhat flexible and very light example packages. They are also dropped a few centimeters above the table and are able to give in a few centimeters, to prevent crushing a hand between the robot gripper and the table. The robot itself was limited to a maximum speed of 0.2 m/s , by its built-in safety features, and the workspace was designed to have a lot of free space around the robot, to prevent squeezing against walls and allow an easy escape. But because the idea is to ensure the safety with the new AI-based system, some of those measures will be ignored in the following risk estimation and classification. For readability reasons, only a selection of the most important risks is highlighted here:

- The robot places a package on the table while the worker has a hand on the drop-off position:
 - $S1$ (suction gripper is somewhat flexible)
 - $F2$ (could occur more than once per 15 minutes)
 - $P1$ (robot moves slowly enough that the worker could pull back the hand)
 - $PL_r = b$ (according to Figure 2.4)
- Robot loses the package, which falls on the worker's foot:
 - $S1$ (parcel weight $< 5\text{ kg}$)
 - $F1$ (occurs less than once per 15 minutes)
 - $P2$ (human reflexes might not be fast enough to pull back the foot)
 - $PL_r = b$
- Robot hits the worker while moving a package from pallet to table:
 - $S1$ (no major squeeze points on the robot path)
 - $F2$ (could occur more than once per 15 minutes)
 - $P1$ (robot moves slowly, a lot of space around robot)
 - $PL_r = b$
- Robot hits the worker while the worker is fetching a package manually:
 - $S2$ (robot could squeeze hand or arm against the pallet)
 - $F1$ (occurs less than once per 15 minutes)
 - $P1$ (robot moves slowly, enough space to avoid collision)
 - $PL_r = c$

- Software has an error and robot leaves the workspace:
 - $S2$ (robot could squeeze the human anywhere in the surrounding area)
 - $F1$ (significantly less frequently than once per 15 minutes)
 - $P1$ (robot moves slowly, a lot of space around robot)
 - $PL_r = c$
- Robot has an error and moves too fast:
 - $S2$ (robot could squeeze the human anywhere in the surrounding area)
 - $F1$ (significantly less frequently than once per 15 minutes)
 - $P2$ (robot moves too fast for human to react)
 - $PL_r = d$

Each risk needs to be handled by a safety measure that has at least the required performance level. One safety system might secure multiple risks together. So in this case, if only one safety system is to be used, a $PL=d$ is required, as *ENISO 10218-2* requires for robotic applications in general if no detailed risk assessment was performed.

3.3 Safety Concept

Following the project requirements, an AI-based safety system shall be designed to reliably ensure the safety of the workers. Therefore, the system is designed to align with the aforementioned safety standards. At the time of development, no specific AI-related standards existed, so those basic standards were followed instead (Section 7.4.1 will describe notable similarities between the concepts presented here and later published official recommendations). To simplify development and achieve a modular and structured design, the safety functions are divided into three parts: safe robot positioning, safe human detection, and distance monitoring. The safe robot positioning ensures that the robot is not moving too fast or at the wrong positions, while the safe human detection and the distance monitoring ensure that the robot stops if a human gets too close.

3.3.1 Safe Robot Position

As found in the risk analysis, leaving the intended workspace or moving too fast are two critical risks that can lead to severe injuries. In particular, excessive acceleration needs to be detected quickly. To reach the required performance level, a redundant system with at least two channels is needed. Often the robot's software already provides safety functions to monitor the robot's position and speed. If this is not the case, or the robot does not output its position redundantly, like the robot used here, a second channel is needed to verify the robot's position. This could, for example, be achieved using an *Inertial Measurement Unit (IMU)* attached to the robot's gripper, a camera system detecting *AprilTags* [176] attached to the robot's joints, or *Wire Draw Encoders* [11] to measure the robot's position directly. Within the *KoARob* project, the project partner *MRK-Systeme GmbH* evaluated several options in this direction. However, since the focus of this work is on reliable AI-based human detection, it will be assumed in the following that the robot's position output is safe and reliable and can be directly integrated into the system.

3.3.2 Safe Human Position

To introduce redundancy and thereby increase reliability, three different AI-based approaches for human pose estimation were developed in this project and subsequently integrated into the safety system: *VoxelKeypointFusion*, *PlaneSweepPosePlus*, and *SimpleDepthPose*. The three models were evaluated across multiple publicly available datasets and compared with several previously existing approaches (Section 3.4.7). It was found that regarding the performance of the models, on datasets yet unseen to them to test their generalization capabilities, they were in summary the most reliable of all. The model architectures and evaluations will be described in detail in the next section. One generally important aspect here is that all three models were designed to use different architectures to detect the human joints, to increase the probability that at least one of them detects the human if the others have problems.

All three algorithms output the 3D position of the different human joints. To ensure that those poses and the robot are in the same coordinate system, the cameras were

calibrated beforehand with a *hand-eye calibration* process with respect to the robot's coordinate system, using their built-in calibration tool.

3.3.3 Distance Monitoring

To simplify the redundant execution of the safety function, to reduce hardware requirements, and to ensure explainability, a simple concept for distance monitoring was chosen. The safety-hulls around the robot and the human are represented by two voxelmaps, which define their space occupancy (see Figure 3.4). If a voxel is occupied by both the robot and the human safety-hulls, the robot is stopped. If the voxel is released again by one of them, the robot can continue moving. In addition, the safety system also ensures that both voxelmaps are up-to-date, so if one voxelization module stops sending new messages, the robot is automatically stopped.

Since the person detectors can differentiate between different body parts, the human pose voxelization process can define different safety distances for each of them. For the scenario, three different distances for the head, torso, and hands were chosen. Thus, the system can ensure a larger distance for the worker's head and a lower distance for the hands.

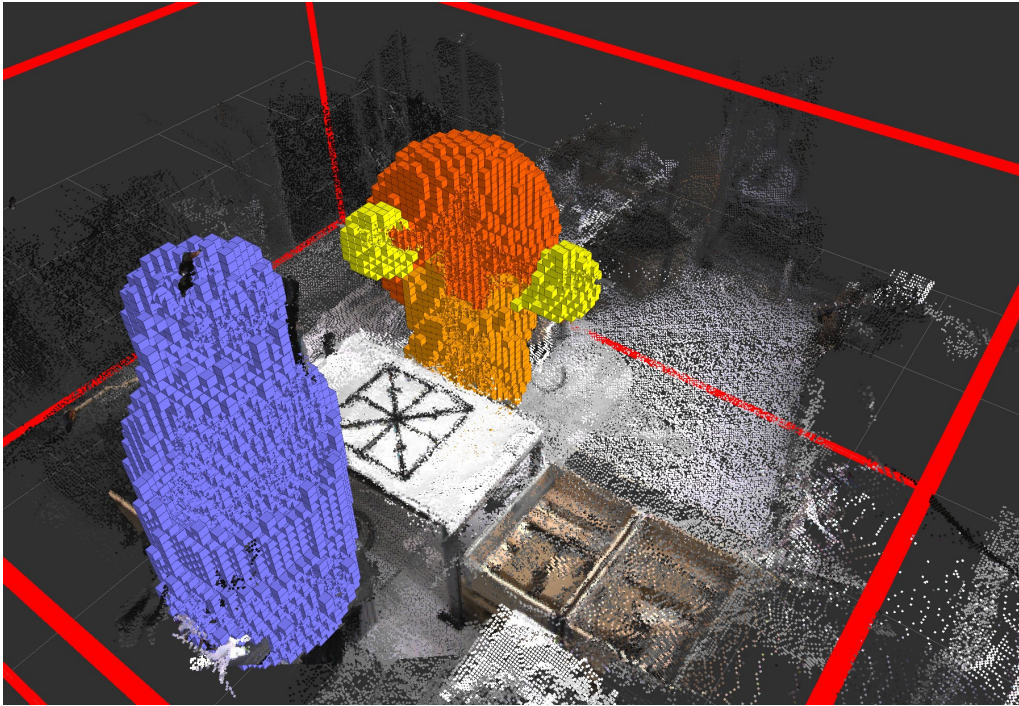


Figure 3.4. Voxel-based distance monitoring. The human is represented by the orange/yellow voxels (depending on the limb type), and the robot by the purple/blue ones. [48][© 2025 IEEE].

An additional buffer was introduced to prevent the robot from rapidly switching between stopping and moving in borderline cases when the human is located directly at the robot's safety boundary zone and boundary voxels regularly alternate between occupied

and free due to slight movements. Therefore, this second safety module aggregates the output of the voxelmap occupancy over a fixed time window (currently 100 *ms*) and only releases the robot if all states in it are "free". This module has the additional advantage that other modules can also trigger a stop from outside the voxelmap architecture. A camera monitoring module, for example, could generate a stop if it detects that the camera has moved or is no longer providing current images.

The occupancy evaluation runs on a single Raspberry-Pi-4 and takes about 1-3 *ms* (at 5 *cm* voxel size). This indicates low computational complexity, and it therefore should also be possible to execute the algorithm on less powerful but safety-certified hardware.

3.3.4 System architecture

Figure 3.5 shows the overall software pipeline and how the three safety parts are combined. The KUKA robot interface *RSI* is used to receive the robot's position and to update the *override* speed to stop it. For communication between the other modules, *ROS2* [155] is used, and the internal system view can be visualized with *RVIZ* (Figure 3.6). The modular software architecture allows an easy exchange of the safety modules.

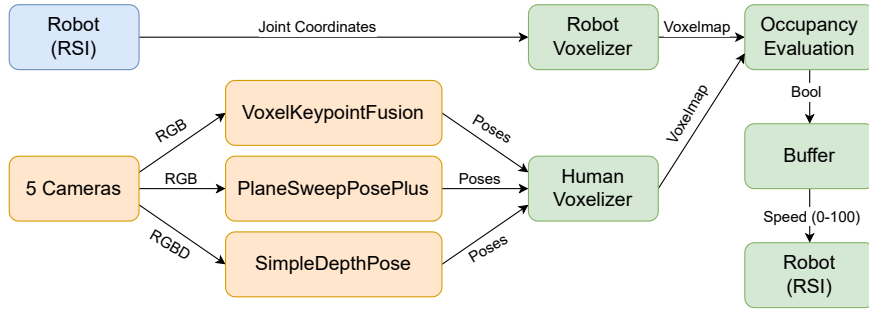


Figure 3.5. Software pipeline, with the three main parts colorized (blue=robot position, orange=human detection, green=distance monitoring). [48][© 2025 IEEE].

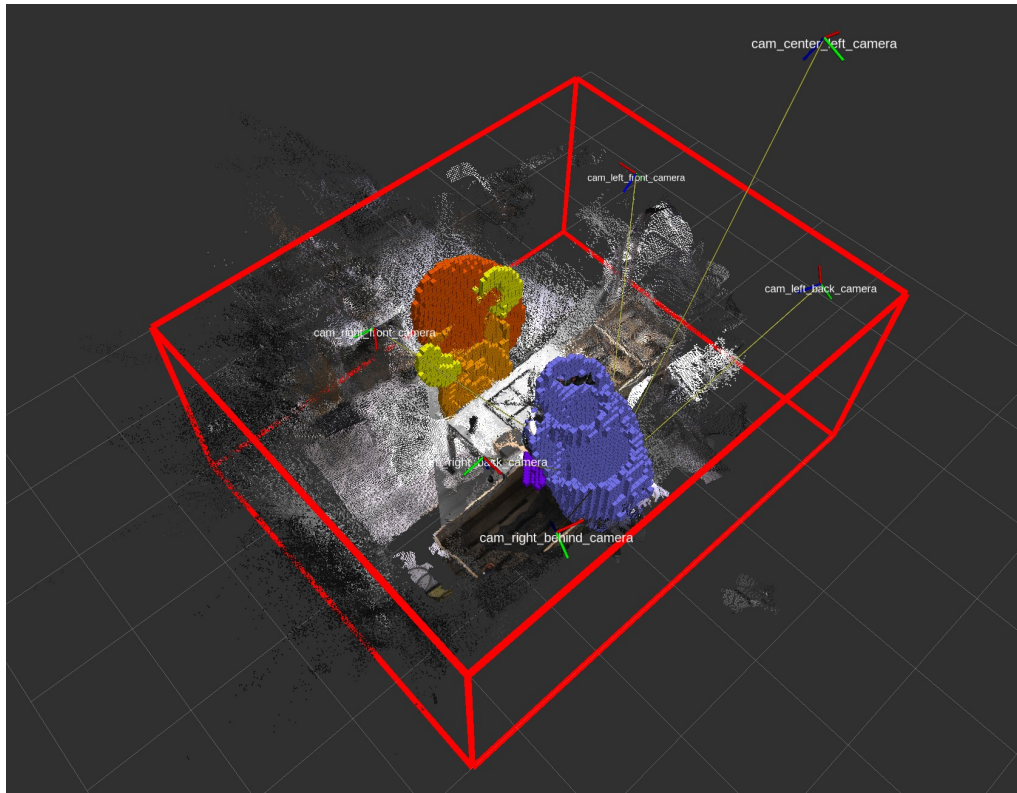


Figure 3.6. Internal visualization of the system using RVIZ. The red lines mark the border of the supervised room, the small dots are from the point-cloud output of the cameras, and their positions are indicated by the labeled coordinate systems.

3.4 Human Pose Estimation

To enable the robot to stop before it collides with a human worker, the core requirement is to detect the human's position first. In this project, the idea was to use images from cameras because they are cost-effective, can be easily mounted in the environment, and can provide extensive information about the human's position and pose. To detect the points of interest (here the human joints), AI-based machine learning methods were used, since they can work without explicit (and complex) feature modeling and can learn to detect the desired patterns from examples instead. The main requirement for such learned approaches is that they need to be trained on a large amount of data, but since human pose estimation is a very common task in computer vision, there are already datasets available for this purpose.

The following section gives an overview of human pose estimation from images in general, then describes the three developed algorithms in more detail, before finally comparing them against previously existing approaches. The section is additionally based on my following works, with some passages directly copied from them:

VoxelKeypointFusion: Generalizable Multi-View Multi-Person Pose Estimation. Bermuth, Daniel and Poeppel, Alexander and Reif, Wolfgang. Preprint. October 2024. [47]

SimpleDepthPose: Fast and Reliable Human Pose Estimation with RGBD-Images. Bermuth, Daniel and Poeppel, Alexander and Reif, Wolfgang. Preprint. January 2025. [50]

3.4.1 Pose Estimation Foundations

Before discussing the details of the developed algorithms, some fundamental concepts of human pose estimation need to be introduced first.

2D Estimation

Two-dimensional pose estimation focuses on predicting the positions of body keypoints in the image plane (Figure 3.7). Most methods follow either a *top-down* or *bottom-up* strategy. Top-down approaches first detect each person, using bounding-boxes for example, and then estimate their keypoints separately. This usually results in high accuracy because, particularly for smaller people in the background, the bounding boxes are automatically zoomed to a larger size. However, they become slower when many people appear in the scene. Bottom-up approaches first detect all keypoint locations in the image and then, in a second step, assemble them into individual poses. If there are many people, this requires a single model pass rather than multiple passes, making bottom-up approaches usually more efficient and better suited for crowded images. If multiple persons are very close to each other, the bounding boxes of a top-down approach might overlap and cause problems, for example, two boxes may merge into one, or duplicate joints in a box may be incorrectly assigned. The most commonly used dataset for this task is the COCO Keypoints dataset [142], which labels each person with 17 keypoints.

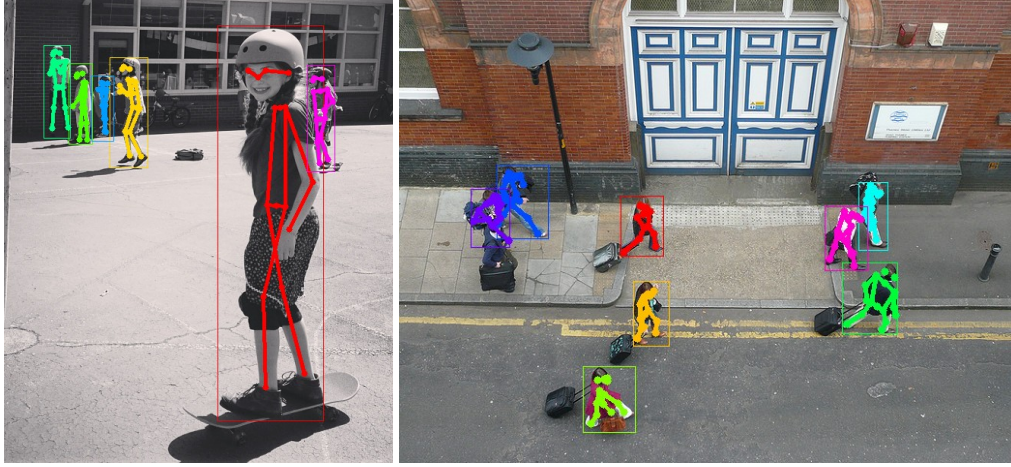


Figure 3.7. Examples of labeled poses from the *COCO* dataset [142].

The most common model architectures use heatmap outputs, which predict confidence maps for each joint (Figure 3.8), such as *ResNet* [250], *HRNet* [224], or *RTMPose* [126]. These heatmaps are then converted into joint coordinates, typically using a simple approach, such as taking the maximum value or calculating a weighted average.

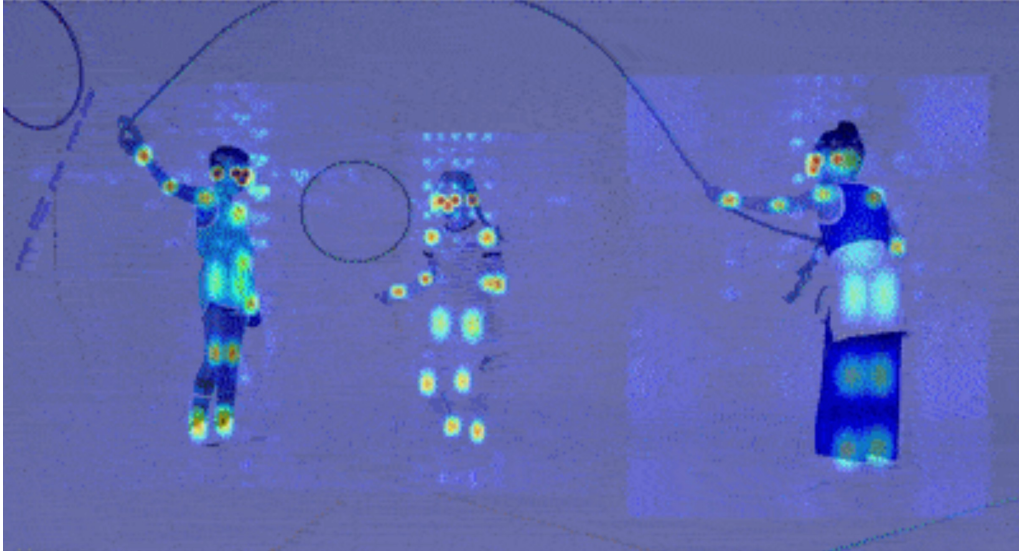


Figure 3.8. Example of a heatmap output for three persons. Image from [126]. The heatmap intensity shows the confidence of each joint. The light blue box areas visualize the bounding box cutouts. While all heatmaps are merged into one image here for visualization purposes, they are normally processed separately (so there would be 3×17 heatmap layers for the three person-boxes and the 17 joints).

3D Estimation

Three-dimensional pose estimation extends the task by predicting joint locations in real-world space, which introduces depth ambiguity and requires stronger geometric reasoning. There are several methods to estimate such poses, for example by attaching markers to the person's body, which can be tracked by special cameras, or by using

standard cameras and estimating the pose from their images. While the marker-based approach is generally more accurate, it is also more complicated because people usually need to wear special clothing [236]. This is not convenient and often not possible, for example when tracking people in public places, like a shopping store, or when special clothing is required, like in operating rooms. The markerless approach, on the other hand, is much more convenient, but also more difficult to compute, since poses must be estimated from images alone.

One common image-based approach is monocular lifting, where a 2D pose is first estimated and then converted into 3D using a learned model or geometric constraints. The main limitation of this approach is that it relies heavily on the accuracy of the initial 2D pose and cannot fully resolve depth ambiguities, especially in cases of occlusion or unusual poses. Multi-view methods overcome this limitation by combining information from several viewpoints. Using some sort of triangulation, the 3D joint positions can easily be calculated from the 2D detections in the different views, resolving the depth ambiguity. If the 2D detections in one image are not accurate, the other views can compensate for this, making the overall system more robust. It is also possible to use depth information from RGBD cameras, which can be used to directly calculate the 3D joint positions from the 2D detections, without the need for triangulation.

In all approaches, temporal information can be used to stabilize the predictions, for example by using a video-based input sequence. This allows for leveraging motion cues and maintaining consistent distances (such as limb lengths) over time, but often at the cost of increased computational complexity. Most monocular lifting approaches use a video-based input for this reason, whereas many multi-view approaches use only single frames, or add simple temporal output smoothing as a post-processing step.

Public Datasets

One of the most commonly used datasets for 3D pose estimation is the *Human3.6M* [122] dataset. It shows a single person performing different actions and is recorded by four cameras placed in the corners of a rectangular room (Figure 3.9).

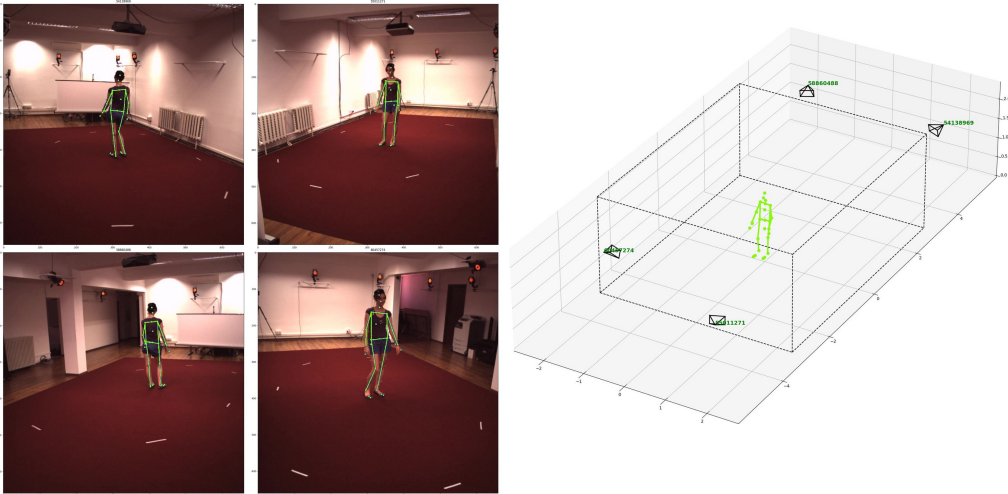


Figure 3.9. Example of a labeled pose from the *Human3.6M* dataset [122].

For multi-view and multi-person approaches, the *Shelf* & *Campus* [40] datasets have been widely adopted. *Shelf* shows multiple persons assembling a shelf from five different viewpoints, while *Campus* shows an outdoor scene with people walking and standing in front of a building, captured by three cameras (Figure 3.10).

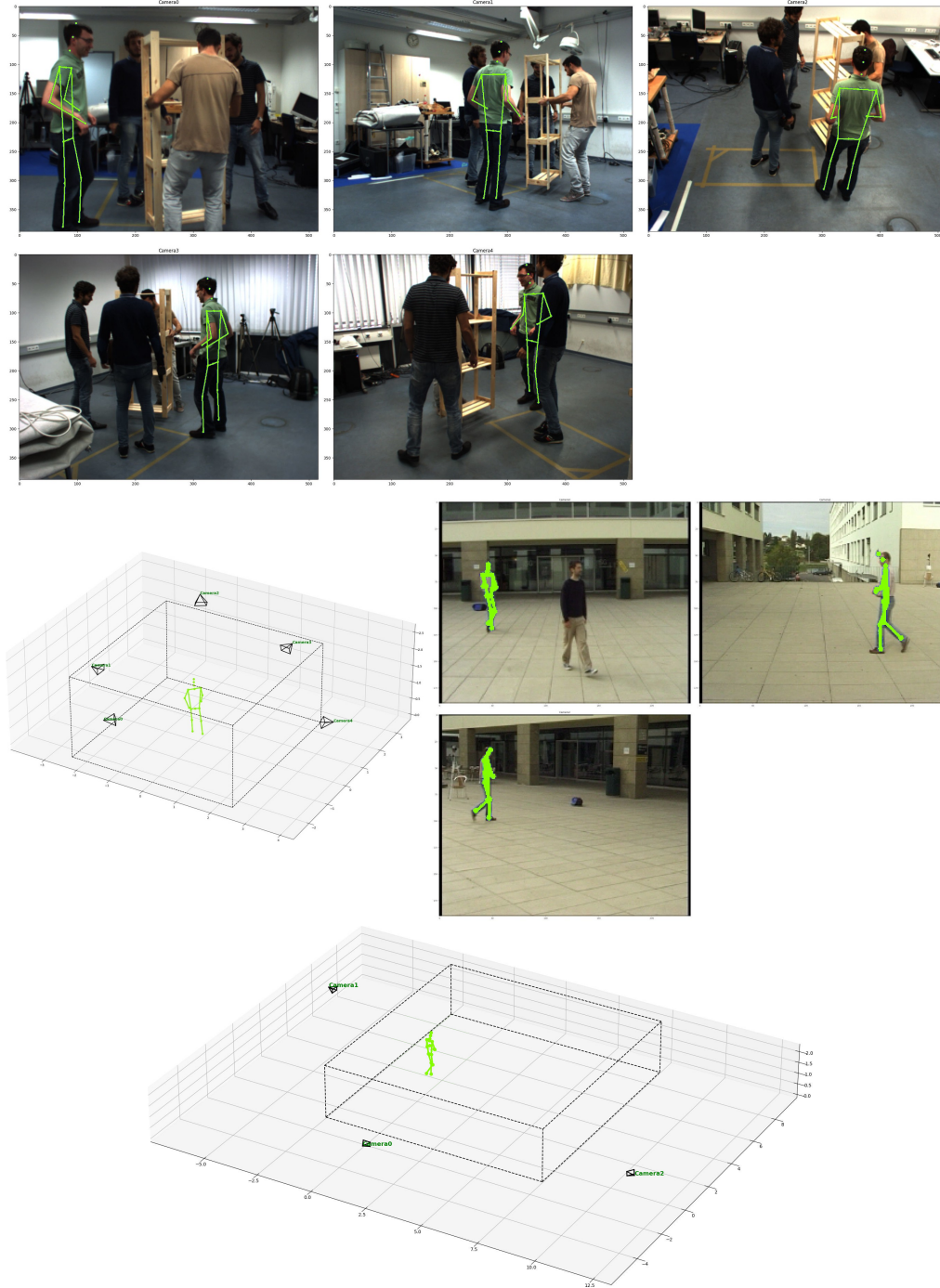


Figure 3.10. Examples of labeled poses from the *Shelf* & *Campus* datasets [40]. In both datasets only a subset of the persons are labeled.

Most recent learning-based multi-view approaches have been trained using the *Panoptic* [127] dataset. Its cameras are mounted in a dome-like structure and point to its center, recording groups of people executing different actions (Figure 3.11). The dataset is very large and has hundreds of cameras, but commonly only five of them are used.

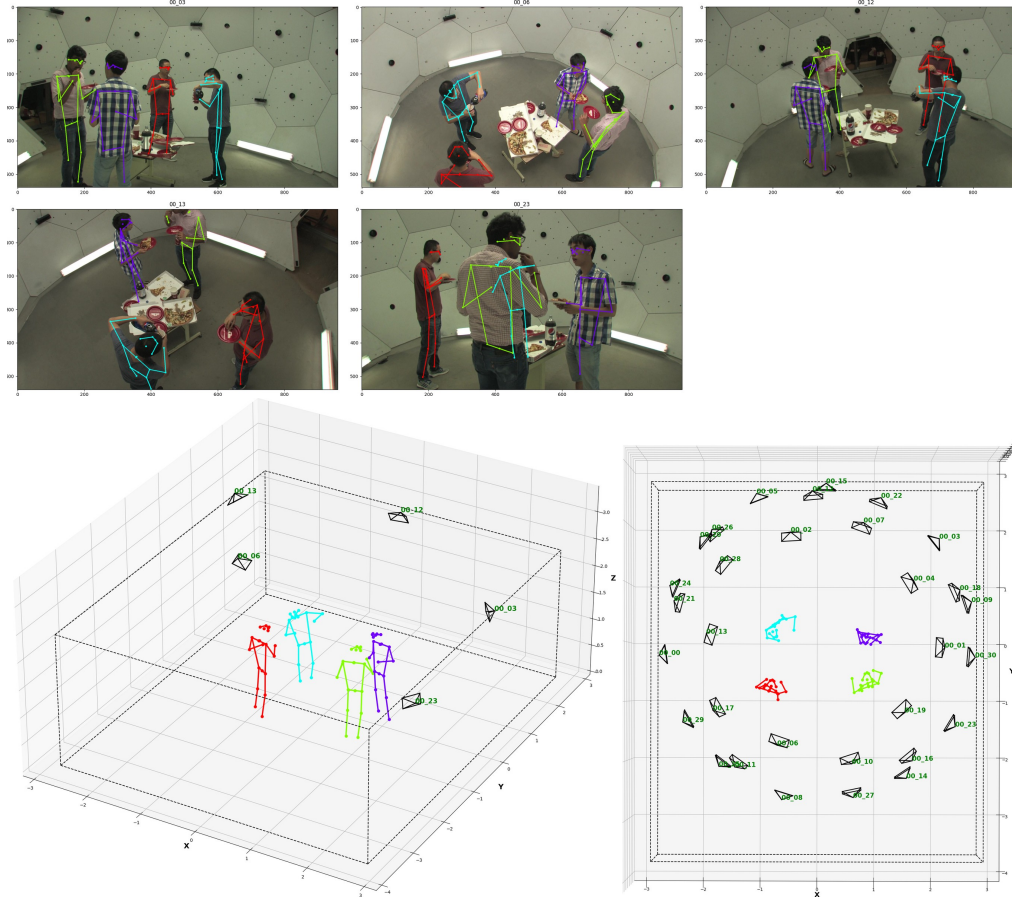


Figure 3.11. Example of a labeled pose from the *Panoptic* dataset [127]. From the 31 Full-HD cameras (in the lower right image) only five (lower left image) are commonly used. Each of the small black dots in the polygons of the walls in the upper image is another lower resolution camera, and the black bars between polygons are additional depth cameras.

The number of multi-view multi-person datasets that also contain depth images is quite low. One more often used in literature is *Multi View Operating Room (MVOR)* [221], which recorded an operating room from three different viewpoints (Figure 3.12). This is a relatively complicated dataset, since there is significant occlusion, and the persons all have similar clothing. Only the upper body of each person is labeled, and most, but not all, persons are labeled.

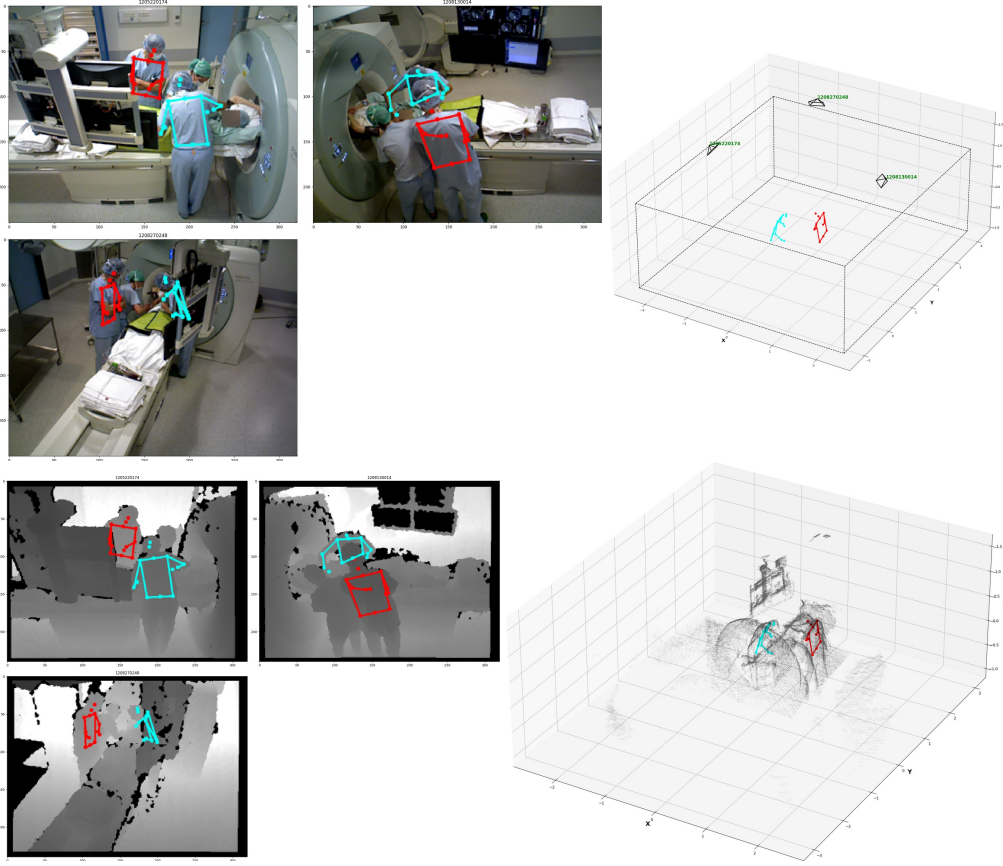


Figure 3.12. Example of a labeled pose from the *MVOR* dataset [221]. The lower images show the depth images and their combined point-cloud.

These five datasets cover a wide range of different scenarios, with each of them having their own interesting challenges. *Human3.6M* [122] is a very clean dataset, with only one person in the scene, and no occlusion, but contains a large variety of different actions. *Shelf* [40] shows multiple persons working together on a common assembly task, making it a relatively average use-case scenario. With *Campus* [40], the influence of few cameras together with large distances and outdoor lighting conditions can be evaluated. *Panoptic* [127] focuses on person interactions, often featuring many individuals in the scene, which results in additional occlusions and close proximity. And *MVOR* [221] can be used to evaluate the robustness of the algorithms in a very difficult scenario, as well as the integration of depth information. Together they make a good selection of datasets that will be used to comprehensively evaluate and compare the newly developed and already existing algorithms, and to identify their strengths and weaknesses.

3.4.2 skelda

Usually, most pose estimation methods have their own dataset-loading tools, which was also the case for the already existing approaches evaluated in the following. However, those datasets are often used in multiple implementations that therefore include many repeatedly implemented components. The goal of my *skelda* library was to extract this into an easy-to-use standalone solution that also contains viewing and evaluation tools. The concept follows some ideas from my audio library *corcua* [44] and splits the task into *readers*, *writers*, and *evals*. In addition, it also contains tools to visualize the dataset samples. The intermediate format is built from JSON-like dictionaries, to make it easy to understand the data, and to simplify adding new datasets as much as possible (Listing 1). This also allows merging labels from multiple datasets to create a larger mixed dataset. All public datasets used in this work (and several others) were integrated into *skelda*, as were the evaluation tools. The library itself was published as open-source software (<https://gitlab.com/Percipote/skelda>). All the evaluated algorithms were extended to use *skelda* to compare their performance on the different datasets.

```

1 {
2   'dataset_name': 'panoptic',
3   'scene': '160906_pizza1',
4   'index': 0,
5   'id': '160906_pizza1_00000000',
6   'cameras': [
7     {
8       'name': '00_00',
9       'type': 'color',
10      'K': [[1633.34, 0.0, 942.256], [0.0, 1628.84, 557.344], [0.0, 0.0, 1.0]],
11      'DC': [-0.220878, 0.189272, 7.8e-05, 0.00074, 0.041804],
12      'width': 1920,
13      'height': 1080,
14      'R': [[0.120147, -0.091578, 0.988523], [-0.992197, 0.022332, 0.122663], [-0.033309, -0.995547, -0.088181]],
15      'T': [[-2.71959], [-0.145746], [1.51792]]
16    }, ...
17  ],
18  'bodies3D': [[[-0.195, 0.015, 1.466, 0.724], [-0.405, -0.005, 1.631, 0.576], [-0.15, 0.042, 0.91, 0.362], ...], ...],
19  'bodyids': [0, 1, 4],
20  'num_persons': 3,
21  'imgpaths': ['/datasets/panoptic/sample/160906_pizza1/hdImgs/00_00/00_00_00000000.jpg', ...],
22  'room_size': [8.0, 8.0, 2.0],
23  'room_center': [0.0, -0.5, 1.0],
24  'joints': ['neck', 'nose', 'hip_middle', 'shoulder_left', 'elbow_left', 'wrist_left', 'hip_left', 'knee_left', 'ankle_left',
25    ↪ 'shoulder_right', 'elbow_right', 'wrist_right', 'hip_right', 'knee_right', 'ankle_right', 'eye_left', 'ear_left',
26    ↪ 'eye_right', 'ear_right', 'shoulder_middle', 'head']

```

Listing 1. Example of the intermediate format of the *skelda* dataset library.

3.4.3 Existing Approaches

Before developing new algorithms, it is essential to review existing approaches in order to understand their underlying methods. In certain scenarios, it is possible to reduce the task of estimating human poses to the prediction of a single person only [39, 77, 123, 263]. This greatly simplifies the problem because there is no need to match multiple joints to different persons having multiple possible options. However, since it allows a broader range of applications and, in the *KoARob* scenario, also allows a coworker to join the robot cell without breaking the safety system, the focus is on multi-person algorithms.

The standard approach to solving the estimation problem is a two-step approach, in which the first step is to estimate the 2D poses for each image, and the second step is to estimate the 3D poses from the combined 2D poses. The algorithms can further be classified by whether they solve the 3D estimation problem using an algorithmic or a learning-based approach. Further distinctions can be made by whether the algorithms make use of temporal information or not, and whether they use depth information or not. All of them have their own strengths and weaknesses, which can be seen in the later evaluations.

Many recent approaches are built on voxel-based representations, which have the advantage that they can easily model the spatial relationships between joints and can be processed with convolutional neural networks. The concept was introduced by *Iskakov et al.* [123] for the single-person case (Figure 3.13).

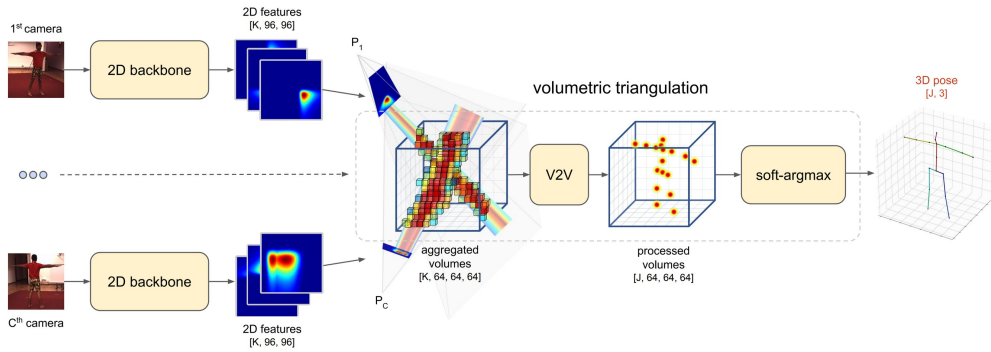


Figure 3.13. Concept of volumetric triangulation. The input for the method is a set of RGB images with known camera parameters. The 2D backbone produces intermediate feature maps that are unprojected into volumes with subsequent aggregation to a fixed size volume. The volume is passed to a 3D convolutional neural network that outputs the interpretable 3D heatmaps. The output 3D positions of the joints are inferred from 3D joint heatmaps by computing soft-argmax. All blocks allow backpropagation of the gradients, so the model can be trained end-to-end. Image and caption from [123].

The authors of *VoxelPose* [231] extended the work of *Iskakov et al.* [123] to multi-person estimations. They added an intermediate stage that does a voxel-based triangulation of the hip joints of all persons in the complete room first, to detect the center of each person, using the *V2V* network structure from [123]. Then they generate a smaller voxel space with finer resolution around each of those person centers, project the corresponding heatmap areas into it, and calculate the final joint locations with a second *V2V* network, same as in [123]. *Faster-VoxelPose* [258] is an improved version of *VoxelPose*, which uses a more efficient network architecture by reducing the 3D-voxel space to multiple 2D and 1D projections. *TesseTrack* [194] also builds upon the concept of *VoxelPose* and adds a time dimension to the voxel space, which allows for tracking persons over time and improving poses by using temporal constraints. Its source-code was not released. *TEMPO* [75] implemented this in a more efficient approach, building upon *Faster-VoxelPose* and tracking bounding boxes instead of adding an additional voxel dimension. *SelfPose3d* [222] is based on self-supervised training. It uses the basic architecture of *VoxelPose* again, but uses predicted 2D poses from a different off-the-

shelf model which are randomly augmented to train the 2D and 3D networks in a self-supervised manner using the additional multi-view information.

Even though most recent approaches are voxel-based, there exist some different concepts as well. *PRGnet* [248] is a graph-based approach that first detects human centers and then refines them with another graph network, which makes it significantly faster than a plain voxel-based approach. *MvP* [241] directly learns to regress the 3D joint coordinates from the 2D keypoint features of the images without an intermediate volumetric projection. *PlaneSweepPose* [141] first calculates a cross-view consistency score between the 2D poses of the different views and then regresses the depth in two stages, first a coarse depth estimation for the person and then a fine depth estimation for each joint. The authors published only parts of their code together with preprocessed inputs, which are enough to reproduce the results but not to use the approach on other datasets.

Regarding the algorithmic approaches, *mvpose* [89] splits the problem into two stages, first finding corresponding 2D poses across the images, by geometric and visual similarity, and then triangulating the matched poses. The method of *mv3dpose* [227] follows a graph matching concept to assign poses using epipolar geometry and also incorporates temporal information to fill in missing joints. *PartAwarePose* [76] uses the poses from the last frame to speed up the matching process and uses a joint-based filter to improve keypoint errors due to occlusions. *4DAssociation* [262] uses a graph-based concept to match body-parts and connections over space and time to assemble the 3D poses. Evaluating this approach was also attempted, but due to missing code and poor documentation, no way was found to create the required inputs using other datasets.

Due to the lower amount of training data and the greater technical requirements, only a few approaches have been developed that incorporate depth information. *OpenPTrack* [67, 167], which is often used in robotic applications, first estimates the 2D keypoints for each image and then uses the depth images to extract the corresponding distance of each joint. For each view, a 3D person proposal is generated which is then transformed into global world coordinates. All proposals are then assigned to persons, and a joint filtering and temporal smoothing step using a *Kalman-Filter* is applied. *MVDeep3DPS* [129] uses a learnable filter to drop invalid person proposals before merging them together in 3D space and refines them afterward using a calculated body-part confidence. *Ryselis et al.* [199] followed the simple approach of just averaging the 3D poses from the different views. *Hansen et al.* [103] predicted keypoint heatmaps from the depth images and used a point cloud to estimate the center of each person before projecting the keypoint heatmaps and the depth information into a voxelized space to generate a 3D pose using a *V2V* [165] network architecture similar to *VoxelPose*. Their code is not available. *PointVoxel* [179] is a recent work that follows a very similar concept, but instead of directly merging keypoint and depth voxelmaps, it has two *V2V*-branches and merges the results afterward. Its code was also not published.

In terms of generalization of the learning-based approaches to unseen datasets, a direct transfer is normally not tested, but *VoxelPose*, *Faster-VoxelPose*, *MvP*, and *TEMPO* implemented a finetuning concept with synthetic data. For this, the 3D poses from another dataset are randomly placed in 3D space and back-projected to the camera views of

the new dataset. Then those 2D poses, after applying some augmentations, are used to learn the 3D reconstruction again. *CloseMoCap* [212] takes this concept a step further by using a larger 3D pose dataset and implementing additional augmentations to better simulate occlusions and estimation errors, but its source-code was not released. The algorithmic approaches normally evaluate the transfer to a few other datasets, which is much easier since no training is required.

As can be seen later in Section 3.4.7, when comparing them on the aforementioned datasets, all these approaches have problems with generalization/transfer to new datasets. This is a major problem for a safety-related application like in the *KoARob* scenario, because the system has to work reliably from the start. It should not require a very time-consuming initial phase of recording and, in particular, labeling a new dataset for training. Therefore, three new approaches have been developed, with the focus on generalization across different setups and scenarios, while having a reasonable inference speed at the same time.

As an additional difficulty, the three new methods all need to have completely different architectures, as mentioned before, to increase the chances that if one fails to detect a person, the others still might detect it. For example, a top-down approach normally achieves better localization results, because it can cut out its bounding-boxes with higher resolutions, while, in contrast to bottom-up approaches, it is more likely to miss the complete person if only very few joints are visible and no box is predicted. Using depth information allows the accurate detection of persons even if they are visible in only one camera, but if the depth information is noisy, the resulting poses can be very inaccurate. A triangulation-based approach needs at least two cameras to detect a person, but can also estimate positions of occluded joints if the 2D models correctly predict/estimate those joints.

3.4.4 SimpleDepthPose

The first of my three new algorithms developed for the *KoARob* system is named *SimpleDepthPose*, and uses both color and depth information. It follows a very simple concept with the following steps:

1. Predict joint coordinates for each color image
2. Extract the distance of each detected visible joint from the aligned depth images
3. Group the 3D pose proposals into persons from the previous time-step, or create new ones if necessary
4. Filter outliers in each proposal group
5. Average remaining proposals to get the final 3D pose

To predict the joint coordinates in step (1), basically every off-the-shelf pose estimator can be used. One important requirement, however, is that the 2D pose estimation model is able to predict only directly visible keypoints, but no occluded ones, because they would result in extracting incorrect depth values. Here the *HigherHrNet* [72] architecture is used, trained and then finetuned on *COCO* [142] to predict only visible joints, and without the refinement step, because this was likely to add occluded joints again.

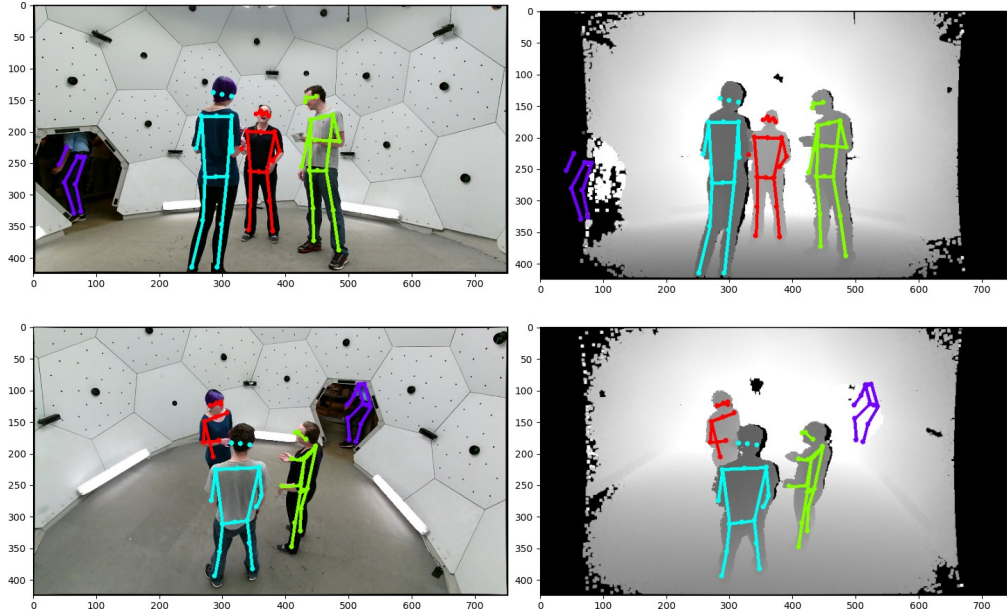


Figure 3.14. Example of a multi-person pose estimation from multiple camera views (from the *Panoptic* dataset [127]). The 2D pose detections from the color images (left), are projected into the depth images (right), so that a distance to the cameras can be extracted from them.

In step (2), the distance is extracted from the depth image (Figure 3.14) by calculating the median value of pixels around the joint coordinates obtained from the RGB-based 2D pose estimator. The depth values are selected using two rectangular cutouts that form a cross-shape, to emphasize the center region and reduce outliers that would occur at the square's edges (Figure 3.15). Afterward, a static per-joint offset is added to the distance, depending on the type of the joint, because a depth camera normally measures the distance to the surface, but the target location is the center of the joint. These offsets are estimated using normal human proportions. For example, 3 cm are added for shoulders and knees, or 1 cm for the wrists. To account for larger persons or (thick) clothing, those default values can be adapted. Finally, the poses are transformed from camera coordinates into world coordinates using the extrinsic calibrations.

In step (3), each 3D person proposal is then assigned to a person from the previous time-step by finding the closest match below a distance threshold. If no match is found, a new person is created. Old persons that were not matched are dropped after a certain number of frames.

Following this, a simple outlier removal step (4) is applied. The filter calculates the distance of each joint proposal to the averaged center of its neighboring joints, and if it is above a threshold, it is discarded. In the case of a knee, for example, the neighbors are the hip and ankle. The idea is to remove proposals that are very far from the other joints, creating impossibly long limbs, which are likely to be wrong. Therefore, the threshold (default 0.5 m) should cover all limb lengths of normal-sized persons ($< 2\text{ m}$).

Then, in the last step (5), if there are enough proposals for a joint, a center between them is calculated, and only the *top-k* (default 3) closest proposals to this center are



Figure 3.15. Visualization of the cross-shape used to extract the depth value for each joint in a zoom-in of the depth image. All pixels inside the cross are used to calculate the median depth distance.

averaged into the new joint location. Wrong joint proposals are either caused by poor keypoint predictions or by errors in the depth image (especially at object edges), both resulting mostly in proposals far from the correct location of the joint. If they were not already removed by the previous outlier filter, they are likely to be far from the center of the remaining proposals and therefore not included in the final average. See Figure 3.16 for an example of the proposals and their fused result.

In a direct comparison with *OpenPTrack*, which is architecturally closest, *SimpleDepthPose* is much simpler, but, as can be seen later in Section 3.4.7, also more accurate. Two key differences in distance estimation are the consideration of joint visibility, which is very important to avoid wrong locations due to occlusions, and the improved concept for depth extraction. Another difference is the filtering and merging approach. *OpenPTrack* uses a *Kalman-Filter* to smooth the joint locations, which can be affected by far-off outliers and can lag behind in fast motions. *SimpleDepthPose*, on the other hand, only uses the joints from the last time-step, the other views, and the joint's direct neighbors to filter outlier joints before merging the remaining proposals into the final results. Unlike the other closed-source depth-based approaches, no learning is required, so it can be directly applied to new camera setups.

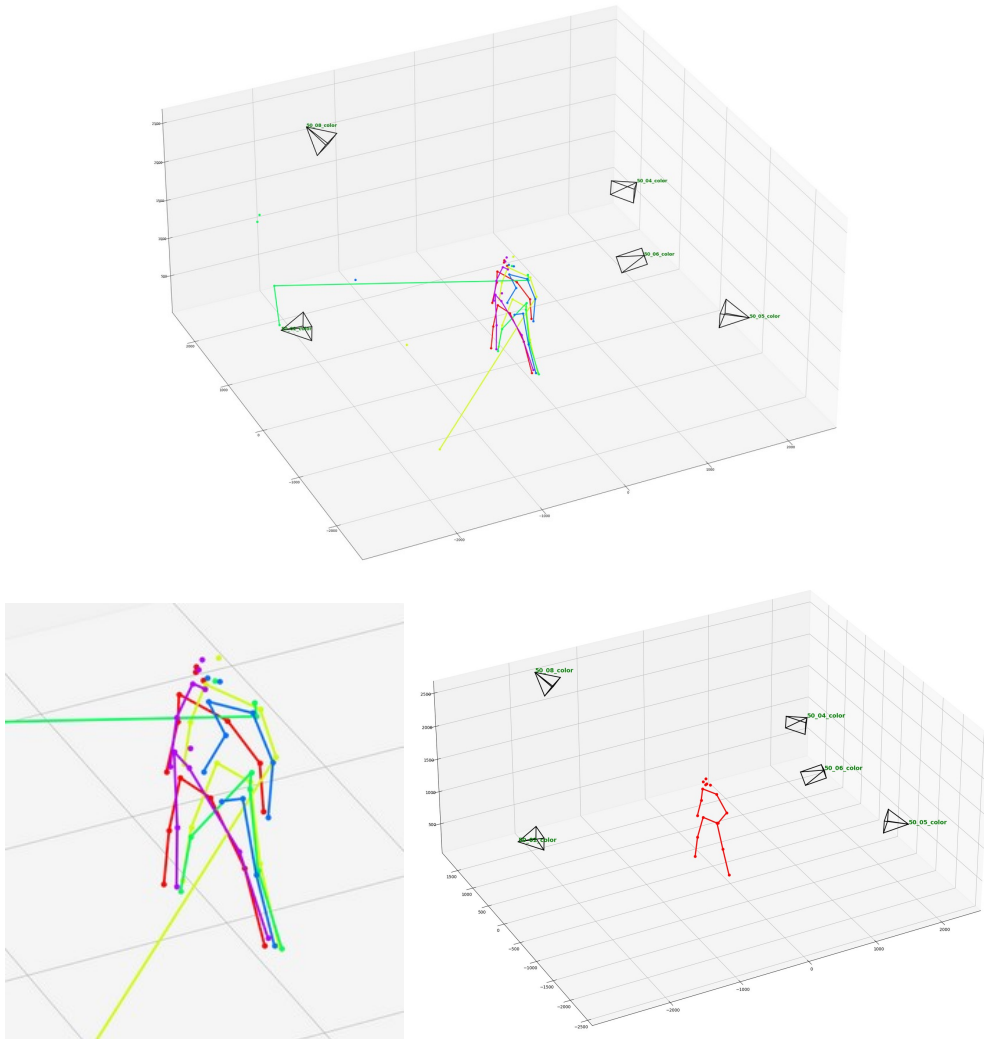


Figure 3.16. Example of the proposals for each view with some joint errors (top), a zoom-in on the per-view poses (left), and their fused result with the final joints (right).

3.4.5 VoxelKeypointFusion

The second new algorithm, *VoxelKeypointFusion*, follows a learning-free voxel-based algorithmic concept. It can be split into two stages as well, with the first one predicting the 2D poses for each image. For this, any 2D pose estimator can be integrated. Here, in accordance with the previously highlighted diversification aspect, *RTMPose* [126] was used. The second stage can be split into the following steps:

1. Generate joint heatmap images for each view
2. Generate person identity images for each view
3. Project all heatmaps into a shared voxelized space
4. Normalize by the number of camera views
5. Find peaks above a threshold for each joint as keypoint proposals using non-maximum-suppression
6. Reproject each proposal into all person identity images and get the person-id at this location if present
7. Group proposals with the same person ids together
8. Merge groups with a large overlap of the person ids
9. Calculate the center of each group and remove outliers
10. Using the best keypoint proposals, build a person from center to outer limbs
11. Drop persons with too few keypoints

In step (1), the keypoints of the 2D pose estimators are used to generate Gaussian heatmap images for each person in the view, which are then combined into a single heatmap. In principle, one could also use heatmaps directly from the pose estimator, but not every model has them, and this also would require additional implementations for every estimator.

In step (2), images of the same shape are created, but instead of keypoint scores, they contain a person-id everywhere the heatmap score is not zero. The ids are generated by simply counting the total number of persons in all images, and every 2D person is assigned a different number (the order does not matter). This can be seen on the two cube walls in Figure 3.17 (a+c). The idea of this step is that the information of which joint belongs to which person is already present in the standard 2D pose estimations. All previous voxel-based approaches do not use this information, but rely on the later neural networks to learn to associate the joints in 3D space again. The association of which person is who in the different views is not relevant yet, this will be solved later in 3D space, only the already available joint-person association is stored in the person-id images.

The voxel-based projection (default resolution $50mm$) in steps (3-5) itself is quite similar to the other voxel-based approaches, except there is no additional neural network later on, no learnable updating/refinement of the scores, and only a single voxel-room is used (see Figure 3.17, b). The Gaussian heatmap beams overlap at certain points, basically everywhere a joint is seen from at least two views, and sometimes with beams from other persons. The score of each voxel is calculated as the average of the heatmap

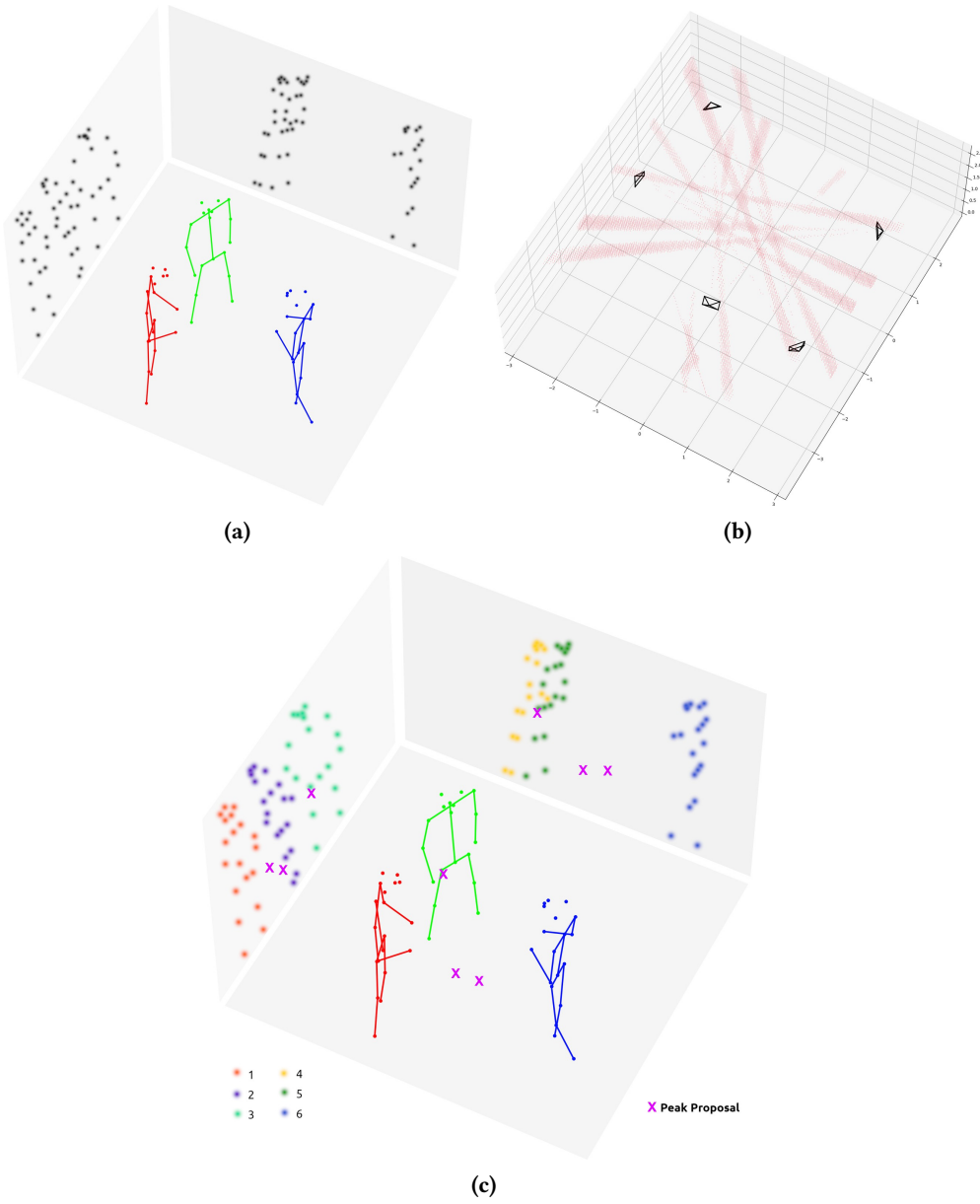


Figure 3.17. Obtaining keypoints and person ids. First, the heatmaps of each view (a, schema for two cameras, the 3D-skeletons are drawn only for better visualization) are projected as beams into the voxelized room (b, five real cameras, only a single hip joint visualized). Then peaks at overlapping beam positions are searched and retrieved as keypoint proposals. Those proposals, here visualized as pink crosses (the three in the center of the image, between the three persons), are projected into each original image (c, with the two cameras from a, for better visualization they do not match to overlaps in b or a). The person association is then gathered from the person-id images (c, color-coded and color-paired only for better visualization, so every pixel with a red color has id=1). The person-ids of these points are extracted, in this case, one proposal has ids {3,5}, and the other two received no ids and thus are discarded. The same is done with the proposals from different joint types (so there could be a right shoulder with ids {3,5}). All proposals with the same id-set are then collected into a person group, which might be merged with other overlapping groups (for example if the left shoulder only got {3}).

beams, so overlapping beams lead to higher scores, and strongly overlapping beams lead to higher scores than poorly overlapping ones. Each voxel above a certain threshold (after non-maximum-neighbor-suppression) is selected as a keypoint proposal. While the voxel-representation reduces the state-space and simplifies the computation, it also introduces a discretization error, because the keypoint proposals are only located at the center of the voxels. The location of each selected keypoint is therefore adjusted with a weighted average of the neighboring voxels, to reach a sub-voxel precision.

In step (6), each keypoint proposal is reprojected to the input views and obtains a set of corresponding person indices from the person-id images if they fall onto a 2D person keypoint (see Figure 3.17, c). If the proposal does not match any person, for example when beams from two different persons overlap somewhere in the space, it is discarded directly. The association to persons then happens in steps (7+8). Ideally, the keypoint proposals of one person all receive the same set of ids and can be clearly matched together. Since this is often not the case, due to occlusions by other persons, which lead to the assignment of wrong ids, groups where the majority of the person ids overlap are merged together, thus ignoring single wrong assignments.

In steps (9-11), one person for each group is created by first taking the best proposal for each keypoint to calculate an initial person center. Then the center is improved by ignoring outliers far away, and then using this second center all keypoints with too large distance in 3D space (above $1.3m$) are dropped. After this creation of a refined person center, the proposals for the torso joints with the best scores are taken to build an initial person. This person is extended with limbs, by first dropping far away outliers again (above $0.6m$ from the parent joint, since the limb lengths of a person are physically restricted) and assigning the remaining best keypoint proposal. At last any persons with too few keypoints (less than 3) are dropped as incomplete, and the remaining persons are returned as the final result.

In a direct comparison with *VoxelPose*, the new *VoxelKeypointFusion* works without any learned part. The voxel-based triangulation already allows finding good keypoint proposals, just through the geometric overlap of the heatmap beams, so the subsequent neural network is not necessary. In contrast to the top-down approach of first detecting persons, then their joints, a bottom-up strategy is used here, which first detects all joints, and then assembles them into persons. This is especially helpful in settings with stronger occlusion, because the risk of totally missing a person is lower. With the new concept of adding person-ids, already present information about the association of keypoints from standard 2D detection models is used, which allows skipping the person detection step. While the previous voxel-based approaches partially integrate the architecture of common 2D pose estimators for their 2D part, they train them in a way that they only output heatmaps, without any person association. Besides the fact that the standard models already output the association of joints to persons, they normally also have a much larger number of parameters and a deeper layer structure than the later 3D voxel networks, so it is expected that they can learn the rather complex association better than the smaller and more shallow 3D networks. In contrast to previous algebraic/line-based triangulation approaches, the voxel-based approach allows partially overlapping heatmap rays, which helps to find a good center between

slightly erroneous 2D predictions. The persons are also assembled directly in 3D space, which allows simpler outlier filtering steps.

For *VoxelKeypointFusion* a simple depth-based masking concept was implemented as well. After generating a point cloud from the depth images, all points that fall into one voxel are counted, and if there are more than a small threshold (to drop noisy points), this voxel is considered filled. The projection space is then masked with the filled depth voxels, to drop all projection voxels that do not match to a real object. This especially helps to remove any voxel-peaks that are generated from 2D detections that do not show the same person but whose heatmap projections still overlap somewhere in 3D space. An example of the masking can be seen in Figure 3.18. The depth views do not have to be aligned to the color views, but can be recorded from any direction and also come from a different number of sensors.

3.4.6 PlaneSweepPosePlus

Regarding the different architectures that are intended, the options for the third algorithm have become limited. *SimpleDepthPose* predicts directly visible 2D keypoints using a bottom-up approach, and then extracts the corresponding distances from depth images. The resulting 3D poses per view are merged, and, after dropping outliers, averaged to get the final 3D poses. *VoxelKeypointFusion* uses a top-down approach to first detect humans in each 2D image by their bounding boxes and then their joints (keypoints) in the box cutouts. The heatmap representations of the keypoints are then triangulated in voxelized space to estimate 3D joint proposals at overlapping positions, optionally filtered using depth information. The resulting proposals are then merged and assigned in a bottom-up approach to complete persons. So the first two algorithms already cover a wide range of different concepts, and the third one needs a different 2D model, and also should not rely on directly extracting depth information, or using (voxel-based) triangulation.

Therefore, a good option for the 3D part was found in *PlaneSweepPose* [141], which, of the remaining previous works, has the most conceptual differences to the first two algorithms. It is a learning-based approach that estimates the depth of a joint in two stages. First, a coarse estimation for the whole person, and then a fine estimation for each joint, in combination with cross-view consistency scoring.

Since the authors only published parts of their code, a new 2D pose estimation model had to be integrated to make it end-to-end runnable. For this, *RTMO* [151] was used, which is a fast bounding-box-based approach to detect the locations of 2D persons with their joints in a single step. Additionally, initial experiments showed that *PlaneSweepPose* was likely to predict clearly invalid persons. Therefore, concepts from the post-processing steps of *VoxelKeypointFusion* were integrated to filter them. The combined approach was named *PlaneSweepPosePlus* and is the third algorithm created for *KoARob*.

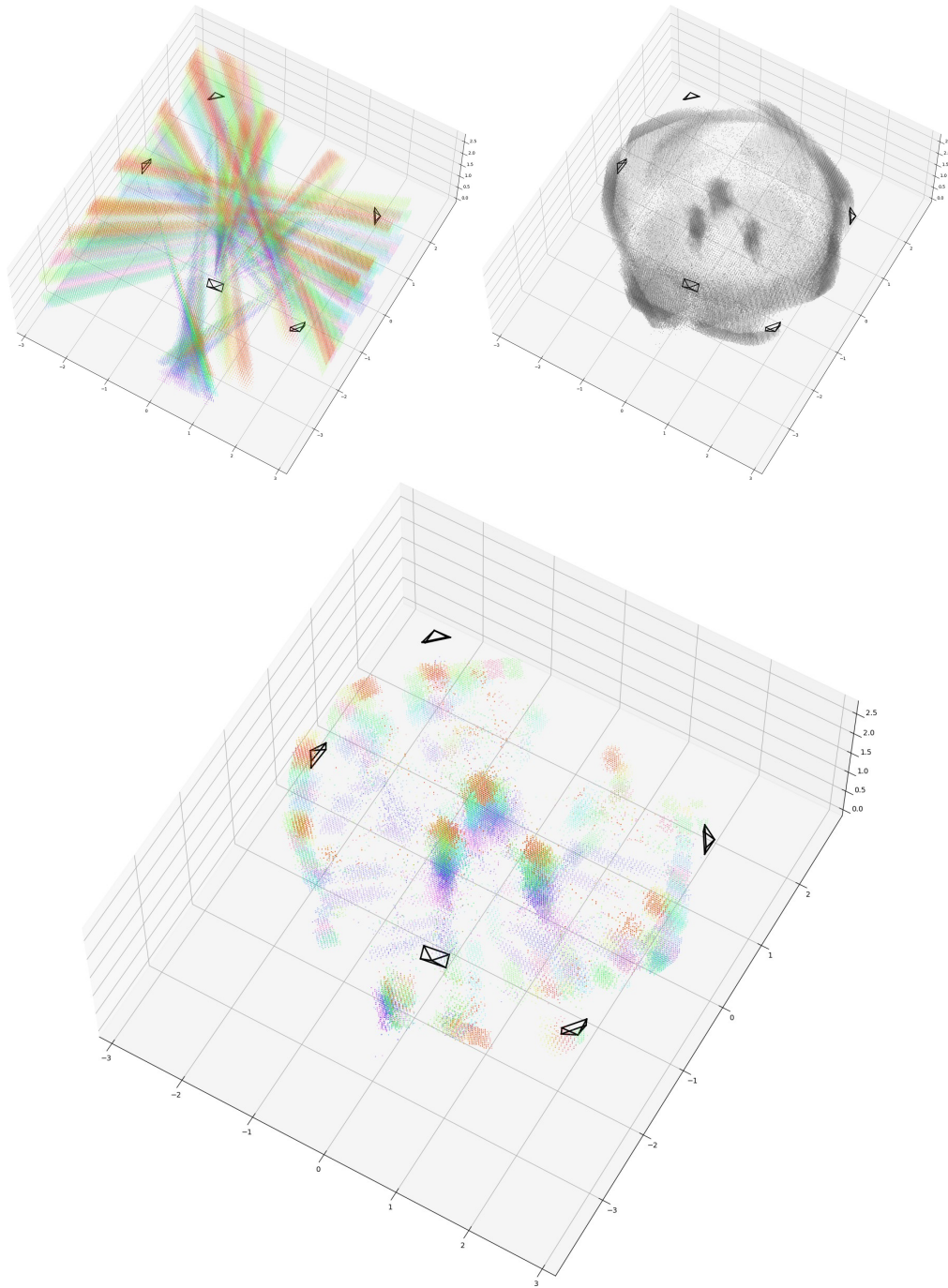


Figure 3.18. Example of depth masking in *VoxelKeypointFusion*. In the top-left image, the keypoint heatmap beam projection of four persons from five camera views into a voxelized room is shown, with one color for each joint type. This is the default input for the peak proposal calculation. The top-right shows the voxelized depth images, in which three persons (the fourth is walking through the entrance) and the room’s wall are clearly visible. On the bottom, the projection was masked with the depth voxels. The three persons in the center of the room are now clearly visible in the projection space as well, and no peaks can be proposed between the persons anymore. Parts of the sphere’s wall are still left, but since the keypoint projections do not overlap well there, no peak proposals are generated from those.

3.4.7 Comparison

To evaluate the performance of the different algorithms, they were compared on the five datasets described previously. In contrast to previous works, this represents a much broader evaluation, as typically only two or three datasets and only a few metrics have been used to evaluate the performance of a new approach. Using more datasets and metrics allows a better assessment of the generalization capabilities of the approaches, and also increases the possibility of finding specific weaknesses. Since for the targeted application a good performance in a previously unseen scenario is crucial, the algorithms were not trained on these datasets (most of them are too small for that anyway), but instead directly applied to them.

For the evaluation, multiple standard metrics from the computer vision domain and previous works were used. *Percentage of Correct Keypoints (PCK)* calculates the percentage of keypoints detected with an error below the given threshold in millimeters (for example *PCK@100* counts the fraction of all keypoints with an error below 100 mm). The *Mean Per Joint Prediction Error (MPJPE)* calculates the mean error of all joints for each person and then averages over all persons, dropping persons with an error above 500 mm as not matched. In total 13 keypoints are evaluated (2 shoulders, 2 hips, 2 elbows, 2 wrists, 2 knees, 2 ankles, 1 nose/head). The *Recall* shows the percentage of persons with an average joint error relative to the ground truth below the given threshold. Note that a prediction is always matched to the closest ground-truth person, and in case it gets matched to a second ground-truth person only the better match is kept, whereas the second ground-truth label is counted as not matched. This varies slightly between the evaluations in other works. *Invalid* counts the percentage of predictions that were not matched to any ground-truth label, and the *F1* score combines it with the *Recall@500* score into a single value. *FPS* states the end-to-end (images-to-poses) inference speed, evaluated on a single graphics card, with a batch-size of 1. In most cases, an *Nvidia-3090* was used, but for *mvpose*, *mv3dpose* and *PartAwarePose* only an *Nvidia-1080Ti* was working because some of their libraries did not support the newer graphics card.

Dataset: Human3.6M

For the first generalization experiment, the *Human3.6M* [122] dataset was evaluated. The learning-based algorithms were first directly evaluated on this dataset, using their pretrained models from the *Panoptic* [127] dataset, and then retrained with synthetic poses for the new camera positions using their included implementations. The algorithmic approaches can be directly transferred, since they do not require extra training. The evaluation frames are from the subset *S9* (commonly *S11* is used as well, but since it has different camera parameters, one would need to run every synthetic training a second time, which was not feasible). To reduce the benchmark size to a level similar to the other ones, the images were sampled at a frame rate of 10Hz with around 3s of consecutive frames and then a larger gap before the next 3s of input.

In Table 3.1 it can be seen that *VoxelPose* and especially *MvP* miss most persons if they are directly transferred to the new setup (*Recall@500*), while *Faster-VoxelPose* and *PRGnet* show much better reliability, but also a relatively high joint prediction

Method	PCK@100/500 (%)		MPJPE (mm)	Recall@100/500 (%)		Invalid (%)	F1 (%)	FPS (Hz)
VoxelPose	15.6	37.3	175	7.0	39.3	11.3	54.5	10.8
Faster-VoxelPose	56.2	94.2	152	58.8	100	69.2	47.1	33.9
MvP	0	0.1	481	0	0.3	99.8	0.2	9.1
PRGnet	52.3	97.2	141	27.5	100	1.0	99.5	13.4
TEMPO	68.6	87.5	85.5	68.5	88.3	3.3	92.3	12.4
SelfPose3d	49.8	86.0	118	38.2	87.3	77.8	35.4	7.6
mvpose	65.0	80.6	79.3	66.0	81.3	16.6	82.4	0.3
mv3dpose	48.1	59.5	99.8	47.3	61.0	0	75.8	2.7
PartAwarePose	81.6	92.6	64.3	83.0	93.0	0.4	96.2	4.3
VoxelPose (5 random)	25.7	50.1	124	16.0	50.2	26.9	59.5	10.2
VoxelPose (4 similar)	30.7	59.0	128	19.8	60.2	0.6	75.0	10.4
VoxelPose (synthetic)	74.8	98.2	92.0	70.8	100	1.5	99.3	16.0
Faster-VoxelPose (synthetic)	75.5	98.8	88.3	75.0	100	0.2	99.5	36.2
PlaneSweepPosePlus	88.3	98.1	61.0	93.2	98.2	0	99.1	10.2
VoxelKeypointFusion	81.7	100	64.3	95.0	100	0	100	8.7

Table 3.1. Transfer to *human36m* [122] without and with viewpoint training.

error ($MPJPE/PCK$). In the untrained *MvP* model, the number of cameras could not be reduced to four, so as a workaround, the first camera was duplicated (in experiments with other datasets this only had a small influence and is not the reason for the poor transfer performance). Interestingly, the algorithmic approaches had problems finding some of the persons, but achieve a better $MPJPE$ than most of the learned algorithms. Only the newly proposed *VoxelKeypointFusion* shows a reliable performance, while also having a decent prediction speed. *PlaneSweepPosePlus* achieves a slightly better $MPJPE$, but also misses a few persons. *SimpleDepthPose* is not included in this evaluation, since it relies on depth images, which are not available in this dataset.

Before training *VoxelPose* with synthetic labels to the new camera setup, a transfer with the original *Panoptic* [127] dataset was tried. Since this dataset has a large number of cameras, one training with five randomly chosen cameras in each training step, and another one with four fixed cameras with a similar room distribution as in *Human36m*, was executed, but both concepts did not improve the performance much. In contrast to this, training with synthetically placed poses and their reprojections to the matching camera views greatly improved the performance. The synthetic training with *Faster-VoxelPose* significantly improved most metrics as well, especially the number of invalid predictions. A finetuning of *MvP* and *TEMPO* was not possible due to unpublished tools and missing documentation about how to generate their synthetic data inputs. The same applies to *PlaneSweepPose*, but since in this case the retraining is more about changed distances to the cameras, the pretrained *Panoptic* model, which has a similar distance distribution, works very well. *SelfPose3d* followed the concept of *VoxelPose*, using data from the *Panoptic* dataset instead of self-supervised data for other datasets than *Panoptic* itself, but those parts were not published either (the reported results in the paper for two different datasets were slightly worse compared to the ones of *VoxelPose*). In their original paper *TEMPO* reported a transfer $MPJPE$ of 63.0 mm [75]. Reasons for the lower performance here may be differences in the data sampling and the published training weights, which, as can be seen later, already showed a lower performance in the *Panoptic* dataset.

Datasets: *Shelf* & *Campus*

The second evaluation was run with the *Shelf* & *Campus* [40] datasets. Both datasets are commonly used for generalization tests and were also used in most models evaluated here, but normally with fewer metrics.

Method	PCK@100/500 (%)		MPJPE (mm)	Recall@100/500 (%)		Invalid (%)	F1 (%)	FPS (Hz)
VoxelPose	78.8	91.6	63.0	88.9	91.6	57.6	58.0	7.7
VoxelPose (synthetic)	86.3	99.4	66.7	91.0	100	48.2	68.3	6.5
Faster-VoxelPose	88.3	100	59.8	99.4	100	50.7	66.0	17.6
Faster-VoxelPose (synthetic)	87.9	100	58.8	99.0	100	48.8	67.7	17.4
MvP	3.2	11.0	415	0	15.1	95.3	7.1	8.7
MvP (synthetic)	94.1	99.7	51.8	97.1	100	82.2	30.2	8.5
PRGnet	86.5	100	64.3	98.7	100	53.3	63.7	6.3
TEMPO	86.4	97.5	57.8	95.6	97.5	50.1	66.0	5.6
SelfPose3d	86.1	97.5	59.9	95.8	97.5	76.1	38.4	4.9
mvpose	88.4	100	57.3	98.1	100	51.8	65.0	0.1
mv3dpose	91.4	98.4	55.8	94.8	98.5	44.3	71.2	1.6
PartAwarePose	92.7	99.0	51.4	98.5	99.2	47.4	68.7	2.1
PlaneSweepPosePlus	89.2	100	55.2	98.7	100	51.4	65.4	8.9
PlaneSweepPosePlus (syn.)	88.5	100	56.6	97.5	100	49.9	66.8	8.8
VoxelKeypointFusion	93.3	100	51.3	98.3	100	49.1	67.4	5.8

Table 3.2. Transfer to *Shelf* [40].

In Table 3.2 it can be seen that the learnable approaches show similar performance on the *Shelf* dataset compared to the algorithmic ones. They are notably faster than most others, which is mainly caused by faster 2D pose estimators, but they tend to predict more invalid persons. Note that in the dataset itself not all occurring persons are labeled, and therefore the percentage of invalid predictions cannot be zero, but the algorithmic models can detect all labeled persons while predicting a lower total number of persons. *MvP* shows an especially poor performance in this regard, even with the synthetically finetuned model, making it essentially unusable for real-world applications (the preprocessed data and tools for *Shelf* & *Campus* were published so it could be trained here). While *VoxelPose* benefits from the synthetic training, the performance difference for *Faster-VoxelPose* is only marginal or even slightly worse in some metrics. The camera setup of *Shelf* is quite similar to the *Panoptic* dataset, which is one reason why the directly transferred models show a decent performance. For *PlaneSweepPosePlus* it even results in an overall slightly lower performance, so for setups with similar camera distances, synthetic training does not seem to be recommendable. In all cases, the models were evaluated end-to-end, instead of using the provided external heatmaps created by unpublished tools, since the generalization of the complete model is of primary interest. Overall, almost all models show a good performance on this dataset, with different models having different strengths and weaknesses in different metrics. Both of the new models are among the top performing architectures here.

On the *Campus* dataset (Table 3.3), most learned algorithms predict even more invalid persons. The algebraic methods generally outperform the learning-based ones on this dataset. *PRGnet* does not seem to generalize to this setup very well, but implementing a similar finetuning concept as the others have might be able to help in this case. The synthetic training of *PlaneSweepPosePlus* significantly improved the performance,

Method	PCK@100/500 (%)		MPJPE (mm)	Recall@100/500 (%)		Invalid (%)	F1 (%)	FPS (Hz)
VoxelPose	1.2	10.7	248	0	11.2	58.8	17.6	17.9
VoxelPose (synthetic)	40.8	80.7	111	26.1	80.9	27.6	76.4	12.9
Faster-VoxelPose	24.3	92.0	142	4.3	92.6	48.3	66.3	23.9
Faster-VoxelPose (synthetic)	41.3	74.4	104	39.6	74.7	2.1	84.8	25.3
MvP	1.2	5.7	301	0	6.9	95.3	5.6	8.9
MvP (synthetic)	64.0	89.3	113	63.6	92.0	80.3	32.4	8.9
PRGnet	0.3	8.2	333	0	9.6	53.2	15.9	16.8
TEMPO	29.8	54.4	106	22.6	54.5	51.5	51.3	7.3
SelfPose3d	6.7	36.9	207	0	39.1	88.2	18.2	6.8
mvpose	70.2	99.9	80.4	82.7	100	25.4	85.5	0.5
mv3dpose	64.4	93.4	135	62.5	94.9	10.1	92.4	2.8
PartAwarePose	78.4	98.9	74.7	93.9	98.9	22.7	86.8	5.8
PlaneSweepPosePlus	30.2	52.3	169	2.4	56.6	65.9	42.6	12.4
PlaneSweepPosePlus (syn.)	73.7	98.3	79.9	89.9	98.7	34.3	78.9	12.7
VoxelKeypointFusion	74.6	99.9	84.4	80.3	100	35.5	78.4	7.8

Table 3.3. Transfer to *Campus* [40].

making it one of the leading methods here, but it is slightly behind *VoxelKeypointFusion* in terms of reliability, as it misses a few persons again. Both methods also have a problem with predicting too many invalid persons. *PlaneSweepPosePlus* often adds extra poses if a person is not visible in multiple views, which is regularly the case in this dataset, and not all of them can be filtered out by the added post-processing steps. In the case of *VoxelKeypointFusion*, it is mostly caused by persons standing directly in front of each other from the camera perspective, thus leading to person groups that are not merged well and resulting in more person outputs. This seems to be a general problem of the voxel-based algorithms. While all learned models improved their performance through synthetic training, they still do not detect all persons, and generally less than the algorithmic ones, which indicates potential issues in their person-proposal algorithms. It is also complicated to declare a single best approach here since all algorithms have strengths and weaknesses.

Dataset: *MVOR*

On the complicated operating room dataset *MVOR* [221], it can be seen that many models are struggling with this setup. In Table 3.4 only a few of them, including the three new ones, *VoxelKeypointFusion*, *PlaneSweepPosePlus*, and especially *SimpleDepthPose*, reach a decent performance on this dataset.

The problem is mainly caused by the many occlusions, which result in some persons, or most parts of them, being visible in only one image. All triangulation-based methods consequently struggle to detect such persons at all. When additionally using depth information, on the other hand, one view can be enough to correctly detect them. In comparison between *VoxelPose* and *Faster-VoxelPose* after synthetic training, it appears that the person-proposal step of the first one is more robust to strongly occluded lower bodies, but on the other hand, it tends to predict much more invalid persons. *OpenPTrack* is quite slow, which is partially caused by the processing of multiple views on a single device and partially because of a motion lag due to the joint smoothing. To reduce the impact in terms of *MPJPE*, the input frame rate was slowed down in a way that the joints became relatively stable before extracting the locations. *Hansen et al.* [103] used

Method	PCK@100/500 (%)		MPJPE (mm)	Recall@100/500 (%)		Invalid (%)	F1 (%)	FPS (Hz)
MV3DReg [128]	-	-	176	-	-	-	-	-
VoxelPose	10.8	35.9	119	19.5	36.8	15.8	51.2	19.2
VoxelPose (synthetic)	27.8	65.9	201	15.2	72.4	76.7	35.3	8.3
Faster-VoxelPose	31.1	55.2	120	29.1	56.0	24.6	64.3	29.3
Faster-VoxelPose (synthetic)	28.1	45.5	109	29.4	46.1	7.7	61.5	30.0
MvP	0	0.1	343	0	0.1	99.9	0.1	8.8
PRGnet	3.6	6.2	120	3.4	6.4	44.3	11.5	11.9
TEMPO	7.9	12.6	102	8.8	12.7	14.0	22.2	20.3
SelfPose3d	36.2	67.7	143	31.1	70.2	36.5	66.7	13.0
mvpose	32.9	60.2	127	27.1	61.3	18.5	70.0	0.8
mv3dpose	0.7	2.8	235	0.4	3.1	57.7	5.8	3.0
PartAwarePose	10.3	25.5	201	6.3	27.7	20.5	41.1	6.9
PlaneSweepPosePlus	42.3	76.4	115	39.0	76.9	20.9	78.0	12.5
VoxelKeypointFusion	43.9	75.1	128	35.9	76.6	24.2	76.2	11.3
MVDeep3DPS [129]	-	-	213	-	-	-	-	-
OpenPTrack	9.9	26.8	323	0.8	33.6	83.9	21.7	1.9
VoxelKeypointFusion	44.2	72.2	119	36.3	73.4	12.9	79.7	10.9
SimpleDepthPose	62.0	94.3	113	54.1	96.6	23.5	85.4	37.2

Table 3.4. Transfer to *MVOR* [221] without and with depth.

4-fold cross-validation, training on three parts of the dataset, while all other models had not seen the dataset before, and also used ground-truth bounding boxes for each person, which is why their score is not directly comparable and therefore not included.

The main improvement of the depth masking of *VoxelKeypointFusion* is the reduction of the number of invalid persons, since now only persons can be predicted where the depth volume is filled. One reason this also leads to some missing joints is that if the keypoint heatmaps are not accurate and create a peak slightly next to a person, it gets cut off. In contrast to *SimpleDepthPose*, this approach does not help with the detection of persons that are only visible in one view, which is the main reason for the significantly better detection reliability of *SimpleDepthPose*. For this dataset, it was also notable that *SimpleDepthPose* has large errors at hip joints, which on average are around 170 *mm* off. In comparison, the upper-body joints have an average error between 60 *mm* and 100 *mm*. This is likely caused by the fact that the hip joints are much more often occluded, which sometimes leads to an incorrect assignment of depth values. Other than that, it significantly outperforms all others in terms of detected persons and keypoints. It is also faster than every other approach.

Dataset: *Panoptic*

Finally, the *Panoptic* [127] dataset is evaluated, and the results can be found in Table 3.5.

Method	PCK@100/500 (%)		MPJPE (mm)	Recall@100/500 (%)		Invalid (%)	F1 (%)	FPS (Hz)
VoxelPose	97.9	98.7	19.3	98.7	98.7	1.1	98.8	8.0
Faster-VoxelPose	98.6	99.9	20.5	99.7	99.9	1.0	99.5	18.0
MvP	97.2	98.3	18.7	98.0	98.5	15.8	90.8	8.9
PRGnet	99.1	99.9	17.1	99.9	99.9	2.0	99.0	6.8
TEMPO	97.4	98.5	16.8	98.4	98.4	2.4	98.0	5.1
SelfPose3d	98.7	99.8	24.9	99.7	99.9	8.0	95.7	7.1
PlaneSweepPosePlus	96.0	98.3	30.2	97.7	98.2	6.7	95.7	8.0
mvpose	75.9	97.5	83.6	73.5	98.5	10.0	94.0	0.1
mv3dpose	79.4	86.1	48.8	81.8	86.4	15.6	85.4	1.3
PartAwarePose	79.9	92.1	60.5	83.1	93.0	1.4	95.8	1.5
VoxelKeypointFusion	94.0	99.7	47.8	97.3	99.9	2.4	98.7	4.2
OpenPTrack	70.9	95.1	97.6	68.9	97.2	15.5	90.4	1.8
VoxelKeypointFusion	90.0	96.9	60.1	85.4	97.8	0.1	98.9	4.0
SimpleDepthPose	91.2	100	45.5	98.6	100	4.7	97.6	17.7

Table 3.5. Replication of *Panoptic* results and transfer without and with depth.

Most learned algorithms are trained using this dataset, and most of their results were closely replicated. From the two with larger discrepancies, *Faster-VoxelPose* published a different backbone model than what was described in its paper, and *PlaneSweepPose* originally used ground-truth bounding boxes for their 2D pose estimator. The evaluated cameras are the same as in the previous works [141, 231, 241, 248, 258], but the frames were changed to contain consecutive motions, same as described for *Human3.6M*. To make the results of all models comparable, only three out of four test scenes were used, since for one of them (*160906_band4*) no depth data was available. Note that since the depth cameras were not time-synchronized, their alignment to the color images and to the pose labels is not perfect. They were considered to belong together if the time difference was below a threshold.

From the RGB-based algorithmic approaches that had never seen this dataset before, *VoxelKeypointFusion* performs best, and is only slightly behind the models that were trained on this dataset in terms of detection reliability.

The results also show that the RGBD-based approaches outperform half of the algorithmic RGB-based ones in terms of the percentage of detected persons (*Recall@500*). *SimpleDepthPose* shows a detection rate of persons and joints that is on the same level as the learned approaches that were trained in this setup, while being faster than most. This is especially relevant in safety-critical applications, like in human-robot collaboration, where inaccurate joint or person estimations can be better handled, for example by generally increasing the required distances, than a missing one. Some of the invalid predictions might be persons entering the room, which are often not labeled. *SimpleDepthPose* already detects them if they are visible in one image, while many other approaches require at least two.

Summary

The general goal of all developed pose estimators is their usage in practical applications. It is not a practical option to create a training dataset matching the exact setup of the application, because every time someone would need to create and label a new dataset. Therefore, the generalization to unseen datasets is a crucial aspect. This is also important since the models have to be robust across a broad range of situations, and not just to the ones they were trained on. As shown in the experiments above, all previously existing models have setups in which they show good performance, and others in which they show poor performance. For a new setup without labels, it is hard to predict which it will be. Therefore, it is important to evaluate the models on a wide variety of datasets to get a better understanding of their capabilities. My three new algorithms, *VoxelKeypointFusion*, *PlaneSweepPosePlus*, and *SimpleDepthPose*, on the other hand, show a good performance across all datasets, even if they might not always be the best.

SimpleDepthPose is a very fast and reliable approach if depth data is available. It shows very good generalization results, and does not require any additional training. In comparison to *OpenPTrack*, the previously described differences in the algorithmic approach have led to a significantly better performance. One limitation, however, is that since there is no neural refinement of the resulting 3D poses, the location accuracy is highly dependent on the accuracy of the depth images. Following the results of the experiments, the use of depth data therefore does not necessarily lead to more accurate results in terms of joint localization, but can significantly increase the number of detected keypoints and persons, especially in strongly occluded settings.

PlaneSweepPosePlus also shows very good generalization results, especially if the camera distances are similar to its *Panoptic* training dataset. *VoxelKeypointFusion* on the other hand provides very good results across all datasets, without requiring any additional training. Integrating the already known joint associations into the voxel-based triangulation and assembling the persons with a bottom-up approach in 3D space has proven to be a very robust concept. Furthermore, it also has shown that a learning-free algebraic approach can still be very competitive in making use of multi-view information.

The papers about *SimpleDepthPose* [50] and *VoxelKeypointFusion* [47] also contain some additional experiments, for example on the influence of the number of cameras or important algorithmic parameters. Please refer to them for more details in this regard.

Since the performance of three new algorithms has now been validated on a wide variety of publicly available datasets, and on average they showed the best results on them, they can be accepted for integration into the safety system, and in the next step, be evaluated in the real-world application scenario.

3.5 Scenario Evaluation

The safety as well as the usability of the system were finally evaluated using the previously described application setup.

3.5.1 Pose Estimation

To assess the reliability of the person recognition in the given scenario, a dedicated dataset was created and labeled. To create the dataset, various people were asked to perform the “returns handling” scenario with the robot. In the scenario, the robot placed several packages with different contents on top of the table in front of the worker, which they then inspected and, depending on whether an object was present or not, sorted into two boxes for reusable returns or waste. Meanwhile, the robot was monitored by the newly developed safety system and stopped automatically if the distance between the person and the robot fell below a certain threshold.

The images from the cameras were recorded during this process, and afterward, parts of the sequences were labeled. A semi-automatic procedure was implemented for this, in which one of the algorithms (regularly switched to prevent biases) created an initial suggestion for the joint positions, which was then corrected manually to repair any misidentifications. A total of 1100 images from 10 people were labeled. The sequences were labeled at a frame rate of 10Hz, with a few seconds of consecutive frames and then a larger gap before the next sequence. See Figure 3.19 for an example of the labeled dataset.

The three models were evaluated using the standard metrics in computer vision, the same as in the previous section. The FPS values were measured on a single *Nvidia-3090* as well.

In Table 3.6, it can be seen that *VoxelKeypointFusion* and *PlaneSweepPosePlus*, which both only use color images, had problems with correctly detecting all persons (*Recall@500*). In contrast, *SimpleDepthPose* recognized all persons and was also the fastest. In comparison to the other datasets evaluated previously, the lack of synchronization of the cameras was found to result in some performance decreases. For hardware reasons, only the most recent image from each camera could be used instead of capturing all images simultaneously, introducing temporal offsets between the views. Consequently, if a person moves quickly, the joints can appear at different positions across the images. *VoxelKeypointFusion* in particular then has problems with triangulation, since the heatmap rays no longer overlap correctly. *SimpleDepthPose* also had problems with the quality of the depth data and additional extensive occlusions. This algorithm had always performed most reliably in the previous comparisons.

Method	PCK@100/500 (%)		MPJPE (mm)	Recall@100/500 (%)		Invalid (%)	F1 (%)	FPS (Hz)
VoxelKeypointFusion	82.8	99.4	59.1	90.2	99.6	0.0	99.8	6.2
PlaneSweepPosePlus	86.5	99.2	45.9	94.8	99.3	0.7	99.3	8.6
SimpleDepthPose	62.5	97.9	112	51.3	100	11.1	94.1	22.7

Table 3.6. Results of single approaches on the new *koarob* dataset. [48][© 2025 IEEE].

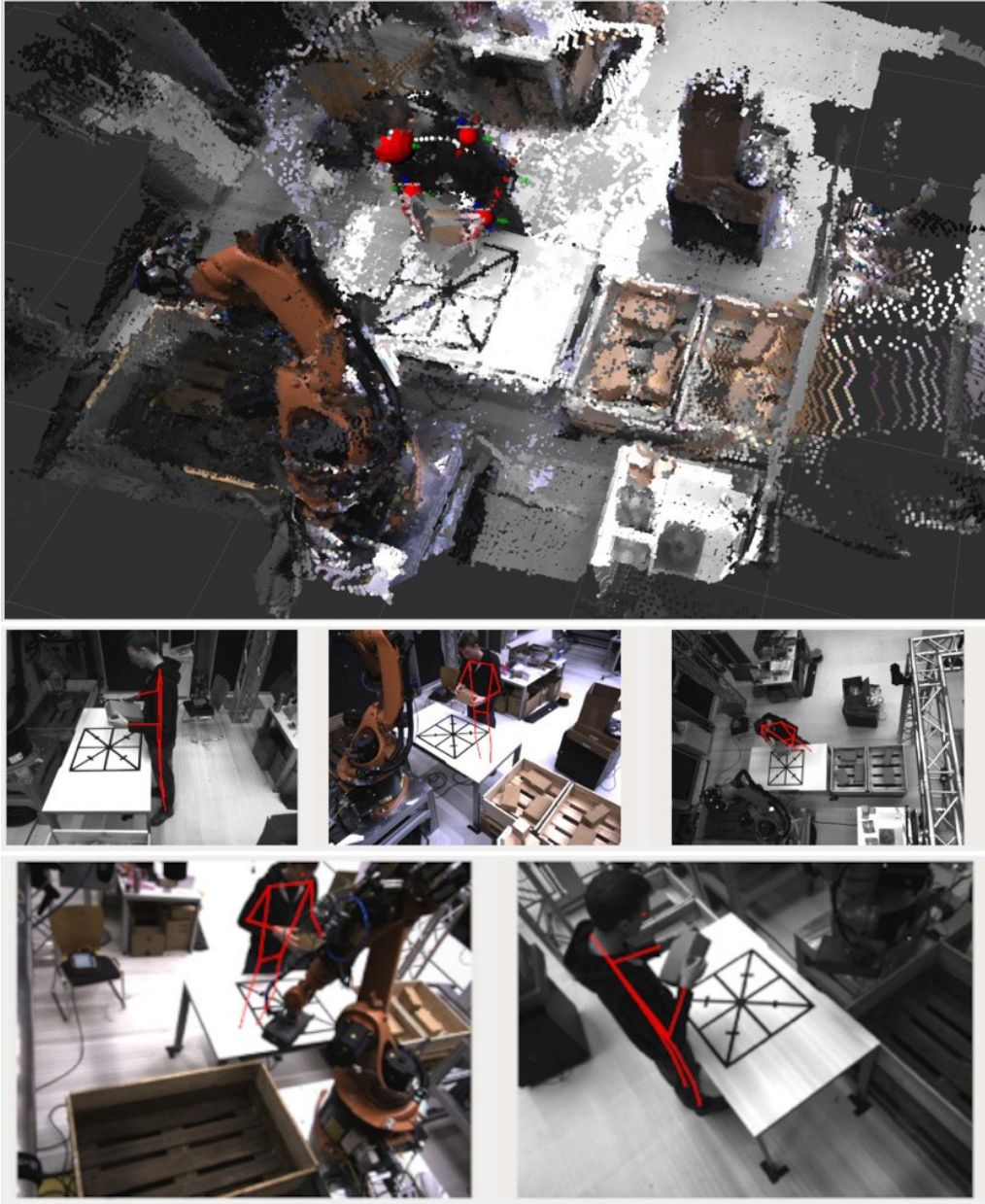


Figure 3.19. Evaluation dataset example of a 3D pose and its projection into the five camera views. To correct the initial label suggestions, the red joint spheres in the upper part can be moved via a drag-and-drop approach. [48][© 2025 IEEE].

As Table 3.6 shows, no algorithm reliably recognizes all joints within the default threshold of 500 mm ($PCK@500$), and therefore they need to be combined to improve the results. Using their diverse architectures, this should lead to better reliability. Two different approaches were investigated for this purpose. The first checks whether all algorithms have recognized the same number of persons and whether all 3D joint coordinates of these persons are also close to each other. If not, a stop is triggered. The second approach simply merges the results of all algorithms and thus detects the persons at all suggested positions (which then can simply overlap).

As can be seen in Table 3.7, both combination approaches now recognize all joints reliably (the minimum distance at which all joints are detected ($\text{minPCK}@100\%$) is less than 0.5 m). With the second approach, however, the minimum required safety space within which all human joints are recognized is somewhat smaller, which allows closer cooperation between humans and robots. Both concepts have their own advantages and disadvantages. The first approach stops the robot in 21% of the images (although these are not expected to be randomly distributed, but position-dependent, which means if the worker enters an area with poor visibility, the robot remains stationary thereafter). On the other hand, it can be assumed that the accepted images have a good pose accuracy. The second approach does not stop the robot directly, but increases the amount of occupied voxels. Approximately 12% of the occupied joint points are further than 500 mm away from the actual ones.

Method	minPCK@99%	minPCK@99.9%	minPCK@100%	non-close-percentage
close persons	0.20m	0.30m	0.45m	21.4% (stop frames)
fused keypoints	0.15m	0.25m	0.30m	11.7% (extra joints)

Table 3.7. Results of merged approaches on the new *koarob* dataset. [48][© 2025 IEEE].

The results per joint in Table 3.8 show that the localization error notably differs between the joint types. Especially the ankles, which were highly occluded in the evaluated scenario, have a larger error. By adding more cameras to reduce occlusions, it is expected that this position error can be reduced. Instead of the partition into head, torso, and limbs, used while running the evaluation (as shown in Figure 3.4), one could also use a custom voxel-occupation radius per joint type. In most applications, only a close distance to the hands and arms is of importance, and the rest of the body can stay further away.

Method	Head	Shoulders		Elbows		Wrists		Hips		Knees		Ankles	
close persons	0.30	0.25	0.20	0.20	0.30	0.20	0.30	0.25	0.25	0.25	0.30	0.45	0.30
fused keypoints	0.25	0.15	0.25	0.15	0.30	0.15	0.25	0.20	0.25	0.25	0.30	0.30	0.30

Table 3.8. Results per joint ($\text{minPCK}@100\%$ in meters, left|right) of merged approaches. [48][© 2025 IEEE].

3.5.2 System Assessment

With those pose estimation results, a system assessment can be performed to evaluate the reliability of the safety system, as previously described in Chapter 2.

Due to the experimental nature of the project, parts of the safety requirements were not fully implemented, such as the specification-guided software development process, or the evaluation of complex components like the used neural network libraries and GPUs. The validation step was also shortened to only check the system's target characteristics and perform the integration test while collecting data for the pose estimation evaluation.

The category of the system can be classified as *B*, since no redundancy is used for the distance monitoring and the three human detection methods were merged into a single channel. While there are a few runtime tests, like for recent timestamps, the overall *Diagnostic Coverage* is below 60%. There were also no special measures taken against *Common Cause Failures* yet, so the score here is below 65 points.

Some common challenges for vision-based detectors, such as unusual clothing, severe occlusions, or challenging lighting that would also affect the above approaches, can be mitigated by using a controlled setup. The company owner can influence the workers' clothing, and the system developer can optimize the number and positions of the cameras to reduce occlusions, as well as the lighting conditions. Therefore, it is recommended to solve these challenges outside the algorithmic concept as far as possible.

Combining the above scores, they would correspond to the first column of Figure 3.20. To reach a certifiable *PL* with the safety system, one would now need to show a longer $MTTF_d$ duration. While the evaluation showed the reliability of the presented safety system for a short time, a much longer testing phase would be needed to show the reliability for at least 3 years to reach a suitable $MTTF_d$ level.

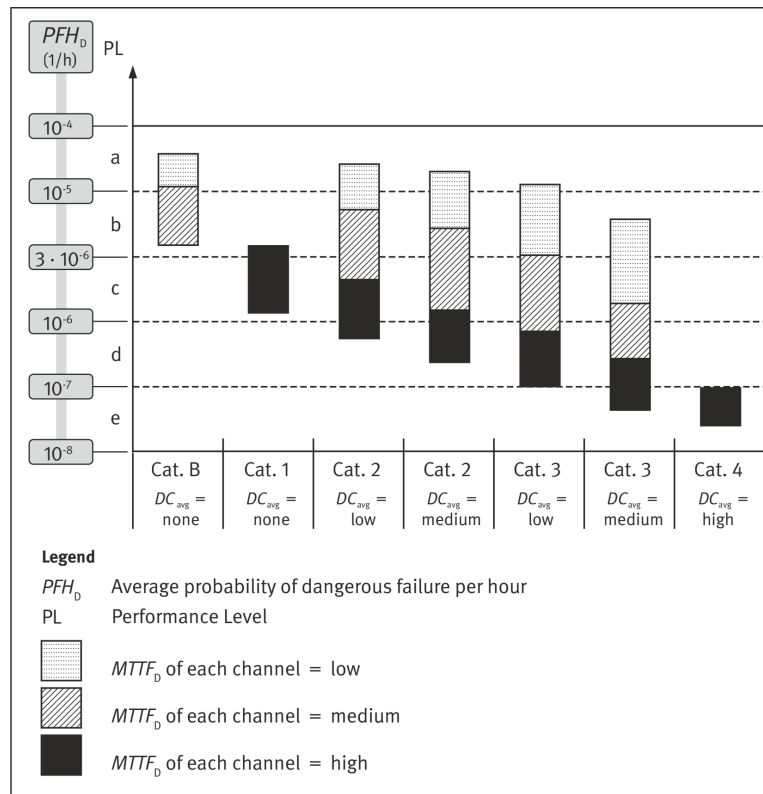


Figure 3.20. System assessment according to ENISO 13849-1, image from [83].

3.6 Related Work

There is already some existing research into tighter safety zones around human workers to compare against:

In [235] multiple *Time of Flight (ToF)*-cameras are used to detect obstacles that trigger a stop if they reach a dynamically adjusted safety zone. In [240] two *ToF*-cameras are used to detect humans in the path of the robot. The authors of [156] detect intrusions into the safety zone and slow down the robot, and as a backup for the two *ToF*-cameras, they also use laser scanners to stop the robot if necessary. Another early research project is [198], in which multiple depth-cameras are used to partition the workspace into three different types, background, robot and human/unknown voxels. Robots and humans receive an additional safety-hull, and if they overlap, the robot is stopped. Similar concepts were presented in [104, 144, 173], with the last two also including additional collision avoidance. All those systems lack safety certification, and thus cannot be used in production environments.

One more recent, commercially available, and safety-certified system following the same approach as the last four research works is *Veo Freemove* [25], which was already described shortly in a previous chapter. It also separates between background, robot, and person/unknown using multiple cameras (Figure 3.21). To reach redundancy, besides redundant calculations, each of the *ToF*-cameras has two sensors that can be cross-checked. The main disadvantage of purely depth-based systems is that they only distinguish between large unknown objects and background, and a finer distinction between human body parts, such as head or hands, is not possible. Therefore, such a system cannot allow acceptable contacts with a human hand and prevent dangerous contacts with the head at the same time.

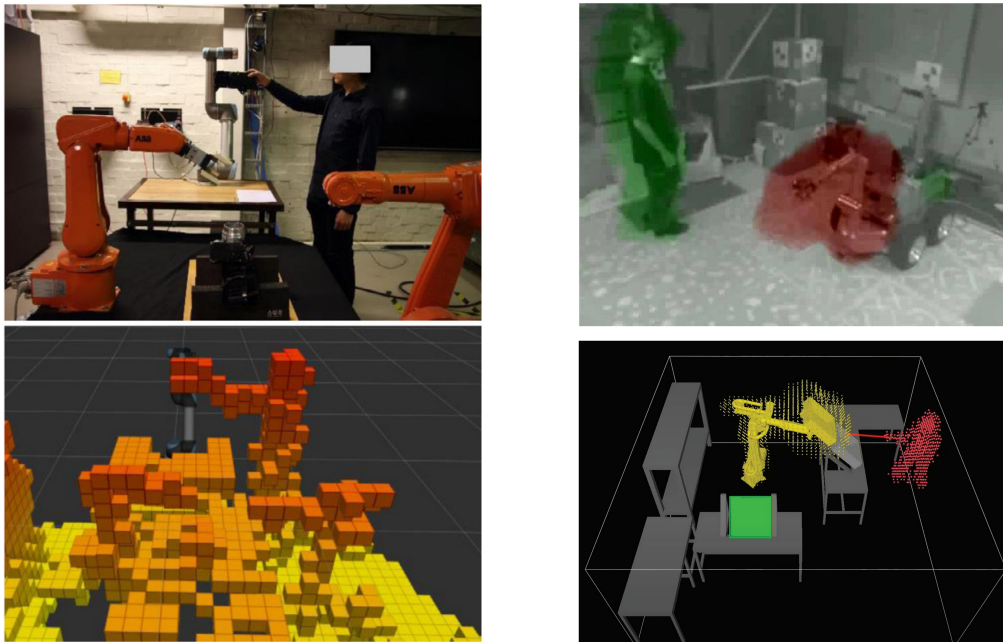


Figure 3.21. Some examples of depth-based safety systems. The two images on the left show the voxel-based concept of [144]. On the top-right, the voxel-separation into background, robot and person/unknown of [198], and on the bottom-right the *Veo Freemove* [25] system (image from [17]).

To distinguish between body parts, one could use a marker-based approach in which the workers are required to wear special clothing that can easily be detected by sensor systems. Examples of such systems are the commercially available *Vicon Motion Capture Suit* or other research works like [34, 58, 73, 138, 226] (Figure 3.22). This might be a very reliable system, even though none of the authors have provided a detailed safety analysis. Such a system, however, has the disadvantage that the workers have to wear special suits, which can be uncomfortable and restrict their movement, or in some use-cases is not possible at all.



Figure 3.22. Some examples of marker-based systems. On top, a *Vicon* system, with the white dots serving as markers which are recorded by multiple cameras (in the background on the trusses). Image from [15]. On the bottom-left, the trackable jacket and helmet with colored markers of [226], and on the bottom-right, the active low-latency tracking glove of [138].

Regarding AI-based safety systems, which would solve the aforementioned problems as they are able to differentiate between body parts without using tracking suits, there are only a few existing research projects so far (Figure 3.23): In [35], a voxel-based concept was extended with information from a human pose estimation system to annotate voxel types. They mentioned a latency of around 0.8 s for voxels of 4 cm size, and only used a single-camera system, which is susceptible to occlusions. The detection reliability was not evaluated. The authors of [78] used multiple modalities, including thermal images, for background removal and segmentation, before detecting human bounding-boxes. Those bounding-boxes were then used to separate the point-cloud into robot, background, and human points. They reported a miss-rate of at least 4% , depending on the modalities they used. In [148], the pose detections of a single camera were used to create ellipsoidal safety zones around the detected joints, with extra tags indicating whether contact with the body-part is allowed or not. The reliability of the method was not evaluated. The approach of [191] used an object detection system to detect body parts in the workcell, and another neural network to predict whether the object is inside the safety distance or not. While they were able to correctly detect all intrusions in their test dataset, the approach is not transferable to other scenarios, since the second network needs to be retrained each time. A similar image classification approach was also followed by [218], reaching an accuracy of only 95% , and [69], reaching 78% zone detection similarity with an expert assessment. In [32], a camera with a top-down view was used to detect the heads of humans in the workspace. A cylindrical safety zone was then created around the person, and, depending on the distance, the robot's speed was reduced. The authors reported a recall of 91.5% for detection in the danger zone. The system is also not able to distinguish between body parts.

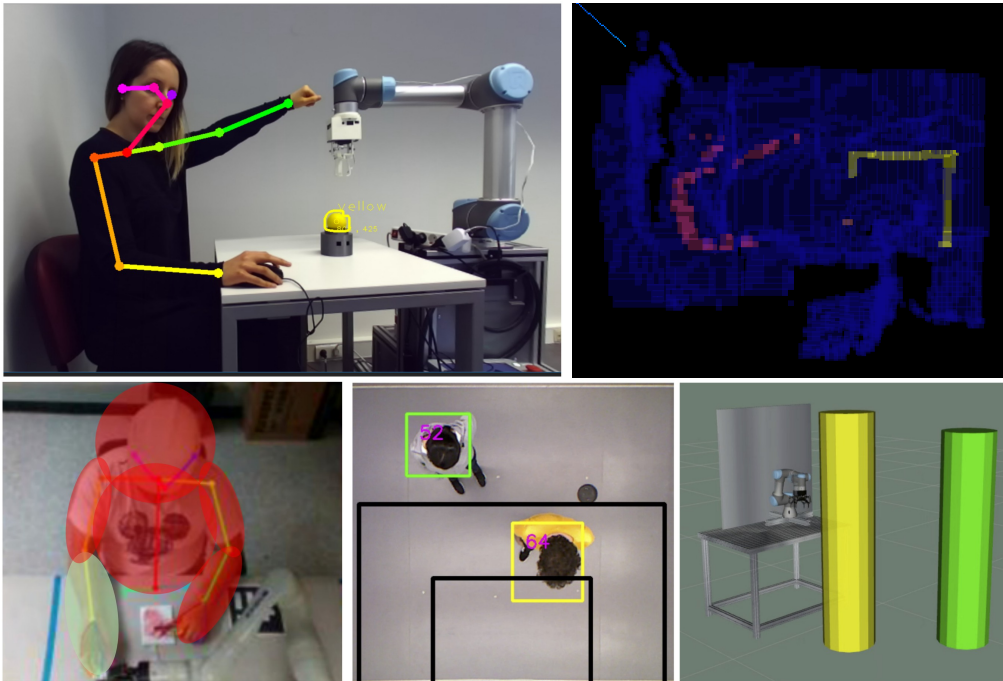


Figure 3.23. Some examples of AI-based systems. On top, the voxel-based concept of [35]. On bottom-left, the ellipsoidal tagged zones of [148], and on bottom-center/right, the cylindrical safety zones of [32].

Unlike those works, the presented *KoARob* system (Figure 3.24) was able to detect all persons as well as all their joints in the workspace with 100% reliability. The evaluation also closely followed the requirements of current safety standards. Furthermore, it allows the definition of different safety distances for different body parts.

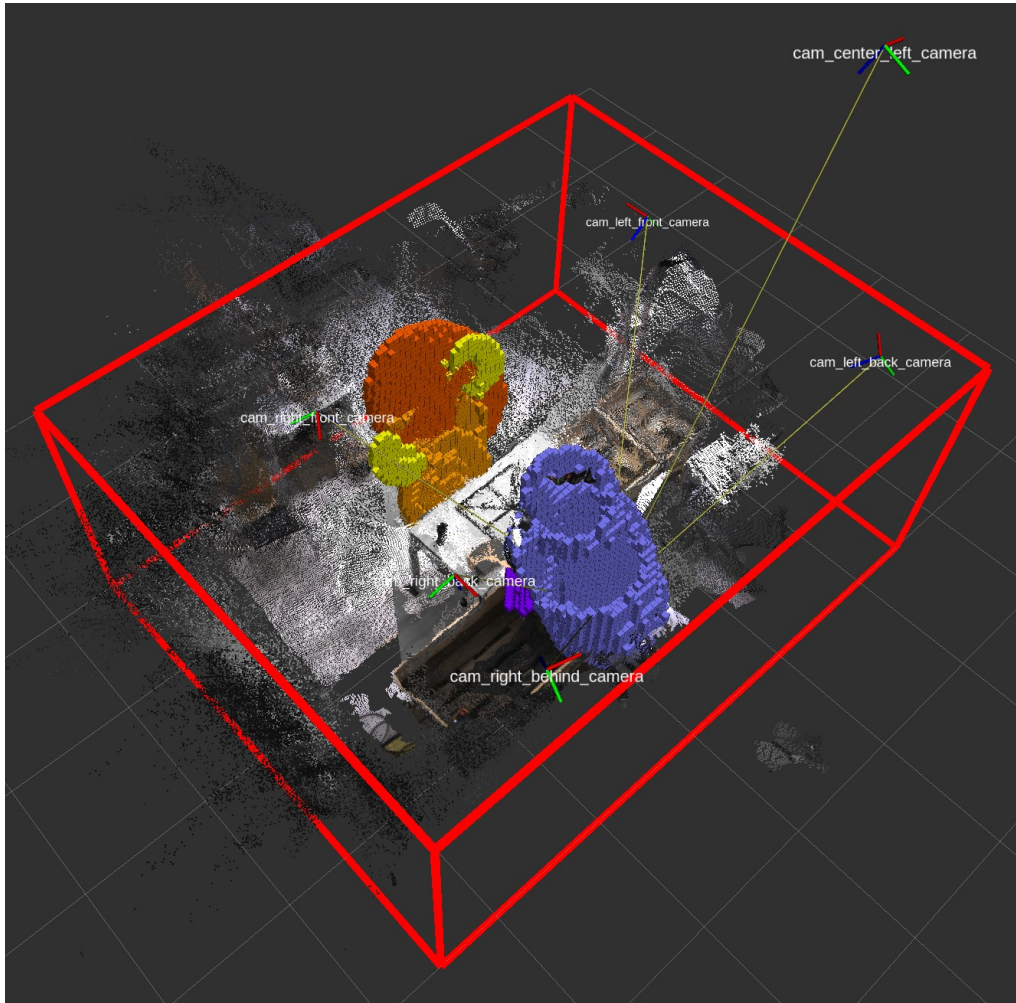


Figure 3.24. Internal visualization of the system using RVIZ. The human is represented by the orange/yellow voxels (depending on the limb type), and the robot by the purple/blue ones. The red lines mark the border of the supervised room, the small dots are from the point-cloud output of the cameras, and their positions are indicated by the labeled coordinate systems.

3.7 Conclusion

This chapter presented a safety concept for human-robot collaboration using AI-based sensor processing, which was evaluated in a real-world scenario. To improve the performance, three different human-pose detectors were combined. The system was able to reliably detect humans and stop the robot if the distance between the robot and any of the human's joints dropped below a certain threshold.

Since the system, and especially the human joint detection algorithms, are not specifically designed or trained for this package-returning scenario, but allow general detection of humans in arbitrary workspaces, the system is not limited to this specific application. Instead, it can be used in a wide range of other applications where a robot needs to work close to a human worker.

The main contribution of this chapter is to show that using AI-based systems for human detection in safety-related contexts is not impossible. Unlike previous works that used AI-based concepts, it has been shown, even though only for a short time so far, that the system detected persons and their joints in the workspace throughout the scenario evaluation with 100% reliability. This is the most important requirement for a new type of safety system, because if it is not completely reliable, the system will not be usable in any production environment.

One of the main remaining challenges is to prove the reliability of the system over a longer time, which is needed to reach a certifiable performance level. The next chapters will present how to get closer to the goal of safety certification by improving the system step by step in an iterative process.

Summary. This chapter provides a concise overview of the safety regulations in the context of human-robot collaboration.

4

Safety Foundations 2: Collaboration

4.1	Methods of Collaboration	74
4.2	Speed and Separation Monitoring	74
4.3	Power and Force Limiting	76

Before starting the next development step, it is important to take a closer look at the safety regulations for human-robot collaboration. The *EN ISO TS 15066* [112] technical specification provides requirements for collaborative robots and is summarized in the following. The risk analysis and system assessment generally follow the same process described earlier in Section 2.2, but specific requirements are added for different collaboration scenarios.

4.1 Methods of Collaboration

As already mentioned in Section 1.2, which presented common safety control systems, four methods of collaboration are defined in *EN ISO TS 15066* [112]:

- **Safety-rated monitored stop:** The robot must cease motion before the human operator enters the collaborative workspace.
- **Hand guiding:** The operator uses a hand-operated device to transmit motion commands to the robot system.
- **Speed and separation monitoring:** The safety system monitors the distance to the human and reduces its speed as necessary to always maintain a safe separation distance.
- **Power and force limiting:** The robot is allowed to have physical contact with the human operator, but the contact forces must not exceed certain limits.

In the context of this work, the last two methods are of particular interest, as they enable a more seamless interaction between humans and robots.

4.2 Speed and Separation Monitoring

In *Speed and Separation Monitoring (SSM)*, the robot system must always maintain at least a protective separation distance from the human operator. If the distance decreases below this limit, the robot must stop its movement. When the distance increases again, the robot can automatically resume motion. To avoid violating the protective separation distance, the robot may also reduce its speed or change its trajectory instead.

The general idea of the protective separation distance is to ensure that the robot can stop its movement completely before it touches the human. This distance is calculated as follows:

$$S_p = S_h + S_r + S_s + C + Z_d + Z_r \quad (4.1)$$

Thus, the protective separation distance S_p is the sum of the possible movement distance of the human S_h after entering the safety zone, the movement distance of the robot until it starts to brake S_r , the braking distance S_s , the intrusion distance C that a body part can move into the safety zone before being detected, the uncertainty of the human's position Z_d , and the uncertainty of the robot's position Z_r (Figure 4.1). To allow closer collaboration, all these components should be reduced as much as possible.

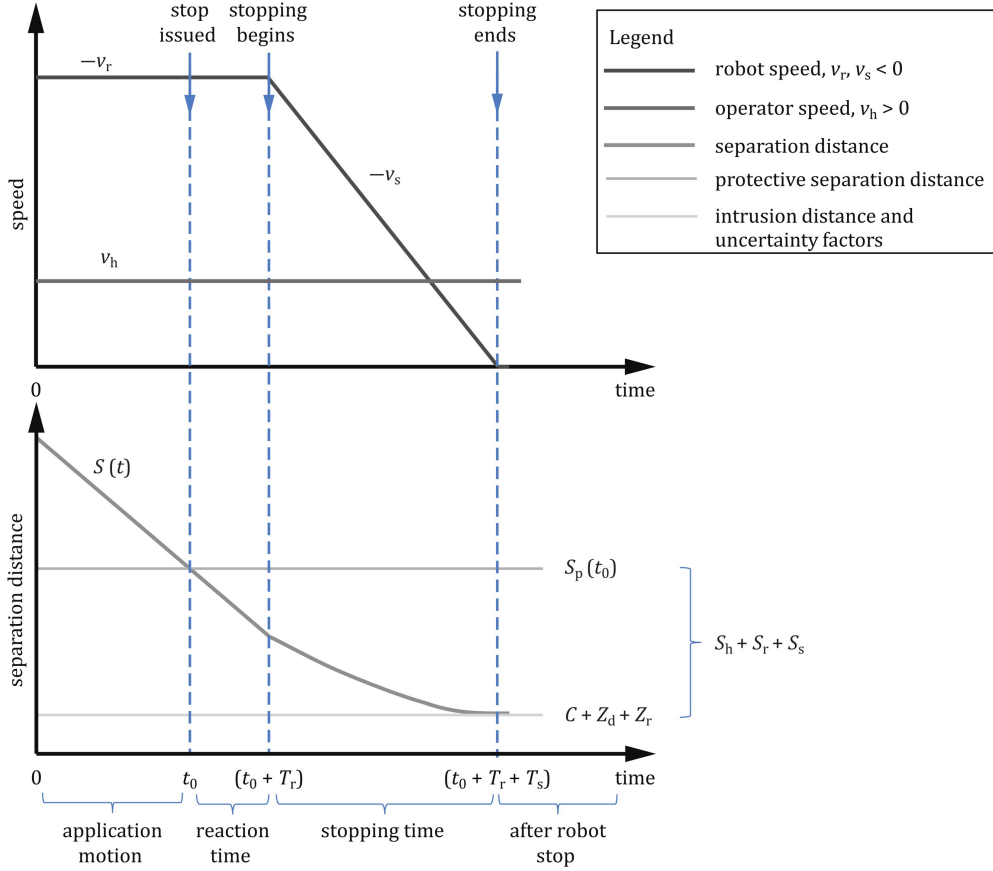


Figure 4.1. Contributions to the protective separation distance. It is generally assumed that the human and the robot move directly toward each other. Image from *EN ISO TS 15066* [112].

The distances S_h and S_r both depend on the delay of the safety system, which measures how long it takes from a human entering the safety zone until the system detects this and sends the stop signal. For a camera-based system, this includes the time to trigger image capture, the camera's processing time, the time for image-processing algorithms and safety calculations, and, since the system is repeatedly evaluated, the time until the next result replaces the previous one. Assuming different parts of the system can run in parallel, but the same parts cannot, the slowest part defines the interleaving delay. The safety delay is then the sum of all these delays, as visualized in Figure 4.2.

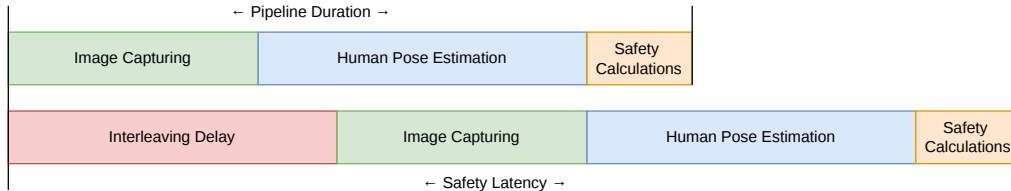


Figure 4.2. Pipeline and safety delay. In this example, the length of the interleaving delay is defined by the *HPE*, which is the slowest non-overlappable part of the system. It can also be understood as the time a person can use to move into the safety zone without being detected by the upper iteration. [51][© 2025 IEEE].

4.3 Power and Force Limiting

In the case of *Power and Force Limiting (PFL)*, the robot is allowed to have physical contact with the human operator (intentionally or unintentionally), but contact forces must not exceed certain limits to avoid injuries. Such contacts can be separated into two categories:

- **Quasi-static contact:** This includes clamping or crushing, where a body part is trapped between the robot and another fixed or moving object in the workspace. Forces are therefore applied for an extended period of time.
- **Transient contact:** This includes impacts where the robot hits a body part of the human operator, but instead of being trapped, the body part can be pushed/moved away. Forces are therefore only applied for a very short duration.

The force applied by the robot depends on the robot's speed, its mass, and the contact area. Maximum pressure and force limits differ between the two contact types, and also depend on the body part in contact, since different body parts have different tolerances (Table 4.1).

Body Part	P_S (N/cm^2)	F_S (N)	P_T (multiplier)	F_T (multiplier)
Middle of forehead	130	65	n/a	n/a
Chest - Sternum	120	140	2	2
Kneecap	220	220	2	2
Lower arm forearm muscle	180	160	2	2
Back of the hand	190	140	2	2
Forefinger pad	270	140	2	2
Forefinger end joint	220	140	2	2

Table 4.1. Examples of experimentally determined biomechanical pressure (P) and force (F) limits (pain onset levels) for static (S) and transient (T) contacts from *ENISO TS 15066* [112].

To reduce contact pressures and forces, possible passive measures include reducing mass, using rounded surfaces to increase contact area, or adding padding to dampen impacts. Additionally, active measures can be implemented, which include using a force/torque sensor to detect contact and then stop or reduce the robot's motion, or using a torque-limited drive system that automatically limits the forces that can be applied by the robot. Especially for transient contacts, the robot's speed is a crucial factor, as contact forces generally increase with impact velocity. To determine the maximum allowed contact speed, the standard provides some example values, which were calculated using a simple physical energy-transfer model [112, Ann. A].

Summary. This chapter presents the *Tutabo-1* project, which investigated the real-time capabilities of an AI-based safety system.

5

Tutabo-1: Reacting in Real-time

5.1	Revisiting KoARob	78
5.2	Improved Implementation	80
5.2.1	Smart Cameras	80
5.2.2	Optimized Pose Estimation	84
5.2.3	Optimized Safety Calculation	85
5.2.4	System architecture	86
5.3	RapidPoseTriangulation	87
5.3.1	Algorithm	87
5.3.2	Dataset Generalization	91
5.3.3	Whole-body Estimation	98
5.3.4	Conclusion	100
5.4	Evaluation	101
5.4.1	Pose Estimation	101
5.4.2	Robot Stop Time	104
5.4.3	Collaboration Distances & Delays	105
5.5	Related Work	108
5.6	Conclusion	109

The evaluation of the *KoARob* system has shown that reliable detection using AI-based methods is achievable, particularly when multiple algorithms are used in parallel. However, as described in the previous chapter, a fast response time of the system is crucial as well, particularly for the usability of the system. Since this was not a core focus in the *KoARob* project, considerable potential for improvement still exists, which this chapter will address.

Because the *KoARob* name is tied to the previously described and already completed research project, my continued research is named *Tutabo*, which is Latin for “*I will protect*”.

The chapter is based on my following published work, with selected text passages directly copied from it:

Tutabo-1: Towards Real-time Capable AI-based Safety Systems for Human-Robot Collaboration. Bermuth, Daniel and Poeppel, Alexander and Reif, Wolfgang. December 2025. In *2025 IEEE International Conference on Advanced Robotics (ICAR)*. Pages 632–639. © 2025 IEEE. [51]

5.1 Revisiting KoARob

To illustrate the use of the safety distance formulas from the previous chapter, and to further motivate the improvements of this chapter, the safety distance for the *KoARob* in the previous evaluation can be calculated as follows:

$$\begin{aligned}
Speed_{human} &= 2.0m/s \\
Speed_{robot} &= 0.2m/s \\
Time_{robot\ stop} &= 0.25s \\
Time_{system\ delay} &= 0.21s_{(cameras)} \\
&\quad + 0.067s_{(sync\ randomness)} \\
&\quad + 0.164s_{(VoxelKeypointFusion)} \\
&\quad + 0.004s_{(voxelization\ \&\ occupancy)} \\
&\quad + 0.164s_{(interleaving)} \\
&= 0.6s
\end{aligned} \tag{5.1}$$

The approach speed of the human (upper-limb movement in this scenario) is specified by *ISO 13855* [22]. The robot speed in the current scenario was set to 10% of the maximum speed, to give the worker some time to inspect the packages. The robot stop time was estimated using the robot’s datasheet [137], which contains standardized time measurements for the first three axes at certain speeds. The *Roboception* cameras were able to deliver images at about 15 Hz, but introduced substantial additional delay for processing the stereo images. Since the cameras were not synchronized, and therefore the images are not received at once, but have some offsets between them, an extra time delay of one frame interval is added for the delay calculation. For the pose estimation, the time of the slowest algorithm from the original evaluation is taken, for simplicity under the assumption that all three pose estimation algorithms can run in parallel

without causing additional slowdown. The slowest non-parallelizable part is the pose estimation, which defines the interleaving delay.

The above variables result in the following distances:

$$\begin{aligned}
 S_{h_1} &= 0.6s * 2.0m/s = 1.2m \\
 S_{h_2} &= 0.25s * 2.0m/s = 0.5m \\
 S_r &= 0.6s * 0.2m/s = 0.12m \\
 S_s &= 1/2 * 0.25s * 0.2m/s = 0.025m \\
 C &= 0.208m \\
 Z_d &= 0.30m + 0.05m = 0.35m \\
 Z_r &= 0.002m
 \end{aligned} \tag{5.2}$$

The possible movement distance of the human was split into two parts: the movement until the safety system triggers the stop signal, and the movement until the robot has stopped. To calculate the robot's braking distance, it was estimated that the robot's speed decreases linearly. The robot's absolute position inaccuracy was estimated as well, but closely matches the measurements of [186]. As intrusion distance, the hand length is taken (208 mm as defined by *ISO 13855* [22]), because the pose estimation only predicted wrist joints (in the original paper evaluation a slightly larger distance of 223 mm was used, which is the 99th percentile from an anthropometric study [180], but using values from the standard seems more appropriate for this chapter). The uncertainty of the human's position is the error of the pose estimation, with an additional offset of 5 cm to account for human-introduced labeling inaccuracies. In summary, the safety distance is:

$$\begin{aligned}
 S_p &= 1.2m + 0.5m + 0.12m + 0.025m + 0.208m + 0.35m + 0.002m \\
 &= 2.405m
 \end{aligned} \tag{5.3}$$

As can be seen, in the original system the required safety distance is quite high. This is mainly due to the high uncertainty of the human's position, and the possible movement of the human during the delay of the system and the long braking time of the robot. The distances can be split into the following parts:

$$\begin{aligned}
 S_{robot} &= 0.5m + 0.025m + 0.002m &= 0.527m \\
 S_{system} &= 1.2m + 0.12m + 0.208m + 0.35m &= 1.878m
 \end{aligned} \tag{5.4}$$

The first one is defined by the robot itself, and is mainly caused by the long braking time. An even lower movement speed should help here, but the robot's datasheet [137] did not contain measurements at low speeds. To reduce this value, the only option is to perform own measurements with robot movements that can occur in the given setup. The second part is mainly caused by the system's delay and the high uncertainty of the human's position. To improve the usability of the system, the required safety distance has to be notably reduced, which will be the focus of the next sections.

5.2 Improved Implementation

To reduce the required safety distance, all parts of the system can be optimized. To achieve this goal, the following sections describe the improvements that were made to the cameras, the pose estimation, and the system architecture.

5.2.1 Smart Cameras

The originally used *Roboception* cameras have the advantage that they can calculate depth images from their stereo view, which can simplify the distance estimation for the 3D human pose prediction. However, the stereo calculation is computationally expensive and results in a large delay. The missing synchronization also adds additional delay, since some images are captured earlier in time than others. In total, the cameras were responsible for almost half of the system's delay.

New cameras

The five original cameras had to be replaced for the above reasons. Regarding the new ones, the following requirements were defined:

- Low latency (less than 20 *ms* from trigger to image output)
- High frame rate (at least one image every 20 *ms*)
- Synchronization capabilities
- High enough resolution (to be able to distinguish between single fingers)
- Affordable price (around 400€ per camera)

Since multi-view pose estimation algorithms generally benefit from more cameras, the number of cameras was doubled to ten. This also allows for better elimination of occlusions, and a better coverage of the workspace, which can further reduce the uncertainty of the human's position. The cameras were placed around the workspace on the trusses again, to cover it from different angles. Figure 5.1 shows the planned camera positions (red numbers). The two cameras 4 and 5 on top are placed at a higher position to obtain a better view of the human's fingers while the worker is working at the table, because both the robot and the human's upper body can cause occlusions in the front and back cameras. Because many pose estimation algorithms, including the one used later, have difficulties with correctly detecting the joints if only a small part of the body is visible (like only the hands), no camera is placed on the robot arm itself. Camera 7 is placed to improve the coverage if the worker fetches a package from the delivery pallet. The intention of camera 10, which records the workspace from behind, is mainly to better cover the lower legs of the human, as they had the highest uncertainty in the first evaluation.

Regarding the image resolution, it needs to be high enough to be able to distinguish between single fingers. For a human, this is easily possible if each finger is around 7 pixels wide in the image, and with increasing effort until around 4 pixels (tested by manually looking at several images from the different evaluation datasets). To calculate the required resolution, three working positions were defined (blue numbers), and

the size (in pixels) of a human finger (thickness around 14 mm) was calculated for different resolutions and lens opening angles (45, 60, 85 degrees) using standard camera projection formulas. Using a *Full-HD* resolution results in a finger size between 7-9 pixels for most camera and working position combinations, with the smallest size for further away combinations of around 4 pixels. Thus, this common resolution is sufficient for the task, and higher resolutions would mainly have a negative effect on processing times, network load, and costs.

Given these requirements, *Basler ace2* cameras [26] were chosen for the replacement, because they are able to deliver images at a rate of up to 160Hz with a resolution of 1920×1200 pixels (lens focal lengths are 4/6/8 mm). The cameras are also technically capable of both hardware and software synchronization, and offer a basic integration into *ROS2* as well. The *ROS2* driver provided by the manufacturer was then optimized to reduce its latency by removing features that are not necessary in this scenario. The driver was further extended with a method to synchronize the image capturing based on the system time, which then can be synchronized across multiple devices using the *Precision Time Protocol*.

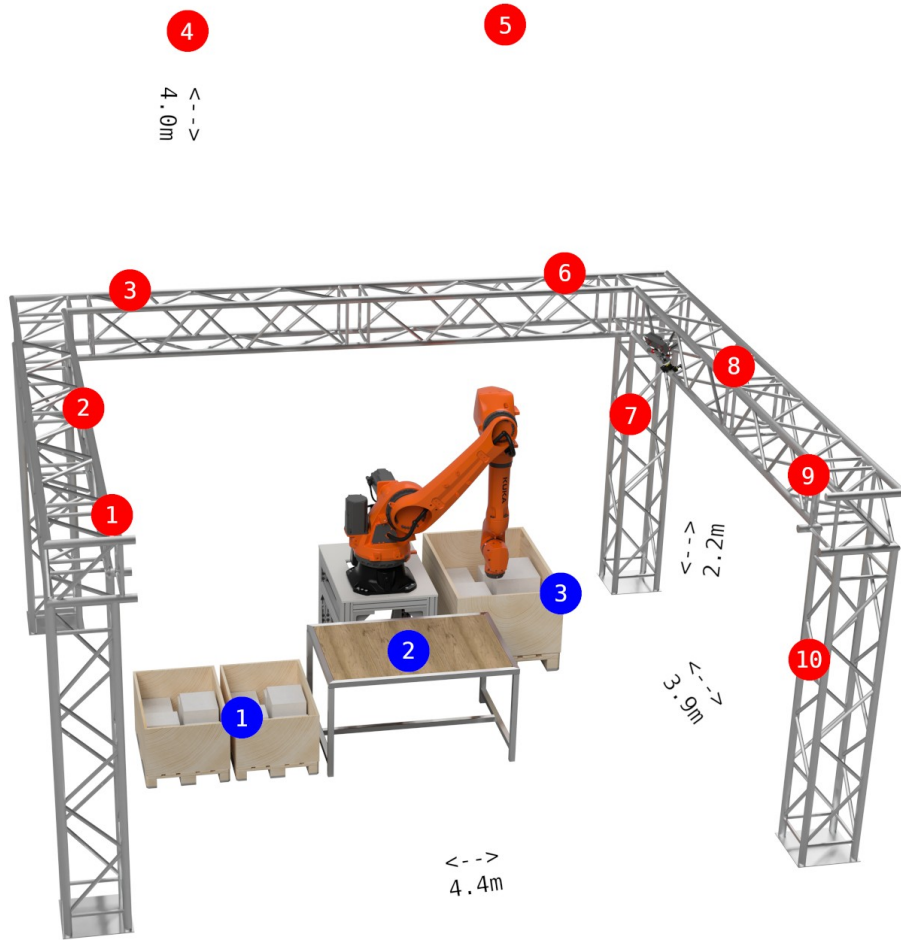


Figure 5.1. Planned camera positions.

Calibration

To be able to use the new cameras for pose estimation, they need to be calibrated, which means their position, or more precisely, the transformation between the camera's coordinate system and the robot's coordinate system needs to be calculated. Previously, this was done using the *Roboception* cameras' built-in calibration tools and their custom calibration board, which is not possible with the new cameras. Therefore, a new calibration tool was implemented, which uses a standard checkerboard pattern as calibration board (Figure 5.2).

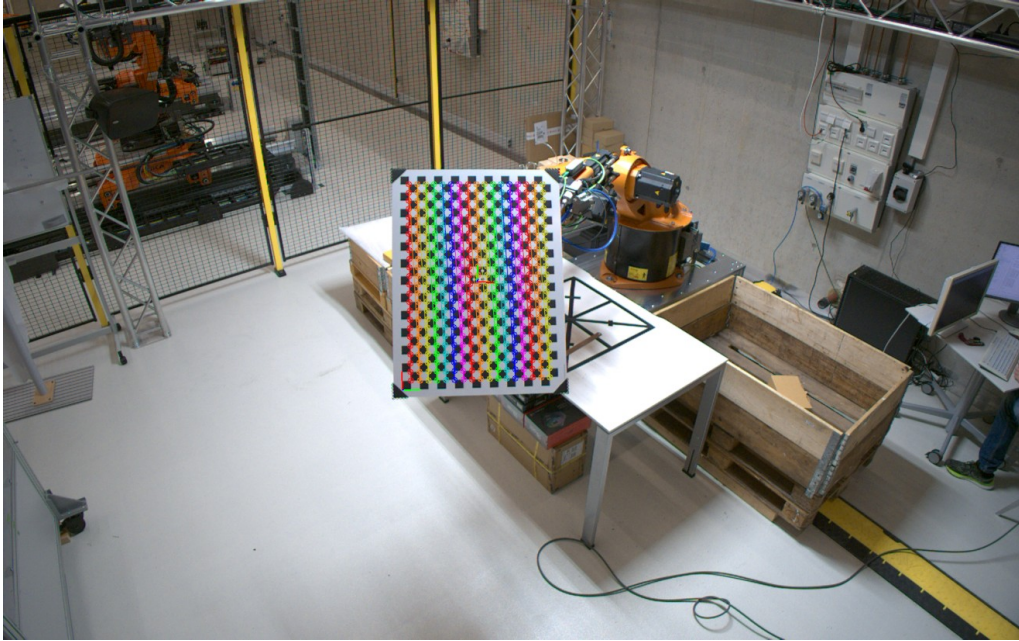


Figure 5.2. Camera calibration using a checkerboard pattern. The camera captures multiple images of the checkerboard at different robot positions, and the corners of the checkerboard are detected in the images (as shown above). The known geometry of the checkerboard and the known pose of the robot allows, if multiple images are taken, to calculate the unknown, but static transformations between the board origin and the robot flange, and the camera and the robot's base coordinate system.

The camera calibration procedure can be split into two steps. The first step is to calculate an *intrinsic* calibration, which describes the camera's internal parameters, such as the focal length and opening angle, the position of the optical center in relation to the image sensor, and the distortions introduced by the camera lens. For this step, the calibration tools of the *OpenCV* library [28] were used. The second step is to calculate the *extrinsic* calibration, which describes the transformation between the camera's coordinate system and the robot's coordinate system (in this context also known as *hand-to-eye calibration*). The general idea is to capture multiple images with different robot poses, to solve a system of equations with known and unknown transformations (Figure 5.2). *OpenCV* already provides multiple methods for this purpose [29], but those results can be further improved by reformulating the optimization problem and also making use of the multi-view setup. In a few experiments with both *hand-to-eye* and *eye-on-hand* setups, this

additional optimization step could further reduce the calculated calibration error (here by about 7%). A thorough evaluation of this optimization procedure is beyond the scope of this chapter and is left for future work, but the full code for the camera calibration and camera integration was released as open-source software (<https://gitlab.com/Percipio/CamStreamer>). In this setup, the calculated *intrinsic* reprojection error of the cameras is between 0.1-0.3 *px* and the final *extrinsic* position and rotation error is between 4-8 *mm* and 0.1-0.2 *deg*, mainly depending on the distance of the camera to the robot (smaller distances generally yield lower errors).

Further improvements

To reduce image exposure times (to 5 *ms*), more lights were added. This also improved the working conditions for the human worker, since the brightness now better follows workspace standards (*ISO/CIE 8995-1* [121]), instead of being rather dark. The shorter exposure times also reduce the impact of motion blur, because the possible movement distance of a fast human motion within a single image exposure is lower.

Due to the high frame rate and resolution, the network latency becomes an important factor. To further reduce the transfer latency, and also to avoid the overhead of in-camera color conversion, the *Bayer* format, which is the default layout of image sensors in most cameras, is used directly. Instead of having three *RGB* channels, this format only has one channel, in which every other pixel has a single red, green, or blue color only (Figure 5.3). This significantly reduces the amount of data sent through the cable to one third, even though the receiver now has to convert it back to *RGB* for further processing.

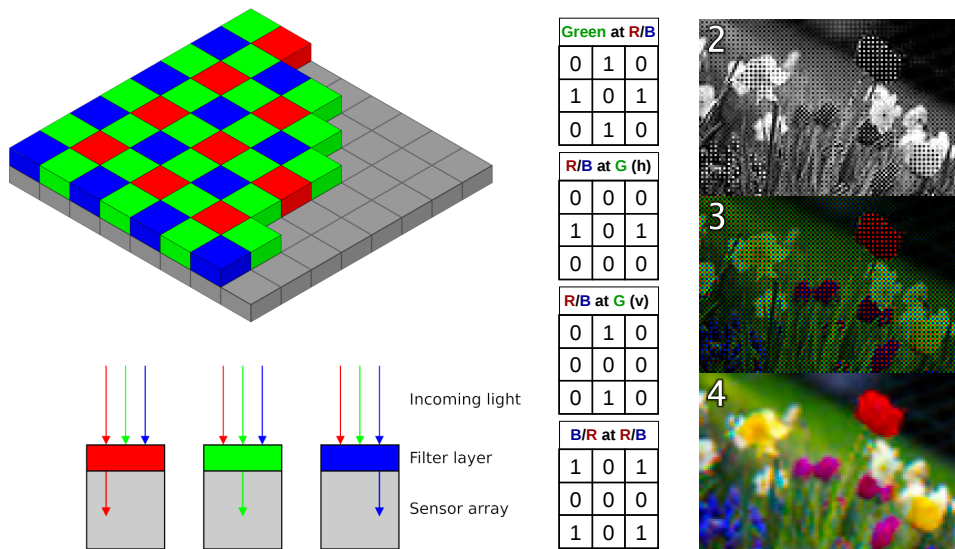


Figure 5.3. Concept of the *Bayer* image format. Each pixel of the image sensor has a color filter in front of it, so it only captures the intensity of the light (0-255) of this one color. The output itself is a color-free intensity image (right, 2). Using the filter layout, it can be converted to a color image with the filter color per pixel (right, 3). To calculate *RGB* values for each pixel (right, 4), the missing colors can be calculated from the neighboring pixels (*demosaicing*). A simple approach is using a linear matrix-based interpolation (center). For example, for a red pixel, the upper matrix (*green at red*) can be used to calculate the missing green value as the average intensity of the four neighboring (green) pixels, the blue color with the bottom matrix (*blue at red*), and as red, the intensity value of the pixel itself is taken. Images on the left and right are from [27].

Local processing

Transferring a single *FullHD* image with $1920 \times 1080 \times 1$ pixels in *Bayer* encoding over a 1 Gb/s network still takes around 17 ms . This adds up for every camera used. While compressing the image could reduce this time, it would require extra processing time and reduce the image quality. Therefore, instead of transferring all ten images to a single host that does all the processing, a distributed approach can be faster. Each camera was therefore connected via 5 Gb/s *USB* to one *Jetson Orin AGX* [174] module, which processes the image locally and transfers only the results to the main computer (Figure 5.4). Since no images need to be transferred, and the pose results are notably smaller, the network load is significantly reduced as well.

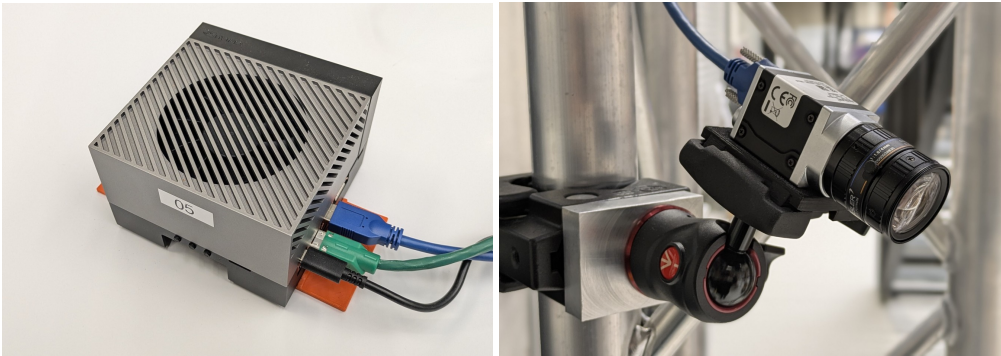


Figure 5.4. Hardware of the smart cameras. On the left a *Jetson Orin AGX*, on the right a *Basler ace2* with its lens mounted on the trusses.

5.2.2 Optimized Pose Estimation

The second part adding a considerable amount of latency to the system is the pose estimation. In the *KoARob* system, three different algorithms were used to estimate the human's pose, and their results were merged to improve the reliability. To reduce the implementation effort and to arrive faster at a minimum viable prototype, only one new algorithm was developed for this iteration. To improve reliability and also reduce occluded areas, the number of cameras was doubled instead. This also aimed at reducing the pose uncertainty of the joint positions, which was also a significant part of the calculated minimum safety distance. For the same reason, the new method also needs to predict the position of the fingers, so that the intrusion distance can be reduced from a hand's length to a finger's thickness.

Thus, the three main requirements for the new pose estimation method were:

- Speed up pose estimation
- Reduce joint error
- Predict finger positions

The newly developed algorithm is called *RapidPoseTriangulation (RPT)*, and it uses a relatively simple but efficient approach to triangulate the 2D poses from multiple cameras

into a 3D pose. It was equipped with a *ROS2* interface, to support the distributed processing of the 2D pose estimation from the images and the subsequent 3D triangulation. Given the requirements of the safety system, the development focus was on prediction speed, and the benchmark results show that my new solution is the most reliable and by far the fastest method for multi-view multi-person human pose estimation currently available. The method's architecture and its evaluation will be described in more detail in Section 5.3.

5.2.3 Optimized Safety Calculation

To reduce the computational complexity, the voxel-based occupancy approach from *KoARob* was replaced by a simpler sphere-collision detection (Figure 5.5).

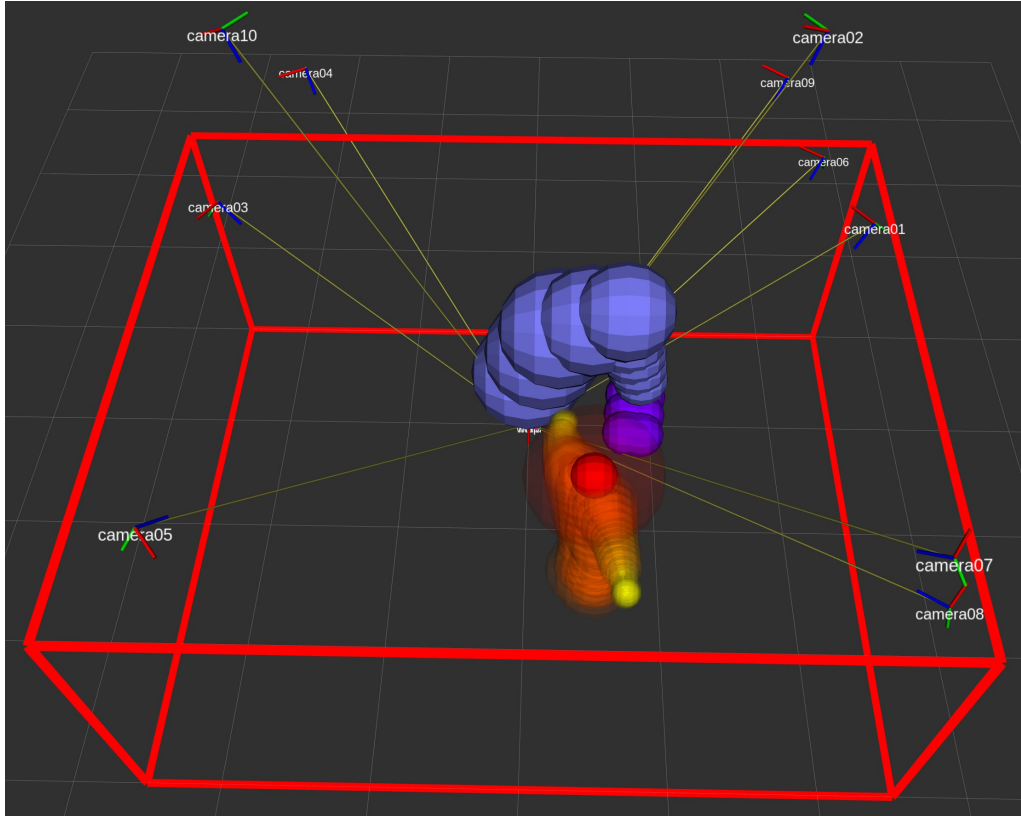


Figure 5.5. Sphere distance monitoring. The human is represented by the orange/yellow spheres (depending on the limb type), and the robot by the purple/blue ones. [51][© 2025 IEEE].

Both the human and the robot are represented by sets of spheres, with different radii for different joint types. If human and robot spheres overlap, a stop signal is published. The new representation also has the advantage that distances do not need to be quantized to a voxel grid (with 50 mm resolution), but full numeric precision can be used instead if the spheres are properly modeled.

5.2.4 System architecture

Figure 5.6 shows the overall software pipeline and how the safety components are combined. The general architecture is still similar to the one from *KoARob* (Figure 3.5), keeping its division into the three main parts: robot position, human detection, and distance monitoring. The modular software design allowed an independent replacement of the components, and only the camera branch received a significant architectural change. As before, the communication between the different modules is done via *ROS2*, and all modules are containerized using *Docker* to improve maintainability and deployment.

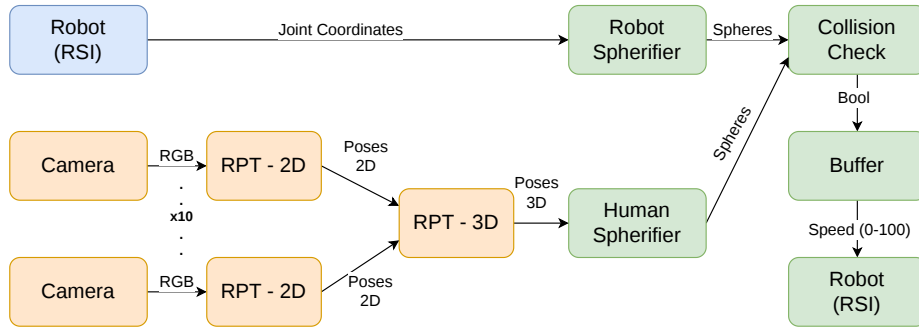


Figure 5.6. Software pipeline of Tutabo-1. [51][© 2025 IEEE].

The cell layout stayed largely the same (Figure 5.7), and the old gripper was replaced by a more customizable *Schmalz* suction gripper.



Figure 5.7. Robot setup of Tutabo-1. [51][© 2025 IEEE].

5.3 RapidPoseTriangulation

In the previous section, a requirement for the new cameras was to reach a frame rate of at least 50Hz. This means the pose estimation needs to be able to process the images at this speed as well. Another requirement was that it can additionally predict the position of the fingers.

The evaluations in the previous chapter have shown that the algebraic method of *VoxelKeypointFusion* is a very robust and reliable method. It also is training-free, which, besides a simpler deployment, additionally allows for easy adjustment of the method to different keypoint types, thus enabling it to predict finger positions as well. The problem is that it does not meet the speed requirement, and therefore a new method needed to be developed.

The new algorithm is called *RapidPoseTriangulation* and will be described in the following. Similar to most other approaches, the algorithm can be split into two stages as well, with the first one predicting the 2D poses for each image. For this, any AI-based 2D pose estimator can be chosen. Here *RTMPose* [126] was used, same as in *VoxelKeypointFusion*. It has the advantage that there already exist some pretrained models that also support predicting finger keypoints. The second stage follows a simple and learning-free triangulation concept. Both stages are optimized for speed and are also designed to be able to run in a distributed way, which is another requirement from the previous sections.

This section is additionally based on my following work, with selected text passages directly copied from it:

RapidPoseTriangulation: Multi-view Multi-person Whole-body Human Pose Triangulation in a Millisecond. Bermuth, Daniel and Poeppel, Alexander and Reif, Wolfgang. Preprint. March 2025. [52]

5.3.1 Algorithm

The second stage, which merges the per-view detections to calculate the 3D pose, is the core of the method and has the most optimization potential. It was designed to be as simple and efficient as possible and can be split into the following steps:

1. Create all possible pairs with poses from other views
2. Filter pairs using the previous 3D poses
3. Select the core joints
4. Triangulate each pair to a 3D proposal
5. Drop proposals outside the room
6. Reproject into 2D views
7. Calculate reprojection error to original 2D poses
8. Drop pairs/proposals with large errors
9. Group the remaining 3D proposals in 3D space
10. Triangulate again, but now with all joints
11. Merge the proposal groups into a single 3D pose
12. Drop clearly invalid poses

13. Optional: Assign persons to tracks over time
14. Optional: Clip joint movements to a maximum speed

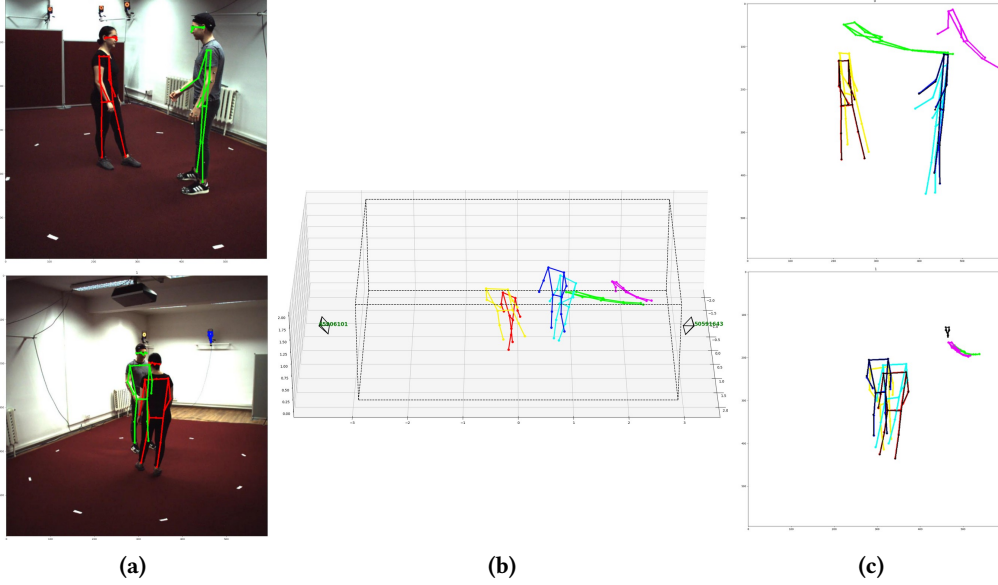


Figure 5.8. Obtaining 3D proposals. The process starts with 2D detections for each view (a, images from *CHI3D* [98], note the small blue colored false positive in the upper-right of the lower image). Then, in step (1), all possible pairs between the views are created. In this case, this leads to six pairs from the two detections above with the three below. The core joints of all pairs are triangulated into 3D proposals (b, steps (3,4)). Then they are reprojected into the 2D views (c, step (6)), and a distance-based error to the original 2D poses (visualized in black in c, tightly overlapping the red and blue poses there) is calculated. As can be seen in the image, the green and pink proposals clearly do not match to their original 2D poses, and receive a very high error. The yellow and light-blue poses resulted from the flipped (man with woman) pairs and also have a notable error. All pairs with errors above a threshold are dropped in step (8). Only the remaining red and dark-blue pairs with low enough errors are used for the further steps (9-12).

In step (1), for each 2D pose, all possible pairs with poses from other views are created, since no association to specific persons is known yet. In the next step (2), the pairs are filtered using the previous 3D poses if available. For this, the 3D poses from the last frame are projected into the 2D views and a pixel-based distance threshold is evaluated. The idea is to reduce the total number of pairs that need to be triangulated. For a pair of 2D person detections in two different views, either both of its 2D poses can match to one of the reprojected 3D poses, only one of them matches, or neither matches. If both 2D poses match, it is likely a valid pair belonging to that 3D person, and it is kept. If only one part matches, a match is unlikely, because the 3D pose should be matchable in both views, so the second part is likely a pose from a different 3D person, and the pair is dropped. If neither matches, it is likely that a new 3D person is present in the current frame, and the pair is kept. In step (3), the core joints are extracted, which are the shoulders, hips, elbows, wrists, knees, and ankles. They are enough to associate the following 3D proposals to a person, and this reduces the computational

complexity in the next steps. In step (4), each pair is triangulated to a 3D proposal (see Figure 5.8b). In the following step (5) clearly invalid proposals outside the observed room are dropped. In step (6), the remaining proposals are reprojected into the 2D views. Step (7) calculates a pixel-distance-based reprojection error to the pair’s original 2D poses, and also prunes clearly invalid limbs. In step (8), pairs/proposals with a large error are dropped, as they are considered invalid matches. In step (9), the remaining 3D proposals are grouped in 3D space. Each of the 2D view pairs created a 3D proposal and if they are close in 3D space, they likely belong to the same person, and are therefore grouped together. Grouping in 3D makes use of physical constraints, which allows a simpler and faster implementation. In step (10), the remaining pairs, which normally are less than in step (4), are triangulated again, but this time with all joints. In step (11), for each group the calculated 3D proposals are merged into a single 3D pose. To filter outliers, first for each joint an average location is calculated, and then for each proposal, the distance of this joint to the average is calculated. If the distance is too large, the joint is dropped. From the remaining joint proposals, the *top-k* closest to the average are selected to drop outliers again, and the final joint location is calculated as the average of those. In general, this late fusion step allows for stronger outlier filtering, because multiple proposals can be used, but also works if a person is only visible in two views. In step (12), clearly invalid poses with too few keypoints, implausible size, or positions outside the room are dropped, and missing joints are filled with their neighboring joints. The general idea behind the algorithmic structure is to use only the minimal necessary information at each step, drop bad proposals early on, and move expensive operations, like the full-body keypoint calculations, to later steps.

Optionally, in steps (13-14) the results can be tracked and smoothed over time. A simple distance-based threshold is used to assign poses to existing tracks. Clipping the joint movements to a maximum speed was preferred to more complex smoothing filters, because it does not introduce response latency to sudden direction changes, which could reduce real-time position accuracy.

Method Comparison

Several differences exist to the other algorithmic approaches in the literature. In comparison to *mvpose* [89], no person identification by appearance similarities is required, as the association is done only by geometric matching. The method of *mv3dpose* [227] matches and groups 2D poses by epipolar geometry instead of the 3D triangulation and grouping used here. *PartAwarePose* [76] groups the 2D poses by assigning them to the 3D poses from the previous view, and uses a temporal filter predicting future joint locations based on previous observations to filter joints, which on the other hand can lead to prediction errors and costs additional computation time. *4DAssociation* [261] follows a bottom-up graph-based concept to match body-parts and connections through space and time to assemble the 3D poses afterward. *Bridgeman et al.* [59] group the 2D poses into persons by first calculating correspondence costs for all pairs of 2D poses, using the minimal distance between two ray projections, then calculate 3D poses and then iteratively calculate costs from the point-to-ray distance to update the 2D–3D matches. The results are further smoothed by a temporal filter. The source-code was not published.

QuickPose [270] splits the detected 2D skeletons into all plausible part-graphs, calculates a ray-based distance between joints in different views, and uses a clustering algorithm to find matching parts. *Chen et al.* [70] iteratively match 2D poses from one view to the already existing 3D persons, and gradually update the matched 3D poses, before continuing with the next view. The last two algorithms are the fastest ones reported in the literature, according to the times stated in their papers, but since their source-code was not published, a detailed comparison is not possible. In comparison to *VoxelKey-pointFusion*, no voxel space is used, which results in much faster calculations, and its person association follows a bottom-up instead of the top-down approach used here.

Performance

Due to the relatively simple concept, the algorithm is very fast. In a test on the *Shelf* [40] dataset, which shows up to four persons in five views, the algorithm requires an average triangulation time of only 0.1 *ms*. The per-step times are listed in Table 5.1.

Step	Time (μ s)
Pose undistortion	14
Pair creation (1)	4
Pair filtering (2)	10
Triangulate and Score (3-7)	28
Grouping (9)	7
Triangulate and Score (10)	28
Merge (11)	12
Post-process (12)	7
Tracking (13-14)	1
Total	115

Table 5.1. Average time for different steps in microseconds.

While the initial version of the algorithm used *OpenCV* [177] functions for most calculations, the current version uses standard *C++* functions only, because the overhead of the *OpenCV* functions was higher than the actual calculation time for the relatively small number of points. Instead of *OpenCV*'s *SVD*-based triangulation, a simple mid-point triangulation was implemented, which was much faster and almost as accurate in the experiments. In summary, the switch to standard *C++* functions led to a significant overall speedup of around two times. Using only the core joints and repeatedly filtering invalid poses helps to keep the subsequent steps efficient. The runtime is further influenced by the number of joints, which are by default 20. In the whole-body case that uses 136 joints, the average runtime increases to 0.4 *ms*. Depending on the integration of the algorithm, data conversion time can lead to further delays. The algorithm itself is implemented in *C++*, but an interface to *Python* (using *SWIG*) is provided as well.

As shown in Table 5.2, the proposed solution is multiple times faster than all the other algorithms, which demonstrates its significant efficiency improvements. Note that all times reported in the different papers were measured on different hardware (see paper for more details). Here a single core of an *AMD-7900X* was used, but the main speedup is clearly caused by the more efficient algorithm.

Method	Time (ms)
mvpose [89]	105
VoxelPose	103
mv3dpose	63
PlaneSweepPose	57
VoxelKeypointFusion	48
Faster-VoxelPose	38
4DAssociation [270]	32
PartAwarePose [76]	10
Bridgeman et al. [59]	9.1
Chen et al. [70]	3.1
QuickPose [270]	2.9
RapidPoseTriangulation	0.1

Table 5.2. Time from 2D to 3D poses on *Shelf* [40] in milliseconds.

5.3.2 Dataset Generalization

As before, the new approach was evaluated on a variety of publicly available datasets. The datasets used for the evaluation were *Human3.6M* [122], *Shelf* [40], *Campus* [40], *MVOR* [221], and *Panoptic* [127], as in Section 3.4.7. Additionally, the datasets *CHI3D* [98], *Tsinghua* [261], and *EgoHumans* [130] were added as well, to allow an even broader evaluation. The metrics used were the same as before and evaluated the same 13 key-points (2 shoulders, 2 hips, 2 elbows, 2 wrists, 2 knees, 2 ankles, 1 nose/head) across each dataset. The *FPS* were measured on an *Nvidia-3090* again. For brevity and readability, for the datasets already evaluated in Section 3.4.7, only the best results/algorithms are listed again.

Standard datasets

On the commonly used datasets, *RapidPoseTriangulation* shows a similar or better generalization performance than most state-of-the-art algorithms, while reaching an outstanding speed at the same time. See Tables 5.3, 5.4, 5.5, 5.6, 5.7.

In the rather simple setup of *Human3.6M* [122], the algorithm performs very well. Much of the *MPJPE* error in this dataset results from different skeleton definitions. The original labels, for example, have higher hips, the nose more inside the head, and the ankles more at the heels than the *COCO* [142] skeleton definition of the 2D pose estimation network. From a human perspective, though, the predictions would often match the images better than the original labels (some examples can be found in the linked code repository [52]).

On the *Campus & Shelf* datasets, the algorithm also performs very well. A visual inspection of samples with large errors showed that a majority of them are caused by label errors, and not by the algorithm itself (some examples in the repository again [52]). Therefore, it needs to be noted that the comparability of the results of well-performing algorithms seems to be somewhat limited.

On *MVOR*, the algorithm suffers, similar to the other RGB-only approaches, from many occlusions and few viewpoints. Even though it is one of the best-performing algorithms, it is still notably behind the depth-based approach of *SimpleDepthPose*.

On the *Panoptic* dataset, its reliability regarding the joint detection (*PCK*) is very high, and only slightly behind the methods that were trained on this dataset. Many of the

Method	PCK@100/500 (%)		MPJPE (mm)	Recall@100/500 (%)		Invalid (%)	F1 (%)	FPS (Hz)
Faster-VoxelPose (synthetic)	75.5	98.8	88.3	75.0	100	0.2	99.5	36.2
VoxelKeypointFusion	81.7	100	64.3	95.0	100	0	100	8.7
RapidPoseTriangulation	88.2	99.9	60.6	94.5	100	0	100	115

Table 5.3. Transfer to *Human3.6M* [122].

Method	PCK@100/500 (%)		MPJPE (mm)	Recall@100/500 (%)		Invalid (%)	F1 (%)	FPS (Hz)
Faster-VoxelPose	88.3	100	59.8	99.4	100	50.7	66.0	17.6
Faster-VoxelPose (synthetic)	87.9	100	58.8	99.0	100	48.8	67.7	17.4
MvP (synthetic)	94.1	99.7	51.8	97.1	100	82.2	30.2	8.5
mv3dpose	91.4	98.4	55.8	94.8	98.5	44.3	71.2	1.6
PartAwarePose	92.7	99.0	51.4	98.5	99.2	47.4	68.7	2.1
VoxelKeypointFusion	93.3	100	51.3	98.3	100	49.1	67.4	5.8
RapidPoseTriangulation	94.4	100	47.5	98.7	100	41.3	74.0	66.6

Table 5.4. Transfer to *Shelf* [40].

Method	PCK@100/500 (%)		MPJPE (mm)	Recall@100/500 (%)		Invalid (%)	F1 (%)	FPS (Hz)
Faster-VoxelPose (synthetic)	41.3	74.4	104	39.6	74.7	2.1	84.8	25.3
mvpose	70.2	99.9	80.4	82.7	100	25.4	85.5	0.5
mv3dpose	64.4	93.4	135	62.5	94.9	10.1	92.4	2.8
PartAwarePose	78.4	98.9	74.7	93.9	98.9	22.7	86.8	5.8
VoxelKeypointFusion	74.6	99.9	84.4	80.3	100	35.5	78.4	7.8
RapidPoseTriangulation	79.9	100	75.2	93.9	100	15.9	91.4	142

Table 5.5. Transfer to *Campus* [40].

Method	PCK@100/500 (%)		MPJPE (mm)	Recall@100/500 (%)		Invalid (%)	F1 (%)	FPS (Hz)
SimpleDepthPose	62.0	94.3	113	54.1	96.6	23.5	85.4	37.2
Faster-VoxelPose (synthetic)	28.1	45.5	109	29.4	46.1	7.7	61.5	30.0
VoxelKeypointFusion	43.9	75.1	128	35.9	76.6	24.2	76.2	11.3
RapidPoseTriangulation	44.6	76.7	120	39.9	77.9	22.1	77.9	121

Table 5.6. Transfer to *MVOR* [221] with depth images (first) and without (others).

Method	PCK@100/500 (%)		MPJPE (mm)	Recall@100/500 (%)		Invalid (%)	F1 (%)	FPS (Hz)
Faster-VoxelPose	98.6	99.9	20.5	99.7	99.9	1.0	99.5	18.0
PRGnet	99.1	99.9	17.1	99.9	99.9	2.0	99.0	6.8
TEMPO	97.4	98.5	16.8	98.4	98.4	2.4	98.0	5.1
VoxelKeypointFusion	94.0	99.7	47.8	97.3	99.9	2.4	98.7	4.2
RapidPoseTriangulation	96.5	99.8	30.6	99.3	99.8	1.8	99.0	57.5

Table 5.7. Replication of *Panoptic* [127] results and transfer without depth.

errors that occurred involve the lower body joints, especially the ankles. While most upper body joints have an average error between 15–25 *mm*, the hips and knees have around 35–40 *mm*, and the ankles around 50 *mm*. This is mainly caused by the position of the cameras, which often cut off parts of the legs. The 2D pose estimation network, however, still predicts those joints at the lower edge of the image (so for example, an ankle is predicted at the height of the knee). This leads to a large error in the triangulation step because the 3D joint is then estimated with a wrong depth, which cannot always be detected as an outlier. For future optimization, this might be better handled if the 2D pose estimation network could predict the visibility of the keypoints as well, so the algorithm could prefer directly visible keypoints over estimated ones.

Additional datasets

To allow an even broader comparison, two further datasets were added, following the same scheme as in Section 3.4.7, and integrated into the *skelda* dataset library.

Method	PCK@100/500 (%)		MPJPE (mm)	Recall@100/500 (%)		Invalid (%)	F1 (%)	FPS (Hz)
VoxelPose (synthetic)	88.4	99.4	72.2	90.5	100	0.5	99.7	15.2
Faster-VoxelPose (synthetic)	88.2	99.2	69.3	94.9	99.6	0.8	99.4	31.9
PRGnet	82.9	99.1	80.1	85.0	100	10.2	94.6	5.0
TEMPO	81.1	90.7	66.5	86.4	91.0	6.6	92.2	8.9
SelfPose3d	71.5	87.2	97.8	66.5	89.7	79.8	32.9	5.1
mvpose	85.2	100	66.7	98.3	100	0.5	99.7	0.2
mv3dpose	53.8	61.1	71.8	55.9	61.4	2.2	75.4	2.6
PartAwarePose	81.6	92.9	77.8	84.0	94.0	8.7	92.5	3.4
PlaneSweepPosePlus	89.9	99.2	63.2	96.9	99.5	0.4	99.6	10.3
VoxelKeypointFusion	81.7	99.1	68.5	95.5	99.4	0.1	99.6	8.0
RapidPoseTriangulation	90.5	99.9	60.3	98.6	100	0	100	97.3

Table 5.8. Transfer to *CHI3D* [98]

Method	PCK@100/500 (%)		MPJPE (mm)	Recall@100/500 (%)		Invalid (%)	F1 (%)	FPS (Hz)
VoxelPose (synthetic)	94.1	99.1	58.9	94.1	99.5	5.7	96.8	5.1
Faster-VoxelPose (synthetic)	87.9	98.6	66.3	89.7	99.0	9.2	94.8	7.7
PRGnet	88.7	99.9	67.8	93.2	100	10.5	94.5	6.3
TEMPO	85.7	90.4	57.3	89.4	90.3	1.4	94.3	5.0
SelfPose3d	83.1	89.2	65.9	83.0	90.2	51.2	63.3	4.9
mvpose	81.5	99.3	79.2	88.6	99.7	4.7	97.4	0.4
mv3dpose	64.8	71.0	63.3	66.8	71.3	14.6	77.7	1.2
PartAwarePose	81.7	94.3	79.7	78.6	94.9	6.7	94.1	0.8
PlaneSweepPosePlus	93.8	98.9	58.0	95.1	98.9	4.3	97.3	6.5
VoxelKeypointFusion	95.9	99.5	51.1	95.1	99.8	0.9	99.4	3.1
RapidPoseTriangulation	96.0	99.8	52.8	95.7	100	0	100	48.4

Table 5.9. Transfer to *Tsinghua* [261]

The first one is the *CHI3D* [98] dataset, which shows two persons interacting with each other. It has the challenge that the persons are often very close to each other, which makes it hard to distinguish between them. The setup is similar to *Human3.6M* and also contains four cameras. The results in Table 5.8 show that generally most persons are detected, with a similar performance as in *Human3.6M*. Regarding *RapidPoseTriangulation*, it was observed that in some cases the 2D pose estimation struggled to

correctly detect joints between very close persons, and the 3D triangulation then did not receive enough inputs to correctly triangulate all joints and could miss a person. One could reduce the reprojection matching threshold to better handle such problematic 2D detections, but at the cost of increasing the number of false positives, or alternatively use the tracking information to fill in missed persons with their last positions. The authors of *CloseMoCap* [212] also reported very good transfer results on this dataset, reaching a $PCK@50$ of 94.3 on average. They also evaluated *VoxelPose*, *Faster-VoxelPose* and *mvpose* on the same dataset and reached much better results for them as well. Additionally, they reported per-joint errors on a subset of the dataset and found that the hip joints had the best results, whereas here the hip joints are among the worst performing joints. In the labels, however, it can be seen that the hips are positioned higher on the body than in the *COCO* skeleton definition, consistent with prior observations in *Human3.6M*. So in conclusion, the mismatch is likely caused by parts of the evaluation or the use of different labels somewhere in *CloseMoCap*, but because their source-code was not published, this assumption cannot be verified.

The second dataset is *Tsinghua* [261], which shows people moving around in a room, while they are being recorded from six different cameras. To prevent multiple training runs for the synthetic models, only the first of the two test sequences was evaluated. The results in Table 5.9 show that almost all models perform relatively well on this dataset, but still many of them miss a few persons. *RapidPoseTriangulation* again shows the best performance, while being multiple times faster than the others at the same time.

Summary

Under the assumption that the general goal of all developed pose estimators is their usage in practical applications, their generalization to unseen datasets is very important. As can be seen in the experiments above and earlier, many models perform well in some setups and poorly in others. For a simpler algorithm comparison, the average performance on all six datasets was calculated. As can be easily seen in Table 5.10, *RapidPoseTriangulation* shows the overall best performance. The previously developed *VoxelKeypointFusion* reaches the second-best average performance, proving its generalization capabilities as well, but is notably slower.

Method	PCK@100/500 (%)		MPJPE (mm)	Recall@100/500 (%)		Invalid (%)	F1 (%)	FPS (Hz)
VoxelPose (synthetic)	68.7	90.5	100	64.6	92.1	26.7	79.3	10.7
Faster-VoxelPose (synthetic)	68.1	86.1	82.6	71.3	86.6	11.5	84.6	24.8
PRGnet	52.4	68.4	134	51.3	69.3	28.8	63.3	9.9
TEMPO	59.9	72.2	79.2	61.9	72.4	21.2	69.7	9.9
SelfPose3d	55.6	77.4	115	52.4	79.0	68.3	42.5	7.0
mvpose	70.5	90.0	81.6	76.8	90.4	19.6	83.3	0.4
mv3dpose	53.9	64.4	110	54.6	65.0	21.5	66.4	2.3
PartAwarePose	71.0	83.9	91.5	74.0	84.6	17.7	79.9	3.9
PlaneSweepPosePlus	72.3	87.5	86.9	70.9	88.4	23.8	80.3	10.1
VoxelKeypointFusion	78.5	95.6	74.6	83.4	96.0	18.3	86.8	7.5
RapidPoseTriangulation	82.3	96.0	69.4	86.9	96.3	13.2	90.5	98.4

Table 5.10. Averaged generalization on all six unseen datasets (*Human3.6M*, *Shelf*, *Campus*, *MVOR*, *CHI3D*, *Tsinghua*).

Looking more closely at the results, both here and in Section 3.4.7, some general observations can be made regarding the different approaches:

The (voxel-based) learned methods all have generalization problems, as they work well on the same dataset (*Panoptic*), but show notable performance losses when transferred to different setups. The synthetic training reduced this somewhat in many cases, but not to the full extent. And as *VoxelKeypointFusion* has shown, the problems do not come from the voxel representation itself.

The previously existing algebraic methods often fail in person association. For example, in Table 5.4 and Table 5.5, the approaches *mv3dpose* and *PartAwarePose* were not able to detect all persons (*Recall@500*). While generally most methods use different 2D pose estimators, improvements over time in those are not the main reason for this problem (as shown in more detail in the paper, which conducted some experiments using the same 2D pose inputs). The same problem can be seen in Table 3.5 or Table 5.9. The better performance of *RapidPoseTriangulation* in this regard can be explained by the different matching strategy. Unlike the others, *RPT* triangulates all possible view-pairs to 3D poses, and then drops those with large errors (steps 1-8). So if a person is seen in only two images, it is very likely to be reconstructed. This contributes substantially to the reliability of *RapidPoseTriangulation*, which is why this approach was chosen, even though it results in a rather high number of computationally expensive triangulations. Grouping all those 3D proposals to persons (step 9) is very robust as well, because it is highly unlikely that two persons have very similar 3D poses while standing at basically the same spot. Because there are no projection-caused similarities, this is more reliable than 2D pose matching across views, especially in more crowded scenes, which have a higher probability of similar-looking persons and therefore more false matches. And compared to appearance-based matching, it has no problems with similar-looking persons (like in *MVOR*) and is orders of magnitude faster. While a larger number of persons also affects the likelihood of wrong 3D proposals generated in *RapidPoseTriangulation*, they can easily be filtered out by setting a higher minimal proposal group size.

Regarding the joint accuracy (*PCK@100*), having multiple 3D proposals per person from the different view-pairs allows for a very efficient but robust outlier detection. At its core, it is based on majority voting, which removes very bad proposals, without allowing them to influence the final position, which would not be possible with iterative matching or pure (weighted) averaging as in other works.

Using more cameras

The previous results demonstrate that the number of cameras has a large impact on the performance of the algorithms. Especially *MVOR* shows that using only three cameras in the presence of strong occlusions leads to a large decrease in performance. From an application point of view, using more cameras is therefore recommended, and as planned in the new setup requirements, the new system needs to handle 10 cameras.

To evaluate the impact of a large number of cameras, the *EgoHumans* [130] dataset was selected (Table 5.11). It shows between 2–8 people in different indoor and outdoor

settings (such as building with lego bricks, fencing, or playing tag, basketball, volleyball, badminton, or tennis). They are recorded by 8–20 cameras. The supervised space is up to $25 \times 14 m$ in size, which is much larger than in the other datasets. The last recording of each setting was selected as the test set, and each recording was subsampled into segments of 3s consecutive motion with larger gaps in between again.

Subset & Method	PCK@100/500 (%)		MPJPE (mm)	Recall@100/500 (%)		Invalid (%)	F1 (%)	Time: Demosaic-2D-3D (ms)		
<i>legoassemble</i>										
VoxelPose (synthetic)	74.4	93.8	94.7	65.3	96.1	71.2	44.4		392	
Faster-VoxelPose (synthetic)	64.1	94.2	107	54.0	95.3	0	97.6		239	
VoxelKeypointFusion	97.2	99.4	37.0	99.4	99.7	0	99.9		205	271
RapidPoseTriangulation	99.0	100	25.3	100	100	0	100	11.2	23.3	0.3
<i>tennis</i>										
VoxelPose (synthetic)	29.5	86.5	128	3.9	87.0	1.3	92.5		1077	
VoxelKeypointFusion	90.6	100	64.6	98.1	100	0	100		434	679
RapidPoseTriangulation	99.5	100	22.7	99.7	100	0	100	26.8	50.1	0.6
<i>tagging</i>										
VoxelKeypointFusion	84.2	92.4	67.1	84.3	94.3	8.0	93.1		207	379
RapidPoseTriangulation	92.9	99.4	47.9	90.0	100	4.1	97.9	11.0	26.5	0.3
<i>fencing</i>										
VoxelKeypointFusion	97.7	100	37.5	100	100	69.0	47.3		335	420
RapidPoseTriangulation	99.6	100	23.9	100	100	50.0	66.7	12.3	27.9	0.4
<i>basketball</i>										
VoxelKeypointFusion	92.2	97.6	53.5	94.8	98.2	50.9	65.5		212	804
RapidPoseTriangulation	96.3	100	40.8	99.7	100	24.8	85.8	11.0	27.1	0.4
<i>volleyball</i>										
VoxelKeypointFusion	94.1	98.0	48.5	96.1	98.6	25.2	85.0		449	1004
RapidPoseTriangulation	99.2	100	27.6	100	100	5.8	97.0	18.0	56.4	1.1
<i>badminton</i>										
VoxelKeypointFusion	99.0	100	36.5	100	100	16.2	91.2		391	846
RapidPoseTriangulation	99.5	100	24.3	100	100	0.0	100	17.5	42.1	0.7

Table 5.11. Transfer to *EgoHumans* [130]

Because the cameras use fisheye lenses, the images are more strongly distorted than in the other datasets. Therefore, a different undistortion method is required. Since this requires larger changes in the pipeline of most algorithms, only a subset of the better performing algorithms was tested on this dataset. In some cases, the poses can be undistorted after the 2D estimation step, while in other cases, the images need to be undistorted instead. *VoxelPose* and *Faster-VoxelPose*, for which the images are undistorted beforehand, both have problems with a direct transfer to the *legoassemble* subset, but again show much better performance after synthetic training. The same holds true for *VoxelPose* on the *tennis* subset, while *Faster-VoxelPose* on the other hand does not work at all here. Because of the large playing field, the synthetic training always failed with an out-of-memory error, even at the lowest possible batch-size. For *mvpose*, the number of views was too high, the six cameras from *Tsinghua* already took up the entire GPU memory. *TEMPO* [75] described an evaluation on this dataset, in which they reported an average transfer error of 120 mm (or 36.7 mm after training) on this dataset, but they did not provide further information about the evaluation in the paper or their code.

VoxelKeypointFusion proved its generalization capabilities with a very good performance, which was only surpassed, especially in terms of speed, by *RapidPoseTriangulation*. The *MPJPE* results show that the outlier-based joint merging approach leads to better joint accuracy than the weighted voxel-based averaging if many proposal options are

available, which is the case with the many cameras here. Regarding the joint detection reliability (*PCK*) of *RapidPoseTriangulation*, it can be seen that the performance is notably better than in the previous datasets, which shows the importance of having enough cameras to cover the scene, especially in the presence of significant occlusion.

About real-time performance

Regarding real-time performance with many cameras, the results in Table 5.11 show that the 3D part of *RapidPoseTriangulation* is very fast, taking at most 1 *ms*, and that the majority of the time is currently spent on 2D pose estimation.

In comparison to the original implementation of *RTMPose*, the preprocessing pipeline and network interfacing were significantly optimized for the above evaluation experiments. Some parts of the preprocessing were moved to the GPU, and the image is transferred using the *uint8* data-type to decrease transfer time. In addition, box cropping and padding were optimized, and for image resizing nearest-neighbor instead of linear interpolation is used for further speedups. The entire pipeline was implemented in *C++* as well. Cropping and resizing directly on the GPU did not show notable speedups, however, as transferring the full image offsets its benefits.

To further reduce the input latency in real applications, the *Bayer* format is used directly. The conversion into *RGB* is now integrated into the preprocessing pipeline. For the dataset evaluation, the *RGB* images were first converted back to *Bayer* encoding. Interestingly, this back-and-forth conversion slightly improved the *MPJPE* score in many datasets. The demosaicing time is already included in the *FPS* values of the previous tables. In the case of *FullHD* resolution, it is around 0.2 *ms* per image. With *4K* inputs, like in the *EgoHumans* dataset, it increases to a notable 1.4 *ms* per image, so demosaicing after cropping and resizing, or directly using *Bayer* images as network input, might be future optimization possibilities.

In real-time applications, as it is the case in this work, it is also important to weigh algorithmic accuracy against latency. Because persons normally move around, estimations with high latency result in a larger error relative to the true position of the human. For example, a person moving with 2 *m/s* would move 20 *mm* in only 10 *ms*, which is already more than the average *MPJPE* difference between the better-performing algorithms from Table 5.10. Algorithmic speed can therefore significantly improve real-time-position accuracy.

Another important latency factor in real-time systems is the transfer time of the image from the camera to the computer, especially if multiple cameras feed one host. As mentioned before, a *smart camera* approach, in which the 2D pose estimation is done on the smart edge device itself, would be optimal for this, as it would only require sending a few keypoint coordinates instead of the full image. Therefore, the algorithm was designed in a way that supports this decentralized approach. Because only a single image is processed per time-step, this would also allow for the use of less powerful hardware. A *Raspberry Pi* with the *AI-Hat* [193] could already be sufficient, though the previously selected *Jetson* is a better choice to achieve lower latency. If the price is not a concern, each camera could even be equipped with one standard GPU. The

RTX-3090 used before takes about 1 *ms* per image and 1 *ms* per person to estimate their 2D pose. With an RTX-4080 this was even 45% faster, and with an RTX-5090 (which has about $2.5\times$ more FLOPS than the RTX-3090), it should be possible to achieve full image-to-pose 3D prediction in only one millisecond.

5.3.3 Whole-body Estimation

One of the requirements for the new pose estimation algorithm was the ability to also predict finger keypoints. For this, both the 2D and 3D parts of any algorithm need to be able to predict and handle those additional keypoints. Regarding the 2D part, there are already some models that can predict whole-body keypoints, for example the *RTMPose* [126] model used before. For multi-view 3D estimation, however, this feature was not yet developed.

One significant benefit of algorithmic approaches over learning-based methods is that they are not restricted to predefined keypoints (typically specified by the training dataset), allowing them to handle arbitrary inputs and making them well-suited for *whole-body* estimation. *VoxelKeypointFusion* can be extended to this by simply running the voxelization steps with additional keypoints, which was already shown in its paper [47]. For learning-based approaches, on the other hand, an additional training dataset with whole-body keypoints would be required, which, due to the large number of keypoints, is very difficult to label. But even if this problem is solved, both learning and voxel-based methods are rather slow (as shown in Table 5.2), which makes them less suitable for real-time applications. Moreover, as *VoxelKeypointFusion* has shown in the next experiment, adding substantially more keypoints makes them even slower.

To evaluate the performance, the *H3WB* dataset [272] was used, which extended a part of the *Human3.6M* dataset with whole-body keypoints (17 body, 6 foot, 68 face, 42 hand). For simplification of the labeling process, the authors used only frames with good visibility of the actor. Therefore, the dataset is somewhat simpler than the original one (the default model of *RapidPoseTriangulation* reaches a *MPJPE* of 23.7 *mm*, and the authors estimated around 17 *mm* labeling error). The whole-body results can be found in Table 5.12. Since this dataset is normally used for single-view 2D-to-3D pose lifting, no other comparable results were found.

Method	PCK@100/500 (%)		MPJPE (All Body Face Hands) (mm)				Recall@100/500 (%)		Invalid (%)	FPS (Hz)
VoxelKeypointFusion	96.8	99.8	40.5	37.3	38.3	40.7	98.5	100	0.5	4.9
RapidPoseTriangulation	88.3	100	45.7	29.1	54.3	41.8	100	100	0	82.6

Table 5.12. Whole-body estimation on *H3WB*.

In general, the whole-body prediction works quite well, though there are still some improvement possibilities. One problem of *VoxelKeypointFusion* are the artifacts of the voxelization process, which, due to the discretization, push some keypoints closer together than they really are. This was especially visible at the finger keypoints, which were often agglomerated together instead of showing distinct finger lines. *RapidPoseTriangulation* does not have this problem anymore, since it directly triangulates in

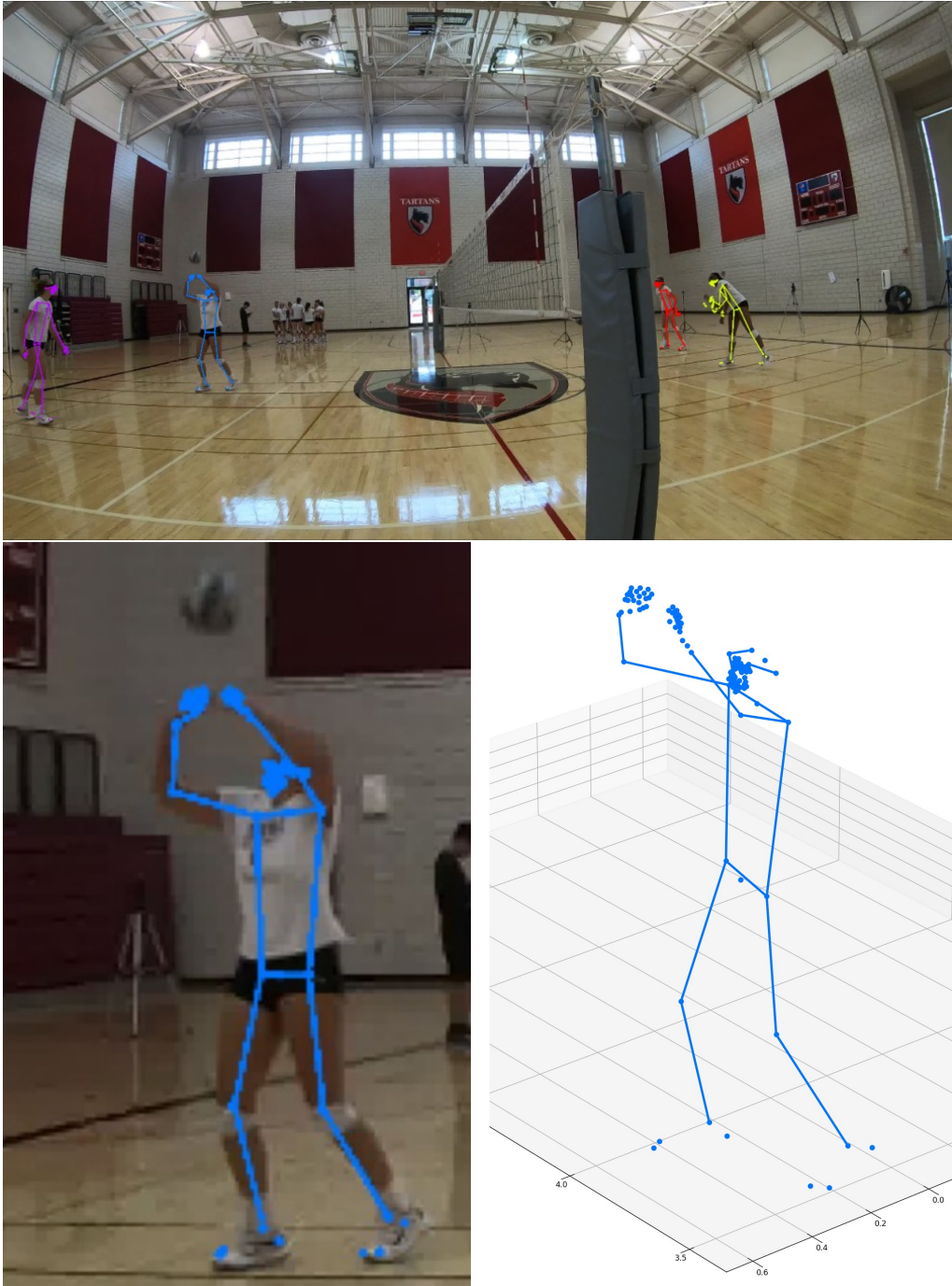


Figure 5.9. Example of a multi-person whole-body pose estimation from multiple camera views in a volleyball game (from the *EgoHumans* dataset [130]). On top, the full image of one camera with the projections of all the detected poses, on the bottom-left a zoom-in on one player, and on the bottom-right her detected 3D pose.

continuous coordinates. On the other hand, smaller errors in the 2D keypoint prediction can lead to larger errors in 3D. For example, if a keypoint is visible from three views, *VoxelKeypointFusion* merges them all at once using heatmap-beams. So if there are small 2D position errors, the beams of all three views still partially overlap. *RapidPoseTriangulation*, on the other hand, triangulates each pair of views separately, and merges them afterward, but the errors do not necessarily cancel themselves out again. This is more severe if the keypoints can only be seen in very few views, such as in this dataset, since it is not possible to detect and drop outliers before the merging step.

A large difference can be seen in the inference speed, especially in the 3D part. *VoxelKeypointFusion* took on average 122 *ms* for the whole-body prediction. *RapidPoseTriangulation* on the other hand only took 0.1 *ms* for this step, which is approximately 1100 $\times$ faster, making it significantly more suitable for real-time applications.

5.3.4 Conclusion

This section presented *RapidPoseTriangulation*, a new algorithm for 3D human pose triangulation. Its basic idea is to triangulate pairs of 2D poses from different views separately, and to group and merge them into final poses directly in 3D space. This relatively simple concept makes it very efficient, and much faster than any previous algorithm.

In a broad evaluation of the algorithm on several datasets, it showed state-of-the-art performance. It scales well with an increased number of cameras and is especially suited for real-time applications. Besides that, the results also show that a relatively simple algebraic approach can still outperform many recent fully differentiable or end-to-end learned multi-view methods. This suggests that the current trend of increasing model complexity does not necessarily translate into better exploitation of the geometric structure.

In summary, it is one of the best and by far the fastest method for multi-view multi-person human pose estimation currently available. In contrast to most prior works, it is also able to predict whole-body poses, which can be used to implement more advanced follow-up tasks. By making the source-code publicly available¹, it is intended to contribute to the development of more intuitive and safer human-computer or human-robot interactions.

The paper about *RapidPoseTriangulation* [52] also contains some additional experiments, for example on the influence of the pose backend or important algorithmic parameters. Please refer to it for more details in this regard.

¹<https://gitlab.com/Percipote/RapidPoseTriangulation>

5.4 Evaluation

The safety as well as the usability of the updated system can now be evaluated, using the same scenario as in *KoARob* again.

5.4.1 Pose Estimation

In order to evaluate the reliability of person recognition in the given scenario, a separate dataset was created and labeled, following the same procedure as in *KoARob*. Since the overall hardware setup was changed, the previous dataset can no longer be used. To create the new dataset, several participants were asked to perform the “returns handling” scenario with the robot. As before, the robot placed several packages with different contents on top of the table in front of the worker, which they then inspected, and, depending on whether an object was present or not, sorted them into two boxes for reusable returns or waste. Meanwhile, the robot was monitored by the new safety system and stopped automatically if the distance between the person and the robot became too small.

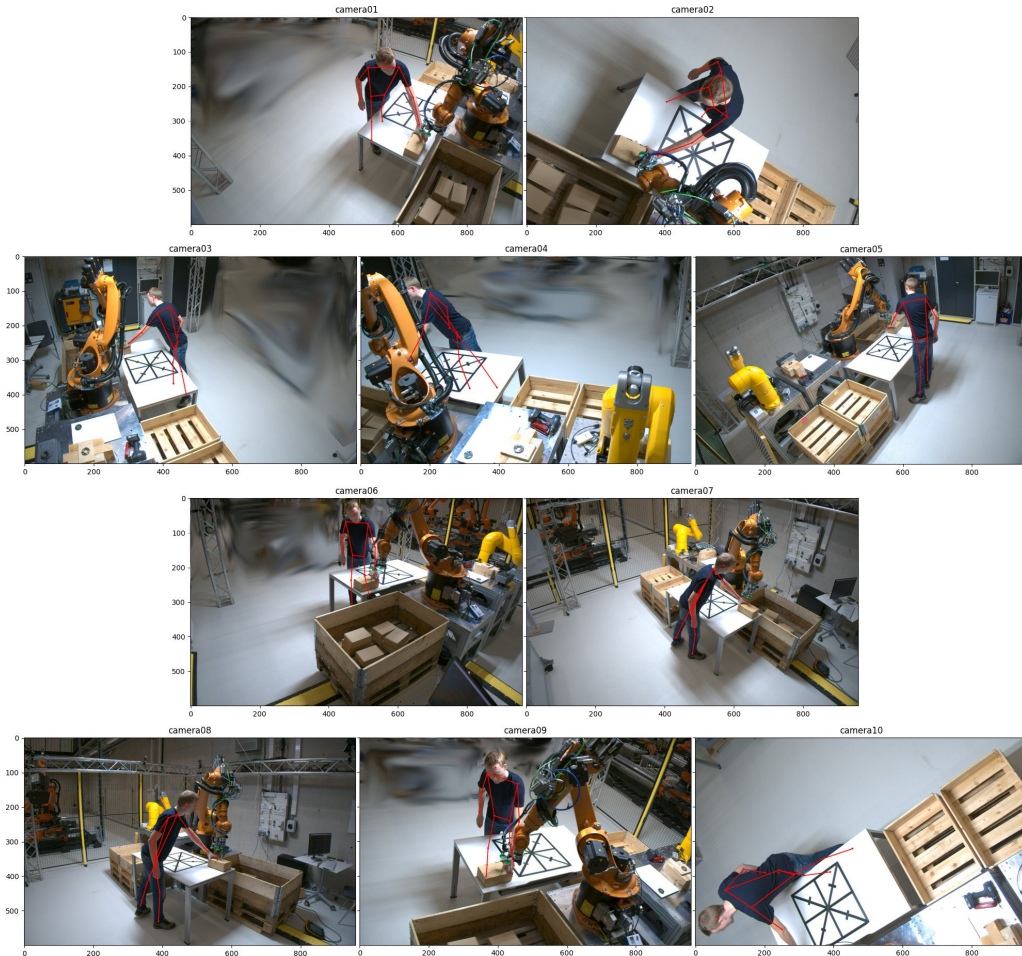


Figure 5.10. Evaluation dataset example of the projections of a labeled 3D pose into the ten camera views. [51][© 2025 IEEE].

The images from the cameras were recorded during this process, and afterward, parts of the sequences were labeled (see Figure 5.10 for an example). A semi-automatic procedure was used again, in which the pose estimator created an initial suggestion for the joint positions, which was then corrected manually to fix any notable errors (greater than around 5 cm). For this, a new annotation tool was developed, which allows for easy adjustment of the joint positions in 3D, while showing live updates of the projections into the camera views, enabling an intuitive and efficient labeling process (Figure 5.11). This tool was integrated into the *skelda* library for anyone to use it to label their own datasets in the future.

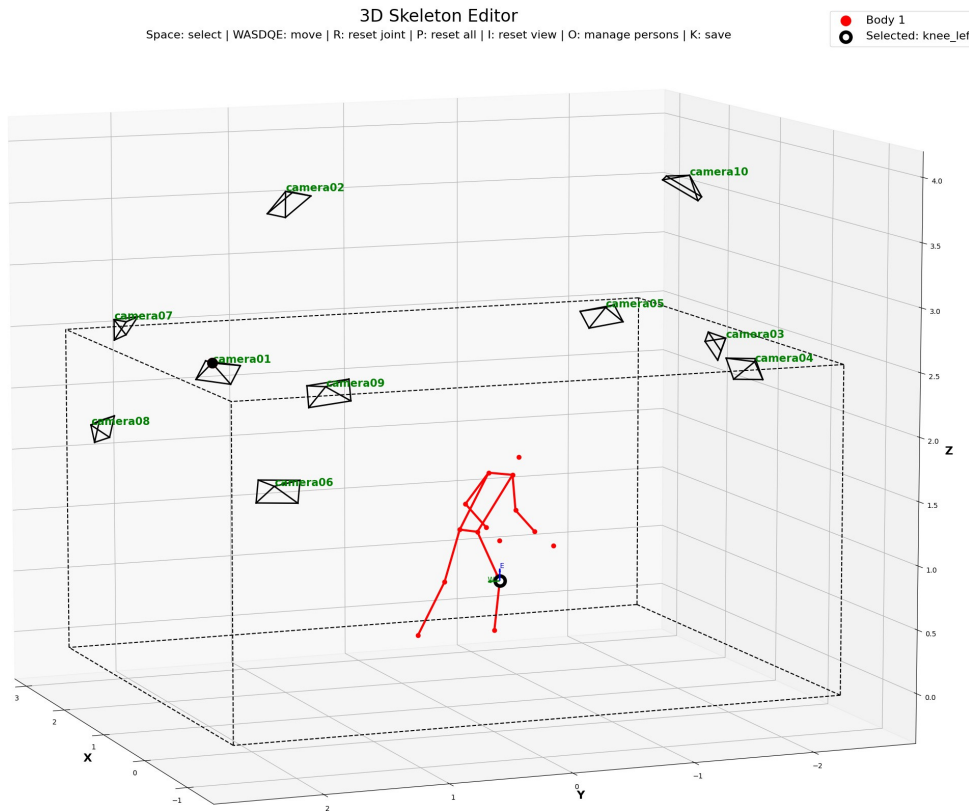


Figure 5.11. To annotate an image set, the initial 3D pose estimation is loaded and the joint positions can be adjusted in 3D using keyboard commands, while the projections into the camera views are displayed and updated live in a second window (similar to Figure 5.10).

To reduce the manual label correction effort, only the 13 main joints and the tip of the left and right middle finger were labeled (*RPT* predicts 20 body, 6 foot, 68 face, and 42 hand joints by default). A total of 1040 images from 6 people were labeled. The recorded sequences have a frame rate of 50Hz and were split into a few seconds of consecutive frames that were labeled, followed by a larger gap before the next labeled segment.

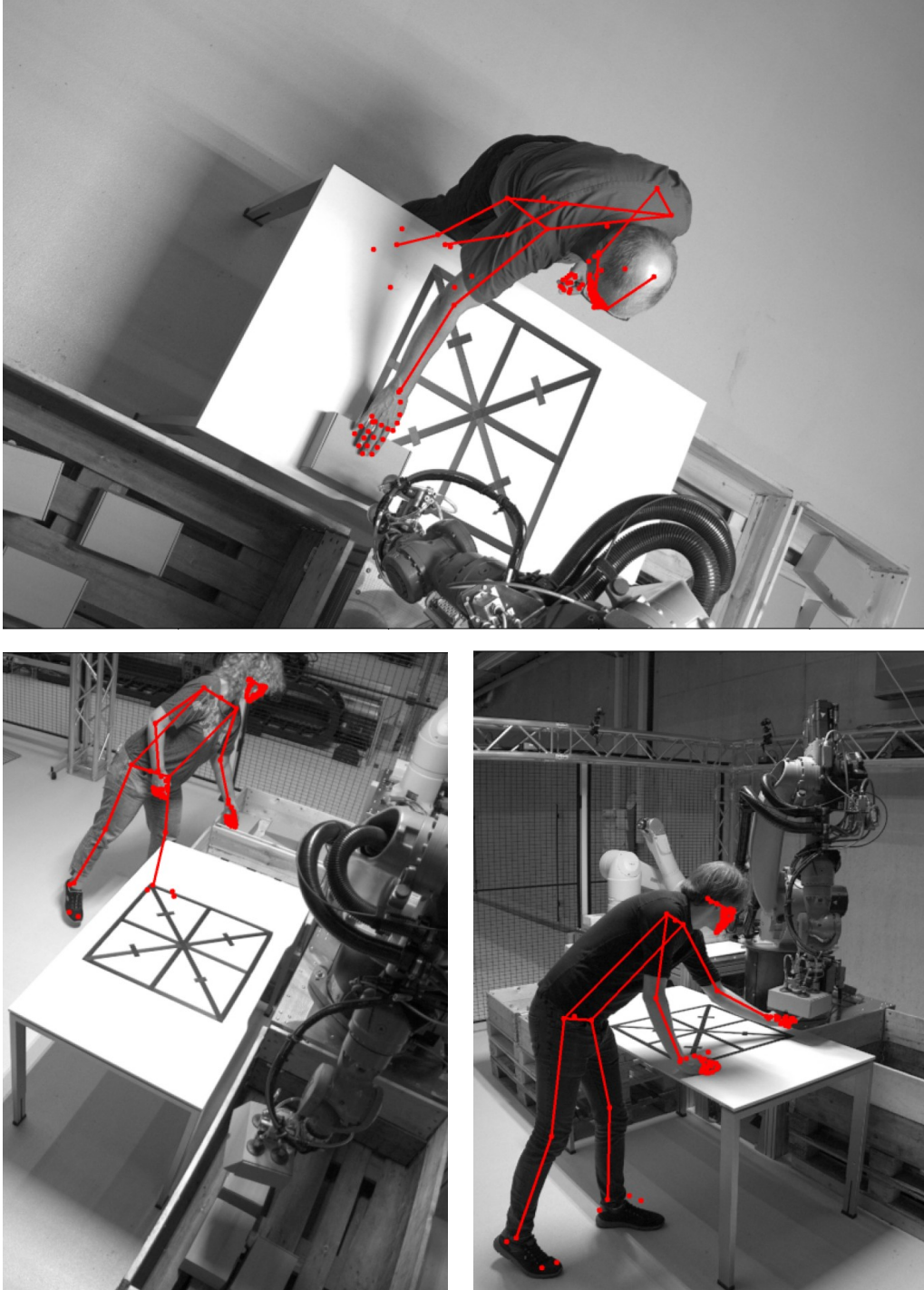


Figure 5.12. Three examples of reprojected whole-body pose detections on the new evaluation dataset.

The results in Table 5.13 show that all joints can be reliably detected ($PCK@500$). On average, the predictions match the labels very accurately ($MPJPE$ of 4.0 mm), but this very low value is heavily influenced by the labeling process, which only corrected larger errors and directly took over the prediction proposals of the estimator in the other cases. As can be seen in the other two PCK scores, the number of outliers with a higher error is very low as well. The FPS measurement on a single GPU serves as a baseline to compare the improvements by the distributed approach, which, as can be seen later, will be notably faster.

Method	PCK@100/250/500 (%)			MPJPE (mm)	Recall@500 (%)	Invalid (%)	F1 (%)	FPS (Hz)
RapidPoseTriangulation	99.3	99.9	100	4.0	100	0	100	37.5

Table 5.13. Pose estimation results on the new evaluation dataset, using standard computer vision metrics. The joint position error thresholds for the *Percentage of Correct Keypoints* and the value of the *Mean Per Joint Prediction Error* are all in millimeters. The *Recall* reports the percentage of how many persons were detected with a $MPJPE$ below the given threshold, and *Invalid* how many persons could not be matched to any ground-truth label, *F1* combines both scores. The *FPS* were measured on a single NVIDIA-RTX4080 and an AMD-9900X. [51][© 2025 IEEE].

In Table 5.14, the maximum detection errors per joint are presented. The head and shoulders are detected with very high accuracy. The ankles are often strongly occluded, resulting in a larger error. The wrists and fingertips are detected with a very high accuracy in most samples, but there were a few outliers from samples with strong occlusions. Section 7.2.1 will take a closer look at those outliers and how they can be reduced, but for now they are accepted as they are.

Head	Shoulders		Elbows		Wrists		Hips		Knees		Ankles		Middle-Finger-Tip	
0.007	0.020	0.013	0.128	0.031	0.236	0.203	0.068	0.059	0.106	0.075	0.210	0.175	0.236	0.332

Table 5.14. Results per joint (thresholds for $PCK@100\%$ in meters, left|right) on the new evaluation dataset. [51][© 2025 IEEE].

5.4.2 Robot Stop Time

As noted before, the time the robot needs to stop was estimated using the robot’s datasheet. To verify this estimation, an experiment was performed, in which the robot movements after a stop had been triggered were recorded and analyzed. The robot moved with the current default speed of 0.2 m/s between the package take-up and drop positions, and was randomly stopped every few seconds by sending the stop signal via the safety system. To simulate the package, a weight of about 4.5 kg was mounted to the robot flange. In total 100 measurements were taken.

Figure 5.13 shows the motion that took the longest. The image shows that after the robot has basically stopped, it still moves slightly due to vibrations. This raises the question of when exactly the robot can be considered as having *stopped*. The previously described ISO-standards do not contain an explicit definition of exceptions, so the strict definition

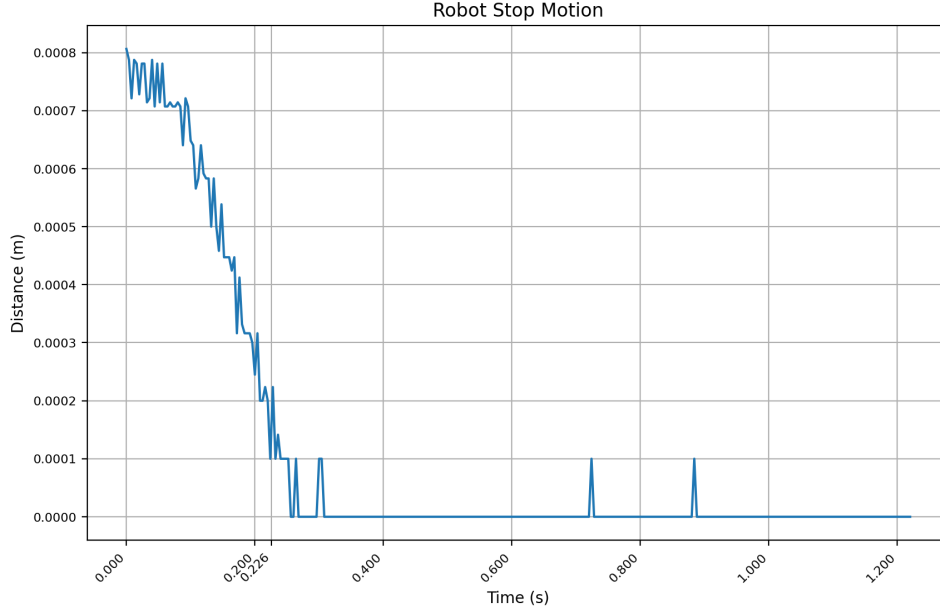


Figure 5.13. Robot motion after stop signal was sent by the safety system. The y-axis represents the robot's velocity, measured as distance moved within 4 *ms* intervals. [51][© 2025 IEEE].

of *all robot motion has ceased* would be that the robot also needs to stop vibrating. The datasheet of a *Mitsubishi* robot [164] states that a stop is reached when the speed falls below 10 *mm/s*. In the *KUKA* datasheet [137], no such statement was found, but its reported stop times suggest that vibrations are likewise excluded. Because the minimal speed that can be measured in this setup is 25 *mm/s* (0.1 *mm* in 4 *ms* intervals), a stop is instead considered at the moment at which the robot does not move more than 1 *mm* in the next second. The maximum stop time was measured as 0.226 *s* and the maximum stop distance as 0.027 *m*, which are both close to the original estimates.

5.4.3 Collaboration Distances & Delays

In the scenario evaluation, the following values were measured:

$$\begin{aligned}
 Speed_{human} &= 2.0m/s \\
 Speed_{robot} &= 0.2m/s \\
 Time_{robot\ stop} &= 0.226s \\
 Time_{system\ delay} &= 0.016s_{(cameras)} \\
 &\quad + 0.013s_{(RapidPoseTriangulation)} \\
 &\quad + 0.001s_{(spherization\ \&\ collisions)} \\
 &\quad + 0.020s_{(interleaving)} \\
 &= 0.05s
 \end{aligned} \tag{5.5}$$

In comparison to the previous values (total delay of 0.6 *s*), the new system's delay has been reduced by about twelve times. The cameras are notably faster now, do not require

an extra synchronization delay, and the new pose estimation is also around $12\times$ faster, even though the number of cameras was doubled. In comparison to the single GPU measurement of Table 5.13 (without any additional image transfer times), the distributed pose estimation is notably faster, using more but less powerful hardware. The distance monitoring module now requires notably less than 1 ms to calculate and check the distances. This shows the success of the aforementioned optimizations implemented in this work.

Each human joint type has a different safety distance, depending on its thickness (intrusion distance), because only the center of the joint is estimated, and on the joint's prediction error. If the wrist joints are taken as an example, the following distances can be calculated:

$$\begin{aligned}
 S_{h_1} &= 0.05s * 2.0m/s = 0.1m \\
 S_{h_2} &= 0.226s * 2.0m/s = 0.452m \\
 S_r &= 0.05s * 0.2m/s = 0.01m \\
 S_s &= 0.027m \\
 C &= 0.03m \\
 Z_d &= 0.236m + 0.05m \\
 Z_r &= 0.002m
 \end{aligned} \tag{5.6}$$

As mentioned before, for the labeling a maximum error of 5 cm is assumed, which is added to the uncertainty of the joint's position. With the current approach, the camera calibration error cannot be measured directly, but only represents an internal consistency error. It is therefore attributed to the labeling error as well, because the labeling process is based on the projections into the camera views, which are directly influenced by the calibration. In summary, the new safety distance is:

$$\begin{aligned}
 S_p &= 0.1m + 0.452m + 0.01m + 0.027m + 0.03m + 0.286m + 0.002m \\
 &= 0.907m
 \end{aligned} \tag{5.7}$$

In comparison to the values of Section 5.1, the required safety distance has improved substantially (the previous value was 2.405 m). It can be split into the following parts again:

$$\begin{aligned}
 S_{robot} &= 0.452m + 0.027m + 0.002m &= 0.481m \\
 S_{system} &= 0.1m + 0.01m + 0.03m + 0.286m &= 0.426m
 \end{aligned} \tag{5.8}$$

Again, the first component, which is defined by the robot itself, is mainly caused by the long braking time. In the second part, the contribution of the system's delay was greatly reduced due to the improvements of this work (the old was 1.878 m). The intrusion distance is also much smaller now. The uncertainty of the human's position has improved as well, but has the most potential of being further reduced with additional research, as will be shown in Section 7.3.

Another option to further reduce the safety distance, which also will be investigated more closely in Section 7.6.2, is making use of the regulations for force-limited collaboration. Instead of requiring the robot to have completely stopped before it contacts a human, it should be possible to allow the robot to reach contact while it is still in the braking phase, provided that force limits for the contacted body-part are not exceeded.

5.5 Related Work

To assess the performance of *Tutabo-1*, it is important to compare it with existing systems. For common light curtains, a system delay of around 11–36 *ms* is typical [213]. Their intrusion distance depends upon the type and resolution of the light curtain. If the beams are so tight that no finger can fit through, it is zero, if a hand fits between, it is 208 *mm*, and if an arm fits through, it is 850 *mm* [22].

Laser scanners that are used for safety applications have a system delay of around 70–90 *ms* and an intrusion distance of around 65–100 *mm* [214]. In case the scanner has multiple layers that are not dense enough, the intrusion distance can be much larger, because a human might fit a whole arm between them. If only one layer at foot-level is used, it will become even larger, because the human could bend over, reach forward with his hand, and after that, start to walk into the safety zone.

Time-of-Flight cameras can have a system delay of around 55 *ms* and an intrusion distance of around 65 *mm* [216]. This type also only has a performance level of PL_c . Radar motion detectors have a system delay of around 100 *ms* [215]. As the other sensors, they also cannot distinguish between objects or humans entering the workspace.

A specialized light-based end-of-arm-safeguard that protects the area around the robot's end-effector has a system delay of around 110 *ms* [217] and also only PL_c .

The *Veo Freemove* system [25], which uses multiple Time-of-Flight cameras, is closest to the proposed system regarding the size of the safety zone around the human worker. According to the datasheet, the latency of the system is 100 *ms* and the resolution 25 *mm*. In a blog post on their website [25] they stated a minimal value of $C + Z_d + Z_r = 0.218m$, together with a minimal object volume of 15000 *cm*³ that can trigger a robot stop (a human with 75 *kg* has a volume of approximately 75000 *cm*³). Using the values from the previous section, this would result in a system contribution of $S_{system} = 0.2m + 0.02m + 0.218m - 0.002m = 0.436m$, which the proposed system achieves as well. As already stated before, the main disadvantage of purely depth-based systems is that they only distinguish between large unknown objects and background, and a finer distinction between human body parts, like head or hands, is not possible.

Most other research works in related fields do not focus on a detailed safety analysis and, apart for a few exceptions, also do not report their system's delay. The work of [138], with the system that attached active light markers to the human hands to interact with a standard industrial robot, reported a system delay of 15 *ms*. The authors of [38] measured a response time of 1052 *ms* to react to a risky situation. In [32], in which a camera with a top-down view detects the heads of humans in the workspace and creates a cylindrical safety zone, the system's average reaction time was reported with 36 *ms*, but a detailed description of what was included is missing, and as mentioned in Section 3.6, it has some reliability issues as well. The authors of [61] evaluated a camera-based pose estimation system with smart edge cameras for non-safety applications and reported an average pipeline delay for the pose estimation between 89–200 *ms*, depending on the number of cameras used. The work of [55] presented a similar edge-based approach that required around 50 *ms* for the pose estimation step. The real-time performance of *KoARob* was already evaluated in depth in Section 5.1, where a system delay of 0.6 *s* and a safety distance S_{system} of about 1.9 *m* were calculated.

5.6 Conclusion

The main focus of this chapter was improving the latency of the proposed AI-based safety system. To speed up the system, both hardware and software optimizations were implemented. On the hardware side, the concept of smart cameras was introduced, which allows distributed local processing of the images. On the software side, a new pose estimation method was developed, which is by far the fastest multi-view multi-person human pose estimation method currently available. The final evaluation showed that the new system is now multiple times faster than before. At the same time, it is still able to reliably detect the human's body parts and to stop the robot in time.

The main contribution of this chapter is to show that AI-based systems can meet the real-time processing constraints required for safety-critical applications. The comparison with available safety-certified systems shows that the proposed system is in a similar latency range, even though it has the most complex information processing.

Following the iterative development process introduced earlier, the next chapters will focus on further improving the reliability and performance of the system.

Summary. This chapter gives a short overview of the safety regulations for using AI in safety-related systems.

6

Safety Foundations 3: Using AI

6.1	Legislation	112
6.2	ISO/IEC-TR-5469	112
6.2.1	Classification of AI systems	113
6.2.2	Desirable Properties	113
6.2.3	Validation	113
6.2.4	Proposed Architectures	114

According to *EN ISO 13849-1*, artificial intelligence can be used in safety-related systems, but it does not specify any additional requirements itself [114, Chap. 7.1]. Some more information can be found in different regulations though, which will be summarized in the following. Note that these documents are very recent and were published during the development of this work, around the time when the *KoARob* system was already finished.

6.1 Legislation

The new *EU-AI-Act* [20] regulates the use of AI in general and states various obligations for systems that are classified as *high-risk* or as having *systemic* impact. A system that is put into service is generally considered *high-risk* if it fulfills two conditions: a) *intended to be used as a safety component* and b) *required to undergo a third-party conformity assessment* [20, Art. 6, (1)]. The new *EU Regulation 2023/1230 on Machinery* [19], which replaces the previous *EU Directive 2006/42/EC on Machinery*, states in this regard that *third-party conformity assessment of software ensuring safety functions [...] should only apply to systems with a fully or partially self-evolving behaviour using machine learning approaches* [19, Rec. 55]. This means that the system developed in this work, which is based on pre-trained models and does not have any self-evolving behavior, should not be classified as a *high-risk* application [19, Ann. 3, Part B, (1)][62], and therefore be subject to less strict regulatory obligations, even though the legal situation regarding this matter is not yet fully clarified [62, 84]. Apart from that, the *EU-AI-Act* would also provide additional exceptions for *open-source* and *research* systems [20, Art. 2, (6),(12)].

The new *EU-Cyber-Resilience-Act* [21] adds further requirements for machinery with certain digital elements and states that they have to comply with the *essential cybersecurity requirements* defined in this regulation [21, Rec. 53]. It *applies to products with digital elements made available on the market, the intended purpose or reasonably foreseeable use of which includes a direct or indirect logical or physical data connection to a device or network* [21, Art. 2, (1)]. The system developed in this work can technically run without any internet connection, in a fully air-gapped mode. Therefore, it would not necessarily be subject to the requirements of the *EU-Cyber-Resilience-Act*. For the future, it however might be worth to consider creating compliance with this regulation as well, to allow adding features such as automatic updates or remote monitoring, but this is outside the scope of this work.

6.2 ISO/IEC-TR-5469

While different AI-related standards already exist (*ISO/IEC 5259* [116], *ISO/IEC 23053* [113], *ISO/IEC 23894* [115]), most are still of a general nature. The new *ISO/IEC Technical Report 5469* [117] instead provides a more detailed analysis of the risks and challenges of using AI in safety-related systems. Due to its classification as a *Technical Report*, it does not define any specific requirements, but rather provides some recommendations for the design and development of such systems. It is planned to be replaced by *ISO/IEC Technical Specification 22440-1* [108] and even later probably by a full standard, but this specification is still under development and not yet available. Therefore, the already published report will be used as a reference, and the most relevant parts for the system developed in this work will be summarized in the following.

6.2.1 Classification of AI systems

In general, the report describes AI systems as a realization of a three-stage process. First data acquisition, then knowledge induction (learning), and finally processing and output generation. From these stages, acceptance criteria can be derived, which can then be used to evaluate the system. It also proposes a classification scheme for AI systems in safety-related applications, which is based on a *Usage Level*, which considers how the AI is used, and a *Technology Class*, which considers the complexity of validating the necessary risk reduction. In the case of the system developed in this work, the *Usage Level* would be classified as *A1*, which means that the AI is involved in the automatic decision-making of the safety function. The *Technology Class* would be classified as *Class II*, because according to the report [117, Ann. B], it is questionable whether the system can be fully validated using only existing functional safety standards, but it is still possible to identify complementary requirements that can be used to validate the desired safety properties to achieve the necessary risk reduction.

6.2.2 Desirable Properties

For each stage of the three-stage development process, first desirable properties should be defined. Those properties are then refined into specific topics, and then into detailed methods and techniques to evaluate them. Finally, acceptance criteria for each of the methods and techniques are defined. This is somewhat similar to the *V-model* approach (Figure 2.9), but better tailored to the specific challenges of AI-based systems. Similar to the example in the report [117, B3], the properties for which requirements should be defined for the system developed in this work are:

- **Specifiability:** What are the requirements for the network? How much training data is sufficient?
- **Domain shift:** How does the system perform in an environment that is different from the one it was trained on?
- **Verifiability:** How is the neural network performance assessed? Which metrics are appropriate? How is it determined whether the verification process is accurate? When is verification complete?
- **Robustness:** How robust is the neural network to input perturbations (lighting conditions, environmental or operational changes, ageing, ...)? What happens in the event of hardware failures?
- **Interpretability:** Are the network outputs understandable? Do they correspond to the expected results?
- **Transparency:** What is the design rationale of the network architecture?

6.2.3 Validation

As a result of the learning-based nature of AI models, the report emphasizes the importance of the data acquisition process. The models can only learn the features that are present in the training data, but also might not learn everything perfectly.

Regarding the data acquisition, the report therefore recommends fulfilling the following requirements [117, Chap. 9.3.3]:

- **Scenario coverage:** The data should cover all relevant scenarios.
- **Risk coverage:** The test data should cover all reasonable variations of situations that can cause the identified risks.
- **Test amount:** The test dataset should be sufficiently large and diverse (especially free from unwanted bias) to completely cover the resulting system states.
- **Variation stability:** The test cases should be stable under the expected variations of the data.

To increase the amount of both training and test data, the report also suggests using simulations to create synthetic data. This approach has the benefit of allowing the creation of test data for scenarios that are difficult to capture in the real world, for example because they are rare or dangerous. However, it is still important to perform real-world tests as well, to ensure the models learn the relevant features.

6.2.4 Proposed Architectures

The report emphasizes the importance of a robust system architecture that can tolerate failures, and states that this is preferable to improving only the AI model itself. It suggests three different possible architectures for AI-based safety systems (Figure 6.1), which can coexist or be used separately.

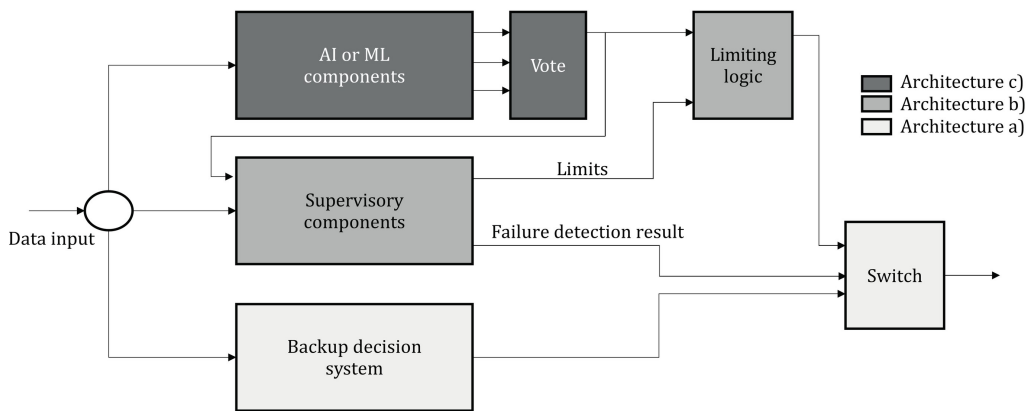


Figure 6.1. Architectural patterns for AI-based systems according to *ISO/IEC TR 5469*, image from [117].

Architecture *a*) uses an AI-free backup function, which can detect unsafe input conditions and then override the AI-based outputs. It has to be safe itself, but can be suboptimal, for example only triggering a stop of the robot. Architecture *b*) instead monitors the AI-based outputs and ensures they remain within predetermined limits to prevent unsafe behavior, using either AI-free or AI-based methods. Architecture *c*) uses redundancy in the AI components, which can then be cross-checked against each other or vote on the final output. To achieve redundancy, multiple approaches are described. One approach of them is to ensemble different neural networks, similar to what was already done in the *KoARob* system.

Summary. This chapter presents the next iteration of the safety concept and an outlook on necessary future steps toward safety certification.

7

Tutabo-2 - Next iteration

7.1	Hardware upgrades	116
7.1.1	Smart Cameras	116
7.1.2	Robot	117
7.2	Stop- and Interaction-Zones	120
7.2.1	Stop-Zones	120
7.2.2	Interaction-Zones	122
7.3	Using the proposals from <i>RPT</i>	124
7.4	Evaluation	126
7.4.1	AI standards	126
7.4.2	Collaboration Distances & Delays	127
7.5	Related Work	129
7.6	Outlook	132
7.6.1	Further steps toward certification	132
7.6.2	Force limiting	137
7.6.3	Economic Considerations	138
7.6.4	Ergonomics	140
7.7	Conclusion	141

The evaluation of the *Tutabo-1* system has shown that real-time performance using AI-based methods is possible. However, the minimum required separation distance is still quite high. Following the iterative development process, the next step is to further refine the system to reduce this distance in order to improve the usability of the system. Furthermore, additional concepts are introduced to increase reliability and add more advanced features, and future steps towards certification along the previously described safety standards are outlined.

7.1 Hardware upgrades

The software implementation of *Tutabo-1* is already strongly optimized for computational efficiency, so there is little potential for further improvement. However, on the hardware side, some notable improvements are possible.

7.1.1 Smart Cameras

Since the footprint of the smart camera devices is not an issue in this scenario, the currently used *Jetsons* were replaced by more powerful but larger computers (Figure 7.1). Again, each *smart* camera was connected to a separate computer, which is equipped with a more powerful *RTX-5070* GPU. This resulted in a significant reduction of the processing time, which is now only around 4 *ms* to predict the 2D-poses of three people, about $3\times$ faster than before. Besides being faster, the new computers also costed notably less than the previous *Jetson AGX-Orin* devices, with the only downsides of being notably larger and consuming more power. If, instead of the three persons with which the delay was tested in the previous evaluations, the system would only need to predict one person (and stop otherwise), this could further reduce the processing time to around 3 *ms*.

To speed up the camera times, their hardware synchronization features could be utilized. Currently, the trigger is software-based, because that was the only option implemented in the *ROS2* driver. However, it has the disadvantage that the camera has to completely process the image before the next one can be captured, whereas the hardware synchronization would allow starting the next image capture while the previous one is still being processed. This would enable using the full frame rate of the cameras and could reduce the required interleaving latency. According to *Basler* [26], the maximum frame rate of the cameras is 164.91Hz at the currently used resolution and image format, which would result in a frame every 6 *ms* instead of the current 20 *ms*. However, as stated before, the *ROS2* driver provided by *Basler* does not support such a high frame rate. Since creating a custom driver requires a lot of effort, it is out of the scope of this work, and the manufacturer's specification of one frame every 6 *ms* will be trusted in the following estimations.

Regarding the image latency (from the capture trigger to image publishing), the new computers slightly reduced the delay with the current sequential image acquisition from 16 *ms* to 15 *ms*. No significant impact on this is expected by parallelizing image capturing and processing for the higher frame rate. However, the way this delay is measured has a notable influence on the results. Previously, it was measured using the official *ROS2* tool *topic delay*, which is implemented in *Python*, and therefore is suspected to have a significant overhead. To get more accurate measurements and

also more metrics, a new tool was implemented in *C++* instead. This tool replacement resulted in a significant reduction of the measured delay, from 15 ms to 12.5 ms .



Figure 7.1. Robot setup of Tutabo-2. On top of the trusses, the new smart-camera computers are mounted.

7.1.2 Robot

A large fraction of the minimal separation distance calculated in Section 5.4 was caused by the robot's long reaction time. The measurements in Figure 5.13 have shown that it takes almost 100 ms until the robot starts to slow down. Therefore, there exists substantial potential to reduce the safety distance by improving this reaction time. Instead of setting the *override* speed over *RSI* to zero, using a dedicated safety input of the robot, under the assumption that this does not impact the rest of its safety system, might lead to a faster braking response. However, the latency may also be caused by the robot's internal controllers, because the stop time measured experimentally in Section 5.4 was close to the previous one estimated from the robot's datasheet, which is measured using the built-in safety inputs.

When comparing the braking ramps of different robots from different manufacturers (Figure 7.2), it can be observed that there are notable differences in the braking behavior and required stopping times. Therefore, it can be said that the choice of the robot has a significant influence on the required safety distances.

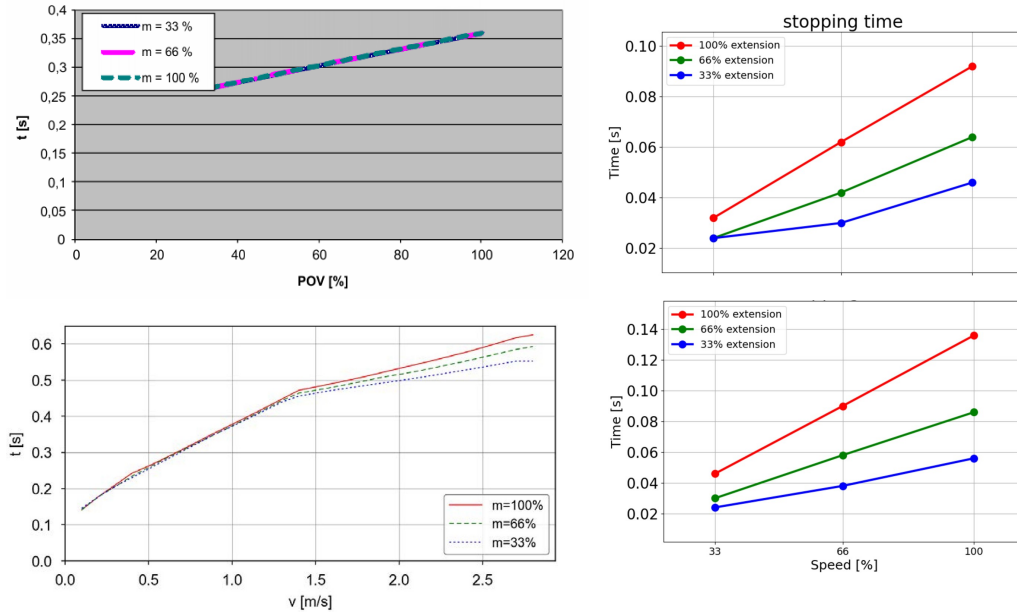


Figure 7.2. Braking ramps of the first axis of different robots. On the top-left, the *KUKA KR-16* robot used in this work, which has a rated load of 16 kg and a range of 1.6 m . The speed in the previous experiments was limited to 0.2 m/s ($POV=10\%$). On the bottom-left, a similar *ABB IRB-2400* robot, with a rated load of 16 kg and a range of 1.5 m . Both images show the measurements at a robot arm extension of 66% . On the right, two measurements of a *Universal Robots UR-16e* robot, with a rated load of 16 kg , a range of 0.9 m , and a maximum speed of 1 m/s , at measured loads of $m = 33\%$ (top) and $m = 100\%$ (bottom) of the rated load. Due to its lower range, the 66% extension of the *KR-16* and *IRB-2400* robots would match a 100% extension of the *UR-16e*. A comparison of the braking ramps shows notable differences in the braking behavior of the different robots. The *IRB-2400* stops earlier than the *KR-16* at low speeds, but takes longer to stop at higher speeds. The *UR-16e*, which is a force-limited collaborative robot, is much faster at braking than the other two. Images from the respective datasheets [30, 137, 233].

Since measuring the differences caused by the robot controller input and buying and evaluating different robots are outside the scope of this work, only an estimation of the potential improvement will be given here. Keeping the same assumptions as before (load around $5\text{ kg} = 33\%$, 66% arm extension, 0.2 m/s robot speed, and 2.0 m/s human speed), the following estimations can be made based on the braking ramps in Figure 7.2:

- *IRB-2400* from datasheet:

$$\begin{aligned}
 Time_{robot\ stop} &= 0.16s \\
 S_{h_2} &= 0.16s * 2.0m/s &= 0.320m \\
 S_s &= 1/2 * 0.16s * 0.2m/s &= 0.016m \\
 Z_r &= 0.002m \\
 S_{robot} &= 0.32 + 0.016 + 0.002 &= 0.338m
 \end{aligned} \tag{7.1}$$

- *KR-16* assuming instantaneous braking start:

$$\begin{aligned}
 Time_{robot\ stop} &= 0.126s \\
 S_{h_2} &= 0.126s * 2.0m/s &= 0.252m \\
 S_s &= 1/2 * 0.126s * 0.2m/s &= 0.013m \\
 Z_r &= 0.002m \\
 S_{robot} &= 0.252 + 0.013 + 0.002 &= 0.267m
 \end{aligned} \tag{7.2}$$

- *UR-16e* from datasheet:

$$\begin{aligned}
 Time_{robot\ stop} &= 0.025s \\
 S_{h_2} &= 0.025s * 2.0m/s &= 0.050m \\
 S_s &= 1/2 * 0.025s * 0.2m/s &= 0.003m \\
 Z_r &= 0.002m \\
 S_{robot} &= 0.050 + 0.003 + 0.002 &= 0.055m
 \end{aligned} \tag{7.3}$$

In comparison to the calculated distance $S_{robot} = 0.481m$ for the *KR-16* in Section 5.4, the above estimations show significantly better results, especially for the *UR-16e* robot, which indicates there is substantial potential to reduce the robot's contribution to the safety distance. However, as noted before, switching to a different robot or optimizing the robots themselves is outside the scope of this work. Therefore, these estimations should be seen only as motivation for future work in this direction.

A small improvement of up to the value of S_s can further be achieved by making the robot's safety hull path-dependent, somewhat similar to *KUKA's* internal *braking before restricted areas* feature [136]. Currently, the robot's safety hull is a fixed distance around the robot, which is the same in all directions. However, the braking movement distance is only relevant in the path direction, which can be pre-calculated because the robot's movement is usually known beforehand, so the safety-hull thickness can be reduced in the other directions. Another option for further improvements could be to create a braking model of the robot, optionally based on neural networks, which then can be used to predict the robot's braking behavior more accurately such as in [219]. This would help to avoid using the worst-case estimation of the braking distance and duration everywhere.

7.2 Stop- and Interaction-Zones

One advantage of the proposed system architecture is that the collision avoidance is completely software-based. This makes it easy to implement additional features which can further optimize the behavior of the robot.

7.2.1 Stop-Zones

The evaluation of the pose estimation in Section 5.4.1 has shown that the system generally detects the joints with a very high accuracy, but there are some outliers which increase the required safety hull around the person. Upon closer inspection, it can be observed that these outliers often occur in frames with many occlusions of a subset of the joints. Many of these cases occur when workers are going to fetch a package from the pallet themselves, thereby walking into a corner of the room with low camera coverage, a case in which the robot is expected to stand still anyway (Figure 7.3).

Creating *stop-zones* around such areas, which trigger a stop if any joint is detected inside them, would allow ignoring joint outliers inside those zones to a certain extent, because the robot will be stopped by the better detected joints already. The size of those zones has to cover the area, and additionally must be increased by the uncertainty of the joints that should trigger the stop-zone. The main advantage of introducing such stop-zones is that, because poor detections inside them can be ignored, the maximum joint detection errors can be reduced, which subsequently allows reducing the safety hull around the person in the rest of the workspace.

Adding a stop-zone around the pallet and the area directly in front of it notably reduces the maximum joint detection errors of some joints, especially for the right middle-finger-tip (Table 7.1). Since it is preferable that the robot stands still if a person is fetching a package anyway, this does not lead to a loss of productivity, but allows the safety hull around the affected joints to be significantly reduced in the remaining workspace, which can lead to fewer unwanted stops and higher productivity there.

Head	Shoulders		Elbows		Wrists		Hips		Knees		Ankles		Middle-Finger-Tip	
0.006 (-0.001)	0.020 (0)	0.013 (0)	0.072 (-0.056)	0.031 (0)	0.236 (0)	0.203 (0)	0.068 (0)	0.059 (0)	0.094 (-0.012)	0.075 (0)	0.184 (-0.026)	0.175 (0)	0.236 (0)	0.112 (-0.220)

Table 7.1. Improvement in joint detection accuracy by adding a stop-zone around the pallet. Above, the new maximum joint detection errors without the outliers in the stop-zone. Below, the improvement compared to the maximum joint errors from the previous evaluation (Table 5.14). Values in *meters*.



Figure 7.3. Examples in which a stop-zone would be helpful. In the images, the persons try to fetch a package themselves. They were detected by the safety system, but due to the significant occlusions and low camera coverage of this location, some joints show notable position errors. In the top image, the two red dots should be at the tips of the middle-fingers, but are substantially displaced. In the bottom image, the left hand is completely misplaced, leading to a high error between the true and detected joint position. In both cases the robot is expected to stand still. In the lower image for safety reasons, and in the upper image because the worker could feel unsafe if the robot moves while it is close and not visible. A cube-sized stop-zone covering both the pallet and the area directly in front of it, which triggers a stop if any joint is detected inside it, would allow those outliers to be ignored, because the robot will be stopped by the better detected joints already.

7.2.2 Interaction-Zones

Following the concept of *stop-zones*, the inverse of it would be *interaction-zones*. The idea is to define zones around the areas in which the robot is expected to interact with the person. In contrast to the stop-zones which trigger a stop if any joint is detected inside them, the interaction-zones would allow the robot to continue moving even if it violates the separation distance, but only if certain joints are inside them.

An example in this scenario, in which such an interaction-zone would be helpful, is when the person is fetching a package the robot has just placed on the table (Figure 7.4). In this case, a contact with the hands and fingers is expected, and if the robot is already retracting after placing the package, there is no risk in this action anymore. So an interaction-zone here can improve the workflow and reduce unnecessary stops. However, the safety system would still detect any other joints being too close to the robot, which could result in unexpected and more risky contact situations, and then trigger a normal robot stop.



Figure 7.4. Example in which an interaction-zone would be helpful. The person is waiting for the robot to place a package on the table, to be able to continue his work. Even if the robot has already dropped the package, the person still has to wait until the robot has moved further away, because otherwise the safety system would detect the person as being too close to the robot and therefore trigger a stop. By defining an interaction-zone around the area, which allows interactions with the hands, and which is activated after the robot has dropped the package, the person can directly fetch the package without triggering an unwanted robot stop.

Another example which could notably benefit from interaction-zones would be hand-over scenarios between a robot and a person. When the robot hands over an object directly to the human worker, it could be allowed to move closer and place it directly into the hands of the worker. The rest of his body, especially the head, will still be

protected by the safety system. After the hand-over, the zone would be deactivated again. In normal *Speed and Separation Monitoring* collaborations, this would not be possible, because the safety system would have to trigger a stop before the robot could reach the hands of the worker. The person then would have to finish the rest of the distance and additionally confirm that the robot can now open its gripper, which would notably reduce the efficiency of the hand-over.

Adding such *interaction zones* would of course require a detailed risk analysis, to make sure the persons are not exposed to potentially dangerous situations. But because the proposed AI-based system is able to distinguish between different body parts, such as the hands and the head, and the zones can be activated and deactivated, those contact situations can be restricted to certain body parts at specific locations and times. This restriction capability notably reduces the possible risks that have to be handled.

Conceptually, those zones could also be easily extended to multi-robot collaborations as well, so that a contact by a human is only allowed in a certain area with $R1$ but not with $R2$, or even to robot-robot contacts, for example to allow a robot to fetch an object from a mobile platform at a certain location with its gripper, but prevent collisions with it in the rest of the workspace.

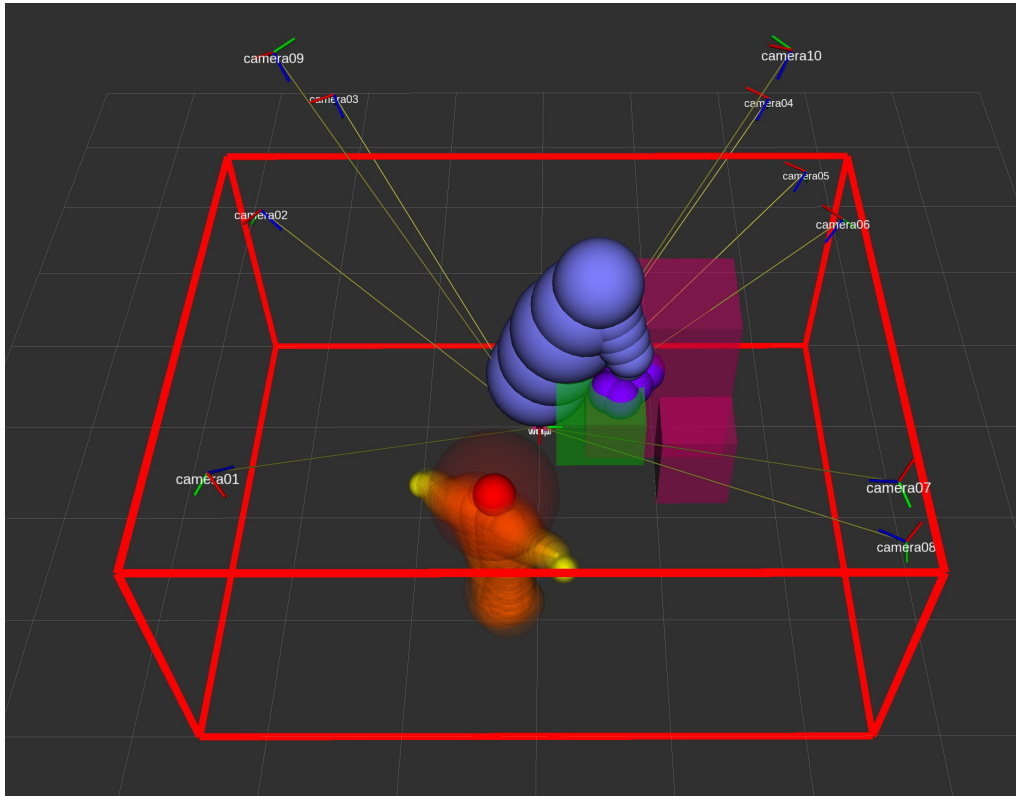


Figure 7.5. Internal visualization of the stop-zones (pink, around the pallet area) and interaction-zones (green, on the table) in the current scenario.

7.3 Using the proposals from *RPT*

In *RapidPoseTriangulation*, the predicted 3D-pose is calculated by merging a set of proposals (Figure 5.8). Those proposals are generated by triangulating the 2D-poses of different camera pairs. The merging step of *RPT* tries to find the best proposals and remove outliers before merging them to a final pose. As shown in its evaluation, this merging step notably improves the pose estimation accuracy. However, it is not perfect, and in the case of many outliers or generally bad proposals, it can result in a poor merge.

Since in this scenario there are many cameras, and they are placed in a way that at least a few of them overlap in every area of the workspace, this means at least a few of them should always have a good view on the person. This in turn means that they also should be able to estimate good 2D-poses, which then should lead to good proposals for those camera pairs with good views. The idea of this section is to evaluate using those proposals directly, instead of merging them, similar to the approach in *KoARob*, in which the three different pose estimation methods were fused together (Section 3.5.1). The safety system would then protect all proposals instead of only the merged pose. An example of the proposals overlaid on the camera images can be seen below (Figure 7.6).

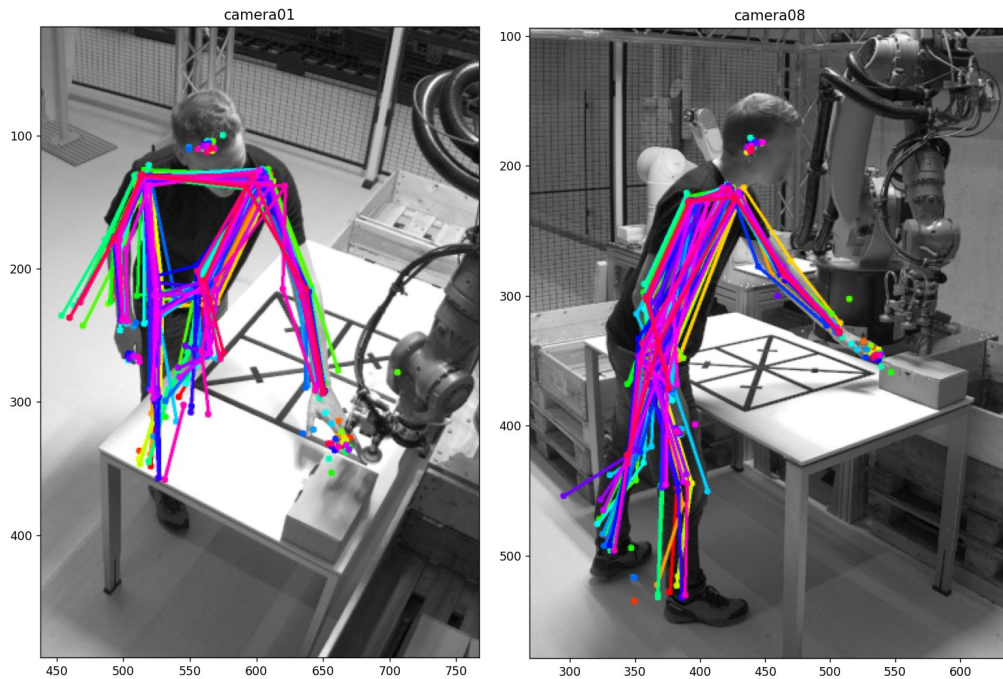


Figure 7.6. Visualizing proposals generated by *RapidPoseTriangulation*.

As shown in the image above, the proposals generally match the true joint positions quite well, but there are a few outliers next to the person. Some other outliers have already been removed automatically, by filtering them using the maximum error of the merged detections (Table 7.1) as a threshold. The error effectively describes a sphere around the detected joint position, in which the true joint position is definitely contained,

so any joint proposal outside this sphere can be safely removed. From the remaining proposals, around 6.8% are further away than 250 *mm* from the true joint position, and around 3.4% of the frames contain at least one such far-away proposal.

The evaluation in Table 7.2 shows that using the proposals directly, without merging and discarding them afterward, significantly reduces the maximum joint detection errors of most joints to only a few centimeters. These results indicate that the predictions of the AI system are generally very good, and that the joints are reliably detected with high accuracy, but the proposal selection and merging process has further optimization potential.

Head	Shoulders		Elbows		Wrists		Hips		Knees		Ankles		Middle-Finger-Tip	
0.006	0.014	0.009	0.011	0.011	0.019	0.046	0.036	0.041	0.056	0.049	0.100	0.081	0.059	0.056

Table 7.2. Maximum joint detection errors (in *meters*) when the proposals are integrated into the safety system.

One possible improvement option would be to better consider the visibility of the joints, and also the visibility of the person in the camera image, to improve the selection of the per-view triangulated proposals. However, since the currently used 2D-pose estimator does not provide such information directly and either has to be substantially modified or be replaced by a different one, this has to be left for future work. Another option, which also will be left for future work, would be to add a second neural network, which takes the proposals and their history as input, and is trained to learn a better selection and estimation of the current 3D-pose from them.

Regarding the integration of the proposals directly into the safety system, it would also be advisable to further evaluate the trade-off between the lower required separation distance due to the better accuracy and the increased protection area due to the outliers, to find the optimal productivity balance. In general, the protected volume should nonetheless shrink, because instead of a larger sphere around the merged pose, there will be multiple smaller spheres around the proposals, even though they are sometimes slightly more spread out. Improving the 3D-pose estimation, however, for example by explicitly considering the visibility information, would also allow the number of outliers to be reduced subsequently by reducing the filter threshold, which would then improve this trade-off as well.

7.4 Evaluation

As in the previous chapters, the updated system was evaluated again. Regarding the pose estimation, the same data as in the previous evaluation of *Tutabo-1* was used to make the results directly comparable.

7.4.1 AI standards

Following the newly proposed procedure of *ISO/IEC-TR-5469* was not the focus of this chapter, as it generally prioritizes an extensive definition of required properties and acceptance criteria for them, with the goal of ensuring their certifiability. This work, however, is more experiment-driven to explore what is currently possible and extend those limits, but some ideas from the previous chapters have notable similarities with the report's recommendations.

Regarding the *desirable properties*, the *interpretability* of the network outputs and the *transparency* of the methods' architectural choices are described in detail in the previous chapters, and the 3D-poses are easily human-understandable. The experiments across multiple datasets demonstrated a certain degree of *robustness* of the used AI-based modules against different inputs and also the capability to handle *domain shifts*. A few further notes on the *robustness* and also the *verifiability*, especially on how much data is necessary to complete the verification, will follow later in Section 7.6.

Regarding the *validation data*, the *scenario coverage* of the recorded data is already quite good for the main scenario, as it includes multiple persons, each with their own movement patterns, executing all steps of the main workflow. The *risk coverage* is not yet complete, as not all reasonable variations of potentially risky situations are included in the data. As already mentioned in the previous chapters, the *test amount* needs to be increased further, and the *variation stability* was only shown to some extent by testing the system on multiple datasets, but not directly in the current scenario. Section 7.6 will take a closer look at some of those points as well.

Regarding the *architectures* proposed in the report (Figure 6.1), some overlaps can be observed. The three redundant AI-based person estimation modules of *KoARob* can be seen as implementing a *voting* architecture, in which each module protects a certain detection area and any vote to stop the robot will be executed. The algorithm of *RapidPoseTriangulation* can also be seen as a voting architecture, in which the proposals are the votes and a more complex, but AI-free merging process implements the voting decision. Alternatively, if the proposals would be directly integrated, a similar voting architecture as in *KoARob* could be implemented. The additional buffer, which was introduced to prevent the robot from constantly jerking in borderline cases, can take over the role of the *switch* module. It allows adding a *backup decision system*, which can check some properties of the inputs, for example if the cameras are sending non-black images, or all modules are sending outputs at the expected rate, and trigger a stop if any of those checks fail. The AI-modules also do not control the robot directly, but only send the detected joint positions to an AI-free collision checking module, which therefore can be seen as implementing some kind of *limiting logic* for the network decisions.

7.4.2 Collaboration Distances & Delays

From the previous sections, the following values for a *speed and separation monitoring* approach can be extracted:

$$\begin{aligned}
 Time_{system\ delay} &= 0.013s_{(cameras)} \\
 &+ 0.004s_{(RapidPoseTriangulation)} \\
 &+ 0.001s_{(spherization\ \&\ collisions)} \\
 &+ 0.006s_{(interleaving)} \\
 &= 0.024s
 \end{aligned} \tag{7.4}$$

In comparison to the previous evaluation, the hardware and measurement optimizations have significantly reduced the system delay, from around 50 *ms* to now 24 *ms*. It is now in the same range as the delay of a light curtain, which is between 11–36 *ms* [213], even though the complexity of the system is much higher and it outputs more information.

Every human joint type will have a different safety distance again, depending on its thickness (intrusion distance), because only the center of the joint is estimated, and on the joint's prediction error. Currently, the joint thickness is estimated using average human body proportions, but in the future it would be recommended to personalize this (if identification reliability can be ensured), to better take into account differences in people's body types. The labeling error is still estimated to 5 *cm* and added to the uncertainty of the joint's position. Using the above information, the following distances can be calculated:

$$\begin{aligned}
 S_{h_1} &= 0.024s * 2.0m/s = 0.048m \\
 S_r &= 0.024s * 0.2m/s = 0.005m \\
 C &= 0.5 * JointThickness \\
 Z_d &= JointError + 0.05m
 \end{aligned} \tag{7.5}$$

Since the previous sections proposed different options, the safety distances are calculated for two different cases. In both cases the safety distance for hands and fingers (using the worst error of those) is calculated. The first only uses the stop-zones (S_{system_1}), and the second additionally integrates the proposals (S_{system_2}). In comparison to the results in *Tutabo-1*, the required safety distance of the system has been notably improved again (the previous value was 0.426*m*).

$$\begin{aligned}
 S_{system} &= S_{h_1} + S_r + C + Z_d \\
 S_{system_1} &= 0.048 + 0.005 + 0.03 + (0.236 + 0.05) = 0.369m \\
 S_{system_2} &= 0.048 + 0.005 + 0.03 + (0.059 + 0.05) = 0.192m
 \end{aligned} \tag{7.6}$$

To calculate the total required safety distance, the robot's contribution has to be added to the system's contribution. For this, one of the values from the previous Section 7.1.2 can be used. When using the best one to estimate a lower bound of the minimum

required safety distance, the *UR-16e*, which is best optimized for fast braking, with $S_{robot} = 0.055m$ leads to a minimal total safety distance of $S_p = 0.247m$. For comparison, this would correspond (after subtracting C again) to a minimum clearance approximately equal to the shorter side of an *A4* paper sheet ($0.210m \times 0.297m$ according to *ENISO 216* [110]) between robot and the surface of the human hand. While only using *speed and separation monitoring* never allows a contact between robot and human, the proposed *interaction-zones* could solve this issue for certain intended interactions.

7.5 Related Work

Regarding the current domain of AI-based safety systems for human-robot collaboration in factory environments that utilize the position of the human, the relevant related work has already been covered in the previous chapters, especially in Section 3.6 and Section 5.5. As shown there, *Tutabo* is currently the leading system in this domain. Therefore, this section will take a closer look at related work in other domains, which could be relevant for the future development of the system.

One domain that already has a very high research activity in the area of AI-based safety systems is autonomous driving. The project *KI Absicherung* [18, 99] explored the use of AI-based safety systems for autonomous vehicles, covering many different aspects and developing both models and architectural concepts for such a system. Multiple car manufacturers also put notable effort into the development of AI-based safety systems for their autonomous vehicles, and *Waymo* already offers commercial driverless taxis in some areas. It should be possible to transfer some concepts and methods, for example system architectures, training and testing procedures, or tools for synthetic data generation, to the domain of human-robot collaboration. The standardization process for autonomous vehicles has progressed further as well, with *ISO/PAS 8800* [118] published as an official standard. From a conceptual point of view, it has some overlaps with the proposals of *ISO/IEC-TR-5469* which were evaluated here, showing the similarities between the two domains again.

Another closely related domain is autonomy in railway trains, which also has strict safety requirements. An important project there is *safe.trAI*n [24], which focuses on developing a driverless regional train. They have also invested considerable effort into the development of safety-certifiable architectures, model training procedures, and synthetic data generation.

While there are notable similarities between the domains of autonomous cars and trains with human-robot collaboration, there are also important differences. The most significant difference is likely the environment. Especially cars have to operate in a very complex and dynamic environment, with many different actors, such as pedestrians, cyclists, or other cars, which can all behave in very unpredictable ways. Additionally, different weather and day conditions, such as rain, snow, fog, bright sunlight, or darkness, can notably affect the measurement quality of the sensors. Furthermore, the environment is openly accessible to anyone, which means that there can be malicious actors trying to attack the AI-based modules, for example by sticking adversarial patterns on signs to trick the system into misclassifying them [124]. In contrast, the environment of human-robot collaboration, especially in industrial settings, is significantly more controllable and predictable. It also has the benefit that sensors can be externally placed, which allows to have a better coverage of the workspace and the human workers, and they also do not need to have a very long visual range. While most car manufacturers and the *safe.trAI*n project as well, rely on a combination of different sensors, such as cameras, radar, and lidar, a camera-only system might be sufficient for the more controllable factory environment. More sensors could of course further improve reliability, but at the cost of higher complexity and price.

A second important difference is in the risk-reduction requirements. For robots, the requirements are very strict. The evaluation has to reduce the risk for all common, but also for reasonably foreseeable use and misuse cases, and in the case of *speed and separation monitoring*, the system must be able to stop the robot before a contact with a human can occur. This strictness can also be explained by the fact that employers have a legal responsibility to provide a safe working environment for their employees and therefore have to make sure that the risk of injuries is reduced as much as possible. In contrast, the safety requirements for autonomous cars are often more about reducing the risk to a socially accepted remaining risk level [99]. A good example to see the difference would be the case of a driver-less car passing by a row of parked cars. A foreseeable misbehavior case would be that a distracted child suddenly runs out between the parked cars, even though it was told not to do so multiple times. Under robot safety requirements, a robot vehicle would have to be able to stop before it can hit the child. Therefore it would have to pass by all parked cars very slowly. Under autonomous car regulations, it instead would likely be tolerated that the car hits the child, to allow it to drive around with reasonable speed, as long as it would respond to the situation better than a human driver would have done [246].

A third difference concerns economic considerations. Especially in the automotive industry, research budgets for the development of autonomous vehicles are very large and can employ many researchers, which allows the development process to proceed much faster. In contrast, the research activity in the domain of human-robot collaboration is comparatively low, as is noticeable in the related work sections. Therefore, to speed up the research in this domain, it is important to broaden the user base of its newly developed systems.

Another very closely related domain to industrial robotics, which has gained substantial traction in recent years, is humanoid robotics. If they are not intended to just replace human workers, but to work together with them, they also have to be equipped with safety systems. Similar to collaborative robots, they often use *power and force limiting* to ensure safety. But for some cases this will not be enough, for example when doing hand-overs of hot or otherwise dangerous objects, where it has to be ensured that the human will grab it only at the right places. Especially if they are designed to perform household work at a human-like level, so basically complicated tasks in a complex and dynamic environment, they presumably need some AI-based systems with safety-standard reliability. An example scenario could be a humanoid vacuum-cleaning around a table with chairs. Usually a human would move the chairs aside to clean the floor below its legs, so a humanoid is expected to do the same, but what if there is something on the chair? Is it a backpack, which can be moved aside together with the chair, or is it a small child sitting on it, which could fall down if the chair is moved? Another example could be a humanoid cutting some food on a kitchen table. If humans cook together, they often steal some pieces as a snack to try from the cutting person, and the cutting person will expect the other to not directly grab into the knife and maybe threaten them a bit with the knife, and the other will expect the cutting person to stop cutting if he miscalculated his move. So this will be a foreseeable misbehavior case, but how should the humanoid react to it, does it always have to stop cutting if

a human is around, and if not, how can it distinguish between cutting a carrot and a human finger? Other examples in which a humanoid needs to be able to reliably detect humans and distinguish between body parts could be handing over a hot cup of tea, which has to be well-placed into the hands of the human to avoid spilling, or the ironing of clothes, in which the iron has to be pressed down on the shirt, but better not onto a human hand next to it. In those four cases and certainly many others as well, only using *power and force limiting* will not be enough. The humanoid will need some understanding of the environment and situation, which is usually provided by AI-based systems, which, if they are intended for safety-relevant interactions as well, also need to prove their reliability. The robot manufacturers will certainly invest considerable effort into addressing this issue, and there also exists independent research on how such dangerous scenarios can be handled [134, 225]. Thus, the research in this work, on reliably detecting the position of human workers and their body parts and using it for safety-relevant decisions, can also be relevant for the development of humanoid robots.

7.6 Outlook

Developing a safety-certifiable AI-based system is a very complex task, and many questions and challenges have to be solved. When taking a look at the closely related domains of autonomous cars and trains, it can be observed that even though many people have been working on this topic for many years, they are still not fully solved. While the topic of AI-based safety systems for robots in factory applications is, as described previously, simpler than those domains because of the better controllable environment, it is still too complex for a single person to solve it alone. So this work can advance the progress in this field, but also will leave open challenges for future research, on which this section will give some suggestions.

7.6.1 Further steps toward certification

Currently, the safety standards assume that the system and its software components are fully specified, deterministic, and fully verifiable. However, AI-based system outputs are usually data-driven, probabilistic, susceptible to input changes, and calculated by a black-box model, which is usually not fully understandable. This makes it very difficult to evaluate their true risk-reducing capabilities and very difficult to certify them with the current standards.

The reliability of a safety system is usually measured by its *average probability of dangerous failure per hour* (PFH_D), as was explained in more detail in Section 2.3, and can be calculated with a simplified approach based on multiple reliability characteristics (*Category*, *MTTF*, *DC*, ...). This simplified approach, however, relies on assumptions that cannot directly be transferred to AI-based systems. Generally, most commercial safety systems are based on well-known fail-safe physical principles, for example measuring the existence of a light beam (in light curtains), or the duration between sending and receiving it (in laser scanners), and (dangerous) failures are usually caused by random hardware failures. In contrast, failures of AI-based systems are usually caused by changes in the input data, making them more systematic than random failures. However, even though for those reasons the simplified approach cannot be applied, and the PFH_D has to be verified directly instead, its reliability characteristics are still helpful to prove the reliability of the proposed AI-based system, as described in the following.

Diagnostic Coverage

Tests to detect dangerous failures of the system are best split into two categories: tests for hardware failures and tests for AI-related data conditions, which increase the probability of wrong model outputs. The first follow the well-established *ISO 13849-1*, while the second follow the idea of the *backup decision system* from *ISO/IEC-TR-5469*.

- **Responsiveness Failures:** To check that all modules are running, and that there also is no processing overload, checking the timestamps of the messages should be an efficient option. Each pipeline cycle (here 6 ms) the final safety modules should receive the messages required for distance checking. If these messages

are not received at the expected rate, it means there is a problem or data loss anywhere earlier in the system, and a stop should be triggered.

- **Message Checksums:** Currently the messages are sent via *ROS2*, which has some built-in checks for message integrity, but from a safety perspective, the entire *ROS* stack has to be verified. It might therefore be more efficient to add a custom checksum-based wrapper for the message data that can be more easily verified, to check that the message content is correctly sent between two *ROS* modules. To check message delivery, the timestamps can then be used, as described in the previous point.
- **Calibration Tests:** A correct calibration of the cameras is crucial for the system to work properly, so it is important to regularly check that the calibration is still correct, and that the cameras did not move because someone accidentally bumped into one or because of vibrations of the robot. This can be done with automatic overnight calibration tests using the calibration board or with *IMU*-sensors mounted on the cameras to detect any movement or drift. Other options include checking the camera images against static workspace structures or fiducial markers, which should always be detected at the same positions, or against a 3D model of the robot, which should perfectly overlap if the calibration is correct.
- **Camera Failures:** To test that the cameras are working properly, it can be checked that slight pixel noise exists between consecutive frames. A comparison against previously captured background images or the robot model, which to some extent (to account for partial occlusions by the human) should match the current images, can verify that the image sensor is working properly. Additionally, the brightness and contrast of the images can be checked regularly to prevent sending *out-of-distribution* inputs to the AI-based models, following the idea of the *backup decision system*.
- **Environment Changes:** The pre-taken background images can also be used to check for any larger changes in the environment, which could create unexpected occlusions or otherwise influence the input data distribution.
- **GPU Tests:** To check that the GPUs are working properly and they have no hardware failures, they can be regularly tested with some known test images showing persons, and the outputs can be compared to the expected and pre-labeled detections.

Multiple Channels

To further increase the reliability of the system, it is recommended to add multiple channels, same as in most *Category-3* systems, which can be implemented in different ways: with other systems, or within the same system.

The robot movement itself can be monitored with already existing and certified solutions, for example *KUKA-SafeOperation*, to ensure it does not violate speed limits and stays within the defined workspace. The outer area of the workspace, where camera coverage does not overlap as well as in the inner area, could be monitored with simpler measures,

for example safe floor mats, to gain earlier awareness of a person approaching the workspace. Optionally, the complete workspace can additionally be monitored with a second safety system, similar to most autonomous cars that combine multiple sensor types, which would then provide the second channel. In this case, however, the worst-case parameters of both systems have to be considered.

Regarding redundancy with multiple AI-based systems, there is the problem that this redundancy is not as strong as in traditional safety systems, because failures are less independent. While it is less likely that two hardware errors happen at the same time, it is more probable that two different AI-based systems fail at the same time if they receive similar inputs, just because of how they were trained. However, redundant AI-based models still offer meaningful benefits and can notably increase the reliability of the system, as the experiments in *KoARob* have shown.

With the current system, multiple approaches for redundancy could be implemented. As in *KoARob*, another pose estimation model with a different architecture could be developed, which optionally also receives its own half of the cameras. Alternatively, some redundancy could be implemented along the time axis, by combining the outputs of the current and previous frames. The space of an erroneous detection could then be protected by a correct detection in a previous frame. While this would increase the system delay, because it now requires multiple measurements as input, the impact would not be too high, as it only adds 6 ms per extra frame. Another option would be to integrate the proposals, similar as described in Section 7.3, because two view-pairs from four different cameras are largely independent. The camera images all have different perspectives, reducing the input-data-based correlation, and the triangulation of the two view-pairs is independent of each other as well.

Similar as in *KoARob*, the proposals can then either be checked for congruence or directly merged to protect the space around all of them. Another option would be to add some additional requirements using the proposals, for example that each joint of a person has to be detected by at least five cameras. If fewer cameras contributed, either a stop could be triggered, or the safety-sphere of the parent joint could be extended by the distance between the joint and its parent joint, to ensure that this joint is still protected, but a forced stop is not necessary. A similar approach could also be used for the *interaction zones*, to ensure collisions inside it are only prevented for joints with high visibility by multiple cameras.

Furthermore, the pre-taken background images could be used and subtracted from the current images, to check for any larger unexpected changes in the environment. In general, the movements of the robot and other controlled elements, like mobile platforms, are known beforehand, so any unexpected remaining parts after the subtraction should be humans. Across multiple cameras, such area outlines could be triangulated again, to a larger cylinder-like shape, and then be protected by default. Only if a 3D person with high confidence is detected in that shape, this area is then unprotected, and the person itself is protected instead. This way, even if the 2D-pose estimation models miss a person, the person would still be protected by the system. Conceptually, this has some overlaps with the idea of a *supervisory component* in ISO/IEC-TR-5469.

Mean Time to Dangerous Failure

Probably the greatest challenge towards certification is to prove that the system, and especially its AI-based components, have a high enough *Mean Time to Dangerous Failure*.

From a hardware perspective, most components integrated into the AI-system are available as consumer products, and their manufacturers usually offer a warranty of around three years. Private computer hardware is also often replaced because it has become outdated but not because it is broken. Therefore, it can be assumed that the life expectancy of those hardware components should be sufficient for a $MTTF_D$ of *low* (3a) and probably even *medium* (10a). Independent of their life expectancy, complex components (microprocessors, GPUs, ...) will not be considered “well-trying” and therefore do not provide points in the CCF evaluation [114, Chap. 6.1.11]. Since not all software and drivers for those components are open-source and verified, it is also necessary to use diverse hardware, for example additionally using AMD graphics cards instead of only NVIDIA, to reduce the risk of *Common Cause Failures* [114, Chap. 7.3.2].

From a software perspective, the most complicated part will be to prove the failure rate of the AI-models. For $PL=a$ the $MTTF_D$ has to be at least 3a, which corresponds to a PFH_D below $10^{-4}/h$. As described earlier, using the simplified approach for higher *Performance Levels* does not work for AI-based systems, so the PFH_D has to be evaluated directly. If $PL=d$ is required, the system has to fulfill either *Cat 3* with PFH_D below $10^{-6}/h$ or *Cat 2* with PFH_D below $10^{-7}/h$ according to the standards [120, Chap. 5.5.3].

So this means that the system has to be reliable for a very long time, and this first has to be verified, which requires a very large amount of testing data, as will be discussed in a moment. However, which makes it a bit simpler, only *dangerous* failures of the AI-models have to be counted and others can be ignored. An optional step in this direction, which follows a similar idea as the proposed *stop zones*, would be to record the robot position in the testing data as well, and then only evaluate the position errors of the joints whose safety-spheres are closest to the robot. Failure detections of joints which are further away can be ignored then, because they are not relevant to measure and adhere to the minimum safety distance.

Simulation

To prove the reliability for such a long time, a very large amount of testing data is required. The number of images the system will process in the minimum required 10^4 hours can be estimated with the following calculation:

$$Images = 10^4 h * 3600 \frac{s}{h} * 164.91 Hz = 5\,936\,760\,000 \quad (7.7)$$

There are two major problems if this amount of data has to be collected for the validation process. First, saving that many images requires substantial storage capacity. At around 2MB per uncompressed image for 10 cameras, it would require around 120PB of storage, which at a price of around 360€/12TB at the time of writing would result in 3.6 million EUR and therefore be rather expensive. Second, which is even more

problematic, is the time it would take to label that many images, which was around 2min per frame for the current labeling process. This would result in the following time requirement for labeling, assuming average German working times with holidays and sick days:

$$Time = \frac{5\,936\,760\,000 \text{ frames} * 2 \frac{\text{min}}{\text{frame}}}{60 \frac{\text{min}}{\text{h}} * 8 \frac{\text{h}}{\text{d}} * (250 - 30 - 20) \frac{\text{d}}{\text{a}}} = 123\,683 \text{ years} \quad (7.8)$$

As can easily be seen, this is slightly above the author's life expectancy. The option of distributing it across multiple researchers is also not really feasible, as with the usual temporary contract for research assistants at German universities (30.5 months on average [16]) and an optimistic estimation of 4h to prepare and sign such a contract, it would alone take 122 years just for contracting, also exceeding the life expectancy of the single administrative staff person who would be responsible for this. So a different approach has to be found for this.

One option could be to equip the workers with a marker-based tracking suit, which would allow automatically obtaining accurate ground-truth joint positions, which then could be used for the system evaluation. The disadvantages of this approach, however, include that it would require a second safety system (as backup of the not-yet-certified AI-based system), the AI predictions could be influenced by the markers and suit, and the workers would have to wear a typically uncomfortable tracking suit for at least three years.

Thus, a better option would be to use a simulated environment, as also suggested in *ISO/IEC TR 5469*, from which an effectively unlimited amount of data with accurate ground-truth labels can be generated. It would also allow easy generation of multiple test cases for dangerous situations, which would be hard to record in real life. Such situations can then, to some extent, be created automatically, for example by training reinforcement learning agents whose purpose is to trick the safety system and hurt themselves [107]. Generally, the simulation should be as realistic as possible, to minimize the sim-to-real gap, and therefore the risk of overestimating the reliability of the system. Regarding the simulation tools, there are already some options from neighboring fields available [90, 94, 175, 205] that could be adapted to the current scenario setting. If the simulation results are also used for training, it is important to be aware that the sim-to-sim evaluation results can improve substantially as a result, reducing the informativeness of the results obtained with simulated testing data.

To verify the model performance, real-world testing data is still required, but it can be significantly reduced, as the simulation can cover a substantial portion of the test cases. To reduce the required labeling effort, it should also be possible to reduce the frame-rate of the real-world data. The main idea is to evaluate the performance of the system with varying scenario events, but with only 6 ms between two frames, there will not be substantial changes between them. Thus, it can be expected that the AI-model performance will not notably change between them either, and therefore a lower frame-rate should be sufficient for significant parts of the evaluation. In the same amount of

labeling time, more scenarios can be covered with this approach, which would give more informative evaluation results.

An option to further reduce the labeling amount for real-world data could be to use a humanoid robot. It could be dressed up as a human and simulate, as well as the humanoid is capable of, it would work and move around like a human. The humanoid's joint positions can then be extracted from its internal sensors and used as ground-truth labels, which should also be more accurate than the human-introduced labeling errors. Compared to a full simulation, this should be closer to later real-world inputs, and therefore have a lower sim-to-real gap. With different clothing, the humanoid could simulate different persons, and it would also be possible to test more dangerous scenarios with it, since in the worst case only a machine is damaged.

Verification

The verification process has to follow the requirements and procedures defined in the relevant standards. The testing data for the AI modules must cover all expected scenarios and risks, and for the other modules, hardware and software error cases must be generated to evaluate the system's failure-detection capabilities.

Regarding the evaluation of the complete system in a real-world environment, multiple options are available, which can also be combined. One option is to test the system in an industrial application, in which there are either no serious risks of injuries to persons or in which protection is provided by other measures and sensors. For example, a force-limited robot could be used, and the AI-based system would receive the task of stopping the robot before it contacts the human, to improve the worker's comfort. If it fails, however, the robot itself would provide the backup safety measure. Any contacts with the robot could then be automatically recorded and later be analyzed to evaluate the failure reasons and failure rates of the system.

Another option, if reliability has already been shown to reach at least $PL=a$, would be to speed up the robot movements or extend the possible scenarios, and then to further evaluate the system to prove that it also meets the requirements for a higher PL . In general, if a detailed risk analysis has been performed, it is possible to split the different risks into parts with different PL requirements and then protect the risks with lower PL with the new system and the remaining risks still with a second backup system. For example, when splitting the risks of the package-handling scenario (as in Section 3.2), some of those can be protected with a $PL=b$ system (or even $PL=a$, if the alternative risk assessment approach from *EN ISO 10218-2* with more accurate borderline classification is used instead). The advantage of such an approach would be that the *Performance Level* (PFH_D) of the system can gradually be improved and validated, while it can already be productively used in the meantime.

7.6.2 Force limiting

Another interesting new feature, enabled by the ability to distinguish between different body parts, is making use of the body-part-specific pressure and force limits (Table 4.1).

Currently, the forces (and resulting pressures) applied by collaborative robots are usually limited to the lowest limits, which are for the head and neck, because in many applications it cannot be reliably determined which body part could come into contact with the robot. With the proposed AI-based system, it would be possible to dynamically adjust the forces the robot applies depending on which body part is close to it.

This could be used in different ways. One example would be to allow faster movements or later braking initiations of the robot when only the hands are close to it, because they can absorb more force on contact. With higher limits, the robot can then be more productive, because it has to stop less often or for shorter times. It could also simplify the design of the robot, because usually contacts that are expected or planned involve the tool or workpiece of the robot, but not its main structure. Thus, it could be sufficient to add a force sensor or limiting mechanism only to the tool, which would be cheaper and easier to implement than a full-body force-limiting robot, and then protect the rest of the robot with the contact-free separation monitoring from before. If reliability has already been shown to reach at least $PL=a$, another technical simplification could be that the force-limiting mechanism only reduces the risk to the level of $PL_r=a$, for example by reducing the severity of injuries, and the AI-based system could then be responsible for the remaining risk reduction.

The force-limiting mechanism could also be combined with the proposed *interaction zones*, to allow certain intended contact interactions at certain positions. The initial example of assembling a bike serves as a good application for this. The robot could hold and move the bike frame to help the worker integrate the cable into the frame, while the person still applies some force on the bike to hold and place the cables. The safety system then can protect this interaction by only allowing a contact between the worker's hands and the bike frame at exactly the planned positions, but no other contacts between different robot or body parts or outside the planned robot trajectory segment. Furthermore, if a force-sensor was integrated into the tool, those contact forces could be monitored additionally. From a safety-standard perspective, such a concept also raises the question whether such planned and more controlled contacts would allow higher limits than currently defined in *EN ISO TS 15066*, which, however, will be left for future research.

7.6.3 Economic Considerations

Especially in the industrial domain, economic considerations are very important for the adoption of new technologies and therefore also for the development and integration of AI-based safety systems. The costs of the system have to be justified by the savings it can provide over time.

Costs of the system

The initial price of the system depends on two main factors, the robot itself and the components of the safety system. The robot price depends heavily on the model and manufacturer, and for the *KUKA-KR16* it can be estimated to around 30000€. For the safety system, the *Basler* cameras cost 359€ each, plus 139€ for the lens, and around 75€

for the mounting adapters and cables. The *RTX-5070* GPUs costed around 570€ each, and the other computer components were around 1100€, though for future production systems, the component selection could be more specifically tailored to the performance requirements of the concrete application to notably reduce the price. So each smart camera module costs around 2200€ and for the ten sets, plus around 3000€ for the trusses, the total price for the current setup is around 25000€. With a better and more optimized hardware selection and general price drops by technical progress, a price below 1500€ per smart camera module seems realistic for the future. This would be further reduced by purchasing components in larger quantities, so a price estimate of around 20000€ for the complete safety system appears reasonable and would also allow paying for software development efforts. Additional expenses such as integration, maintenance and capital costs are not considered here, as they are highly application-specific, but they should also be taken into account in a detailed economic evaluation of a specific system.

Besides the initial price, the running costs of the system also have to be considered, which mainly include the energy consumption. The robot itself already has a relatively high energy consumption, around 1.2kW at the current 10% speed, and up to 2.7kW at full speed, according to the experiments in [209]. The smart cameras require around 130W each, so for ten cameras it would be around 1300W. In comparison to less complex safety systems, like light curtains, laser scanners, or radar sensors, which usually require around 10W each [213–215], the energy consumption of the proposed system is notably higher. In comparison to the more complex *Veo Freemove* system, which requires around 1600W for eight cameras [25], the power consumption is, however, acceptable. With a current price for industrial electricity of around 0.16€/kWh for new contracts [171], the energy costs of the safety system would be around 0.21€/h and around 0.19€/h for the robot, both of which are relatively low.

Savings by the system

Given the above hardware costs, an assumption of 5 years of productive use and 200 working days of 8 hours per year, the robot itself would cost around 3.75€/h, and the safety system around 2.50€/h. So in combination with the energy costs, the total costs for the robot would be around 3.94€/h and 2.69€/h for the safety system. While higher utilization in multi-shift operation and the possibility of meeting higher production demands within the same factory space are not considered here, they would further reduce the effective costs and should likewise be taken into account in a detailed economic evaluation. To justify the above hourly costs, the new system has to provide savings of at least that amount by increasing the productivity of the human workers.

In Germany, the average hourly labor cost in the manufacturing industry amounted to approximately 43.40€ in the year 2024 [223]. Based on these values, if adding a robot increases the per-hour productivity of a worker by more than 9.08% and adding this safety system to a robot cell increases the collaboration efficiency by more than 6.20%, the system can already be justified from an economic perspective.

Since there already exist multiple research studies showing that the cycle time of human-robot collaborative assembly can be notably reduced by using intelligent sensor systems

and algorithms, similar experiments were not repeated in this work. In [63], which replaced a light curtain with dynamic distance monitoring by a laser scanner, the cycle time was reduced between 4% and 18%, depending on their scenarios. The authors of [207] reported a cycle time reduction of around 18% in comparison to multiple single-access zones by implementing a skeleton-based human detection and collision avoidance system. In the automotive case study of [163], which integrated a high-payload robot together with a *Pilz-SafetEYE* system for dynamic safety zones, a cycle time reduction of around 19% was achieved. Those experiments have shown large productivity increases through intelligent collaborations. Therefore, it is reasonable to expect that the proposed system can provide a productivity increase that would justify its costs as well.

Regarding the cycle time in the current scenario, the speed of the robot movements could be further increased to up to $1 \frac{m}{s}$, without increasing the risk probability (same avoidance parameter P), if the approaching hazard is easily recognizable and the human worker can easily escape it ([114, Tab. A.1]). This can be restricted to certain areas and working steps, but if the surrounding is modeled, it could also be solved more dynamically because the system knows the position of the human and can check escape possibilities and direction of sight.

Besides improving productivity, both the robot, with its ability to take over heavy and repetitive tasks, and the proposed safety system, with its ability to create dynamic safety zones and distinguish between different body parts, can also enable new use cases and applications, which would not be possible with traditional systems, and therefore can provide additional benefits by enabling new applications (like the bike assembly example described in the introduction).

7.6.4 Ergonomics

The safety standards also focus on the ergonomics of the system over the long term, to ensure that it does not cause any health issues for the workers [86, 109, 111]. The proposed system can contribute to this in multiple ways.

The ability to safely operate robots with higher reach and payloads in collaborative applications allows more physically demanding tasks to be transferred from human workers to the robot. This improves long-term ergonomics and also enables older workers to remain active in such work environments.

In addition to these robot-based capabilities, the system can also support the optimization of workflow ergonomics. One method that is commonly used to evaluate the ergonomics of a workplace is the *RULA (Rapid Upper Limb Assessment)* method [81, 160], which evaluates the posture of a worker and gives a score regarding its ergonomic acceptability. There already exist multiple research works that automatically evaluate the ergonomics of human postures and optimize the robot's behavior based on it [87, 133, 182, 252]. Therefore, this topic was not further investigated in this work. But since the proposed system can reliably detect the position of different body parts, those could easily be used to automatically calculate the *RULA* score, allowing a simple extension of the system to also monitor and improve the ergonomics of the workers.

7.7 Conclusion

In this chapter, the *Tutabo* system was further optimized to improve its usability in real-world applications and those optimizations were evaluated again.

By improving the hardware of the smart cameras, the latency of the pose estimation, and therefore of the overall safety system, was notably reduced. A comparison between different robot types highlighted the potential that could be achieved by an informed choice of the robot model.

To make better use of the scenario workflows, the concepts of *stop-zones* and *interaction-zones* were introduced. The *stop-zones* help to protect areas in the workspace, in which the robot is expected to stand still anyway if a human is within it. This allows ignoring detection errors of joints with high occlusions and lower detection reliability inside those zones, since the better-detected joints will instead ensure the stop and this way can allow reducing their required safety distances in the rest of the workspace. The *interaction-zones* work on the opposite principle and can allow contacts and interactions between the robot and certain body parts at certain positions, for example in hand-over scenarios. This can notably increase the productivity of the system, because it allows avoiding unnecessary stops, while still ensuring the safety of the workers.

The experiments integrating the *proposals* of *RapidPoseTriangulation* into the safety calculations have shown potential for further improvements of the pose estimation itself or an option to further increase the detection reliability of the system.

In the evaluation, notable similarities between the concepts of the current *Tutabo* system and the recommendations of the new *ISO/IEC-TR-5469* report were observed, which show that the proposed system is already following many of the requirements and suggestions of this technical report, and therefore is a good basis for further improvements towards certification. The evaluation of the system delay and required safety distances then showed significant improvements due to the presented optimizations.

Finally, since the task of developing a certifiable AI-based safety system is very complex, and cannot be solved by a single person alone, an extensive outlook was given on the next steps towards certification. This included suggestions for proving the reliability of the system over a longer time and about the necessity of using simulated environments to generate the large amount of testing data required for safety verification.

The following chapters will now take a step away from the requirements of the safety system and will look into the benefits the research of the previous chapters can provide for neighboring fields and also into improving the human-robot-interactions beyond the safety-related aspects.

Summary. This chapter examines the benefits that the previous research can provide in the context of understanding human behavior based on movement patterns, which in turn can be used to optimize collaboration efficiency.

8

Understanding Human Movements

8.1	Skarimva: Human Action Recognition	145
8.1.1	Related Work	145
8.1.2	Improving Pose Estimates	146
8.1.3	Results	147
8.1.4	Conclusion	151
8.2	Scriboora: Human Pose Forecasting	152
8.2.1	Related Work	152
8.2.2	Additional real-time Metrics	153
8.2.3	Evaluation under Unified Protocol	154
8.2.4	Scriboora	155
8.2.5	Noisy Coordinates	157
8.2.6	Conclusion	159

The outlook of the previous chapter showed that safety verification of AI-based systems will require substantial effort, which is beyond what a single researcher can realistically accomplish within a reasonable time frame. Therefore, it is important to show that improvements made to increase the reliability of the safety system also have benefits for other applications, to motivate more researchers to work on this topic.

One important aspect, which also has economic benefits, is the optimization of task planning in human-robot collaboration. Through this, the time and capabilities of both the human and the robot can be used more efficiently, which then can lead to higher productivity. A basic requirement in this context is the ability of the robot system to understand human intentions and actions, and to react accordingly. Since the algorithms integrated into the proposed safety system already provide a reliable pose estimation of the human, this information can easily be used as a basis to analyze the human movement patterns and subsequently interpret the human behavior and respond to it.

Regarding efficiency gains in human-robot collaboration, many prior studies have shown that predicting the human's next movements can notably improve collaboration efficiency and cycle times [190, 203, 232, 259]. Since, from a scientific point of view, demonstrating such efficiency gains in this scenario as well does not provide substantial new insight, the focus will instead be placed on the underlying algorithms and the influence of reliable pose estimation on them.

8.1 Skarimva: Human Action Recognition

To allow a computer system to intelligently react to human behavior, it is important that it can reliably recognize the actions performed by the human. For example, if a collaborative robot system would be aware of what the worker is currently doing, it could better plan the next robot motions or already prepare the next working step. Regarding the task of understanding the human movements, skeleton-based action recognition, which receives a sequence of skeleton poses and classifies it as a distinct action, has become a popular approach. This is because skeleton data is compact, robust to changes in human appearance and surroundings, and still preserves relevant motion information.

However, most research in this field towards increasing action recognition reliability has focused on improving the machine learning algorithms, while comparatively less attention has been given to the input data itself. In fact, progress in recent years appears to have plateaued, with only incremental accuracy gains being reported, despite the use of increasingly complex models. In the following, it is demonstrated that by rethinking the input data acquisition process, significant accuracy improvements can be achieved with existing models, leading to new state-of-the-art results under standard evaluation protocols. Therefore, the input data quality seems to have been a limiting factor for the performance of current skeleton-based action recognition models.

Following these results, it is proposed to treat skeleton-based action recognition as a multi-view application, instead of the current single-view standard. While using multiple cameras increases the system complexity slightly, the setup of additional cameras is relatively simple and worth the additional effort in most practical applications, if they are not already available for other purposes, as in this work. A multi-view strategy is therefore an effective way to improve action recognition reliability in practical systems.

This section summarizes my following published work, with some text passages directly copied from it:

Skarimva: Skeleton-based Action Recognition is a Multi-view Application. Bermuth, Daniel and Poeppel, Alexander and Reif, Wolfgang. February 2026. In *The 20th IEEE International Conference on Automatic Face and Gesture Recognition (FG)*. © 2026 IEEE. [53]

8.1.1 Related Work

Most research in skeleton-based action recognition has focused on developing new model architectures, while only a few works have investigated the influence of input skeleton quality so far. This will therefore be the focus in the following. Since the number of action-recognition datasets with multi-view image sequences, which are required for this investigation, is very limited, the following experiments are based on the *NTU-RGBD* dataset [147, 210], which is also the only dataset of this type that is widely used in the field. It is one of the most popular action-recognition datasets, which has the benefit of allowing an experimental comparison with many previous works in the following.

The original *NTU-RGBD* dataset [147, 210] was created with three non-calibrated *Kinect*-RGB-D cameras, from which each camera’s 3D skeleton estimation was processed independently. Although the skeleton quality was sufficient at the time of release, it contains a notable number of poor joint estimations, caused by occlusions, depth ambiguities, and the limited accuracy of the pose estimation algorithms available at the time.

The authors of *PoseConv3D* [93] found that using 2D poses from a more recent pose estimator already improves the action recognition accuracy notably, despite discarding the 3D distance information. In *NTU-X* [229], a different approach was followed, in which new whole-body 2D and 3D skeletons were created with a single-view 2D-to-3D lifting approach. This also resulted in accuracy improvements.

Both methods, though, are limited by the single-view setting, which is susceptible to depth ambiguities and occlusion problems. In contrast, this work explores a multi-view triangulation approach to even further improve the skeleton quality, which leads to significantly better action recognition results. A more detailed comparison with these previous works is provided later in Section 8.1.3.

8.1.2 Improving Pose Estimates

One of the main problems that prevents directly using a multi-view approach on the *NTU-RGBD* dataset is that the camera calibrations and synchronizations are not available. Therefore, a reconstruction procedure was developed to recover intrinsic and extrinsic camera parameters by minimizing multi-view skeleton reprojection and pose alignment errors (using the original camera-aligned skeletons) with iterative outlier removal, as well as to estimate temporal alignment between the video streams. Additional details about this procedure are provided in the appendix of the original paper [53] and the source-code which was integrated into the *skelda* library.

To calculate new skeleton poses, *RapidPoseTriangulation* can now be used. The pose estimation, with about 130 FPS on a single Nvidia RTX4080, runs faster than the dataset’s camera frame-rate of 30 FPS. To create motion sequences again (Figure 8.1), the skeletons are matched to individuals over time using a simple dataset-optimized tracking and outlier removal approach.

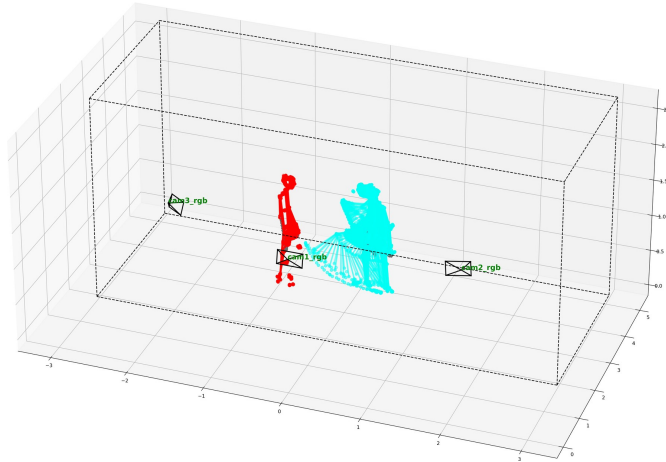


Figure 8.1. Example of a *kick other person* action with the new multi-view whole-body skeletons. [53][© 2026 IEEE].

8.1.3 Results

To evaluate the benefits of the new multi-view skeletons, three recent skeleton-based action recognition models were trained and evaluated on the *NTU-RGBD-60* [210] and *NTU-RGBD-120* [147] datasets. The models used are *MSG3D* [150], *DG-STGCN* [91], and *ProtoGCN* [145], which are all graph convolutional network (GCN) based models that have shown strong performance on these datasets. For the first two models, the implementations from *PySKL* [92] were used. In general, only a few modifications were necessary to train the models with the new skeletons, mostly related to the different joint counts and types. The training hyperparameters were kept identical to the original implementations as far as possible, to ensure comparability. As augmentations, only random rotations around the z-axis (which is now straightforward with the world-aligned skeletons) and size scaling were used.

Improvements with new whole-body poses

Table 8.1 shows that using the new multi-view whole-body skeletons leads to significant accuracy improvements for all three models on both dataset splits.

Method	original	wb25	wb137	wb69	wb31	body
<i>MSG3D</i>	90.8	96.5	94.0	96.3	96.4	94.0
<i>DG-STGCN</i>	92.0	96.7	94.4	96.5	96.9	94.6
<i>ProtoGCN</i>	91.8	97.1	94.3	95.6	96.6	95.5
<i>MSG3D</i>	87.0	93.6	90.4	93.3	93.5	88.2
<i>DG-STGCN</i>	88.3	94.6	92.7	94.9	94.1	89.7
<i>ProtoGCN</i>	88.7	94.8	91.6	94.4	94.3	89.9

Table 8.1. Classification accuracy results with the new whole-body poses on *NTU-RGBD-60* and *NTU-RGBD-120*. The split *wb25* has the same keypoints as the original one, *wb137* has all whole-body keypoints, *wb69* has no additional face keypoints, and *wb31* only adds the other three fingertips. [53][© 2026 IEEE].

An interesting observation is that adding more finger and face keypoints does not necessarily improve the accuracy further. This could be an indication that the models start to overfit on keypoints that are not particularly relevant for the action classes. This effect was also observed in *NTU-X* [229]. Additionally, the models become significantly slower with more input joints due to the large increase in graph edges. These findings raise questions for future work about how models can be designed to use whole-body keypoints more efficiently and focus on the relevant keypoints automatically themselves.

Comparison with other skeleton recalculations

Besides the approach followed in this work, two other methods for improving the skeletons of *NTU-RGBD* have already been evaluated. As described previously, *Pose-Conv3D* [93] introduced higher-quality 2D body-only skeletons from a more recent 2D estimator and *NTU-X* [229] estimated new whole-body skeletons with a single-view 2D-to-3D lifting approach. The results in Table 8.2 show that the multi-view triangulation approach proposed in this work outperforms both previous methods by a significant margin.

Method	body	wb25	body+fingers	body+hand+face
<i>original</i> [150]	-	91.5	-	-
<i>PoseConv3D</i> [93]	92.5	-	-	-
<i>NTU60-X</i> [229]	91.3	-	91.8	91.1
<i>Skarimva</i>	94.0	96.5	96.3	94.0
<i>original</i> [150]	-	86.9	-	-
<i>PoseConv3D</i> [93]	87.2	-	-	-
<i>NTU120-X</i> [229]	84.5	-	87.1	-
<i>Skarimva</i>	88.2	93.6	93.3	90.4

Table 8.2. Accuracy comparison with other skeleton representations using the *MSG3D* model. Note that there are small differences in the joint types and ensembling methods, but they are not expected to significantly impact the results, as explained in more detail in the paper. [53][© 2026 IEEE].

While the results of *PoseConv3D* [93] with the better body-only skeletons show that improving the quality of the 2D pose estimates is already beneficial, the authors also noticed a larger drop in performance when projecting the original 3D poses back to 2D ([93], Table 16), caused by the loss of helpful 3D information. This can also be seen when comparing the accuracy to the new 3D body-only poses of this work, which is clearly higher (while both methods used different 2D pose estimators, their performance is on a similar level, so the influence from this should be minimal).

The results of *NTU-X* [229] show that adding fingers to the skeletons is important to improve the average accuracy, but their lifting-based 3D reconstruction method is notably inferior to the multi-view approach proposed in this work. Another experiment in *PoseConv3D* also showed that lifting itself does not help ([93], Table 13). An explanation for this effect could be that the action recognition model already receives the pose input over time, so its input is similar to that of the lifting models. Therefore, if doing so would be helpful, the action model could learn to perform the lifting internally. Under this assumption, an additional 2D-to-3D lifting model would not add any new information, but instead could introduce additional errors from the lifting process. The only benefit of an external lifting model could be that it can be pre-trained with general human motions on a larger dataset, because no action labels are required. However, such a pre-training would be possible for the multi-view approaches as well. In fact, most of the lifting datasets are created from multi-view datasets already. In theory, depth triangulation is possible in a single-viewpoint video over time as well, using invariant distances (like limb lengths) moving in front of the camera, but multi-view triangulation is much easier and more computationally efficient. Using multiple views also has the important advantage that (self-) occlusions can be partially avoided, whereas a video model would need to infer occluded keypoints from context, which is susceptible to errors.

Multi-stream ensembling

The common evaluation procedure of previous works uses multi-stream ensembling of multiple input modalities (*joint*, *bone*, *joint motion*, *bone motion*, ...), combined with creating multiple randomized samples from a single test sequence, to improve the accuracy further. Note that since this procedure is computationally very intensive, it is not suited for real-time applications. The results with this evaluation method are given in Table 8.3 for completeness and set new state-of-the-art results. The model performance on the

single modalities is provided in the appendix of the original paper [53]. When comparing the results of the previous methods, it can be observed that the accuracy improvements between the different model architectures are relatively small, and sometimes the performance even decreased over time. The gain from the new skeletons, on the other hand, is much larger and could reduce the error of the *ProtoGCN* model by around 50%, which hints that the skeleton quality is a limiting factor for model development.

Method		xsub	Method		xsub
<i>MSG3D</i> [150]	'2020	91.5	<i>MSG3D</i> [150]	'2020	86.9
<i>CTR-GCN</i> [71]	'2021	92.4	<i>PoseC3D</i> [93]	'2022	86.9
<i>ST-GCN++</i> [92]	'2022	92.6	<i>ST-GCN++</i> [92]	'2022	88.6
<i>MotionBert</i> [271]	'2023	93.0	<i>LLM-AR</i> [188]	'2024	88.7
<i>InfoGCN</i> [74]	'2022	93.0	<i>CTR-GCN</i> [71]	'2021	88.9
<i>BlockGCN</i> [268]	'2024	93.1	<i>DG-STGCN</i> [91]	'2022	89.6
<i>MMP-ST</i> [267]	'2025	93.1	<i>SkateFormer</i> [88]	'2024	89.8
<i>LGS-Net</i> [178]	'2025	93.2	<i>InfoGCN</i> [74]	'2022	89.8
<i>DG-STGCN</i> [91]	'2022	93.2	<i>HD-GCN</i> [139]	'2023	90.1
<i>HD-GCN</i> [139]	'2023	93.4	<i>MMP-ST</i> [267]	'2025	90.2
<i>SkateFormer</i> [88]	'2024	93.5	<i>BlockGCN</i> [268]	'2024	90.3
<i>LA-GCN</i> [255]	'2023	93.5	<i>Shap-Mix</i> [260]	'2024	90.4
<i>TDSN-GCN</i> [143]	'2025	93.6	<i>MSA-GCN</i> [31]	'2024	90.6
<i>MSA-GCN</i> [31]	'2024	93.6	<i>LA-GCN</i> [255]	'2023	90.7
<i>DE-GCN</i> [169]	'2024	93.6	<i>JMDA</i> [249]	'2025	90.9
<i>JMDA</i> [249]	'2025	93.7	<i>Hyper-GCN</i> [269]	'2025	90.9
<i>Hyper-GCN</i> [269]	'2025	93.7	<i>ProtoGCN</i> [145]	'2025	90.9
<i>Shap-Mix</i> [260]	'2024	93.7	<i>DE-GCN</i> [169]	'2024	91.0
<i>ProtoGCN</i> [145]	'2025	93.8	<i>TDSN-GCN</i> [143]	'2025	91.1
<i>PoseC3D</i> [93]	'2022	94.1	<i>POTR</i> [64]	'2025	91.1
<i>Hulk</i> [244]	'2025	94.3	<i>SkeletonAgent</i> [146]	'2025	91.7
<i>SkeletonAgent</i> [146]	'2025	94.5	<i>3Mformer</i> [239]	'2023	92.0
<i>3Mformer</i> [239]	'2023	94.8			
<i>LLM-AR</i> [188]	'2024	95.0			
<i>POTR</i> [64]	'2025	95.3			
<i>ProtoGCN+Skarimva</i>		97.5	<i>ProtoGCN+Skarimva</i>		95.4

Table 8.3. Classification accuracy on *NTU-RGBD-60* (left) and *NTU-RGBD-120* (right). [53][© 2026 IEEE].

Few-shot learning

In real-world applications, it is often unlikely to have large amounts of labeled training data (here around 200 training examples per class), because labeling is time-consuming and costly. Therefore, the ability to classify new actions with only a few examples is an important feature for action recognition models. Thus, the following experiments evaluate few-shot learning capabilities with the new skeleton poses.

Because the source-code of the three top-performing previous works in this task was either incomplete or not available, the currently used classification models were extended instead. In particular, their default linear-layer classification head was replaced by a combined contrastive embedding and classification head, and the data sampling process was modified to support the new training scenario. The idea of the embeddings is to map samples of the same class closer together in the embedding space, while pushing samples of different classes further apart. At evaluation time, when the 20 new classes are introduced, their embeddings can be calculated from the single example, and new samples can then be classified by calculating the shortest distance to the example embeddings.

Table 8.4 shows that the current state-of-the-art performance for one-shot learning can be improved notably with the new skeletons as well.

Method	
<i>APSR</i> [147]	45.3
<i>TCN OneShot</i> [202]	46.5
<i>SL-DML</i> [161]	49.6
<i>Skeleton-DML</i> [162]	54.2
<i>MotionBert</i> [271]	61.0
<i>Koopman</i> [242]	68.1
<i>M+C-scale</i> [257]	68.7
<i>SkeletonX</i> [264]	69.1
<i>ProtoGCNce+Skarimva</i>	76.0

Table 8.4. One-shot transfer classification accuracy on *NTU-RGBD-120*. [53][© 2026 IEEE].

However, using only one labeled example is considered unreasonable, because in practical applications it should be possible to provide a few more examples with low effort, especially if accuracy benefits from it. Therefore, Table 8.5 investigates the results with five examples per action class, a scenario that leads to a notably better performance.

Method	
nearest neighbor	84.7
5-nearest neighbor	84.9
prototype matching	84.9

Table 8.5. Few-shot transfer accuracy with five examples per action. [53][© 2026 IEEE].

Real-time Applications

As was highlighted by the previous chapters, many applications require responding to human actions with only a small delay. Therefore, the prediction speed of the models is of interest as well. As can be seen in Table 8.6, the three tested models are rather small and can predict actions very quickly, making them suitable for real-time applications. In the context of the *Tutabo* setup, they are still slightly slower than the camera and skeleton outputs, but even if a GPU upgrade is not an option, it would be possible to skip every second or more skeleton-detections, which should not have a notable impact on the accuracy, or to run the action classifier only every ten frames instead.

Method	Size	FPS
<i>MSG3D</i>	3.14M	189
<i>DG-STGCN</i>	2.04M	140
<i>ProtoGCN</i>	4.46M	150

Table 8.6. Model size and inference speed on an Nvidia-RTX4080 using batch-size 1. [53][© 2026 IEEE].

In the standard evaluation protocol, the input of the models is a complete action sequence. From each, a fixed number of frames (here 100) is randomly sampled and fed into the model. For this approach, the start and end of an action have to be known beforehand, but this is not very realistic for streamed camera inputs in a live setting. Therefore, to

evaluate such a setting, the following experiment samples a fixed-length continuous sequence at a random starting position from the action sample instead. This would be similar to a windowed streaming approach. The results in Table 8.7 show that this leads to a slight performance decrease, growing with smaller window sizes, but the accuracy is still reasonably high.

Method	3s	2s	1s
<i>DG-STGCN</i>	95.7	94.7	92.8
<i>ProtoGCN</i>	95.6	94.7	92.8

Table 8.7. Comparison of the effect on classification accuracy of different input durations on *NTU-RGBD-60*, using a single random continuous subsequence of each sample. [53][© 2026 IEEE].

8.1.4 Conclusion

The experiments performed in *Skarimva* showed that improving the input data quality is an effective way to boost the performance of skeleton-based action recognition models. In particular, it was demonstrated that leveraging multiple camera views to triangulate 3D skeletons leads to substantially better skeleton poses and subsequently to large accuracy gains. Across multiple settings, the classification error could be reduced by over 50% compared to the original skeletons, achieving new state-of-the-art results on the popular *NTU-RGBD* dataset.

Based on these findings, and the comprehensive discussion in the paper [53], it can be concluded that skeleton-based action recognition should be treated as a multi-view application, even though most research in this field uses single-view skeletons so far. While a precise calibration of the cameras (like in *Tutabo*) is recommended to achieve the best results, the experiments have shown that even the rough calibration obtained for *NTU-RGBD* already leads to significant improvements, which simplifies an initial integration in practical applications. To support future research, the code and models used in this work were made publicly available¹.

In the context of the *Tutabo* setup, integrating action recognition can provide important insights into the human’s current and most likely future behavior. A possible approach could be that the system builds an internal workflow model of the human actions by watching them a few times, to estimate a probability distribution over the action order. For example, if the system recognizes that the human is currently checking the contents of a package, it can then predict that the next action will likely be that the worker will sort the package into one of the two boxes on his left. This means that the worker will move away from the table, making it an optimal moment for the robot to place the next package on the table (which then can even be done with higher robot speed since the worker is further away).

Other options, which were already investigated in different research works, are to use the action recognition output for progress monitoring [206, 245], or to support learning robot movements from human demonstrations [125, 172, 237].

¹<https://gitlab.com/Percipote/Skarimva>

8.2 Scriboora: Human Pose Forecasting

While recognizing the current human action helps with predicting the human’s future behavior over longer time horizons, it is also important to have a more direct short-term prediction of the human’s future movements, which can be used, for example, for collision avoidance in close proximity.

Since human pose forecasting has many applications, there are already many research papers that propose different solutions for this problem, of which some will be described more closely in the following. Although rapid progress is reported, the state of the field remains difficult to assess. Results are often produced with heterogeneous preprocessing and metric implementations, partially released code, and sometimes contain evaluation errors. Comparisons across papers therefore become unreliable, and claimed improvements can vanish once consistent protocols are applied. Previous works also only compare against a limited set of specialized baselines, but miss the question of whether mature models from neighboring domains already provide strong baselines. At the same time, the evaluations rarely reflect deployment conditions. In practical systems, input poses often need to be obtained from markerless estimators, because outfitting pedestrians with tracking suits is not an option and outfitting workers in a factory is not feasible. Prediction errors of such estimators differ from synthetic noise used in some prior studies and have not been investigated yet.

Therefore, a broad reproducibility audit under a unified protocol is needed to clarify the actual state of the field, which also results in the finding that the current state-of-the-art performance can be improved by utilizing baseline model architectures from a neighboring domain. To better account for real-world conditions, the impact of model latency and realistic skeleton-pose errors have to be investigated as well.

This section summarizes my following research work, with some text passages directly copied from it:

Scriboora: Rethinking Human Pose Forecasting. Bermuth, Daniel and Poepfel, Alexander and Reif, Wolfgang. Preprint. November 2025. [49]

8.2.1 Related Work

Most recent pose forecasting approaches have specialized into two categories: (a) Predicting a relative future pose, where the center of the person, typically its hip joints, stays at the same position over all the time (*HisReplItself* [247], *STSGCN* [220], *SPGSN* [140], *PGBIG* [154], *MotionMixer* [57], *siMLPe* [102], *EqMotion* [254], *DePOSit* [200], *AuxFormer* [253], *GMFNet* [211], *GCNext* [243], *DeformMLP* [106]). (b) Predicting an absolute future pose (Figure 8.2) that also contains global human motion trajectories (*SoMoFormer* [234], *TBiFormer* [181], *JRTransformer* [256], *IAFormer* [251], *EMPMP* [266]). Please see the paper [49] for a more detailed description of the related work architectures and evaluation protocols, for readability reasons it is omitted here.

Depending on the two different categories, the commonly used datasets and evaluation protocols differ as well. For relative pose forecasting, the standard dataset is the *Human3.6M* [122] dataset, which was already described in Section 3.4.1. The absolute

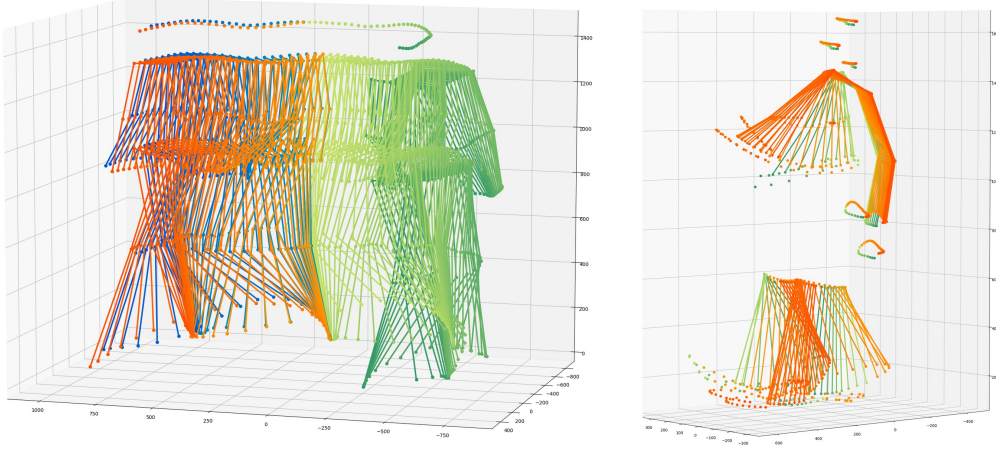


Figure 8.2. Example of an *absolute* (left) and *relative* (right) pose forecast of a walking person. The green skeletons visualize the input sequence of the prediction model, the red ones the predicted future poses, and the blue ones the ground-truth labels.

pose forecasting approaches, on the other hand, are usually evaluated on the *3D Poses in the wild* (3DPW) dataset [238], which consists of 60 action sequences in a real-world setting, or the *CMU-MoCap* [66] dataset, in which the actors have been captured by twelve cameras in a $3m \times 8m$ room.

Most forecasting approaches focused on using ground-truth pose labels from the dataset, and only a few approaches [80, 200] also added artificial errors. For several reasons, which are explained in more detail in Section 8.2.5, such artificial errors differ from the pose estimation errors. No paper was found that evaluated the influence of noisy joint coordinates obtained from pose detectors, which therefore will be investigated later on.

Before comparing all models on the same dataset, the first research question was: *Can the results of the approaches described in their respective papers be reproduced?* This should have resulted in a short answer of *yes*, but in approximately half of the evaluated approaches some severe problems were encountered. Those findings, which are described in detail in the paper [49], highlight the importance of publishing open-source code for reproducibility tests.

8.2.2 Additional real-time Metrics

Forecasting is a real-time-only task, therefore the runtime speed of the algorithm is of high importance. For example, it does not make sense to use a very large model that takes more than a second to predict the next second of human motion, because in this case one could just use the real measurements instead. This is not captured by the commonly used *Mean Per Joint Prediction Error* at a certain forecast timestep t ($MPJPE_t$), so to take this into account, two new metrics are introduced.

The *forecast after delay error* (FADE, Equation 8.1) focuses on the average performance, and has the general idea that if the prediction takes some amount of time (measured as *predictions per second* (PPS)), the forecast needs to be longer by the same amount

of time. Because it would be too complicated to always change the output sequence length of each model accordingly, it will be approximated by a simpler approach that assumes a linear increase of the error with the time.

$$FADE_t = MPJPE_t + MPJPE_t \cdot \frac{1000 \text{ ms}}{t} \cdot \frac{1}{PPS} \quad (8.1)$$

The *fast change estimate* (*FCE*, Equation 8.2) targets movement/direction changes, which are especially important for future motion planning. For example, if a person is standing still for the whole model-input-time, the forecast will likely predict that it will continue to stand still. But if the person suddenly starts to walk, the current forecast will be completely wrong, therefore the system needs to make a new forecast as soon as possible. The *FCE* describes the movement a person can do until a new forecast is available, using the upper human limb movement speed defined in *EN ISO 13855* [22].

$$FCE = 2000 \frac{\text{mm}}{\text{s}} \cdot \frac{1}{PPS} \quad (8.2)$$

Note that in real applications, the image-to-pose prediction model will also take a notable amount of time (basically $Time_{system \text{ delay}}$), which is not included in those metrics here, because it is independent of the pose forecasting model itself, and therefore not required for their comparison.

8.2.3 Evaluation under Unified Protocol

Forecasting absolute human poses is important for a variety of applications, for example to improve the motion planning of a robot in proximity to a human. If a person moves around, their absolute position in the world can contain important information that is not captured by relative pose forecasts, and such models would therefore require an additional trajectory prediction model. Instead of building a more complex two-model system that stacks trajectory and relative forecasting models and then also their prediction errors, the idea here was to evaluate if the already existing relative forecasting algorithms can be adapted to directly predict absolute poses as well. Since the data format of both tasks is very similar, only small adjustments to the model architecture had to be made in most cases, which are described in more detail in the paper [49].

For the performance comparison, the *Human3.6M* [122] dataset is used, with 50 input and 25 output timesteps at a frame-rate of 25Hz, resulting in 1s of motion forecast. The models are trained on only 13 joints (2 hips, 2 shoulders, 1 nose, 2 knees, 2 ankles, 2 elbows, 2 wrists) to simplify usage with other datasets that have fewer ground-truth joint labels. The input and output format is then directly a sequence of 3D joint coordinates.

As can be seen in Table 8.8, most converted relative models can predict absolute motion forecasts fairly well. A new static approach was added for comparison as well: *last delta average* repeats the average of the last input-pose delta, so all joints keep moving in the direction of the last overall movement (the average delta is used to preserve the human shape over time). As a classic baseline method, a simple *Ridge Regressor* was trained.

Method		Type	Size	MPJPE	PPS	FCE	FADE
<i>Repeat last frame</i>		-	-	301	-	-	301
<i>Last delta average</i>		-	-	264	-	-	264
<i>Ridge Regression</i>		-	-	185	-	-	185
TBiFormer	'2023	abs	5.64M	301	293	7	302
PGBIG	'2022	rel	2.65M	279	35	57	287
GMFnet	'2024	rel	11.3M	279	51	39	284
SPGSN	'2022	rel	5.21M	262	19	105	276
STSGCN	'2021	rel	202K	275	1405	1	275
HisReplItself	'2020	rel	2.93M	233	142	14	235
DePOsit	'2023	rel	1.17M	197	6	333	230
GCNext	'2024	rel	4.95M	175	38	53	180
siMLPe	'2022	rel	129K	178	158	13	179
SoMoFormer	'2022	abs	4.88M	179	586	3	179
MotionMixer	'2022	rel	37.2K	168	632	3	168
IAFormer	'2024	abs	2.53M	158	28	71	164
AuxFormer	'2023	rel	1.27M	164	403	5	164
JRTransformer	'2023	abs	3.74M	162	595	3	162
DeformMLP	'2024	rel	1.29M	159	512	4	159
EMPMP	'2025	abs	314K	150	215	9	151
EqMotion	'2023	rel	3.36M	148	45	44	151

Table 8.8. Comparison of *absolute* pose forecasting on *Human3.6M*. *MPJPE* and *FADE* are measured at timestep 1000 *ms*. The error metrics are in millimeters. *PPS* is tested on a computer with AMD-9900X and a single Nvidia-RTX4080. A batch-size of 1 was used for testing. Note that the size and speed of some models depends on the number of input and output timesteps (here: 50 in, 25 out).

An interesting finding that can be seen here is that most models using graph neural networks (*HisReplItself*, *STSGCN*, *SPGSN*, *PGBIG*, *GMFnet*) show lower performance in this setting than the others using a different network structure. It seems their task-optimized architectures do not handle the task switching too well. The results however show that the general idea of switching the task of the relative models was successful, because some of those models could outperform the absolute-pose algorithms of *SoMoFormer*, *JRTransformer*, and *EMPMP*.

8.2.4 Scriboora

The results in Table 8.8 showed that models with diverse architectures can achieve similar results, and also that task switching (from *relative* to *absolute*) is not a problem for some models. The idea behind *Scriboora* is based on the question: *Can speech-to-text models be retasked as well?* While the transfer from speech to poses might seem far-fetched at first, the speech models play an important role in human-robot interaction as well (more on that in Section 9.2), so they are already implemented in the system, and their architectures have interesting properties that could be beneficial for pose forecasting as well. From a high-level perspective, both tasks are similar, a sequence of input numbers is transformed into a sequence of output numbers, except that the input and output are characters or words, not joint coordinates. The speech models are designed to learn both local patterns between neighboring timesteps and global patterns across the complete sequence, and there is also some focus on computational efficiency, both of which are interesting properties for pose forecasting as well. To reach a good data transformation, the model should be able to learn the general idea of the observed data, either the meaning of sentences, or the movement concept (like walking or waving) of a motion sequence. For the experiments with different speech-to-text

models (*DeepSpeech* [166], *QuartzNet* [135], *Conformer* [101], *Squeezeformer* [132]), the implementations from my *Scribosermo* [44] project were used.

Network Changes

The main changes needed were to adapt the data pipeline, so that motion instead of audio sequences could be loaded. The models themselves required only minor changes. The hidden dimensions and filter shapes were adjusted to keep the layer and model sizes balanced. All *BatchNorm* layers were removed, because they resulted in poor training results. For *Conformer* and *Squeezeformer*, the initial subsampling layers, which compress the (audio) input sequences to increase model throughput, were updated to perform only a $2\times$ temporal downsampling (instead of $4\times$) to match the required output sequence length.

Performance Results

As can be seen in Table 8.9, the results of the converted speech models are surprisingly good and can compete with the best forecasting models. The *Conformer* models combine convolutional, linear, and attention layers to capture both local and global patterns and are therefore among the leading approaches in speech recognition. This relatively generic architecture also appears to adapt well to the new task.

Method		Size	MPJPE	PPS	FCE	FADE
DeepSpeech	'2014	2.35M	198	653	3	198
QuartzNet	'2020	3.17M	168	1300	2	168
Squeezeformer	'2022	3.31M	149	960	2	149
Conformer	'2020	3.22M	147	1049	2	147
MotionConformer		9.23M	143	929	2	143

Table 8.9. Results of converted speech models on *Human3.6M*.

As can be seen in Table 8.9 as well, most models are very fast, so this leaves room to further increase the size of future models. With that in mind, the *Conformer* model was further improved by increasing the layer dimensions. As another change, because the motion sequence length is notably shorter than the average audio sequence length, the time downsampling at the start of the model was moved to the end of the model, so that information is retained longer in the model. The resulting model is called *MotionConformer* in the following. Although it is the second-largest model overall, the training time of $1h$ on the Nvidia-RTX4080 is at the lower end among the compared architectures. As can be seen in Table 8.9, it achieves the best results of all tested models, with a very high inference speed as well.

The paper also contains ablation studies on the model updates, and further experiments on the *CMU-MoCap* [66] dataset, with $1s$ and $3s$ forecast durations, and on the *CHI3D* [98] dataset, to evaluate forecasts of multi-person interactions. In all cases, the new *MotionConformer* is the best performing forecast model. Further details on these experiments are omitted here for brevity and can be found in the paper [49].

8.2.5 Noisy Coordinates

As mentioned earlier, the use of noisy joint coordinates obtained from pose detectors can introduce additional uncertainty in the forecast estimation process, which has not been evaluated before. Especially in markerless settings, like in the *Tutabo* system for example, they might inflict notable errors to the forecast which can decrease the overall effectiveness of the system.

Previous Approaches

With [80] and [200], there are already two studies that investigated the impact of artificially generated noise. Both of them randomly dropped some of the joints and also added Gaussian noise to the joint positions. Only *DePOSit* [200] released their code for further use. Dropping random joints is a type of noise that normally does not occur in markerless settings, because most pose estimators always attempt to estimate joints that are not visible, without explicitly marking those estimated joints (which would allow dropping them afterwards). Regarding the Gaussian noise, across multiple frames (as in the model inputs) the random noise averages around the true location, while pose estimators rather have more structural errors (like forearm too short in all frames) in which the noise does not follow a Gaussian distribution. Due to those reasons, this work will investigate a more realistic type of error, namely the one caused by a preceding pose estimation network.

New Dataset

To simplify the comparison between the performance with and without such noisy joint labels, a reasonably large dataset is needed which also contains ground-truth labels. The authors of [203] already created a pose forecasting dataset with predicted joint labels (using *VoxelPose* [231]), but it has no ground-truth labels, and more importantly contains almost no global movement of the persons, since it was created for use with a *relative* forecaster. The human workers are normally standing at a fixed position in front of a robot and mostly only move their arms. Therefore, for the following experiments, a new variant of the *Human3.6M* dataset was generated. The joints were predicted by *RapidPoseTriangulation*, which achieved a fairly good zero-shot performance on the unknown dataset of *52mm MPJPE* across all frames. In the very rare case that the person was not detected, or only with a score below a threshold (of 0.1 on a 0–1 scale), the joints of this frame were replaced with the joints of a previous valid frame. In case multiple persons were detected, the one with the best score was kept. The pose prediction was very fast, with an average of 160 FPS on a single Nvidia-RTX4080 GPU. However, for comparability with the initial experiments, the input of the forecasting model was set to 25 frames for 1000ms again.

Noise Experiments

To keep the evaluation as close to real use-cases as possible, the models were pre-trained on a large mixture of datasets first (here *CMU-MoCap* [66] and *BMLmovi* [100],

BMLrub [230], *KIT* [159] from the *AMASS* [157] collection, which together equate to around 11:30h of motion). The 13 joints from before were used again, but in the case of *CMU-MoCap* the *nose* is replaced with the *upper-head* joint position since no *nose* joint exists in this dataset. The original frame-rates of those datasets are 120fps or 30fps, but to match the target dataset, the forecast horizon of one second is represented with 25 output steps. Even though the motions are slightly slower this way, the model should still be able to learn the general patterns of motion.

One very important requirement for machine learning models is their generalization performance. Table 8.10 shows that the performance of the pre-trained models on the *Human3.6M* dataset is quite close to the results from the last section (Table 8.8, Table 8.9), where this dataset was used for training too. In conclusion, the models show a good generalization capability to the new dataset.

Method	MPJPE		
	40 /	400 /	1000
EMPMP	4.3 /	59.7 /	165
EqMotion	4.0 /	52.0 /	156
MotionConformer	4.0 /	51.6 /	149

Table 8.10. Zero-shot transfer capabilities from a large mixture of datasets to *Human3.6M*, measured as *MPJPE* in millimeters at the given timesteps in milliseconds.

Table 8.11 shows a strong negative influence of skeleton joints generated by the 3D-pose-estimation network on the model performance. While the left half of the table shows the performance drop due to noisy joints alone, the right half reports error with respect to the ground-truth targets and therefore also includes the error caused by the prior pose estimator (here on average 52 mm at time zero). This would be the real error of the prediction, while the left one would be the only one that can be measured in a system that only uses predicted poses.

Method	400 / 1000		400 / 1000	
EqMotion	513 /	842	532 /	856
EMPMP	100 /	215	123 /	229
MotionConformer	94.2 /	214	118 /	228

Table 8.11. Influence of skeleton joints generated by a 3D-pose-estimation network on model performance, measured as *MPJPE* in millimeters at the given timesteps in milliseconds. The left section uses the network predictions as inputs and targets (which would be the measurable error in live systems), and the right one the predictions for input and ground-truth labels as targets (which would be the real error).

Even though adding artificial Gaussian noise to the joint coordinates does not match the errors from a pose estimation system, it can still help to prepare the forecasting network for such inputs, and it is more effective than training solely on clean data. Finetuning the networks with random noise (*std*: 25mm, clip at 125mm) notably improves the zero-shot performance for longer (1000ms) predictions, as can be seen in Table 8.12. The training of *EqMotion* always diverged mid-training, so no results are available for this model here. As a side note, further experiments with the other two models showed that

in both cases, the performance on the clean ground-truth dataset dropped slightly after adding noise, but this is to be expected because of the then lower train-test similarity.

Method	400 / 1000	400 / 1000
EMPMP	82.5 / 188	108 / 204
MotionConformer	82.4 / 185	106 / 199

Table 8.12. Improvements through artificial noise pre-training. The left/right columns are separated into prediction/groundtruth targets, as in Table 8.11, showing the measurable/real error.

In the upper part of Table 8.13, the model is finetuned on the new dataset variant. This simulates that the system is already running and has collected some movement observations (the trainset has around 1:55h) and can now be optimized with those observations for the current environment. Since the system shall only use a camera-based pose estimation approach, the network is finetuned on the noisy pose predictions instead of using the ground-truth labels. As expected, the performance greatly improved in the left part of the table, which only uses predicted poses. The performance regarding the ground-truth positions also improved, but not as much as with the predicted poses.

Method	400 / 1000	400 / 1000
MotionConformer	66.9 / 159	94.8 / 177
EMPMP	76.4 / 179	104 / 195
EqMotion	68.5 / 168	96.1 / 185
MotionConformer	69.1 / 165	98.3 / 183

Table 8.13. Improvements through unsupervised learning. Finetuning the pre-trained model from Table 8.12 on noisy predictions in the upper part, training all models from scratch in the lower part. The left/right columns are separated in the same way as in Table 8.11

Because in the other two approaches no finetuning implementation was provided, and the pre-training of *EqMotion* diverged anyway, another experiment in which all models were trained from scratch on the noisy dataset variant was conducted (lower part of Table 8.13). The comparison between the two *MotionConformer* results shows that pre-training on a large dataset is beneficial, but the performance increase is modest. The finetuning converged much faster though, and from a practical point of view, a pre-trained model is required for the initial system setup anyway. In terms of *MPJPE* the results of *EqMotion* and *MotionConformer* are very similar, but taking the runtime-speed into account as well (as in Table 8.8 and 8.9) increases the performance advantage of *MotionConformer* again.

8.2.6 Conclusion

Human pose forecasting was re-examined with a focus on reliable evaluation and deployment realism. The replication attempts of recent model trainings uncovered several issues in prior evaluation code and preprocessing. By re-evaluating them on a unified dataset and task, a more accurate comparison of their performance was achieved, which can serve as a stable reference for future work.

By viewing forecasting from a high-level viewpoint as a sequence-to-sequence problem, architectures developed for speech recognition, another sequence-to-sequence task, were transferred to this new setting. The adapted *Conformer* and the further optimized *MotionConformer* developed in this research achieved new state-of-the-art accuracy while maintaining real-time throughput.

To assess deployment performance, first, two new metrics, *FADE* and *FCE*, were introduced, which also take the real-time nature of the forecasting task into account. Second, the robustness of the models against realistic input noise was evaluated. These experiments revealed that a significant performance drop will occur in real-world applications if the models are trained and tested only on clean motion-capture data. However, it was also shown that this performance drop can be mitigated by unsupervised finetuning on the noisy input data, which can be easily collected in a running system.

Overall, the results suggest that rethinking the approach to human pose forecasting has proven effective. Following a more holistic view of the task, from model architecture to evaluation and deployment, can lead to more robust and practical solutions. Improving the reproducibility and evaluating models under more realistic conditions should become a standard practice for future research in this area. To support this, all code and datasets created in this work are made publicly available².

In the context of the *Tutabo* setup, the pose forecasting system can be used to predict the future movements of the human worker and optimize the robot's motion planning accordingly. Generally, optimizing the robot paths based on human movement forecasts is already a well-researched topic [149, 190, 203, 232, 259], but an often neglected problem is the safety impact of replanning paths. The new paths must not lead to new collision risks, which requires a reliable position estimation of the human worker, which the proposed AI-based system of *Tutabo* could provide. Regarding the motion forecast itself, the better the pose predictions are, the less impact their noise will have on the forecast accuracy.

With slight modifications, the *MotionConformer* model could also be adjusted to predict the current position of the human worker instead, using the proposals from *Rapid-PoseTriangulation* from the last timesteps as input, making it an alternative method to the current proposal merging approach. However, experiments in this direction are left for future work.

In the context of safety verification, predicting the future movements of the human worker, using both pose forecasting and action recognition methods, might also help to optimize the verification process. The idea would be that if the system can predict the future human position, it could optimize the robot's path to increase the distance between the human and the robot, and thus reduce the possibility of *dangerous* errors close to the robot. If the pose estimation errors are outside the minimal separation distance, they are technically not *dangerous* anymore, which could help to further reduce the number of truly *dangerous* failures, which must be below the required *Performance Level* threshold. However, evaluations in this regard are left for future work as well.

²<https://gitlab.com/Percipote/Scriboora>

Summary. This chapter examines concepts for improving human-robot interactions beyond the safety-related aspects of the current *Tutabo* system.

9

Improving Human-Robot Interactions

- 9.1 RoboJack: Guidelines for improving Interactions 163**
- 9.2 Jaco-Assistant: A privacy-aware Voice Interface 166**
 - 9.2.1 Jaco-Assistant 166
 - 9.2.2 finstreder 169
 - 9.2.3 slungt 171
 - 9.2.4 Benchmarks 172
 - 9.2.5 Conclusion 174
- 9.3 Crazyflie Swarms: Beyond Industrial Robots 176**

The previous chapters heavily focused on the technical aspects of developing an AI-based safety system for human-robot collaboration, but the interaction between humans and robots is also an important aspect that requires attention. Users of a collaborative robot system should be convinced of the advantages of the system so they adopt it on their own initiative and prefer it over conventional methods without external pressure. While this is essential for any service-related application, it is also important in the manufacturing context, as it can lead to higher worker motivation. In the context of the industrial revolutions described in the introduction of this thesis, improving interaction quality can be seen as an important aspect of the transition towards *Industry 5.0*, with its focus on human-centric applications.

9.1 RoboJack: Guidelines for improving Interactions

In my master's thesis, I have explored the development of simple but comprehensive guidelines for improving human-robot interactions using an already existing robot system as a reference case. Such guidelines can then ideally be used without requiring extensive background knowledge to analyze and improve similar human-robot interactions, and they were tested exemplarily with the robotic system *RoboJack* (Figure 9.1). Due to their general nature, these guidelines are interesting for improving the interaction quality in a wide range of human-robot collaborative systems, including that of *Tutabo*. This section briefly summarizes my master's thesis, with selected text passages directly copied and translated from it:

Methodiken zur Analyse und Verbesserung der Interaktionsqualität in Mensch-Roboter-Kollaborationsanwendungen (Methods for analyzing and improving the interaction quality in human-robot collaboration applications). Bermuth, Daniel. October 2020. [42]

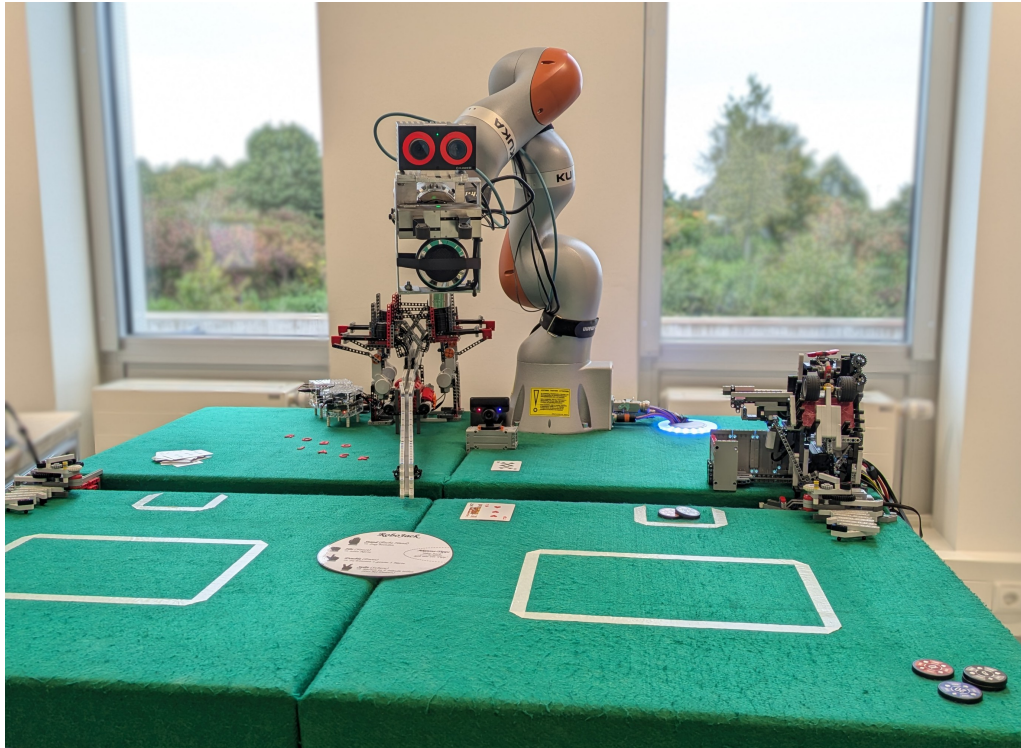


Figure 9.1. The *RoboJack* system playing the role of the dealer in the game *BlackJack*. Two players can sit at the table and play against the robot, which interacts with them by asking for their bets, dealing them cards, and executing the game actions signaled by the players via hand gestures in the large rectangles.

RoboJack takes over the tasks of the dealer in the game *BlackJack*. The goal of the casino game *BlackJack* is to have more points in your hand than the dealer, but not more than 21. Up to two players can sit at the table in front of the robot and play against it. After detecting the players via face recognition, the robot asks for their bets and then

deals them cards. When all cards are dealt, the players can request the game actions they want to execute (like *hit* or *stand*) using hand gestures, which the robot then recognizes and executes. Optionally, the player can also ask *RoboJack* for a hint before showing a hand gesture and be told by the system which action would be the best according to the basic strategy at the current game state. At the end of the game, *RoboJack* tells the players their winnings or losses, cashes out the bets, and collects the playing cards again.

The core focus of the thesis was to develop guidelines for improving the interaction quality in human-robot collaboration applications. They were developed based on the one hand on the experience with the *RoboJack* system and on the other hand on already existing literature. Following an example-based approach ensures a more global perspective and can still result in generalizable concepts for many characteristics. In contrast to that, it was found that existing literature often provides more detailed reasoning for specific subtasks, but can lack a holistic perspective. The developed guidelines (with more details and reasoning in the original thesis [42]) are summarized as follows:

1. **System works reliably:** A system that does not work reliably will not be used in the long term. Since normally not all errors can be excluded beforehand, the system should also be able to ask for help if it cannot solve the problem itself.
2. **Users feel safe in front of the robot:** The system should not only be safe, but also give users the feeling of being safe. This can be achieved by keeping an appropriate distance and by optimizing the robot's postures and movements.
3. **Robot works in an understandable way:** The movement and interaction patterns should be designed in a way that the robot's behavior is predictable and easy to understand for the users. The system should also give useful feedback.
4. **Equivalent or better task performance:** The system needs to perform the task as well as a human or provide extra benefits. This is considered as a very important aspect for the long-term acceptance of the robot. If this is not possible for a particular task, it is necessary to compensate for this with other capabilities.
5. **Comfortable environment:** Since first impressions and the environment influence the users' perception of the system, the surrounding environment should also be optimized.
6. **Intuitive interactions:** Make the interactions similar to how a human would interact with a user, as those are normally intuitive and follow common standards.
7. **System is personalized:** To feel more appreciated by the system, especially in long-term interactions, the system should be personalized.

The effectiveness of the guidelines was evaluated with the help of two iterative user studies. The first was done with the original *RoboJack* system, and the second with an updated version of the system, which was improved by implementing updates derived from the guidelines. In the first study 18 university students participated, and in the second study, which was further increased in the time between the thesis and this work, a total of 27 subjects participated. After playing some rounds of *BlackJack* with

the robot, the participants were asked to fill out a questionnaire, which contained questions regarding the different guideline items and some general questions about their experience with the system, which could be rated on a scale from 1 (worst) to 5 (best). In summary (with more details again in the original thesis [42]), the results of the user studies showed the success of using the guidelines for improving the interaction quality of the system. The evaluation itself was split into two parts: the first part focused on the single guideline items, and the second part on the overall user experience. The user response showed that improving the system along the proposed guidelines led, as expected, to an improvement in the questionnaire ratings targeting those specific items. The more important finding, however, was that splitting the improvements into the different guideline items and optimizing them separately not only led to improvements in those specific items, but also to a significant improvement in the overall user experience. In conclusion, *RoboJack* was significantly better accepted by users after analyzing and improving the existing interaction problems with the help of the proposed guidelines. Players had significantly more fun in the game, felt safer while playing, found the interactions more intuitive, and the user-friendliness of the system, calculated with the *System Usability Scale (SUS)* [60], has also improved significantly. Therefore, the results of the second study have shown that the developed guidelines can be successfully applied to improve human-robot interactions in an already existing application. While there already existed other robots that could play *BlackJack* before [2, 3], and even though some others were also developed after the *RoboJack* project was finished [1, 68, 97], the *RoboJack* system is currently the most advanced *BlackJack*-playing robot. Even though the proposed guidelines were tested only with this single system, they were not especially tailored to *RoboJack* but formulated in a more general manner, so it is expected that they are usable as a tool for improving the interaction quality in other human-robot collaboration applications as well.

In the context of the *Tutabo* setup, several of the developed methods can also be used to improve the interactions according to the proposed guidelines. For example, knowing the current and future position of the human workers, and optionally their gaze direction, can help to optimize the robot's movements to increase the perceived safety of the users. Integrating the action recognition system, which allows the system to understand what the human worker is doing, can help to make the interactions more intuitive. It can also help to personalize the system, by being able to react to specific actions of the users, by learning individual workflow preferences, or by providing personalized feedback. The skeleton-based pose estimations can be used as a second channel to the face recognition system to further increase the reliability of user recognition, or to give feedback on the human worker's posture to optimize ergonomics and comfort over longer periods of time. Furthermore, all these suggestions can already be integrated to optimize the human-robot interactions, even if the system does not take over the full responsibility of a safety system yet.

9.2 Jaco-Assistant: A privacy-aware Voice Interface

The sixth guideline of *RoboJack* suggested making the interactions with the robot more intuitive, which often can be done by making them similar to how a human would interact with a user. Besides gestures, one of the most common ways for humans to interact with each other is through speech. Therefore, a robotic system should also be able to understand and speak natural language to allow for more intuitive interactions.

In the context of the *Tutabo* setup, for example, the system could explain its behavior to users (in accordance with the third guideline, it already tells them if they are too close and therefore prevent the robot from moving again), give workflow instructions to support users in their tasks (optionally automatically if the system detects that they are taking unusually long for a specific action), or let users ask questions about the system or the current task.

Important requirements for such a voice interface include reliable command understanding, low-latency responses, offline operation for privacy reasons, and straightforward integration into an existing robotic system. To meet these requirements, a custom voice interface was developed under the name *Jaco-Assistant*.

This section summarizes my following published work, with selected text passages directly copied from it:

Jaco: An Offline Running Privacy-aware Voice Assistant. Bermuth, Daniel and Poeppel, Alexander and Reif, Wolfgang. March 2022. In *17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. © 2022 IEEE. [46]

9.2.1 Jaco-Assistant

Most smart voice assistants run online as cloud services and are not always known for a good protection of users' privacy. In typical cloud solutions, users have no guarantee that the voice commands sent to online servers are handled safely there. The only option in such cases would be to fully trust the cloud provider. To improve privacy, one important concept is therefore that the assistant can run completely offline. While offline voice assistants have been developed before, they often had other limitations, for example regarding the training process, source-code accessibility, the skill store, supported languages, or host-device resource usage, which are described in more detail in the original paper [46].

The new voice assistant *Jaco* has the following core advantages: (a) It is open-source and can run completely offline. This is in contrast to *Snips* [79] where only the usage was offline, but the training of the assistant had to be done online (after the company's acquisition, their servers were shut down), to *Rhino* [184] which provides a similar cloud-based training, to *Mycroft* [168] which used the online speech recognition service from Google, similar as its open-source successor (the company went bankrupt), and to *Alexa* [33] which runs completely in the cloud. (b) By adding skills to the assistant, it can be easily extended with new features or integrated into other frameworks like *ROS2*. (c) While maintaining an architectural focus on protecting users' privacy, the

developers are not restricted from using all of the host device’s resources. (d) It supports multiple languages, currently German, English, Spanish, and French, and can easily be extended with new ones. (e) It is competitive with other voice assistant solutions and outperforms them in various benchmarks.

The general architecture of a typical voice assistant is shown in Figure 9.2. Usually an interaction starts with a user speaking a *keyword* (here: *computer*) that triggers the assistant to listen to the following spoken request. A *Speech-to-Text* module transcribes the request and the transcription is sent to a *Natural-Language-Understanding* module, which extracts the useful information from the full sentence. The extracted intent and the entities are then handled by a skill, which performs the appropriate action, and informs the user upon success by sending a textual answer to a *Text-to-Speech* service, which answers the user.

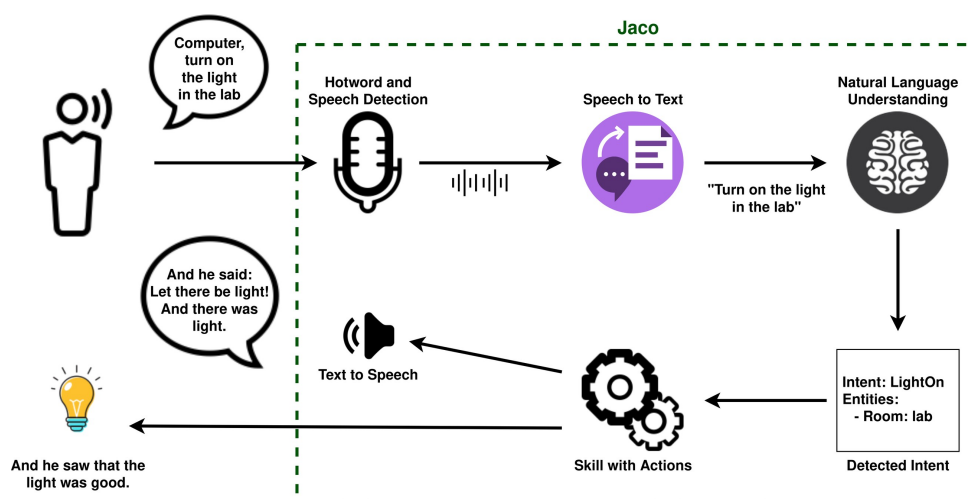


Figure 9.2. General architecture of a voice assistant. [46][© 2022 IEEE].

The architecture of the proposed assistant *Jaco* is presented in greater detail in Figure 9.3. The general architecture is split into two parts: one or multiple lightweight satellite devices, which can be installed in different rooms, and a single central node, which handles the main processing tasks. The example consists of two different *Satellites* (*Alpha*, *Beta*) and the *Master*. Both satellites have a microphone from which the *Audio Streamer* reads the audio input and streams it to the *Wake Word* modules. If they detect the wake word in the stream, they send a message to the *Dialog Manager*. After checking which satellite detected the wake word first, the *Dialog Manager* sends an activation request to the *Speech to Text* module. The transcription of the speech is then started and continues until the user stops speaking. Afterwards, the detected text is sent to the *NLU Parser*, which extracts the intent and sends it back to the *Dialog Manager*. From there, it is sent to the skill that handles the detected intent type. The *skill* can perform various actions (like turning on the light in the lab) and responds with a text message. The *Dialog Manager* then forwards the message to the *Text to Speech* module. There,

the text is converted into an audio file, which is afterwards sent to the *Satellite* specified in the message. The *Audio Streamer* then plays the file to the user through the speaker.

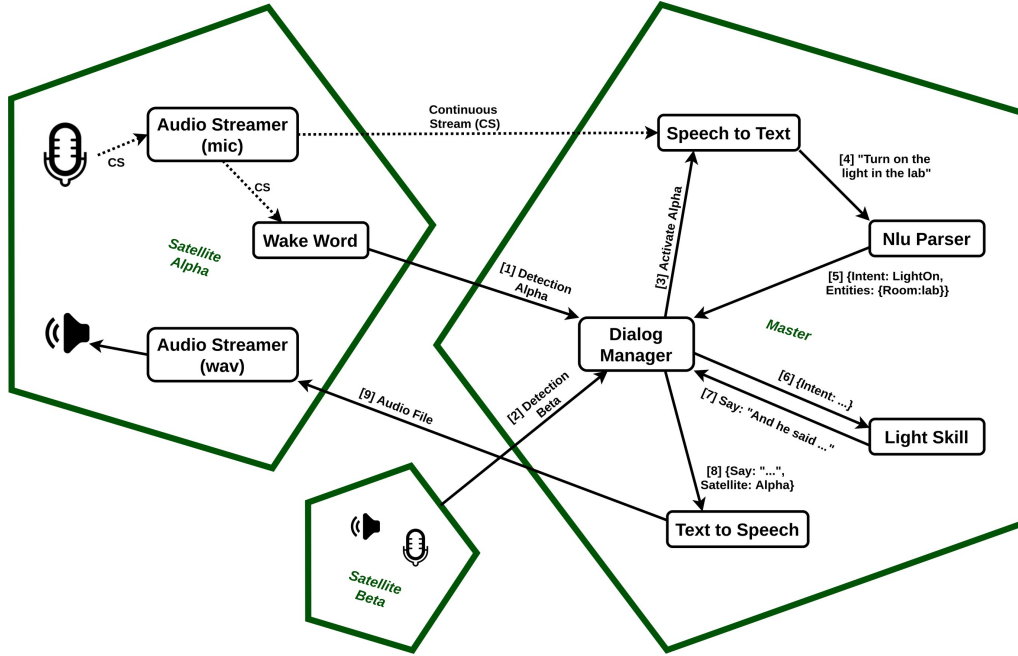


Figure 9.3. Process of a voice interaction using *Jaco*. [46][© 2022 IEEE].

All the modules are container-based and run in their own individual container, which has the advantage of allowing for a more flexible development and easier installation and integration of the system. The module-based approach also allows simple replacement of specific modules with other solutions, which can be useful for researchers to test their current research in a complete system workflow.

For wake-word recognition, *Porcupine* [183] is used. The speech-to-text module uses the pre-trained models from my *Scribosermo* [44] project, which was developed explicitly for use with *Jaco* and, at the time of its development, was one of the best-performing offline speech recognition solutions (please refer to its paper for details). The NLU-parsing is done with *Rasa* [192] and *Duckling* [96], and the text-to-speech generation uses *picotts* [170]. Communication between the modules is executed over MQTT. These solutions were selected because they can all run completely offline on a *RaspberryPi*.

One of the most important features of a smart voice assistant is the possibility to add new capabilities in the form of user-created skills. The architecture of *Jaco* was designed in a way that this is very easy to achieve, and, in contrast to the other assistants, the skills can also access all the hardware resources of the executing device. Besides that, multiple features were implemented to protect privacy when third-party skills are used, for example by encrypting the module communications and implementing a simple permission system. Please refer to the original paper for more details on the architecture and the privacy-related features of *Jaco* [46].

To compare *Jaco*'s performance relative to other voice assistants, the performance of the core *Speech and Language Understanding (SLU)* components, which provide the assistant's key functionality of understanding user commands, was evaluated against different solutions on multiple benchmarks. These experiments, summarized at the end of this section, showed that *Jaco* is competitive with other solutions even though it can also run on low-compute devices like a *RaspberryPi*.

9.2.2 *finstreder*

Jaco allows easy integration of new skills, adding new capabilities to the assistant and allowing a high degree of customizability to users' needs. To improve the understanding reliability, both *STT* and *NLU* modules are optimized for the currently installed skills, and therefore have to be retrained when new skills are added. Since the *Rasa* NLU module, especially on low-compute devices, can take multiple minutes to retrain, this is unfavorable for the user experience and therefore required optimization.

To optimize the retraining time, a new method called *finstreder* was developed, which allows retraining in a few seconds instead of minutes. As the core method, *Finite State Transducers (FSTs)* are used, which are a well-known approach for efficient string processing and have been used in various applications, including speech recognition and natural language processing in the early stages of the field, but have received less attention in recent years due to the rise of deep learning methods.

This section briefly summarizes my following work, with selected text passages directly copied from it:

Finstreder: Simple and fast Spoken Language Understanding with Finite State Transducers using modern Speech-to-Text models. Bermuth, Daniel and Poeppel, Alexander and Reif, Wolfgang. Preprint. June 2022. [45]

The core idea of *finstreder* is that voice commands usually follow a specific structure, which can be efficiently exploited with *FSTs*. Given a voice assistant skill that could help children choose their favorite pet, a developer would define possible user commands in a simple *JSON* format, which is shown in Listing 2. The developer would then add some code to handle the detected intent and entities, upload the skill to the skill store, and the users could then install the skill on their assistant and retrain the assistant to use it.

From those command definitions, multiple *FSTs* are generated, which can then be used to efficiently tag intents and entities that are required for skill execution (Figure 9.4). The output of the *STT* module is a probability distribution over characters or tokens at certain time steps of the input audio. Those can then be used together with the list of possible words defined in the skill to calculate a probability score for each word over time, which then can be combined with the *FSTs* for tagging. The final output of the process is then a list of probable tagged commands, together with their probability scores, which can be used to select the most likely command and execute the corresponding skill action. Please refer to the original paper for more details on this process [45].

```

1 {
2   "intents": {
3     "get-looks": [
4       "(is a|are) [---](animal) cute"
5     ]
6   },
7   "lookups": {
8     "animal": [
9       "whitemargin stargazer",
10      "atlantic stargazer",
11      "aye aye",
12      "(hairy frogfish)->striated frogfish"
13    ]
14  }
15 }

```

Listing 2. Example skill definition.

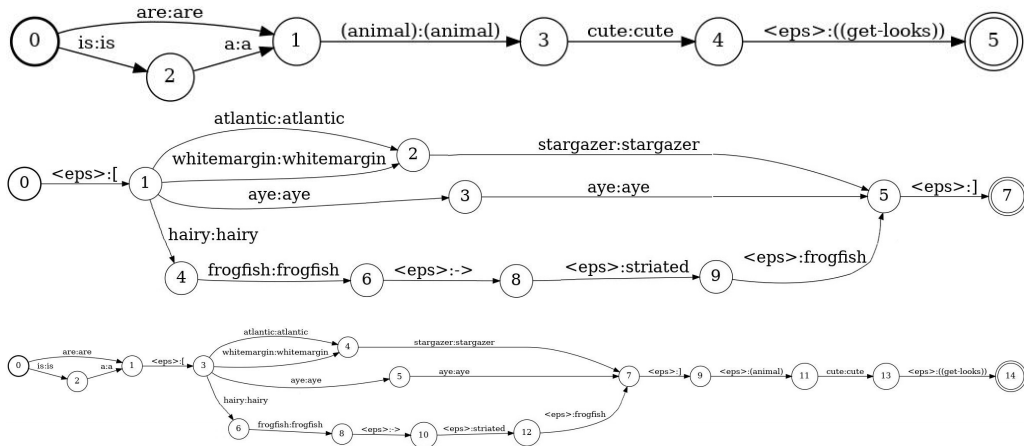


Figure 9.4. Intent and entity tagging with *finstredet*. At first, multiple FSTs are generated from the skill definition. The first one on top checks whether a list of words matches the defined intent structure, and the second one checks the presence of entities. Both FSTs are then merged into one combined FST (by inserting the second one into the first one at the *animal* transition), resulting in the third FST. Now a user command like “is a whitemargin stargazer cute” can be processed by the merged FST to check and tag it. Starting at state (0), the first transition matches the word “is”, because it is the same as the part on the left of the “:”, so the next state is (2). Then, with the next word, it gets to state (1) and automatically to state (3) because of the condition-free epsilon transition. This process is continued until the end of the command, and if the last state is a final state, the command is valid. The output of the process is then the combination of the parts right of the “:” in the matched transitions, which in this case would be “is a [whitemargin stargazer](animal) cute ((get-looks))”, from which the intent and entities can easily be extracted. While this example considers transitions from words to tagged sentences (the key element for command understanding), it is important to note that the speech model does not output words directly, but instead probabilities for characters or tokens. Technically, each word transition in the figure is another nested graph, encoding the paths from tokens to words. Please refer to the original paper for more details on this [45].

9.2.3 slungt

Even though *finstreder* showed outstanding results in the benchmarks, and is faster than most other alternative solutions, there is still potential for improvement. The *FST*-based approach of *finstreder* basically builds a graph of possible outputs and a second graph of most probable input tokens, using the token probabilities as transition costs. Then it combines the two graphs into a single very large graph and searches for the shortest path in it, which is the most probable command.

Instead of creating a single large graph and searching for the best path in it, *slungt* uses a *beam-search* approach to find the best path step by step, separating the problem along the time axis. This results in a much smaller number of possible paths which need to be checked, because bad paths can be discarded earlier.

This section briefly summarizes my following work, with selected text passages directly copied from it:

Slungt: Even Faster Spoken Language Understanding with N-Grams and Tries. Bermuth, Daniel and Poeppel, Alexander and Reif, Wolfgang. April 2025. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. © 2025 IEEE. [43]

In the decoding process, a *trie*-based data structure is used to check whether a word suggested by the speech-to-text model is part of the vocabulary of a certain intent. A *trie* is a tree-like data structure where each node represents a single character, and its children are the next characters of the word. The last character of a word is marked as a leaf node. See Figure 9.5 for an example.

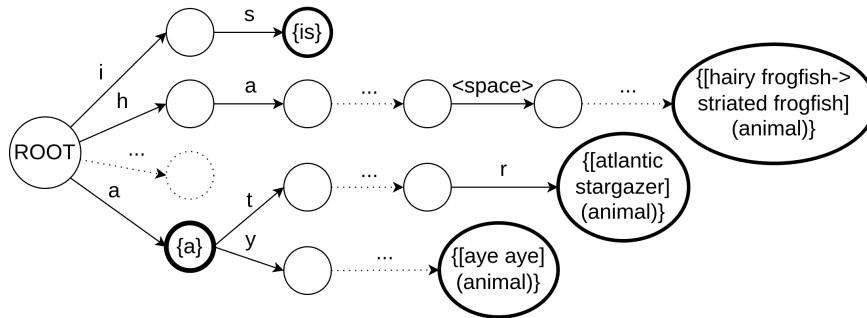


Figure 9.5. Entity tagging with *slungt*. Given a list of characters, the trie is traversed starting from the root node, following the transitions by those characters. If a leaf node is reached with the last character of the suggestion, the word is valid and complete. If only a non-leaf node was reached, the suggestion is not complete yet. If a node does not have a child with the next character of the suggestion, the suggestion is invalid. In contrast to the standard procedure, a leaf node does not return a single word, but instead a set of word options, which can also include the entity tags. [43][© 2025 IEEE].

The input of the decoding process is the token probabilities for each time-step from the speech-to-text model. The process starts with one empty beam per intent. Then, for each time-step, the following steps are executed:

1. For each token with a probability above a threshold, a new beam is created from each of the previous beams.
2. For each new beam, the new token is added to the beam's temporary text.
3. All beams with the same content are merged.
4. Each temporary text is evaluated with the word trie of the corresponding intent.
5. If a new word is found, it is scored with the intent's n-gram model and added to the beam's transcription. Entity values are scored as their entity type.
6. The number of beams is pruned to the maximum beam-size, keeping only the best ones.

At the end of the decoding process, unfinished beams are penalized. The beams are sorted again and then returned. The one with the highest score is considered the final result. The intent was already stored as an attribute while creating the initial beams, and the slots are extracted from the beam's transcription. Please refer to the original paper for more details on the overall process [43].

9.2.4 Benchmarks

To compare the performance of the new methods of *Jaco*, *finstreder*, and *slungt* relative to other Speech and Language Understanding (SLU) solutions, multiple benchmarks with these and some other methods were performed, which are summarized in the following. For details on the different model variants that were tested, and the exact training and evaluation process, please refer to the original papers [43, 45, 46].

A command is considered correctly predicted if the intent and all entities are correctly detected, otherwise, it is counted as an error. The *Word Error Rate (WER)* measures the percentage of incorrectly predicted words in the transcribed text, and the *Real Time Factor (RTF)* measures the time it takes to process the audio input relative to the duration of the audio itself, which defines the response latency of the system, and therefore also is an important aspect for the user experience.

The *SmartLights* benchmark from *Snips* [201] tests the ability to control lights in different rooms. It consists of 1660 requests which are split into five partitions for a 5-fold evaluation. A sample command could be: “*please change the [bedroom] lights to [red]*” or “*i'd like the [living room] lights to be at [twelve] percent*”. The benchmark results are presented in Table 9.1. The original approach of *Jaco* was on par with the best performing approaches at the time, even though it could run completely offline on a *RaspberryPi*. With the later *finstreder* and *slungt* methods, the performance could be further improved. While the approach of *finstreder* is slightly better than *slungt* in terms of accuracy, the approach of *slungt* is notably faster due to the earlier discarding of bad decoding paths. The *Real Time Factor* (decoding-time / audio-duration) was measured on a desktop computer with an *AMD-3700X* CPU, no GPU, in an end-to-end (audio-to-intent) manner.

The *SmartSpeaker* benchmark tests the performance of reacting to music player commands in English as well as in French. The benchmark is from *Snips* [201] too, and is the only one found that includes a language other than English. It has the difficulty

Method	Accuracy	WER	RTF
<i>Google</i> [201]	0.793	—	—
<i>Snips</i> [201]	0.842	—	—
<i>Alexa</i> [46]	0.792	—	—
<i>Houndify</i> [46]	0.545	0.108	—
<i>Jaco</i> [46]	0.854	0.108	—
<i>AT-AT</i> [197]	0.849	—	—
<i>Lugosch et al.</i> [153]	0.714	—	—
<i>Finstreder</i> [43]	0.902	0.075	0.283
<i>Slungt</i> [43]	0.893	0.051	0.062

Table 9.1. Results on the *SmartLights* dataset. [43][© 2025 IEEE].

of containing many artist or music tracks with uncommon names in the commands, like “play music by [a boogie wit da hoodie]” or “i’d like to listen to [Kinokoteikoku]”. As shown in Table 9.2, the performance of *Jaco* on this benchmark was again on par with the best performing approaches at the time, and is further improved with *finstreder* and *slungt*. The performance improvement of *finstreder*’s SLU approach compared to *Jaco*’s STT+NLU concept can be explained to some extent by a different handling of the artist names in the *n*-gram language models. The *n*-grams of *Jaco* directly include the names, which allows the parser to leave out some parts of the names or mix them up, whereas the ones of *finstreder* only have a placeholder which then has to be matched exactly. One reason why *slungt* does not completely match *finstreder* is that the latter is able to correct errors early in the name if the complete match improves at the end, whereas *slungt* might have already pruned such a beam.

Method	English	French
<i>Snips</i> [201]	0.687	0.751
<i>Google</i> [201]	0.478	0.423
<i>Jaco</i> [46]	0.627	0.480
<i>Alexa</i> [46]	0.455	0.889
<i>Finstreder</i> [43]	0.879	0.865
<i>Slungt</i> [43]	0.847	0.868

Table 9.2. Accuracy on the *SmartSpeaker* dataset. [43][© 2025 IEEE].

FluentSpeechCommands [152] is the most commonly used benchmark in this domain and tests simple voice assistant requests. It includes commands like “turn up the [bathroom] temperature”, “switch the lights on” or “go get me my [shoes]”. Table 9.3 shows that my new *slungt* and *finstreder* methods are competitive with the best performing approaches on this benchmark, even though in their default versions they were not trained with the audio files on this dataset, unlike most of the other approaches. Similar to before, *slungt* slightly falls behind *finstreder* again, as it cannot force match the input tokens to command options.

Most of the other approaches did not provide any information about the hardware used, which suggests that resource efficiency was not a highlightable feature, but the ones that did, used high-end GPUs, whereas *finstreder* and *slungt* can run on a *RaspberryPi* and are

Methods	Accuracy
<i>Alexa</i> [45]	0.987
<i>Cao et al.</i> [65], <i>Radfar et al.</i> [189], <i>Benazir et al.</i> [41]	0.990
<i>Slungt (sp / cb)</i> [43]	0.983 / 0.992
<i>Reptile</i> [228], <i>CMMC</i> [265]	0.992
<i>AT-AT</i> [197]	0.995
<i>Finstreder (sp / cb)</i> [45]	0.995 / 0.994
<i>Borgholt et al.</i> [56]	0.996
<i>Seo et al.</i> [208], <i>Qian et al.</i> [187], <i>Kim et al.</i> [131]	0.997
<i>Slungt (cb) + AMT</i> [43]	0.997
<i>UniverSLU</i> [36]	0.998
<i>Finstreder (cb) + AMT</i> [45]	0.998

Table 9.3. Accuracy on the FSC dataset. [43][© 2025 IEEE].

optimized in a few seconds. After using the audio files of *FSC* for training as well (the *AMT* model [45], finetuned with training audio for around 3h on one Nvidia-1080Ti), *finstreder* could set a new state-of-the-art value on this benchmark. Technically it also reached the best possible accuracy, because the remaining accuracy error is due to audio files with wrong labels (see linked repository for details [45]).

The original papers of *Jaco* [46], *finstreder* [45] and *slungt* [43] include more results with outstanding performance on further benchmarks, which are skipped here for brevity reasons, as well as further ablation studies on the different model variants, which are also not included here, and runtime results on the *RaspberryPi*, so please refer to the original papers for more details.

9.2.5 Conclusion

This section presented my work on implementing a privacy-aware voice assistant, which can run completely offline on low-compute devices, and two methods for optimizing the spoken language understanding process of such an assistant. *Jaco* provides all the core capabilities of a modern voice assistant, and can be easily extended with new skills, which can be shared with other users through a skill store.

The new methods *finstreder* and *slungt* optimized the spoken language understanding process of the assistant, which basically is its core functionality, and achieved outstanding results on multiple benchmarks, even though they can run on low-compute devices and do not require any training with the audio files of those benchmarks.

The presented methods are especially interesting for use-cases that have frequent domain changes, relatively small datasets, or restricted training possibilities. For example, this could be the case in customizable smart home assistants running on edge devices or on smartphones, where using a cloud service is undesirable, for example to be independent of unstable internet connections. Running everything offline on the user's local device ensures that users don't have to trust their cloud provider that voice commands or accidentally recorded sounds (if the wake-word was triggered through a false positive) are handled safely there.

To support further research in this domain, or the integration of the presented methods into other systems, the source-code of *Jaco*, *finstredder* and *slungt* was made publicly available¹.

In the context of the *Tutabo* setup, a voice assistant like *Jaco* can be used to implement more intuitive interactions with users or better explain the system's behavior. Besides the features already mentioned at the beginning of this section, users could give the robot instructions to do certain tasks (like "*please fetch the next package*"), or directly adjust the robot poses (like "*please rotate the bike by ten degrees*"). Furthermore, it could be used as a tool to better personalize the interactions with users, in correspondence with the seventh guideline from *RoboJack*.

From a safety perspective, however, it is questionable whether a voice assistant should be used to give safety-related commands to the robot, for example via a "*stop*" command. The reliability of the detection of the command would have to be verified as well, which, due to the complexity of the involved AI models, would be a very difficult task. How would a simulated test of billions of "*stop*" calls be designed? What happens under loud background noise? Would the robot need to stop because the command does not work then? Does this effort make sense for a single voice command? A plausible compromise could be to offer convenience commands like "*please wait*", which do not sound as safety-critical, but still offer the same functionality. Such a solution would not create a false sense of safety for users, who might otherwise rely on the existence of a "*stop*" command to stop the robot in case of an emergency, but then find out that it does not work in such a situation, which could lead to dangerous situations. Therefore, the suggested solution would be to use the voice assistant only for convenience features to improve the robot interactions and optimize the user experience.

¹<https://gitlab.com/Jaco-Assistant>

9.3 Crazyflie Swarms: Beyond Industrial Robots

While the previous chapters and sections mainly focused on human-robot collaboration in industrial settings, the underlying system modules can also be used in many other contexts. An example in which the pose estimation method was already integrated is the control and interaction with small drones (Figure 9.6).

In particular, a set of small *Crazyflie* drones [54] that create an interactive and self-organizing swarm formation [158] can be hand-guided by a person, using the hand position estimated by *RapidPoseTriangulation*. The pose estimation system can replace the previously used handheld stick with markers on top and thus allows for more intuitive control and interaction with the drones.

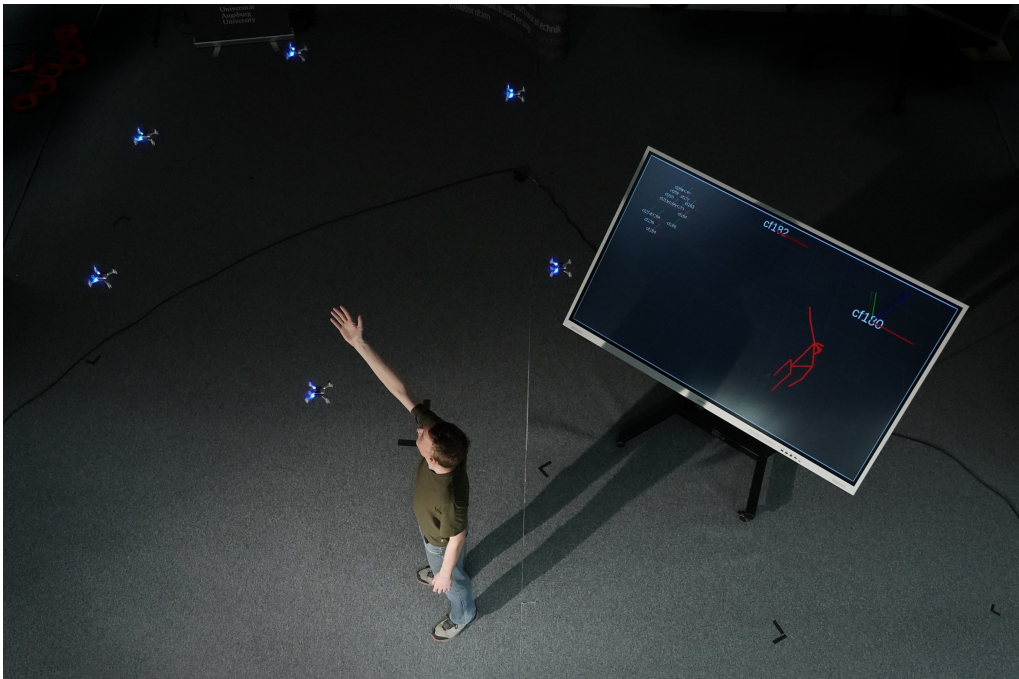


Figure 9.6. Hand-guiding a swarm of small drones. The drones build a cycle formation centered around the right hand of the person. The person can now walk around and move his hand freely, while the drones follow the hand movement and maintain the formation.

While currently only a simple position-based control is implemented, it is planned for future work to also include more sophisticated gesture and action recognition methods to allow for more complex interactions with the drones. Additionally, the voice assistant *Jaco* could be integrated to allow using voice commands to control the drones, for example to change the swarm formation. Furthermore, the complete *Tutabo* system is planned to be integrated again for its original purpose, by providing safety measures to protect the persons and avoid collisions between the drones and humans in the area. This shows that the developed system modules are not limited to industrial use-cases only, but can also be easily integrated in many other contexts, which is an important aspect to support the long-term continuous improvement of the system.

Summary. This chapter summarizes the research results obtained in this work.

10

Conclusion

This thesis presented an AI-based safety system for human-robot collaboration, which enables accurate real-time detection and localization of human workers in the collaborative workspace. Due to its ability to create dynamic safety zones around both the robot and the human worker and to distinguish between different body parts, the system allows for more efficient and closer human-robot interaction.

The proposed system was developed in an iterative process. The *KoARob* prototype demonstrated the general feasibility of using AI-based methods to detect humans and stop the robot when the distance between them falls below a defined threshold. In a realistic application scenario, the system showed that, even though only evaluated over a short time so far, it can reliably detect persons and their joints in the workspace, which is a key requirement for a new type of safety system. In *Tutabo-1*, the system's efficiency was significantly improved, and it was shown that the system can achieve real-time responses, while still reliably detecting the human's body parts and stopping the robot in time. The safety system was further optimized in *Tutabo-2*, demonstrating that an AI-based system can reach a latency comparable to that of a light curtain, which is among the fastest safety-certified systems, even though the complexity of the new system is much higher and it provides more information.

With the proposed *interaction-zones*, the system can selectively approve contacts and interactions between the robot and certain body parts at certain positions, for example in hand-over scenarios. This allows the merging of the concepts of *speed and separation monitoring* and *power and force limiting*, which are currently the two main concepts for safety in human-robot collaborative applications. As a result, this can notably increase system productivity while still ensuring the safety of the workers, and new application scenarios become feasible.

One of the main remaining challenges is to prove the reliability of the system over a longer period of time, which is required to reach a certifiable performance level. The outlook in *Tutabo-2* outlines multiple suggestions toward this goal. In this context, the importance of following an iterative development process, as applied in *Tutabo*, should be emphasized. It allows improving the system step by step, with a focus on creating intermediate results that are directly usable and provide practical benefit, which is essential for a task as complex as developing a certifiable AI-based safety system. In software engineering practice, iterative processes such as *Scrum* have been widely adopted for similar reasons already and have proven highly effective.

The research for *Tutabo* demonstrated the general feasibility of using AI-based methods for safety-related applications. Although a safety certification requires further work, many of the obtained results are already useful in other contexts. With *RapidPoseTriangulation*, the fastest currently available multi-view multi-person whole-body human pose estimation method was developed. The choice of detecting persons as 3D skeletons provides several advantages, as they are intuitive to interpret, transferable to other application domains, and support explainability in safety-critical systems. The method can therefore also be used in a wide range of applications beyond the safety context, for example to control swarms of drones. In *Skarimva* it was demonstrated that using *RapidPoseTriangulation* to predict human skeleton sequences significantly improved the current state-of-the-art on the task of human action recognition. Besides achieving lead-

ing performance in human pose forecasting, the experiments with *Scriboora* additionally highlighted the importance of reliable human pose estimation for forecast accuracy. These methods can then be used in a next iteration to further improve the usability and efficiency of the *Tutabo* system. For example, by leveraging predicted human movements, the robot’s trajectory can be optimized to further increase the productivity and cost-effectiveness of the system. The *RoboJack* project then highlighted the importance of interaction quality for increasing user acceptance of the system, which is a crucial aspect for its success. Several of the methods presented in this thesis can also be used to improve the interactions according to the guidelines proposed in *RoboJack*.

In this context of improving interactions, the voice assistant *Jaco* was developed to provide the system with an additional way to communicate with the users. *Jaco* offers all the core functionalities of a modern voice assistant, but can run completely offline on resource-constrained devices and has a focus on privacy. The new methods *fnstreder* and *slungt* then optimized the spoken language understanding process of the assistant, achieving leading performance in command understanding on multiple benchmarks.

In the introduction of this thesis, where the main goals (robot agnosticism, tool and workpiece safety, dynamic safety zones, distance measurements, body-part specific reactions, and safety certification) for the new safety system were defined (Section 1.3), a bike assembly scenario was used as an example to illustrate the relevance of these goals. Figure 10.1 shows the same scenario again, but now from the system’s perspective, illustrating their successful implementation.

To support further research and facilitate integration of the presented methods into other systems, the source-code of all developed methods has been made publicly available¹. Overall, the presented work demonstrates that AI-based safety systems are a viable and promising approach for efficient human-robot collaboration, offering practical advantages over existing approaches. Beyond its immediate safety contribution, the presented system provides an enabling technology for advanced human-centric smart factories, where flexibly organized robot-based automation can be supported by human workers, allowing the production of goods that robots alone cannot yet manufacture.

¹<https://gitlab.com/Percipiote>

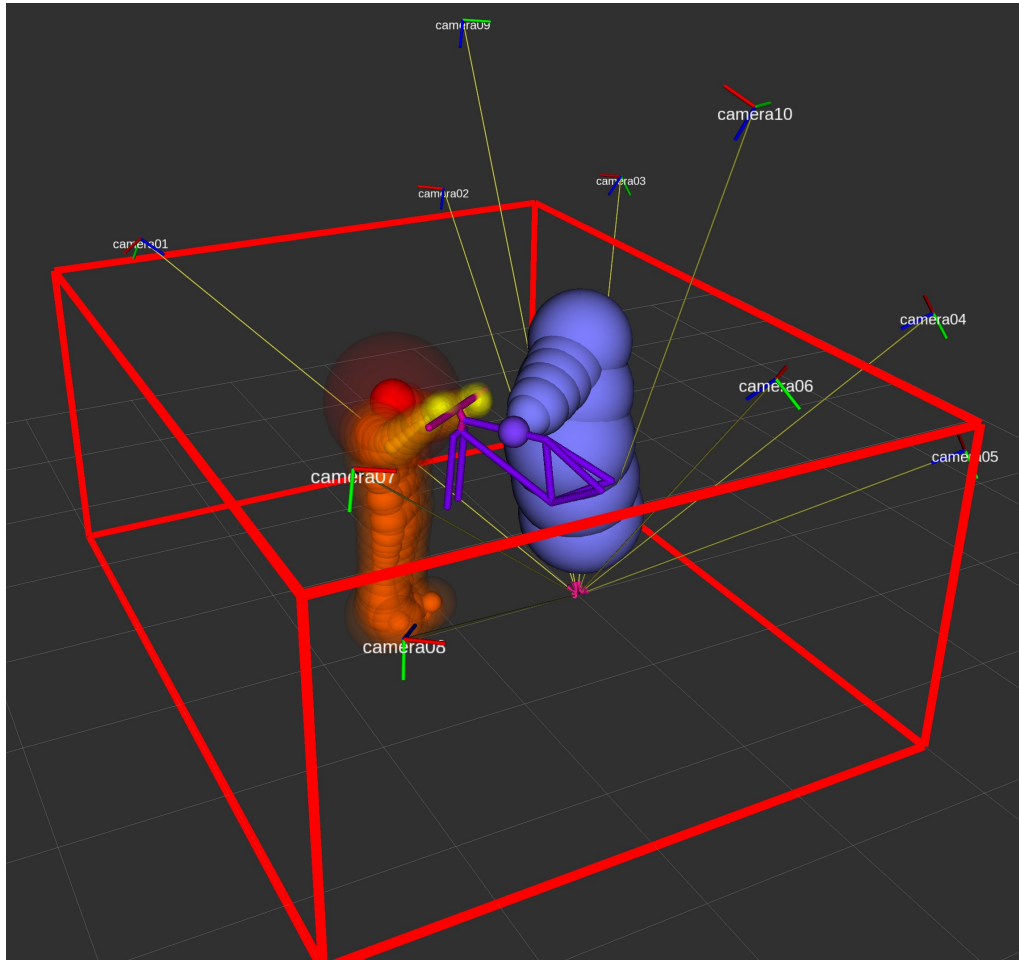


Figure 10.1. The motivating scenario from the introduction of this thesis from the system’s perspective. Using multiple cameras, the proposed AI-based safety system can reliably detect the human worker and his body parts, and then can create dynamic safety zones around the robot, the workpiece, and the human. If necessary, the system can selectively approve interactions, for example, allowing the worker to touch the handlebar of the bike frame, while stopping the robot when anything else gets too close to the worker.

Bibliography

- [1] CES 2026: Blackjack Dealing Robot (Real Time Speed, Dual Camera), . URL <https://www.youtube.com/watch?v=sKN0620CRSA>. Accessed: 13-April-2026.
- [2] Robot Blackjack Dealer - Ai11, . URL https://www.youtube.com/watch?v=Thm_hGRFpZA. Accessed: 13-April-2026.
- [3] Robot blackjack at automate 2013, . URL https://www.youtube.com/watch?v=bNe93_9jmcY. Accessed: 13-April-2026.
- [4] AIRSKIN. URL <https://airskin.io/products/airskin-modules.html>. Accessed: 12.03.2026.
- [5] Bosch APAS: Intelligent systems for the flexible factory 4.0. URL <https://www.bosch.com/stories/industry-4-0-flexible-production-line/>. Accessed: 13.03.2026.
- [6] FANUC Collaborative Robot Series. URL <https://www.fanucamerica.com/products/robots/series/collaborative-robot>. Accessed: 12.03.2026.
- [7] KUKA UserTech KCP. URL https://www.kuka.com/en-my/products/robotics-systems/software/simulation-planning-optimization/kuka_usertech. Accessed: 12.03.2026.
- [8] Mayser Safety Mats. URL <https://www.mayser.com/en/safetytechnology/surfacesensors>. Accessed: 12.03.2026.
- [9] PILZ PSENradar Safe Radar System. URL <https://www.pilz.com/en-INT/products/sensor-technology/safe-radar-systems/psenradar-safe-radar-system>. Accessed: 12.03.2026.
- [10] SICK Safe Robotics Area Protection. URL <https://www.sick.com/hk/en/catalog/products/safety/safety-systems/safe-robotics-area-protection/c/g463851>. Accessed: 12.03.2026.
- [11] Sick wire draw encoders. URL <https://www.sick.com/de/en/catalog/products/motion-control-sensors/wire-draw-encoders/c/g286652>.
- [12] TOF example image. URL <https://www.e-consystems.com/images/eCAM/tof-camera/tof-camera-sample-image.png>. Accessed: 12.03.2026.
- [13] Neunte Verordnung zum Produktsicherheitsgesetz (Maschinenverordnung) (9. ProdSV), 1993. https://www.gesetze-im-internet.de/gsgv_9/9._ProdSV.pdf.
- [14] Directive 2006/42/EC of the European Parliament and of the council - of 17 May 2006 - on machinery, and amending Directive 95/16/EC, 2006. 63 pages, <https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32006L0042>.
- [15] What Mocap Suit Suits You?, 2019. URL <https://vfxvoice.com/what-mocap-suit-suits-you/>. Accessed: 25.02.2026.
- [16] DGB-Hochschulreport - Arbeits- und Beschäftigungsbedingungen an Hochschulen in Deutschland. Deutscher Gewerkschaftsbund, 2020.
- [17] Using Intel FPGAs in Veo FreeMove System to Enable Safe, Dynamic, and Flexible Human-Robot Collaboration, 2020. URL <https://www.intel.com/content/dam/www/central-libraries/us/en/documents/fpgas-in-veo-freemove-case-study.pdf>. Accessed: 25.02.2026.

- [18] Abschlussbericht. *KI Absicherung: Safe AI for Automated Driving*, 2022. URL <https://www.ki-absicherung-projekt.de/en/>.
- [19] Regulation (EU) 2023/1230 of the European Parliament and of the council - of 14 June 2023 - on machinery and repealing Directive 2006/42/EC of the European Parliament and of the Council and Council Directive 73/361/EEC, 2023. 102 pages, <https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32023R1230>.
- [20] Regulation (EU) 2024/1689 of the European Parliament and of the council - of 13 June 2024 - laying down harmonised rules on artificial intelligence and amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and Directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act), 2024. 144 pages, https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=OJ:L_202401689.
- [21] Regulation (EU) 2024/1689 of the European Parliament and of the council - of 23 October 2024 - on horizontal cybersecurity requirements for products with digital elements and amending Regulations (EU) No 168/2013 and (EU) No 2019/1020 and Directive (EU) 2020/1828 (Cyber Resilience Act), 2024. 81 pages, https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=OJ:L_202402847.
- [22] Safety of machinery - Positioning of safeguards with respect to the approach of the human body (ISO 13855:2024), 2024. 95 pages.
- [23] rc-visard - Smart 3D Stereo Sensor - Technical Specifications. *roboception GmbH*, 2024. https://roboception.com/wp-content/uploads/Datasheet-rc_visard-24.05-web.pdf.
- [24] safe.train. *safe.train: Safe AI using a driverless regional train as an example*, 2024. URL <https://safetrain-projekt.de/en/>.
- [25] Freemove. *Veo Robotics Inc*, 2024. <https://www.veobot.com/freemove>, <https://www.veobot.com/blog/2020/10/16/how-collaborative-is-your-robot-a-practical-approach>.
- [26] Basler ace2 - a2A1920-160ucBAS, 2025. <https://www.baslerweb.com/de-de/shop/a2a1920-160ucbas/>, <https://www.baslerweb.com/de-de/tools/frame-rate-calculator/#camera=a2A1920-160ucBAS>.
- [27] Bayer filter, 2026. https://en.wikipedia.org/wiki/Bayer_filter, https://en.wikipedia.org/wiki/Bayer_filter#/media/File:Bayer_pattern_on_sensor.svg, https://en.wikipedia.org/wiki/Bayer_filter#/media/File:Bayer_pattern_on_sensor_profile.svg, https://en.wikipedia.org/wiki/Bayer_filter#/media/File:Colorful_spring_garden_Bayer+_RGB.png, Accessed: 10.03.2026.
- [28] Camera Calibration, 2026. https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html, https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html, Accessed: 25.03.2026.
- [29] calibrateHandEye, 2026. URL https://docs.opencv.org/4.5.4/d9/d0c/group__calib3d.html#gaebfc1c9f7434196a374c382abf43439b. Accessed: 25.03.2026.
- [30] ABB Robotics. Product specification - Robot stopping distances according to ISO 10218-1, 2025.
- [31] K. C. Alowonou and J.-H. Han. MSA-GCN: Exploiting Multi-Scale Temporal Dynamics With Adaptive Graph Convolution for Skeleton-Based Action Recognition. *IEEE Access*, 2024.

- [32] L. M. Amaya-Mejía, N. Duque-Suárez, D. Jaramillo-Ramírez, and C. Martinez. Vision-based safety system for barrierless human-robot collaboration. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7331–7336. IEEE, 2022.
- [33] Amazon. Alexa, 2021. URL <https://developer.amazon.com/en-US/alexa>. [accessed 23-March-2021].
- [34] A. Amorim, D. Guimares, T. Mendona, P. Neto, P. Costa, and A. P. Moreira. Robust human position estimation in cooperative robotic cells. *Robotics and Computer-Integrated Manufacturing*, 67:102035, 2021.
- [35] L. Antao, J. Reis, and G. Gonçalves. Voxel-based space monitoring in human-robot collaboration environments. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 552–559. IEEE, 2019.
- [36] S. Arora, H. Futami, J.-w. Jung, Y. Peng, R. Sharma, Y. Kashiwagi, E. Tsunoo, and S. Watanabe. UniverSLU: Universal Spoken Language Understanding for Diverse Classification and Sequence Generation Tasks with a Single Network. *arXiv preprint arXiv:2310.02973*, 2023.
- [37] Automation.at. Pilz SafetyEYE: Dreidimensionale Freiheit. URL https://www.automation.at/bericht/sicherheitskameras/safetyeye_macht_pressen_und_stanzen_sicher-2017-05-09. Accessed: 12.03.2026.
- [38] G. Barbosa, C. Ledebour, A. T. de Oliveira Filho, I. R. Rodrigues, D. Sadok, J. Kelner, and R. Souza. Hosa: An end-to-end safety system for human-robot interaction. *Journal of Intelligent & Robotic Systems*, 105(4):95, 2022.
- [39] K. Bartol, D. Bojanić, T. Petković, and T. Pribanić. Generalizable human pose triangulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11028–11037, 2022.
- [40] V. Belagiannis, S. Amin, M. Andriluka, B. Schiele, N. Navab, and S. Ilic. 3D pictorial structures for multiple human pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1669–1676, 2014.
- [41] A. Benazir, Z. Xu, and F. X. Lin. Speech Understanding on Tiny Devices with A Learning Cache. In *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services*, pages 425–437, 2024.
- [42] D. Bermuth. Methodiken zur Analyse und Verbesserung der Interaktionsqualität in Mensch-Roboter-Kollaborationsanwendungen (Methods for analyzing and improving the interaction quality in human-robot collaboration applications), 2020. Master’s thesis, 18. October 2020, University of Augsburg. German. <https://www.youtube.com/watch?v=vYSBguVT54o>, Accessed: 13-April-2026.
- [43] D. Bermuth and W. Reif. Slungt: Even Faster Spoken Language Understanding with N-Grams and Tries. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2025.
- [44] D. Bermuth, A. Poeppel, and W. Reif. Scribosermo: Fast Speech-to-Text models for German and other Languages. *arXiv preprint arXiv:2110.07982*, 2021.
- [45] D. Bermuth, A. Poeppel, and W. Reif. Finstreder: Simple and fast spoken language understanding with finite state transducers using modern speech-to-text models. *arXiv preprint arXiv:2206.14589*, 2022.

- [46] D. Bermuth, A. Poeppel, and W. Reif. Jaco: An Offline Running Privacy-aware Voice Assistant. In *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 618–622. IEEE, 2022.
- [47] D. Bermuth, A. Poeppel, and W. Reif. VoxelKeypointFusion: Generalizable Multi-View Multi-Person Pose Estimation. *arXiv preprint arXiv:2410.18723*, 2024.
- [48] D. Bermuth, A. Poeppel, and W. Reif. KoARob: Towards AI-based Safety in Human-Robot-Collaborations. In *2025 IEEE 21st International Conference on Automation Science and Engineering (CASE)*, pages 837–844. IEEE, 2025.
- [49] D. Bermuth, A. Poeppel, and W. Reif. Scriboora: Rethinking human pose forecasting. *arXiv preprint arXiv:2511.15565*, 2025.
- [50] D. Bermuth, A. Poeppel, and W. Reif. SimpleDepthPose: Fast and Reliable Human Pose Estimation with RGBD-Images. *arXiv preprint arXiv:2501.18478*, 2025.
- [51] D. Bermuth, A. Poeppel, and W. Reif. Tutabo-1: Towards real-time capable ai-based safety systems for human-robot collaboration. In *2025 IEEE International Conference on Advanced Robotics (ICAR)*, pages 632–639. IEEE, 2025.
- [52] D. Bermuth, A. Poeppel, and W. Reif. RapidPoseTriangulation: Multi-view Multi-person Whole-body Human Pose Triangulation in a Millisecond. *arXiv preprint arXiv:2503.21692*, 2025.
- [53] D. Bermuth, A. Poeppel, and W. Reif. Skarimva: Skeleton-based action recognition is a multi-view application. In *2026 IEEE 20th International Conference on Automatic Face and Gesture Recognition (FG)*, pages 1–5. IEEE, 2026.
- [54] Bitcraze AB. Bitcraze - Homepage. URL <https://www.bitcraze.io/>.
- [55] M. Boldo, M. De Marchi, E. Martini, S. Aldegheri, D. Quaglia, F. Fummi, and N. Bombieri. Real-time multi-camera 3d human pose estimation at the edge for industrial applications. *Expert Systems with Applications*, 252:124089, 2024.
- [56] L. Borgholt, J. D. Havtorn, M. Abdou, J. Edin, L. Maaløe, A. Søgaaard, and C. Igel. Do We Still Need Automatic Speech Recognition for Spoken Language Understanding? *arXiv preprint arXiv:2111.14842*, 2021.
- [57] A. Bouazizi, A. Holzbock, U. Kressel, K. Dietmayer, and V. Belagiannis. MotionMixer: MLP-based 3D Human Body Pose Forecasting. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 791–798. International Joint Conferences on Artificial Intelligence Organization, 7 2022.
- [58] S. Brending, M. Lawo, J. Pannek, T. Sprodownski, P. Zeising, and D. Zimmermann. Certifiable software architecture for human robot collaboration in industrial production environments. *IFAC-PapersOnLine*, 50(1):1983–1990, 2017.
- [59] L. Bridgeman, M. Volino, J.-Y. Guillemaut, and A. Hilton. Multi-person 3d pose estimation and tracking in sports. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019.
- [60] J. Brooke et al. SUS - A quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.
- [61] S. Bultmann and S. Behnke. Real-time multi-view 3D human pose estimation using semantic feedback to smart edge sensors. In *Proceedings of Robotics: Science and Systems (RSS)*, 2021.

- [62] Bundesanstalt für Arbeitsschutz und Arbeitsmedizin (Federal Institute for Occupational Safety and Health). Maschinen mit Künstlicher Intelligenz: Anforderungen an Hersteller nach der neuen EU-MVO und Bezüge zur KI-VO (Machines with Artificial Intelligence: Requirements for Manufacturers Under the New EU Machinery Regulation and Links to the AI Regulation), 2025. German, <https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32023R1230>.
- [63] C. Byner, B. Matthias, and H. Ding. Dynamic speed and separation monitoring for collaborative robot applications—concepts and performance. *Robotics and Computer-Integrated Manufacturing*, 58:239–252, 2019.
- [64] L. Cao, S. Huai, and J. Gai. Reenvisioning Skeleton-based Action Recognition Through the Lens of NLP. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2025.
- [65] Y. Cao, N. Potdar, and A. R. Avila. Sequential End-to-End Intent and Slot Label Classification and Localization. In *Proc. Interspeech 2021*, pages 1229–1233, 2021. doi: 10.21437/Interspeech.2021-1569.
- [66] Carnegie Mellon University. CMU Graphics Lab Motion Capture Database. URL <http://mocap.cs.cmu.edu/>.
- [67] M. Carraro, M. Munaro, J. Burke, and E. Menegatti. Real-time marker-less multi-person 3D pose estimation in RGB-depth camera networks. In *Intelligent Autonomous Systems 15: Proceedings of the 15th International Conference IAS-15*, pages 534–545. Springer, 2019.
- [68] G. S. casino equipment. Card Dealing Robot: Blackjack for online casino. URL <https://www.youtube.com/watch?v=aT7ExmC0G3E>. Accessed: 13-April-2026.
- [69] E. Chebotareva, M. Mustafin, R. Safin, T. Tsoy, E. A. Martinez-García, H. Li, and E. Magid. Camera-based safety system for collaborative assembly. *Journal of Intelligent Manufacturing*, 36(8):5593–5611, 2025.
- [70] L. Chen, H. Ai, R. Chen, Z. Zhuang, and S. Liu. Cross-view tracking for multi-human 3d pose estimation at over 100 fps. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [71] Y. Chen, Z. Zhang, C. Yuan, B. Li, Y. Deng, and W. Hu. Channel-wise topology refinement graph convolution for skeleton-based action recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 13359–13368, 2021.
- [72] B. Cheng, B. Xiao, J. Wang, H. Shi, T. S. Huang, and L. Zhang. Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5386–5395, 2020.
- [73] A. Cherubini, R. Passama, A. Crosnier, A. Lasnier, and P. Fraisse. Collaborative manufacturing with physical human–robot interaction. *Robotics and Computer-Integrated Manufacturing*, 40:1–13, 2016.
- [74] H.-g. Chi, M. H. Ha, S. Chi, S. W. Lee, Q. Huang, and K. Ramani. InfoGCN: Representation learning for human skeleton-based action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20186–20196, 2022.
- [75] R. Choudhury, K. M. Kitani, and L. A. Jeni. TEMPO: Efficient multi-view pose estimation, tracking, and forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14750–14760, 2023.

- [76] H. Chu, J.-H. Lee, Y.-C. Lee, C.-H. Hsu, J.-D. Li, and C.-S. Chen. Part-aware measurement for robust multi-view multi-human 3d pose estimation and tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1472–1481, 2021.
- [77] S. Chun, S. Park, and J. Y. Chang. Learnable human mesh triangulation for 3d human pose and shape estimation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2850–2859, 2023.
- [78] M. Costanzo, G. De Maria, G. Lettera, C. Natale, and D. Perrone. A multimodal perception system for detection of human operators in robotic work cells. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 692–699. IEEE, 2019.
- [79] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril, et al. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*, 2018.
- [80] Q. Cui and H. Sun. Towards accurate 3d human motion prediction from incomplete observations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4801–4810, 2021.
- [81] Deutsche Gesetzliche Unfallversicherung e. V. (DGUV). Das Rapid Upper Limb Assessment (RULA), 2007. URL <https://www.dguv.de/medien/ifa/de/pub/rep/pdf/rep07/biar0207/rula.pdf>. German, Accessed: 07.04.2026.
- [82] Deutsche Gesetzliche Unfallversicherung e. V. (DGUV). DGUV Information 209-074 - Industrial robots. 2015. 80 pages, <https://publikationen.dguv.de/regelwerk/dguv-informationen/3150/industrial-robots/>.
- [83] Deutsche Gesetzliche Unfallversicherung e. V. (DGUV). Functional safety of machine controls - Application of EN ISO 13849. 2019. 324 pages, <https://publikationen.dguv.de/forschung/ifa/ifa-report/4150/ifa-report-2/2017e-functional-safety-of-machine-controls-application-of-en-iso-13849>.
- [84] Deutsche Gesetzliche Unfallversicherung e. V. (DGUV). Weiterreichende Fragen zur neuen Maschinenverordnung (Further Questions About the New Machinery Regulation), 2025. German, https://www.dguv.de/medien/dguv-test-medien/_pdf_zip_doc_ppt/eu_maschinenverordnung/2025-29-09_faq-ak-masch-zur-mvo-pdf-tabelle_fuer_dgvu-test-homepage.pdf.
- [85] Deutsche Gesetzliche Unfallversicherung e. V. (DGUV). Software-Assistent SIS-TEMA: Safety Integrity Software Tool for the Evaluation of Machine Applications. 2025. <https://www.dguv.de/ifa/praxishilfen/practical-solutions-machine-safety/software-sistema/index.jsp>.
- [86] Deutsches Institut für Normung e. V. (DIN). Safety of machinery - Human physical performance - Part 1-5 (DIN EN 1005), 2009. German, 188 pages.
- [87] N. Dimitropoulos, T. Togias, N. Zacharaki, G. Michalos, and S. Makris. Seamless human-robot collaborative assembly using artificial intelligence and wearable devices. *Applied Sciences*, 11(12):5699, 2021.
- [88] J. Do and M. Kim. Skateformer: skeletal-temporal transformer for human action recognition. In *European Conference on Computer Vision*, pages 401–420. Springer, 2024.
- [89] J. Dong, W. Jiang, Q. Huang, H. Bao, and X. Zhou. Fast and Robust Multi-Person 3D Pose Estimation from Multiple Views. 2019.

- [90] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.
- [91] H. Duan, J. Wang, K. Chen, and D. Lin. DG-STGCN: dynamic spatial-temporal modeling for skeleton-based action recognition. *arXiv preprint arXiv:2210.05895*, 2022.
- [92] H. Duan, J. Wang, K. Chen, and D. Lin. Pyskl: Towards good practices for skeleton action recognition. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 7351–7354, 2022.
- [93] H. Duan, Y. Zhao, K. Chen, D. Lin, and B. Dai. Revisiting skeleton-based action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2969–2978, 2022.
- [94] S. Erfanian Ebadi, Y.-C. Jhang, A. Zook, S. Dhakad, A. Crespi, P. Parisi, S. Borkman, J. Hogins, and S. Ganguly. Peoplesanspeople: A synthetic data generator for human-centric computer vision. 2021.
- [95] European Commission and Directorate-General for Research and Innovation. *Industry 5.0 - Towards a sustainable, human-centric and resilient European industry*. Publications Office of the European Union, 2021. doi: doi/10.2777/308407.
- [96] Facebook Inc. Duckling, 2021. URL <https://github.com/facebook/duckling>. [accessed 09-June-2021].
- [97] N. Fiedler, J. Gldenstein, T. Na, M. Grner, N. Hendrich, and J. Zhang. A multimodal robotic blackjack dealer: Design, implementation, and reliability analysis. In *International Conference on Intelligent Autonomous Systems*, pages 271–283. Springer, 2023. <https://www.youtube.com/watch?v=0XWj5tyZmnY>, Accessed: 13-April-2026.
- [98] M. Fieraru, M. Zanfir, E. Oneata, A.-I. Popa, V. Olaru, and C. Sminchisescu. Three-dimensional reconstruction of human interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7214–7223, 2020.
- [99] T. Fingscheidt, H. Gottschalk, and S. Houben. *Deep neural networks and data for automated driving: Robustness, uncertainty quantification, and insights towards safety*. Springer Nature, 2022.
- [100] S. Ghorbani, K. Mahdavian, A. Thaler, K. Kording, D. J. Cook, G. Blohm, and N. F. Troje. MoVi: A large multi-purpose human motion and video dataset. *Plos one*, 16(6):e0253157, 2021.
- [101] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.
- [102] W. Guo, Y. Du, X. Shen, V. Lepetit, A.-P. Xavier, and M.-N. Francesc. Back to MLP: A Simple Baseline for Human Motion Prediction. *arXiv preprint arXiv:2207.01567*, 2022.
- [103] L. Hansen, M. Siebert, J. Diesel, and M. P. Heinrich. Fusing information from multiple 2D depth cameras for 3D human pose estimation in the operating room. *International journal of computer assisted radiology and surgery*, 14:1871–1879, 2019.
- [104] A. Hermann, F. Drews, J. Bauer, S. Klemm, A. Roennau, and R. Dillmann. Unified gpu voxel collision detection for mobile manipulation planning. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4154–4160. IEEE, 2014.

- [105] A. Hoffmann, A. Poeppel, A. Schierl, and W. Reif. Environment-aware proximity detection with capacitive sensors for human-robot-interaction. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 145–150. IEEE, 2016.
- [106] H. Huang, C.-M. Pun, H. Li, M. Liu, J. Xiong, and H. Gao. DeformMLP: dynamic large-scale receptive field MLP networks for human motion prediction. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5200–5204. IEEE, 2024.
- [107] T. P. Huck, M. Kaiser, C. Cronrath, B. Lennartson, T. Kröger, and T. Asfour. Reinforcement Learning for Safety Testing: Lessons from A Mobile Robot Case Study. *arXiv preprint arXiv:2311.02907*, 2023.
- [108] International Organization for Standardization (ISO). Artificial intelligence - Functional safety and AI systems - Part 1: Requirements (ISO/IEC CD TS 22440-1). <https://www.iso.org/standard/89535.html>.
- [109] International Organization for Standardization (ISO). Ergonomics - Evaluation of static working postures (ISO 11226:2000), 2000. 19 pages.
- [110] International Organization for Standardization (ISO). Writing paper and certain classes of printed matter - Trimmed sizes - A and B series, and indication of machine direction, 2007. 12 pages.
- [111] International Organization for Standardization (ISO). Safety of machinery — General principles for design — Risk assessment and risk reduction (ISO 12100:2010), 2010. 90 pages.
- [112] International Organization for Standardization (ISO). Robots and robotic devices - Collaborative robots (ISO/TS 15066:2016), 2016. 40 pages.
- [113] International Organization for Standardization (ISO). Framework for Artificial Intelligence (AI) Systems Using Machine Learning (ML) (ISO/IEC 23053:2022), 2022. 36 pages.
- [114] International Organization for Standardization (ISO). Safety of machinery - Safety-related parts of control systems - Part 1: General principles for design (ISO 13849-1:2023), 2023. 164 pages.
- [115] International Organization for Standardization (ISO). Information technology - Artificial intelligence - Guidance on risk management (ISO/IEC 23894:2023), 2023. 26 pages.
- [116] International Organization for Standardization (ISO). Artificial intelligence - Data quality for analytics and machine learning (ML) - Part 1-5 (ISO/IEC 5259:2024), 2024. 128 pages.
- [117] International Organization for Standardization (ISO). Artificial intelligence - Functional safety and AI systems (ISO/IEC TR 5469:2024), 2024. 82 pages.
- [118] International Organization for Standardization (ISO). Road vehicles - Safety and artificial intelligence (ISO/PAS 8800:2024-12), 2024. 172 pages.
- [119] International Organization for Standardization (ISO). Robotics - Safety requirements - Part 1: Industrial robots (ISO 10218-1:2025), 2025. 104 pages.
- [120] International Organization for Standardization (ISO). Robotics - Safety requirements - Part 2: Industrial robot applications and robot cells (ISO 10218-2:2025), 2025. 234 pages.
- [121] International Organization for Standardization (ISO). Light and lighting - Lighting of work places - Part 1: Indoor (ISO/CIE 8995-1:2025), 2025. 109 pages.

- [122] 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*, 36(7):1325–1339, jul 2014.
- [123] K. Isakov, E. Burkov, V. Lempitsky, and Y. Malkov. Learnable triangulation of human pose. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7718–7727, 2019.
- [124] W. Jia, Z. Lu, H. Zhang, Z. Liu, J. Wang, and G. Qu. Fooling the eyes of autonomous vehicles: Robust physical adversarial examples against traffic sign recognition systems. *arXiv preprint arXiv:2201.06192*, 2022.
- [125] Z. Jia, M. Lin, Z. Chen, and S. Jian. Vision-based robot manipulation learning via human demonstrations. *arXiv preprint arXiv:2003.00385*, 2020.
- [126] T. Jiang, P. Lu, L. Zhang, N. Ma, R. Han, C. Lyu, Y. Li, and K. Chen. RtmPose: Real-time multi-person pose estimation based on mmpose. *arXiv preprint arXiv:2303.07399*, 2023.
- [127] H. Joo, H. Liu, L. Tan, L. Gui, B. Nabbe, I. Matthews, T. Kanade, S. Nobuhara, and Y. Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3334–3342, 2015.
- [128] A. Kadkhodamohammadi and N. Padoy. A generalizable approach for multi-view 3d human pose regression. *Machine Vision and Applications*, 32(1):6, 2021.
- [129] A. Kadkhodamohammadi, A. Gangi, M. de Mathelin, and N. Padoy. A multi-view RGB-D approach for human pose estimation in operating rooms. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 363–372. IEEE, 2017.
- [130] R. Khirodkar, A. Bansal, L. Ma, R. Newcombe, M. Vo, and K. Kitani. Ego-Humans: An Ego-Centric 3D Multi-Human Benchmark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19807–19819, 2023.
- [131] S. Kim, G. Kim, S. Shin, and S. Lee. Two-stage Textual Knowledge Distillation to Speech Encoder for Spoken Language Understanding. *arXiv preprint arXiv:2010.13105*, 2020.
- [132] S. Kim, A. Gholami, A. Shaw, N. Lee, K. Mangalam, J. Malik, M. W. Mahoney, and K. Keutzer. Squeezeformer: An efficient transformer for automatic speech recognition. *Advances in Neural Information Processing Systems*, 35:9361–9373, 2022.
- [133] W. Kim, L. Peternel, M. Lorenzini, J. Babič, and A. Ajoudani. A human-robot collaboration framework for improving ergonomics during dexterous operation of power tools. *Robotics and Computer-Integrated Manufacturing*, 68:102084, 2021.
- [134] D. Kóczy and J. Sárosi. Safety engineering for humanoid robots in everyday life—scoping review. *Electronics*, 14(23):4734, 2025.
- [135] S. Kriman, S. Beliaev, B. Ginsburg, J. Huang, O. Kuchaiev, V. Lavrukhin, R. Leary, J. Li, and Y. Zhang. Quartznet: Deep automatic speech recognition with 1d time-channel separable convolutions. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6124–6128. IEEE, 2020.
- [136] KUKA Deutschland GmbH. KUKA.SafeOperation 3.6 - Assembly and Operating Instructions, 2024.
- [137] KUKA Robotics. Spez KR CYBERTECH-2 V4, 2019.
- [138] P. A. Lasota, G. F. Rossano, and J. A. Shah. Toward safe close-proximity human-robot interaction with standard industrial robots. In *2014 IEEE international Conference on Automation Science and Engineering (CASE)*, pages 339–344. IEEE, 2014.

- [139] J. Lee, M. Lee, D. Lee, and S. Lee. Hierarchically decomposed graph convolutional networks for skeleton-based action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10444–10453, 2023.
- [140] M. Li, S. Chen, Z. Zhang, L. Xie, Q. Tian, and Y. Zhang. Skeleton-parted graph scattering networks for 3d human motion prediction. In *European Conference on Computer Vision*, pages 18–36. Springer, 2022.
- [141] J. Lin and G. H. Lee. Multi-view multi-person 3d pose estimation with plane sweep stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11886–11895, 2021.
- [142] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.
- [143] D. Liu, Y. Hu, K. Hua, Y. Lu, Z. Zhang, X. Ma, Z. Zhong, and P. Chen. TDSN-GCN: Transformerify Overall Structure Decaying Static Graph Embedding NAS-guided GCN for Skeleton Action Recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 2025.
- [144] H. Liu and L. Wang. Collision-free human-robot collaboration based on context awareness. *Robotics and Computer-Integrated Manufacturing*, 67:101997, 2021.
- [145] H. Liu, Y. Liu, M. Ren, H. Wang, Y. Wang, and Z. Sun. Revealing key details to see differences: A novel prototypical perspective for skeleton-based action recognition. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 29248–29257, 2025.
- [146] H. Liu, Y. Liu, C. Wang, Y. Wang, and Z. Sun. SkeletonAgent: An Agentic Interaction Framework for Skeleton-based Action Recognition. *arXiv preprint arXiv:2511.22433*, 2025.
- [147] J. Liu, A. Shahroudy, M. Perez, G. Wang, L.-Y. Duan, and A. C. Kot. Ntu rgb+ d 120: A large-scale benchmark for 3d human activity understanding. *IEEE transactions on pattern analysis and machine intelligence*, 42(10):2684–2701, 2019.
- [148] R. Liu, R. Chen, A. Abuduweili, and C. Liu. Proactive human-robot co-assembly: Leveraging human intention prediction and robust safe control. In *2023 IEEE Conference on Control Technology and Applications (CCTA)*, pages 339–345. IEEE, 2023.
- [149] Y. Liu and H. Jebelli. Intention-aware robot motion planning for safe worker–robot collaboration. *Computer-Aided Civil and Infrastructure Engineering*, 39(15):2242–2269, 2024.
- [150] Z. Liu, H. Zhang, Z. Chen, Z. Wang, and W. Ouyang. Disentangling and unifying graph convolutions for skeleton-based action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 143–152, 2020.
- [151] P. Lu, T. Jiang, Y. Li, X. Li, K. Chen, and W. Yang. RTMO: Towards High-Performance One-Stage Real-Time Multi-Person Pose Estimation. *arXiv preprint arXiv:2312.07526*, 2023.
- [152] L. Lugosch, M. Ravanelli, P. Ignoto, V. S. Tomar, and Y. Bengio. Speech model pre-training for end-to-end spoken language understanding. *arXiv preprint arXiv:1904.03670*, 2019.
- [153] L. Lugosch, B. H. Meyer, D. Nowrouzezahrai, and M. Ravanelli. Using speech synthesis to train end-to-end spoken language understanding models. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8499–8503. IEEE, 2020.

- [154] T. Ma, Y. Nie, C. Long, Q. Zhang, and G. Li. Progressively generating better initial guesses towards next stages for high-quality human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6437–6446, 2022.
- [155] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022. doi: 10.1126/scirobotics.abm6074. URL <https://www.science.org/doi/abs/10.1126/scirobotics.abm6074>.
- [156] E. Magrini, F. Ferraguti, A. J. Ronga, F. Pini, A. De Luca, and F. Leali. Human-robot coexistence and interaction in open industrial cells. *Robotics and Computer-Integrated Manufacturing*, 61:101846, 2020.
- [157] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black. AMASS: Archive of Motion Capture as Surface Shapes. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2019. URL <https://amass.is.tue.mpg.de>.
- [158] V. Malke, O. Kosak, S. Rossi, M. Schörner, C. Wanninger, and W. Reif. Swarmcage: Development sandbox for interactive swarms, formation flights, and continuous observation using crazyflies, ros, and vicon. *Currently under review*, 2026.
- [159] C. Mandery, Ö. Terlemez, M. Do, N. Vahrenkamp, and T. Asfour. The kit whole-body human motion database. In *2015 International Conference on Advanced Robotics (ICAR)*, pages 329–336. IEEE, 2015.
- [160] L. McAtamney and E. N. Corlett. Rula: a survey method for the investigation of work-related upper limb disorders. *Applied ergonomics*, 24(2):91–99, 1993.
- [161] R. Memmesheimer, N. Theisen, and D. Paulus. Sl-dml: Signal level deep metric learning for multimodal one-shot action recognition. In *2020 25th International conference on pattern recognition (ICPR)*, pages 4573–4580. IEEE, 2021.
- [162] R. Memmesheimer, S. Häring, N. Theisen, and D. Paulus. Skeleton-dml: Deep metric learning for skeleton-based one-shot action recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3702–3710, 2022.
- [163] G. Michalos, N. Kousi, P. Karagiannis, C. Gkournelos, K. Dimoulas, S. Koukas, K. Mparis, A. Papavasileiou, and S. Makris. Seamless human robot collaborative assembly—an automotive case study. *Mechatronics*, 55:194–211, 2018.
- [164] Mitsubishi Electric. Collaborative Robot Safety Manual.
- [165] G. Moon, J. Y. Chang, and K. M. Lee. V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. In *Proceedings of the IEEE conference on computer vision and pattern Recognition*, pages 5079–5088, 2018.
- [166] Mozilla. Project DeepSpeech, 2021. URL <https://github.com/mozilla/DeepSpeech>. Accessed: 26-February-2021.
- [167] M. Munaro, A. Horn, R. Illum, J. Burke, and R. B. Rusu. OpenPTrack: People tracking for heterogeneous networks of color-depth cameras. In *IAS-13 Workshop Proceedings: 1st Intl. Workshop on 3D Robot Perception with Point Cloud Library*, pages 235–247. Citeseer, 2014.
- [168] Mycroft AI. Mycroft, 2021. URL <https://mycroft.ai/>. [accessed 17-June-2021].
- [169] W. Myung, N. Su, J.-H. Xue, and G. Wang. Degcn: Deformable graph convolutional networks for skeleton-based action recognition. *IEEE Transactions on Image Processing*, 33:2477–2490, 2024.

- [170] naggety. picotts, 2021. URL <https://github.com/naggety/picotts>. [accessed 23-March-2021].
- [171] NBDEW Bundesverband der Energie- und Wasserwirtschaft e.V. BDEW-Strompreisanalyse Januar 2026, 2026. URL <https://www.bdew.de/service/daten-und-grafiken/bdew-strompreisanalyse/>. German, Accessed: 07.04.2026.
- [172] S. Nikolaidis, R. Ramakrishnan, K. Gu, and J. Shah. Efficient model learning from joint-action demonstrations for human-robot collaborative tasks. In *Proceedings of the tenth annual ACM/IEEE international conference on human-robot interaction*, pages 189–196, 2015.
- [173] N. Nikolakis, V. Maratos, and S. Makris. A cyber physical system (cps) approach for safe human-robot collaboration in a shared workplace. *Robotics and Computer-Integrated Manufacturing*, 56:233–243, 2019.
- [174] NVIDIA Corporation. NVIDIA Jetson Orin, 2022. URL <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/>.
- [175] NVIDIA Corporation. NVIDIA Isaac Sim, 2026. <https://developer.nvidia.com/isaac/sim>, https://docs.isaacsim.omniverse.nvidia.com/4.5.0/replicator-tutorials/tutorial_replicator_agent.html, Accessed: 02.04.2026.
- [176] E. Olson. AprilTag: A robust and flexible visual fiducial system. In *2011 IEEE international conference on robotics and automation*, pages 3400–3407. IEEE, 2011.
- [177] OpenCV. OpenCV Homepage. URL <https://opencv.org/>.
- [178] Q. Pan and X. Xie. Language-guided temporal primitive modeling for skeleton-based action recognition. *Neurocomputing*, 613:128636, 2025.
- [179] Z. Pan, Z. Zhong, W. Guo, Y. Chen, J. Feng, and J. Zhou. PointVoxel: A Simple and Effective Pipeline for Multi-View Multi-Modal 3D Human Pose Estimation. *arXiv preprint arXiv:2312.06409*, 2023.
- [180] S. Paquette. *Anthropometric survey (ANSUR) II pilot study: methods and summary statistics*. Anthrotrch, US Army Natick Soldier Research, Development and Engineering Center, 2009.
- [181] X. Peng, S. Mao, and Z. Wu. Trajectory-aware body interaction transformer for multi-person pose forecasting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 17121–17130, 2023.
- [182] L. Peternel, C. Fang, N. Tsagarakis, and A. Ajoudani. A selective muscle fatigue management approach to ergonomic human-robot co-manipulation. *Robotics and Computer-Integrated Manufacturing*, 58:69–79, 2019.
- [183] Picovoice. Porcupine, 2021. URL <https://picovoice.ai/platform/porcupine/>. [accessed 23-March-2021].
- [184] Picovoice. Rhino, 2021. URL <https://picovoice.ai/platform/rhino/>. [accessed 11-October-2021].
- [185] Pilz GmbH and Co. KG. SafetyEYE. Perfect 3D Protection. URL https://airlinemedia.airlinehyd.com/Literature/Manufacturer_Catalogs/Pilz/SafetyEYE-114142003.pdf. Accessed: 12.03.2026.
- [186] M. Płaczek and Ł. Piszczek. Testing of an industrial robot’s accuracy and repeatability in off and online environment. *Eksploracja i Niezawodność*, 20(3):455–464, 2018.

- [187] Y. Qian, X. Bian, Y. Shi, N. Kanda, L. Shen, Z. Xiao, and M. Zeng. Speech-language pre-training for end-to-end spoken language understanding. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7458–7462. IEEE, 2021.
- [188] H. Qu, Y. Cai, and J. Liu. Llms are good action recognizers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18395–18406, 2024.
- [189] M. Radfar, A. Mouchtaris, S. Kunzmann, and A. Rastrow. FANS: Fusing ASR and NLU for On-Device SLU. In *Proc. Interspeech 2021*, pages 1224–1228, 2021. doi: 10.21437/Interspeech.2021-793.
- [190] M. Ragaglia, A. M. Zanchettin, and P. Rocco. Trajectory generation algorithm for safe human-robot collaboration based on multiple depth sensor measurements. *Mechatronics*, 55:267–281, 2018.
- [191] H. Rajnathsing and C. Li. A neural network based monitoring system for safety in shared work-space human-robot collaboration. *Industrial Robot: An International Journal*, 45(4): 481–491, 2018.
- [192] Rasa Technologies Inc. Rasa, 2021. URL <https://rasa.com/>. [accessed 23-March-2021].
- [193] Raspberry Pi Foundation. Raspberry Pi AI HAT+, 2026. URL <https://www.raspberrypi.com/products/ai-hat/>.
- [194] N. D. Reddy, L. Guigues, L. Pishchulin, J. Eledath, and S. G. Narasimhan. Tesseract: End-to-end learnable multi-person articulated 3d pose tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15190–15200, 2021.
- [195] Robotiq, Mathieu Bélanger-Barrette. Universal Robots Release their New Generation of Collaborative Robots, 2014. URL <https://blog.robotiq.com/bid/71494/universal-robots-release-their-new-generation-of-collaborative-robots>. Accessed: 12.03.2026.
- [196] Robotiq, Mathieu Bélanger-Barrette. Cobot, Force Limited Robot, Collaborative Robot, What’s the Deal?, 2015. URL <https://blog.robotiq.com/bid/73295/cobot-force-limited-robot-collaborative-robot-what-s-the-deal>. Accessed: 12.03.2026.
- [197] S. Rongali, B. Liu, L. Cai, K. Arkoudas, C. Su, and W. Hamza. Exploring Transfer Learning For End-to-End Spoken Language Understanding. *arXiv preprint arXiv:2012.08549*, 2020.
- [198] P. Rybski, P. Anderson-Sprecher, D. Huber, C. Niessl, and R. Simmons. Sensor fusion for human safety in industrial workcells. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3612–3619. IEEE, 2012.
- [199] K. Ryselis, T. Petkus, T. Blažauskas, R. Maskeliūnas, and R. Damaševičius. Multiple Kinect based system to monitor and analyze key performance indicators of physical training. *Human-Centric Computing and Information Sciences*, 10:1–22, 2020.
- [200] S. Saadatnejad, A. Rasekh, M. Mofayez, Y. Medghalchi, S. Rajabzadeh, T. Mordan, and A. Alahi. A generic diffusion-based approach for 3D human pose prediction in the wild. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8246–8253. IEEE, 2023.
- [201] A. Saade, A. Coucke, A. Caulier, J. Dureau, A. Ball, T. Bluche, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, et al. Spoken language understanding on the edge. *arXiv preprint arXiv:1810.12735*, 2018.

- [202] A. Sabater, L. Santos, J. Santos-Victor, A. Bernardino, L. Montesano, and A. C. Murillo. One-shot action recognition in challenging therapy scenarios. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2777–2785, 2021.
- [203] A. Sampieri, G. M. D. di Melendugno, A. Avogaro, F. Cunico, F. Setti, G. Skenderi, M. Cristani, and F. Galasso. Pose forecasting in industrial human-robot collaboration. In *European Conference on Computer Vision*, pages 51–69. Springer, 2022.
- [204] L. Sanneman, C. Fourie, and J. Shah. The state of industrial robotics: Emerging technologies, challenges, and key research directions. 10 2020. doi: 10.48550/arXiv.2010.14537.
- [205] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.
- [206] F. Schirmer, P. Kranz, J. Schmitt, and T. Kaupp. Anomaly detection for dynamic human-robot assembly: Application of an lstm-based autoencoder to interpret uncertain human behavior in hrc. In *Companion of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*, pages 333–337, 2023.
- [207] L. S. Scimmi, M. Melchiorre, M. Troise, S. Mauro, and S. Pastorelli. A practical and effective layout for a safe human-robot collaborative assembly task. *Applied Sciences*, 11(4):1763, 2021.
- [208] S. Seo, D. Kwak, and B. Lee. Integration of Pre-trained Networks with Continuous Token Interface for End-to-End Spoken Language Understanding. *arXiv preprint arXiv:2104.07253*, 2021.
- [209] H. N. M. Shah, M. Sulaiman, K. S. M. Isa, Z. Kamis, and M. R. Baharon. Optimization of energy consumption in kuka kr 16 articulated robot manipulator. *Int. J. Recent Technol. Eng*, 8:6470–6476, 2019.
- [210] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1010–1019, 2016.
- [211] J. Shi, J. Zhong, Z. He, and W. Cao. Gradient multi-foci networks for 3D skeleton-based human motion prediction. *Neural Computing and Applications*, 36(24):14627–14642, 2024.
- [212] Q. Shuai, Z. Yu, Z. Zhou, L. Fan, H. Yang, C. Yang, and X. Zhou. Reconstructing Close Human Interactions from Multiple Views. *ACM Transactions on Graphics (TOG)*, 42(6): 1–14, 2023.
- [213] Sick AG. Sick light curtain, 2026. <https://www.sick.com/at/en/catalog/products/safety/safety-light-curtains/c4000-advanced/c40s-1204ca010-c40e-1204cb010/p/p702?tab=detail>, <https://www.sick.com/at/en/catalog/products/safety/safety-light-curtains/m4000-advanced-curtain/m40s-65a503aa0-m40e-65a503rb0/p/p54993?tab=detail>, Accessed: 02.03.2026.
- [214] Sick AG. Sick laser scanner, 2026. URL <https://www.sick.com/at/en/catalog/produkte/safety/sicherheitslaserscanner/microscan3/mics3-ccaz55anlp01/p/p678415?tab=detail>. Accessed: 02.03.2026.
- [215] Sick AG. Sick radar, 2026. URL <https://www.sick.com/at/en/catalog/produkte/safety/sichere-radarsensoren/saferssafers3/sra2-aar140akzi/p/p655280?tab=detail>. Accessed: 02.03.2026.

- [216] Sick AG. Sick ToF camera, 2026. URL <https://www.sick.com/at/en/catalog/produkte/safety/sichere-3d-kameras/safevisionary2/v3sa2-abbabbaan1/p/p666143?tab=detail>. Accessed: 02.03.2026.
- [217] Sick AG. Sick End-of-Arm-Safeguard, 2026. URL <https://www.sick.com/sg/en/catalog/products/safety/safety-systems/end-of-arm-safeguard/c/g585667>. Accessed: 02.03.2026.
- [218] I. R. Silva, G. B. Barbosa, C. C. Ledebour, A. T. O. Filho, J. Kelner, D. Sadok, S. Lins, and R. Souza. Assessing deep learning models for human-robot collaboration collision detection in industrial environments. In *Brazilian conference on intelligent systems*, pages 240–255. Springer, 2020.
- [219] H. Smith, N. Fatehpour, B. Matthias, D. Clever, and ABB AG. Vorhersage des Bremsverhaltens für Robotersicherheit - Prediction of Stopping Behavior for Robot Safety.
- [220] T. Sofianos, A. Sampieri, L. Franco, and F. Galasso. Space-Time-Separable Graph Convolutional Network for Pose Forecasting, 2021.
- [221] V. Srivastav, T. Issenhuth, A. Kadkhodamohammadi, M. de Mathelin, A. Gangi, and N. Padoy. MVOR: A multi-view RGB-D operating room dataset for 2D and 3D human pose estimation. *arXiv preprint arXiv:1808.08180*, 2018.
- [222] V. Srivastav, K. Chen, and N. Padoy. SelfPose3d: Self-Supervised Multi-Person Multi-View 3d Pose Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2502–2512, June 2024.
- [223] Statistisches Bundesamt (Destatis). Arbeitskosten im Vergleich - Pressemitteilung Nr. 154 vom 30. April 2025, 2026. URL https://www.destatis.de/DE/Presse/Pressemitteilungen/2025/04/PD25_154_624.html. German, Accessed: 07.04.2026.
- [224] K. Sun, B. Xiao, D. Liu, and J. Wang. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5693–5703, 2019.
- [225] Y. Sun, R. Chen, K. S. Yun, Y. Fang, S. Jung, F. Li, B. Li, W. Zhao, and C. Liu. SPARK: Safe protective and assistive robot kit. IFAC Symposium on Robotics, 2025.
- [226] J. T. C. Tan and T. Arai. Triple stereo vision system for safety monitoring of human-robot collaboration in cellular manufacturing. In *2011 IEEE International Symposium on Assembly and Manufacturing (ISAM)*, pages 1–6. IEEE, 2011.
- [227] J. Tanke and J. Gall. Iterative Greedy Matching for 3D Human Pose Tracking from Multiple Views. In *German Conference on Pattern Recognition*, 2019.
- [228] Y. Tian and P. J. Gorinski. Improving End-to-End Speech-to-Intent Classification with Reptile. *arXiv preprint arXiv:2008.01994*, 2020.
- [229] N. Trivedi, A. Thatipelli, and R. K. Sarvadevabhatla. NTU-X: an enhanced large-scale dataset for improving pose-based recognition of subtle human actions. In *Proceedings of the Twelfth Indian Conference on Computer Vision, Graphics and Image Processing*, pages 1–9, 2021.
- [230] N. F. Troje. Decomposing Biological Motion: A Framework for Analysis and Synthesis of Human Gait Patterns. *Journal of Vision*, 2(5):2–2, Sept. 2002. doi: 10.1167/2.5.2.
- [231] H. Tu, C. Wang, and W. Zeng. VoxelPose: Towards Multi-Camera 3D Human Pose Estimation in Wild Environment. In *European Conference on Computer Vision (ECCV)*, 2020.

- [232] V. V. Unhelkar, P. A. Lasota, Q. Tyroller, R.-D. Buhai, L. Marceau, B. Deml, and J. A. Shah. Human-aware robotic assistant for collaborative assembly: Integrating human motion prediction with planning in time. *IEEE Robotics and Automation Letters*, 3(3):2394–2401, 2018.
- [233] Universal Robots A/S. User Manual - UR16e, 2024.
- [234] E. Vendrow, S. Kumar, E. Adeli, and H. Rezatofighi. SoMoFormer: Multi-Person Pose Forecasting with Transformers. *arXiv preprint arXiv:2208.14023*, 2022.
- [235] F. Vicentini, N. Pedrocchi, M. Giussani, and L. M. Tosatti. Dynamic safety in collaborative robot workspaces through a network of devices fulfilling functional safety requirements. In *ISR/Robotik 2014; 41st International Symposium on Robotics*, pages 1–7. VDE, 2014.
- [236] Vicon Motion Systems. How is markerless motion capture different from marker-based motion capture? URL <https://www.vicon.com/resources/blog/what-is-markerless-motion-capture/>. Accessed: 02.05.2026.
- [237] D. Vogt, S. Stepputtis, R. Weinhold, B. Jung, and H. B. Amor. Learning human-robot interactions from human-human demonstrations (with applications in lego rocket assembly). In *Humanoids*, pages 142–143, 2016.
- [238] T. Von Marcard, R. Henschel, M. J. Black, B. Rosenhahn, and G. Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *Proceedings of the European conference on computer vision (ECCV)*, pages 601–617, 2018.
- [239] L. Wang and P. Koniusz. 3Mformer: Multi-order Multi-mode Transformer for Skeletal Action Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5620–5631, 2023.
- [240] L. Wang, B. Schmidt, and A. Y. Nee. Vision-guided active collision avoidance for human-robot collaborations. *Manufacturing Letters*, 1(1):5–8, 2013.
- [241] T. Wang, J. Zhang, Y. Cai, S. Yan, and J. Feng. Direct Multi-view Multi-person 3D Human Pose Estimation. *Advances in Neural Information Processing Systems*, 2021.
- [242] X. Wang, X. Xu, and Y. Mu. Neural koopman pooling: Control-inspired temporal dynamics encoding for skeleton-based action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10597–10607, 2023.
- [243] X. Wang, Q. Cui, C. Chen, and M. Liu. Gcnex: Towards the unity of graph convolutions for human motion prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 5642–5650, 2024.
- [244] Y. Wang, Y. Wu, W. He, X. Guo, F. Zhu, L. Bai, R. Zhao, J. Wu, T. He, W. Ouyang, et al. Hulk: A universal knowledge translator for human-centric tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.
- [245] Z. Wang and J. Yan. Deep learning based assembly process action recognition and progress prediction facing human-centric intelligent manufacturing. *Computers & Industrial Engineering*, 196:110527, 2024.
- [246] Waymo LLC. A commitment to transparency and road safety: event overview, 2026. URL <https://waymo.com/blog/2026/01/a-commitment-to-transparency-and-road-safety/>. Accessed: 30.03.2026.
- [247] M. Wei, L. Miaomiao, and S. Mathieu. History Repeats Itself: Human Motion Prediction via Motion Attention. In *ECCV*, 2020.

- [248] S. Wu, S. Jin, W. Liu, L. Bai, C. Qian, D. Liu, and W. Ouyang. Graph-based 3d multi-person pose estimation using multi-view images. In *ICCV*, 2021.
- [249] L. Xiang and Z. Wang. Joint mixing data augmentation for skeleton-based action recognition. *ACM Transactions on Multimedia Computing, Communications and Applications*, 21(4):1–24, 2025.
- [250] B. Xiao, H. Wu, and Y. Wei. Simple baselines for human pose estimation and tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 466–481, 2018.
- [251] P. Xiao, Y. Xie, X. Xu, W. Chen, and H. Zhang. Multi-person Pose Forecasting with Individual Interaction Perceptron and Prior Learning. In *European Conference on Computer Vision*, pages 402–419. Springer, 2024.
- [252] Z. Xie, L. Lu, H. Wang, B. Su, Y. Liu, and X. Xu. Improving workers’ musculoskeletal health during human-robot collaboration through reinforcement learning. *Human Factors*, 66(6):1754–1769, 2024.
- [253] C. Xu, R. T. Tan, Y. Tan, S. Chen, X. Wang, and Y. Wang. Auxiliary tasks benefit 3d skeleton-based human motion prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9509–9520, 2023.
- [254] C. Xu, R. T. Tan, Y. Tan, S. Chen, Y. G. Wang, X. Wang, and Y. Wang. EqMotion: Equivariant Multi-agent Motion Prediction with Invariant Interaction Reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1410–1420, 2023.
- [255] H. Xu, Y. Gao, Z. Hui, J. Li, and X. Gao. Language Knowledge-Assisted Representation Learning for Skeleton-Based Action Recognition. *arXiv preprint arXiv:2305.12398*, 2023.
- [256] Q. Xu, W. Mao, J. Gong, C. Xu, S. Chen, W. Xie, Y. Zhang, and Y. Wang. Joint-relation transformer for multi-person motion prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9816–9826, 2023.
- [257] S. Yang, J. Liu, S. Lu, E. M. Hwa, and A. C. Kot. One-shot action recognition via multi-scale spatial-temporal skeleton matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(7):5149–5156, 2024.
- [258] H. Ye, W. Zhu, C. Wang, R. Wu, and Y. Wang. Faster VoxelPose: Real-time 3D Human Pose Estimation by Orthographic Projection. In *European Conference on Computer Vision (ECCV)*, 2022.
- [259] A. M. Zanchettin, A. Casalino, L. Piroddi, and P. Rocco. Prediction of human activity patterns for human–robot collaborative assembly tasks. *IEEE Transactions on Industrial Informatics*, 15(7):3934–3942, 2018.
- [260] J. Zhang, L. Lin, and J. Liu. Shap-mix: Shapley value guided mixing for long-tailed skeleton based action recognition. *arXiv preprint arXiv:2407.12312*, 2024.
- [261] Y. Zhang, L. An, T. Yu, x. Li, K. Li, and Y. Liu. 4D Association Graph for Realtime Multi-person Motion Capture Using Multiple Video Cameras. In *IEEE International Conference on Computer Vision and Pattern Recognition, (CVPR)*, 2020.
- [262] Y. Zhang, L. An, T. Yu, X. Li, K. Li, and Y. Liu. 4d association graph for realtime multi-person motion capture using multiple video cameras. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1324–1333, 2020.
- [263] Z. Zhang, C. Wang, W. Qiu, W. Qin, and W. Zeng. Adafuse: Adaptive multiview fusion for accurate human pose estimation in the wild. *International Journal of Computer Vision*, 129:703–718, 2021.

- [264] Z. Zhang, W. Cai, Q. Liu, and Y. Wang. SkeletonX: Data-Efficient Skeleton-based Action Recognition via Cross-sample Feature Aggregation. *arXiv preprint arXiv:2504.11749*, 2025.
- [265] B. Zheng, M. Ablimit, and A. Hamdulla. Cross-Modal Alignment for End-to-End Spoken Language Understanding Based on Momentum Contrastive Learning. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 10576–10580. IEEE, 2024.
- [266] Y. Zheng, R. Yu, and J. Sun. Efficient Multi-Person Motion Prediction by Lightweight Spatial and Temporal Interactions. *arXiv preprint arXiv:2507.09446*, 2025.
- [267] L. Zhou and X. Jiao. Multi-modal and multi-part with skeletons and texts for action recognition. *Expert Systems with Applications*, page 126646, 2025.
- [268] Y. Zhou, X. Yan, Z.-Q. Cheng, Y. Yan, Q. Dai, and X.-S. Hua. BlockGCN: Redefine topology awareness for skeleton-based action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2049–2058, 2024.
- [269] Y. Zhou, T. Xu, C. Wu, X. Wu, and J. Kittler. Adaptive hyper-graph convolution network for skeleton-based human action recognition with virtual connections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12648–12658, 2025.
- [270] Z. Zhou, Q. Shuai, Y. Wang, Q. Fang, X. Ji, F. Li, H. Bao, and X. Zhou. Quickpose: Real-time multi-view multi-person pose estimation in crowded scenes. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022.
- [271] W. Zhu, X. Ma, Z. Liu, L. Liu, W. Wu, and Y. Wang. MotionBERT: A Unified Perspective on Learning Human Motion Representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15085–15099, 2023.
- [272] Y. Zhu, N. Samet, and D. Picard. H3WB: Human3.6M 3D WholeBody Dataset and Benchmark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 20166–20177, October 2023.

List of Figures

1.1	Introducing robot example	2
1.2	Example of a robot system with physical separation	3
1.3	Examples of stop-on-entry systems	4
1.4	Examples of hand-guiding systems	5
1.5	Examples of speed and separation monitoring systems	6
1.6	Example of a tighter safety zone around the human	7
1.7	Examples of capacitive sensor robots	8
1.8	Examples of collaborative robots	9
1.9	Example of additional safety cushions	10
1.10	Motivating robot with bike example	12
2.1	Structural organization of safety standards	16
2.2	Examples of hazards	18
2.3	Risk reduction process for machine design	20
2.4	Risk assessment according to <i>ENISO 13849-1</i>	21
2.5	Categories <i>B</i> and <i>1</i> according to <i>ENISO 13849-1</i>	25
2.6	Category <i>2</i> according to <i>ENISO 13849-1</i>	26
2.7	Categories <i>3</i> and <i>4</i> according to <i>ENISO 13849-1</i>	26
2.8	System assessment according to <i>ENISO 13849-1</i>	27
2.9	Software development process according to <i>ENISO 13849-1</i>	28
3.1	Close collaboration at handling packages	31
3.2	The robot setup	32
3.3	The rc-visard stereo-camera	32
3.4	Voxel-based distance monitoring	36
3.5	Software pipeline of KoARob	37
3.6	Internal visualization	38
3.7	Pose estimation on images	40
3.8	Pose estimation with heatmaps	40
3.9	Dataset example of <i>Human3.6M</i>	41
3.10	Dataset example of <i>Shelf & Campus</i>	42
3.11	Dataset example of <i>Panoptic</i>	43
3.12	Dataset example of <i>MVOR</i>	44
3.13	Learnable voxel triangulation	46
3.14	Aligned color and depth images in <i>SimpleDepthPose</i>	49
3.15	Extracting depth in <i>SimpleDepthPose</i>	50
3.16	Merging proposals in <i>SimpleDepthPose</i>	51
3.17	Concept overview of <i>VoxelKeypointFusion</i>	53
3.18	Depth masking in <i>VoxelKeypointFusion</i>	56
3.19	Evaluation dataset example of a 3D pose and their projection	65
3.20	System assessment according to <i>ENISO 13849-1</i>	67

3.21	Related Work: Depth	68
3.22	Related Work: Markers	69
3.23	Related Work: AI-based	70
3.24	Internal visualization	71
4.1	Contributions to the <i>protective separation distance</i>	75
4.2	Pipeline and safety delay	75
5.1	Planned camera positions	81
5.2	Camera calibration	82
5.3	Concept of the <i>Bayer</i> image format	83
5.4	Smart Camera Hardware	84
5.5	Sphere distance monitoring	85
5.6	Software pipeline of Tutabo-1	86
5.7	The robot setup of Tutabo-1	86
5.8	Concept of RapidPoseTriangulation	88
5.9	Example of a multi-person whole-body pose estimation	99
5.10	Evaluation dataset example	101
5.11	Dataset annotation tool	102
5.12	Pose detection examples	103
5.13	Robot motion after stop signal	105
6.1	Architectural patterns for AI-based systems from <i>ISO/IEC TR 5469</i>	114
7.1	The robot setup of Tutabo-2	117
7.2	Different robot braking ramps	118
7.3	Examples of outliers motivating stop-zones	121
7.4	Example motivating interaction-zones	122
7.5	Visualization of stop- and interaction-zones	123
7.6	Visualizing proposals generated by RPT	124
8.1	Example poses of an action sequence	146
8.2	Example of a pose forecast of a walking person	153
9.1	RoboJack	163
9.2	General architecture of a voice assistant	167
9.3	Process of a voice interaction using <i>Jaco</i>	168
9.4	Intent and entity tagging with <i>finstreder</i>	170
9.5	Entity tagging with <i>slungt</i>	171
9.6	Hand-guiding small drones	176
10.1	Motivating robot with bike example	180

List of Tables

2.1	Definition of <i>Performance Levels</i> according to <i>EN ISO 13849-1</i>	23
2.2	Definition of <i>Mean Time to Dangerous Failure</i> levels in <i>EN ISO 13849-1</i>	24
2.3	Definition of <i>Diagnostic Coverage</i> levels in <i>EN ISO 13849-1</i>	24
2.4	Measures against <i>Common Cause Failures</i> in <i>EN ISO 13849-1</i>	25
3.1	Transfer to <i>human36m</i>	58
3.2	Transfer to <i>Shelf</i>	59
3.3	Transfer to <i>Campus</i>	60
3.4	Transfer to <i>MVOR</i>	61
3.5	Results on <i>Panoptic</i>	62
3.6	Results of single approaches on the new <i>koarob</i> dataset	64
3.7	Results of merged approaches on the new <i>koarob</i> dataset	66
3.8	Results per joint on <i>koarob</i>	66
4.1	Biomechanical force limits according to <i>EN ISO TS 15066</i>	76
5.1	Average time for different steps of <i>RapidPoseTriangulation</i>	90
5.2	Time from 2D to 3D poses on <i>Shelf</i> in milliseconds	91
5.3	Transfer to <i>Human3.6M</i>	92
5.4	Transfer to <i>Shelf</i>	92
5.5	Transfer to <i>Campus</i>	92
5.6	Transfer to <i>MVOR</i>	92
5.7	Results on <i>Panoptic</i>	92
5.8	Transfer to <i>CHI3D</i>	93
5.9	Transfer to <i>Tsinghua</i>	93
5.10	Averaged generalization on all six unseen datasets	94
5.11	Transfer to <i>EgoHumans</i>	96
5.12	Whole-body estimation on <i>H3WB</i>	98
5.13	Pose estimation results on the new evaluation dataset	104
5.14	Results per joint	104
7.1	Joint detection improvement by stop-zones	120
7.2	Joint detection errors with proposals	125
8.1	Results with the new whole-body poses on <i>NTU-RGBD</i>	147
8.2	Comparison with other skeleton representations	148
8.3	Results on <i>NTU-RGBD-60</i> and <i>NTU-RGBD-120</i>	149
8.4	One-shot transfer classification on <i>NTU-RGBD</i>	150
8.5	Few-shot transfer with five examples per action	150
8.6	Model size and inference speed	150
8.7	Comparison of different input durations	151
8.8	Comparison of <i>absolute</i> pose forecasting on <i>Human3.6M</i>	155

8.9	Results of converted speech models on <i>Human3.6M</i>	156
8.10	Zero-shot transfer capabilities from a large mixture of datasets	158
8.11	Influence of skeleton joints generated by a 3D-pose-estimation model	158
8.12	Improvements through artificial noise pre-training	159
8.13	Improvements through unsupervised learning	159
9.1	Results on the <i>SmartLights</i> dataset	173
9.2	Accuracy on the <i>SmartSpeaker</i> dataset	173
9.3	Accuracy on the <i>FSC</i> dataset	174

Own Publications

1. **KoARob: Towards AI-based Safety in Human-Robot-Collaborations.** Bermuth, Daniel and Poeppel, Alexander and Reif, Wolfgang. August 2025. In *2025 IEEE 21st International Conference on Automation Science and Engineering (CASE)*. Pages 837-844. © 2025 IEEE. [48]
2. **VoxelKeypointFusion: Generalizable Multi-View Multi-Person Pose Estimation.** Bermuth, Daniel and Poeppel, Alexander and Reif, Wolfgang. Preprint. October 2024. [47]
3. **SimpleDepthPose: Fast and Reliable Human Pose Estimation with RGBD-Images.** Bermuth, Daniel and Poeppel, Alexander and Reif, Wolfgang. Preprint. January 2025. [50]
4. **Tutabo-1: Towards Real-time Capable AI-based Safety Systems for Human-Robot Collaboration.** Bermuth, Daniel and Poeppel, Alexander and Reif, Wolfgang. December 2025. In *2025 IEEE International Conference on Advanced Robotics (ICAR)*. Pages 632–639. © 2025 IEEE. [51]
5. **RapidPoseTriangulation: Multi-view Multi-person Whole-body Human Pose Triangulation in a Millisecond.** Bermuth, Daniel and Poeppel, Alexander and Reif, Wolfgang. Preprint. March 2025. [52]
6. **Skarimva: Skeleton-based Action Recognition is a Multi-view Application.** Bermuth, Daniel and Poeppel, Alexander and Reif, Wolfgang. February 2026. In *The 20th IEEE International Conference on Automatic Face and Gesture Recognition (FG)*. © 2026 IEEE. [53]
7. **Scriboora: Rethinking Human Pose Forecasting.** Bermuth, Daniel and Poeppel, Alexander and Reif, Wolfgang. Preprint. November 2025. [49]
8. **Methodiken zur Analyse und Verbesserung der Interaktionsqualität in Mensch-Roboter-Kollaborationsanwendungen.** Bermuth, Daniel. Master's thesis, October 2020. [42]
9. **Jaco: An Offline Running Privacy-aware Voice Assistant.** Bermuth, Daniel and Poeppel, Alexander and Reif, Wolfgang. March 2022. In *17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. © 2022 IEEE. [46]
10. **Scribosermo: Fast Speech-to-Text models for German and other Languages.** Bermuth, Daniel and Poeppel, Alexander and Reif, Wolfgang. Preprint. October 2021. [44]
11. **Finstreder: Simple and fast Spoken Language Understanding with Finite State Transducers using modern Speech-to-Text models.** Bermuth, Daniel and Poeppel, Alexander and Reif, Wolfgang. Preprint. June 2022. [45]
12. **Slungt: Even Faster Spoken Language Understanding with N-Grams and Tries.** Bermuth, Daniel and Poeppel, Alexander and Reif, Wolfgang. April 2025. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. © 2025 IEEE. [43]