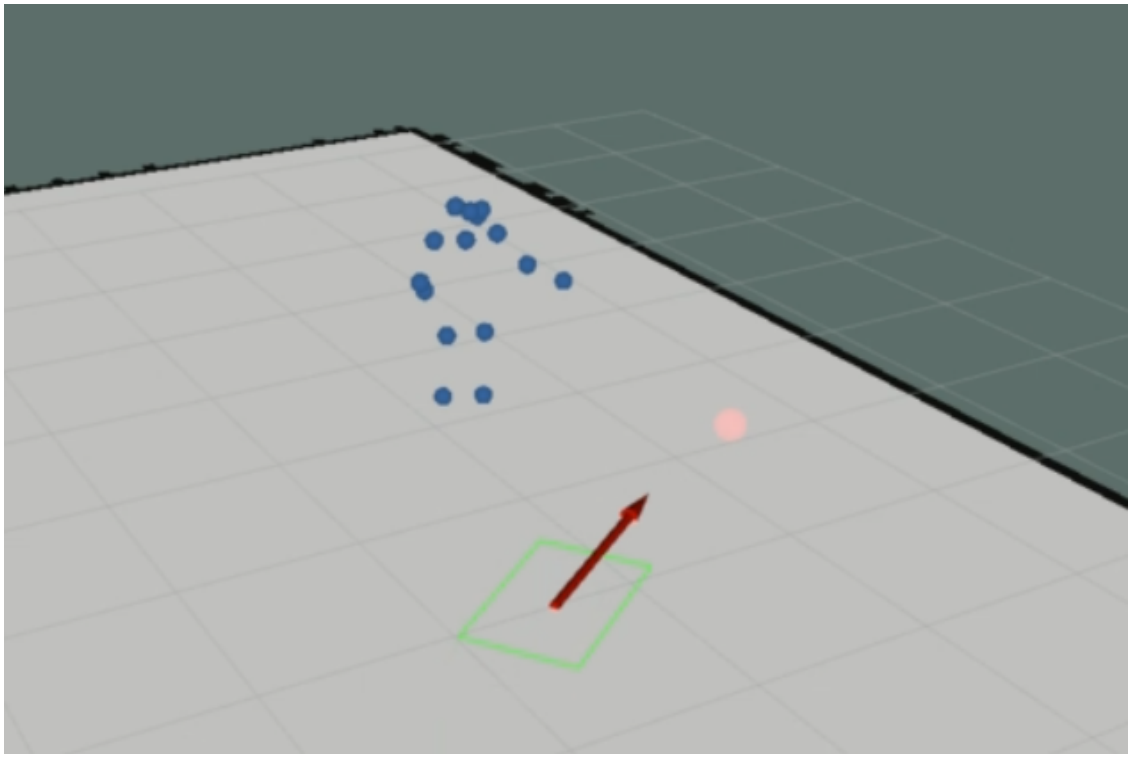


Evaluation of Human-Robot Interaction with Autonomous Mobile Robots using Human Pose Estimation

Master's Thesis in Systems Control and Mechatronics

DANIEL JAKOBSSON & MAX VAHLSTRÖM

Department of Some Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021

Master's Thesis 2021

Evaluation of Human-Robot Interaction with Autonomous Mobile Robots using Human Pose Estimation

DANIEL JAKOBSSON & MAX VAHLSTRÖM

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2021

Evaluation of Human-Robot Interaction
with Autonomous Mobile Robots
using Human Pose Estimation
DANIEL JAKOBSSON & MAX VAHLSTRÖM

Supervisor: Kristofer Bengtsson, Department of Electrical Engineering
Supervisor: Simon Bokesand, Kollmorgen Automation AB
Examiner: Knut Åkesson, Department of Electrical Engineering

Master's Thesis 2021
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Visualization of directing a robot by gesture.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

Evaluation of Human-Robot Interaction with Autonomous Mobile Robots
using Human Pose Estimation

DANIEL JAKOBSSON & MAX VAHLSTRÖM
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The aim of this master thesis is to evaluate possible improvements of the perceived behavior of an Autonomous Mobile Robot (AMR) system using an RGB-D camera for Human Pose Estimation (HPE). This topic is approached by conducting an initial study of the current challenges and needs of humans working in shared local environments with AMR-systems. A study on how different HPE methods perform in an AMR system has also been conducted, which show that different methods could have very differing performance. Based on the results of the studies, a prototype implementation of some interaction concepts is presented, including the understanding of the human intentions as well as how an AMR should respond to these intentions.

The results of this thesis show that there are common issues for human users who are working along with current AMR systems. A proof of concept on how some of these issues could be solved is proposed, which potentially could improve the perceived behavior of the AMR significantly. Suggestions of HPE methods that have shown the most promising performance from tests is also given.

Keywords: Human Pose Estimation; Human-Robot Interaction; Computer vision; Human-Aware Navigation; Autonomous Mobile Robots;

Acknowledgements

We wish to extend our deepest gratitude to Kollmorgen Automation and specifically the AMR team for having us during the work in this master's thesis. Simon Bokesand has assisted us in solutions and Mikael Bohman has been of great help in anything software related. Johanna Turesson has helped us with analyzing the market and Zihua Yang has assisted us with her UX-expertise. Their expertise and guidance have been of great assistance all the way from the formulation of the problem to the end result.

We also would like to thank our supervisor at Chalmers, Kristofer Bengtsson, for his invaluable insight and guidance during this project. It has been a pleasure having him as a supervisor.

Thank you all very much!

Daniel Jakobsson & Max Vahlström, Gothenburg, March 2021

Contents

List of Acronyms

AGV Automated Guided Vehicle
AI Artificial Intelligence
AMR Autonomous Mobile Robot
AP Average Precision
CPU Central Processing Unit
CV Computer Vision
DL Deep Learning
DoF Degrees of Freedom
FoV Field of View
GPU Graphical Processing Unit
HMI Human-Machine Interaction
HPE Human Pose Estimation
HRI Human-Robot Interaction
IMU Inertial Measurement Unit
LiDAR Light Detection and Ranging
OpenVINO Open Visual Inference and Neural network Optimization
PAF Part Affinity Fields
ROS Robot Operating System
TCP Transmission Control Protocol

1
Introduction

Automation is a continuously growing field in today's society. One part of this growth is the use of Automated Guided Vehicle (AGV) systems for automating internal logistics in factories which has increased a lot during the last decades and will continue to do so in the future [1]. This has led to significant changes in the work environment for human logistic operators who need to adapt to the new technology since AGVs are often run in the same local environment. There are various forms and sizes of AGVs, where some are large and heavy which could make people uncomfortable if they are not behaving in a way that feels intuitive and safe for human presence. It is for example common for AGVs to not slow down or stop for an obstacle until a certain security distance where it uses emergency brakes to stop, which is a behavior that is generally not interpreted as intuitive or safe for humans.

Due to the vast development of AI and microprocessors during the last decade, AGV systems can now sense their surrounding environment and interpret their sensor data in new ways that allow them to distinguish humans from other surrounding objects. The potential improvement of using this ability is however not yet fully understood. If AGVs would be more adaptable to human behavior and give the possibility for people to interact with them in an intuitive way, it could ease the working situation for human users who are working in the same local environment significantly. This thesis project is therefore investigating the possibility of an interface that enables AGVs to adapt to human presence and lets people perform basic interactions with the AGV.

1.1 Background

Kollmorgen Automation AB is a company that during the last decades has been developing world-leading software solutions for AGVs [2]. In parallel with their development in this field, they have recently also started looking at solutions for Autonomous Mobile Robots (AMRs). The major difference compared to the AGV is that the AMR is controlled completely by hardware and software located on the robot itself, while the AGV requires support features to navigate e.g. reflectors placed in its surroundings. This gives AMRs increased freedom and flexibility in the path planning compared to the AGVs which are only driving in predefined lanes.

AGVs have mainly been using a two-dimensional laser scanner for localization and

have not had the flexibility to change its route if there is an obstacle obstructing it. Instead, AGVs are relying on that the surrounding environment does not put obstacles in its way, and if it does the AGV simply stops. AMRs are however flexible enough to detect and avoid obstacles in real-time, which requires some additional sensors to the laser scanner since it only detects objects located at the same height on which the laser scanner is mounted on. To be able to see obstacles with different heights an RGB-D camera can then be used as a complement for obstacle detection.

Due to the large progress in Deep Learning (DL) during the last decade as mentioned above, the use of an RGB-D camera enables additional possibilities since it can be used for detecting more specific objects than just obstacles in general. This opens up a potential of an adjustment of the behavior of the vehicle depending on what kind of obstacle is in the way, which is a potential that Kollmorgen would like to examine.

1.2 Aim and research questions

This project aims to examine the possibilities regarding single- and multi-person Human-Machine Interaction (HMI) and human-aware navigation using an RGB-D camera on the AMR. Using Human Pose Estimation (HPE) the AMR could potentially be able to interpret for example a human's intention, if a human is trying to communicate with the AMR or if humans are likely to make room for the robot or not. These abilities could make the AMR flexible and adaptable in an environment populated by human operators, which could make the work environment more convenient for the operators. The expected result of this project is a conceptual showcase of what improvements can be made to the behavior of the AMR system when using HPE and human-aware navigation functionalities.

The research questions that should be answered in this thesis were the following:

- Q1: From the perspective of a person working in the same area as AMRs, what features could improve the behavior of the AMR system using HPE?
- Q2: Among available Deep Learning alternatives of online Human Pose Estimation, what are the strength and weaknesses of the different approaches? What method is suitable to be used in real-time on an AMR system?
- Q3: Given a functional HPE system, how should the interactions according to Q1 be designed, and is the AMR response intuitive from a human's perspective?

1.3 Method

To retrieve an understanding of the challenges when working in environments around AMRs, an assessment was conducted with people working like that. The aim was to determine whether people have encountered problematic situations with AMRs, if they have come across any thoughts of what they think would be good features to have regarding HMI up until now, and finally what kind of features they would like to see in the future if they would be given the task of designing them. Additionally,

inputs and ideas have been taken from different people such as supervisors at both Chalmers and Kollmorgen, and also the sales department at Kollmorgen who knows what their competitors have been working on. These inputs resulted in a set of interaction concepts to be implemented and further examined.

The interaction concepts were the basis of what requirements there were on the performance of the HPE methods. A study on different HPE methods was then conducted to examine which ones fulfilled those requirements the best. Since the hardware of the AMR system consisted of a CPU only, the methods were primarily evaluated when run on a CPU. Additionally, a perspective on how much the performance could improve if one would decide to add a GPU to the system has been given by a comparison between methods run on a GPU. This gives guidelines on which kind of HPE models are most suitable to work with for real-time estimation applications, and also what possible benefits there could be to add a GPU to the AMR hardware.

Based on the input of what problems there were for humans located in AMR environments and the capabilities of the methods that were examined, a proposed implementation of the interaction concepts has been developed. The implementation has been evaluated based on the resulting changes in the behavior of the AMR system. A final recommendation has then been presented to Kollmorgen on how the use of HPE could improve the current AMR solution.

1.4 Scope

The scope of this thesis was to evaluate what potential improvements that the use of HPE could give to an AMR system from a user's perspective. The initial study was mainly based on interviews with people who in different ways have experience working with AGVs or AMRs. Due to an ongoing pandemic of Covid-19, the study has been limited to actions that are done remotely. Therefore, no visits have been made to sites using AGV or AMR systems, and the interviews have been conducted online. The study was also limited to inputs and experiences that have been retrieved by people at Kollmorgen or the use of products given out by Kollmorgen.

The thesis also conducts a study on what different DL methods for HPE there are and how well they are suited to run in a real-time application on the AMR system. No design or training of DL models has been done during this thesis. The study is therefore limited to DL methods for HPE that are already designed by others and available online as open-source software. It is also limited to previously trained model graphs that are available along with the open-source software for each DL method. The study is done from the perspective of what methods are most suitable for a real-time application on the AMR system at Kollmorgen. The study is therefore based on the current hardware available on this particular AMR.

An implementation of the retrieved feature concepts was conducted. It was designed to be easily integrated into the current software that Kollmorgen has in its AMR

system specifically.

1.5 Previous work

The task of navigating a robot based on humans is called Human Aware Navigation. It intersects the fields of Human-Robot Interaction as well as robot navigation [3]. The robot is developed within the Robot Operating System (ROS) framework which has many useful tools for movement such as navigating a robot to a desired position.

However, the field of Human-Robot Interaction is less generalized. The task of interpreting human data and deciding what to do varies from purpose to purpose. The design of these interactions will be a focus of this thesis.

To gather data about the humans in the robot surrounding, the cameras will be used to estimate the poses of each person in the field of view of the robot's camera. The task of interpreting human poses is called Human Pose Estimation (HPE) which is a young field with many available methods to choose from. A second focus of this thesis is an evaluation of which of these methods to use.

In the field of Human Aware Navigation, research has been done in improving the path planning around moving humans both using Light Detection and Ranging (LiDAR) scanners for tracking movement of a whole-body [4] as well as eye-gaze glasses for predicting the movement of a whole body[5]. Additionally, work has been done toward incorporating detected humans into the navigation stack in Robot Operating System (ROS) [6].

When a user wants to communicate with a drone, one study shows that the primary mode of communication without instruction is by using gestures [7]. One important gesture used in this thesis is a pointing gesture. An earlier work successfully gave directions to a robot in real-time using hand gestures. In that implementation, they defined the pointed direction by the hand direction by having detected a hand silhouette [8]. Another work used an HPE-method within ROS to detect what object a user is pointing at. In this work, the object pointed at is an object that coincides with the 2D line formed by the forearm drawn on each frame [9].

Not much research available on the usage of HPE for Human-Robot Interactions has been found. Two relevant studies that used gesture control are [8] and [9] who used different ways of estimating the point a user was directing a robot to.

1.6 Ethical implication

The result of the features developed and implemented in this thesis is used to facilitate the integration of AMRs in environments where humans are present. The goal of the thesis is a system that is perceived as safer, and one that promotes the integration of them into a wide array of sectors. Further, the development of better

human-machine interactions with AMRs can contribute to the development of better interactions with autonomous road vehicles, and this can in turn lead to fewer road accidents. The *three Ds of automation* [10], the Dirty, the Dangerous, and the Dull jobs are the jobs that people envision that robots will replace humans in. In the case of the AMR, the job that is automated can be considered a dull job, and for many jobs in this category, one can argue that they should be automated.

The United Nations has defined 17 sustainability goals [11] that Chalmers is interested in working towards. The nature of one of these goals is to ensure healthy lives and promote well-being for people of all ages. A target to reach this is to reduce the total amount of road accidents [11], and all advancements in the interaction between humans and autonomously mobile robots work towards this goal. Further, an advantage of the AMR system is its usability in non-factory environments, such as healthcare facilities. The automation of tasks in these areas could provide more affordable healthcare which is another of the goals [11]. Lastly, a third relevant goal is to promote sustained inclusive and sustainable growth. A target for reaching this is by fostering increased productivity in labor-intensive sectors [11], where logistics is one of these.

2
Theory

The topic of this thesis spans many relevant areas. In this chapter, we will describe the relevant theory to understand and motivate our methods and conclusions.

2.1 Automated guided vehicles

An Automated Guided Vehicle is a mobile type of robot which has become an important part of the increased automation of the industry. An AGV is usually a logistics robot on wheels with the purpose of transporting material from one point to another. There are multiple ways for AGVs to facilitate localization, for example by external markers, LiDAR, vision cameras, and magnets. It is also common to fuse different kinds of sensor data together to get a better interpretation of the surrounding environment. Based on the interpretation a control algorithm calculates suitable driving paths for the robot to enable it to go from one place to the other.

There are many different types and applications for AGVs. They are in general used in industrial applications for transporting materials in a systematic way. They can be deployed in many different environments for example warehouses, offices, factories, etc. It is common for AGVs to be equipped with some kind of additional feature such as lifting mechanisms. A common application is the AGV version of a forklift, which does the same thing as a manually driven forklift but instead controlled autonomously.

2.1.1 History of AGVs

The first era of AGVs came in the early 1950s where vehicles first were configured to follow wires drawn above or below it. They AGVs navigated with simple track-guided systems and tactile sensors which were controlled with mechanical switches [12]. A few years later the AGV business was started in Europe and Japan as well. Until the 1970s the systems used simple electric and magnetic sensor systems which performed sufficient results. However, the systems did not offer any flexibility, and there were big limitations for the actions that the vehicles could take since they practically could only go in one direction along a predefined track [13].

The second era of AGV systems took place from the 1970s until the early 1990s where electronics were introduced on the computers in simple onboard computers

and control cabinets for block section control. This was a time when big advances were made in electronics and sensory technology, which was something that the manufacturers of AGV systems took advantage of. Technical innovations for example high-performance electronics and microprocessors and more powerful batteries increased the possibilities for the manufacturers and improved the performance of the systems even more [13]. During the 1980s, breakthroughs in laser technology were also made, which revolutionized the AGV industry. This vastly increased the ability to change the layout at the factory site without major changes in the software of the AGV systems [14]. It was also in this era where the company Netzler & Dahlgren Co AB started working on their first AGV systems, which is the company that has evolved into what Kollmorgen Automation is today [15].

The third era lasted from the mid-1990s to 2010. At this time standards for AGV systems were set, and the devices have electronic guidance and contact-free sensors. In this era, the AGV systems became acknowledged as reliable and powerful autonomous solutions that gave economic profit overtime for the customers. AGVs were developed to be able to handle almost any kind of load, which broadened the applications from mainly the automotive industry to a big variety of customers.

2.1.2 Pros and cons with AGVs

One reason for a business to implement an AGV system is that it reduces labor costs over time since an AGV is a single expense that will then perform work that otherwise would be done by a human. A human is usually significantly more expensive to hire over time. Using AGVs could also increase the safety at the worksite since AGVs are programmed with safety mechanisms in contrast to manual work where accidents happen from time to time. Since AGVs in general are cheaper over time than hiring humans, it could also increase the accuracy and productivity of the business when being able to deploy more work units for the same price [16].

A problem with deploying an AGV system is that it comes with a high initial investment, and it takes time for the investment to pay off. Depending on the quality of the system there could also be maintenance and reparation costs from time to time. AGVs are also, along with most robotic systems, not suitable for non-repetitive tasks where the outside of what the AGV is programmed to do [16].

2.1.3 Autonomous Mobile Robots

An Autonomous Mobile Robot (AMR) is a special kind of AGV which has the property of not being dependent on any external features for example a fixed driving path or external reflexes. Instead, AMRs navigate completely based on internal software and the environment given by sensors located on the robot itself to find an optimal path to its target destination. This property makes AMRs more flexible than AGVs which must follow preset routes and enables a lot of potentials when designing the behavior of the AMR [17]. The flexibility eases the requirements from manual workers when putting load or material such that the robots could pick them up, where

AGV systems usually require very precise placement. It also enables algorithms and applications of Artificial Intelligence for optimizing the performance of the AMRs, for example, it could learn to drive in spaces that are not used as much or where it minimizes the traffic. A downside with not having external support for the sensors such as reflexes is that the measurement errors that occur in the sensors are bigger, which leads to a decreased accuracy of positioning of the AMR in relation to surrounding objects. This limits the ability of AMRs to perform tasks that require high precision and accuracy when approaching a load or target.

Typical applications of AMRs are transporting material or load within factories and warehouses. This is a time-consuming and simple task for workers, which easily can be automated. While AGV systems are effective, they are typically labor, floor space, and capital-intensive. This can then be solved by AMR systems instead, which are designed to fill the efficiency gap in functionality [17]. The flexibility of AMRs also enables more collaborative behavior with human workers. Order picking is an expensive task that is performed in most warehouses or factories, where humans have to walk between shelves to fetch material which is very time-consuming. With AMR systems the robots could instead make the shelves come to the worker and then put it back into the storage structure, which increases productivity significantly. Another solution for order picking could be that the AMR follows next to the human to assist with storage possibilities or tools needed to perform the task effectively. When this is done to pick parts for a kit, the process is called *kitting*.

2.2 Computer vision

The term Computer Vision (CV) is an active field of research that refers to how computers and autonomous systems can interpret information from digital images or videos. One could say that the famous quote "A picture is worth a thousand words" is literally taken into the application here [18].

The development of CV is believed to have started at universities that were researching the field of Artificial Intelligence in the late 1960s [19]. To make the computer system able to mimic the behavior of a human, they wanted to give the computer the ability to see like the human visual system does by attaching a camera to the system and then to try to make the system able to interpret information from the images [20]. The difference between CV and the already existing field of digital image processing was the attempt to extract 3-dimensional objects from the scene to achieve a full understanding of the image. Several fundamental techniques of CV algorithms were later found in the 1970s for example edge detection, labeling of lines, and motion estimation [19]. Many of these techniques are still used today.

2.2.1 Applications of CV

There are applications of CV in several fields where different kinds of information can be retrieved. It might for example be how far an object is to a camera if a vehicle drives in the middle of its lane or how many people are in the view of a camera.

These kinds of questions and many more are challenges that are included in the field and should be answered based on the information given by digital images and videos [21]. Apart from the field of autonomous vehicles, there are also applications of CV in other areas like medicine and the military [22].

Typical tasks to solve in CV are for example object detection, classification, and identification. In object detection, the challenge is to identify whether there are objects, such as obstacles, in an image. In object classification, the task is to recognize certain pre-specified object classes in images, for example, cars, houses, humans, etc. In object identification, an individual instance of an object is recognized for example identifying a specific person.

2.2.2 3D Computer Vision

When working with CV it is convenient to work with three-dimensional computer graphics to model the scene. This could be done by using RGB-D cameras, which compared to normal RGB images has the feature of offering an estimated depth value for each pixel in an image. In the Intel Realsense camera this is done by using multiple camera modules together with a projected laser pattern to capture the depth in the image [23]. From these kinds of cameras, one can retrieve several different three-dimensional scene representations. One can for example use point clouds which is a set of data points plotted in space, or depth maps which are images that have a certain value of each pixel in the image depending on the distance to the camera. From these kinds of image representation, one can model a scene in 3D which is very useful in applications such as autonomous vehicles.

2.3 Robot Operating System

The Robot Operating System [24] is a flexible framework for developing robot software. It is an open-source collection of tools and libraries that has been created to simplify the development of complex and robust robotic systems in a wide field of applications [24]. Since ROS came in the late 2000s, researchers generally do not bother to create their own robotic system from scratch anymore, but instead uses the libraries and tools that are continuously built and shared within the ROS community. It has also encouraged robotic researchers to share their code as open-source, which has sped up the development of more innovations and findings in the field [25].

2.3.1 Nodes

A ROS system generally consists of a set of *nodes* which is computation processes written in for example C++ or Python. A complete robot control system will often consist of several nodes to control different components of the system. This means that the nodes can work separately, regardless of programming language. Communication between nodes is handled through *topics*. Topics are communication channels where data is sent in the form of *messages*. Messages are a simplified description

language in ROS for describing data values. An overview of the schematics of a simple ROS application can be seen in Figure 2.1.

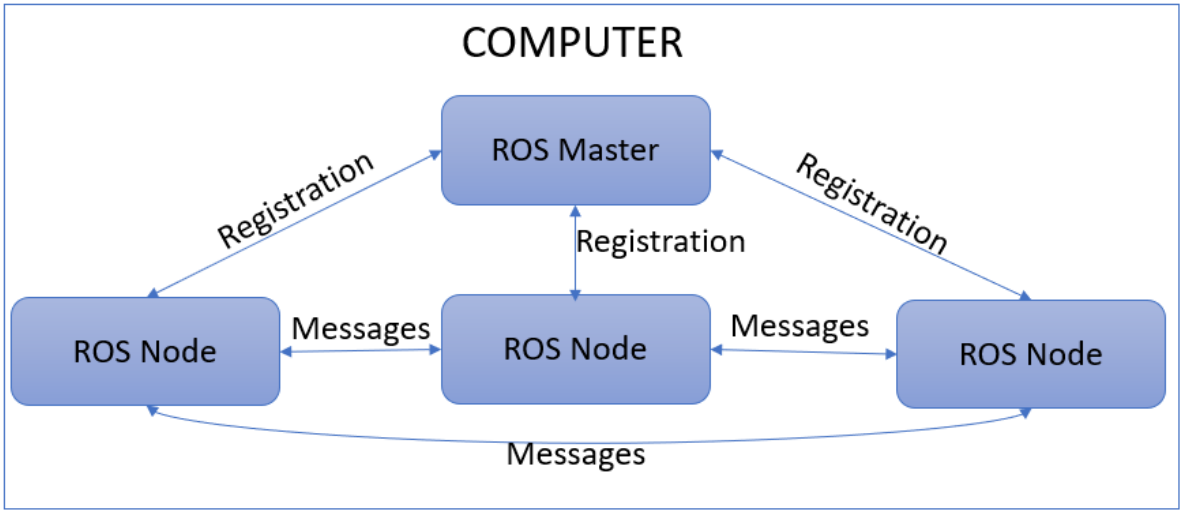


Figure 2.1: ROS schematics

2.3.2 Node to node communication

The most fundamental way of sending messages is through publishers and subscribers. This message stream is one way and suitable for continuous streams of data [26]. Actions, on the other hand, are useful for example sending movement commands or other messages that take time to execute and where a continuous update on the status is needed [26]. The state of an issued action can be listened to and overridden. All messages between nodes are sent through Transmission Control Protocol (TCP) which allows for distributing the nodes, not only in separate processes in a single computer but also across network-connected computers. The translation to ROS messages and the sending of the actual data, instead of pointers to data, comes with a degree of overhead computational cost and latency. Nodelets can be used to avoid this by sending pointers to the data instead, but this is only possible between nodes that are both on the same hardware and written in the same language [27].

2.3.3 ROS Navigation Stack

The robot has a Navigation Stack package called *move_base* that handles the full chain from sensor output to feasible and collision avoiding control command messages. This Navigation Stack enables one node to send desired go-to points as actions that will automatically be carried out, and if necessary canceled.

2.4 Deep learning

Deep Learning is a type of method in the family of machine learning methods that learns to perform complex tasks. It is based on the concept of artificial neural

networks, which are computing systems that are inspired by the biological neural networks which form animal brains [28]. Deep learning can be divided into three categories supervised, semi-supervised, and unsupervised learning [29]. Supervised learning is used when one has a big set of data that is used to train the system for example predicting a certain object in an image. When training a model supervised, there is always a ground truth for the model to compare with. This ground truth is used to see how the model performs and to tune the model based on it[29].

Unsupervised learning is used when one wants a computer system to learn behavior or skill from scratch [30] based on a given environment that it interacts with for example a computer game. The system will then have some reward system that gives it a positive reward when it does something considered as good and vice versa. Semi-supervised learning is a mix between the previously described methods where the model is given some data with a ground truth included, and then tries to learn by itself from that point [31].

2.4.1 DL on CPU vs GPU

An essential aspect when running a DL model is what hardware it will be run on. The Central Processing Unit (CPU) is the processor which is the main controller in computers today [32]. They are designed to handle a wide range of tasks quickly and executes tasks in sequential order. In comparison, the Graphical Processing Unit (GPU) was initially created to perform fast calculations to render images for computer graphics and video games [32]. The GPU consists of many small processing cores that can execute calculations in parallel, which makes it a powerful tool for specific tasks where parallelism is beneficial. Machine learning and AI are tasks of this kind since they are based on calculating big sets of data [32].

Because of this, a GPU can however not fully replace a CPU since it lacks the task scheduling abilities and the ability to run multiple general-purpose computing tasks simultaneously that the CPU has. The GPU must therefore be considered as a complement to the CPU [32], which from a user's perspective is an additional expense that has to be taken into consideration when designing a system.

2.4.2 Convolutional neural networks

Convolutional neural networks (CNN) are a type of artificial neural network that is mostly used for pattern recognition when analyzing digital images [33]. As the name implies the difference with this architecture is that there are convolutional layers, which are layers that are performing convolutional filtering on the matrix representation of an image. This is done by defining small squared matrices for example 3×3 which convolves over the entire image matrix and computes the dot product for each 3×3 matrix in the image [34]. Simply put, one could think of convolutional filtering as pattern detectors, which detect certain features like edges, squares, or circles in an image when applied multiple times on the same image. This property makes CNNs suitable for applications such as image recognition and

objects classification [35]. It is, therefore, a very appropriate tool for this project.

2.4.3 The evolution of Deep Learning

The early foundation for deep learning can be traced back to the 1940s, where advancements were made in *cybernetics*. This was a field of research that was trying to understand how the brains of humans and animals learn and to replicate this functionality in computational models [36]. In cybernetics, it was then attempted to mimic a biological neuron with the *McCulloch-Pitts Neuron*, which was one of the first attempts of mathematically modeling the biological counterpart [37]. The introduction of cybernetics yielded a big interest in self-organizing systems during the 1950s and 1960s. In the 1950s, an electrical machine called *Perceptron* was developed by Frank Rosenblatt which was designed to perform image recognition and learned how to classify patterns into categories for example letters of the alphabet. This was a remarkable machine at the time but did not fulfill all the bold claims that were made [38].

Connectionism [39] was another field of research that became popular in the 1980s, which is an approach of cognitive science that tries to explain intellectual abilities using artificial neural networks. It was during this time the concept of hidden layers in neural networks was introduced, which is layers of neurons in between the input and the output of a network. Significant improvements were also accomplished in back-propagation and other concepts were made which are still key components in DL. In the mid-1990s a wave of startups based on AI occurred, but due to the lack of computational resources, it was hard to deliver sufficient products which led to a dip in interest [36].

Another breakthrough in DL was made in 2006 when Geoffrey Hinton showed that many-layered feedforward neural networks could be trained one layer at a time [40]. This discovery enabled researchers to train neural networks with more and more hidden layers in them, which led to the popularisation of the term *Deep* Learning. In parallel, during this time a large increase in computational power was achieved, which improved the ability to train the networks by using larger and larger datasets [36]. With a continuously increasing number of people using online services in the world, more and more data is produced which could be used for training DL networks.

For a long time, deep learning was a field of research that made significant advances but was not very popular in the industry due to unreliability and limitations in what results it could achieve. By 2011 the computing power of GPUs had increased significantly, which made it possible to train deep neural networks more effectively [41]. It was therefore first during this time that the combination of deep CNNs and GPUs made progress on computer vision [42]. Until 2011 CNNs was not performing any better than other shallow machine learning methods, but in 2012 it could be shown that CNNs run on GPUs could improve the performance of DL in CV dramatically [43]. Since this time the use of DL has outperformed other alternative

methods, and many vision benchmark records have been set and continuously gets increased with better and better approaches coming up.

2.4.4 Machine learning inference

The process of running live data points through a trained machine learning model is called inference [44]. When training a machine learning model, the inference is always performed since it is performing estimation based on the training data. The difference when running pure inference is that the model is no longer evaluating and trying to improve its own performance, but instead only produces the specific output it is designed for such as a numerical value or a position of an object in an image [44]. The data running through the model is then no longer the training and test data that has been used during the development of the model, but instead data from an actual application that the model was designed to execute. A machine learning model normally requires the input data structured in a specific way to be able to deploy, such as a certain size and format of an input image given to the model. The input data to a system, therefore, has to be modified before it can be executed through the machine learning model [44]. When performing inference the model is usually a part of a bigger application, where there are processes that are depending on the output data and are performing actions based on it.

2.4.5 Optimization

Machine learning models designed for difficult tasks can consist of dozens or hundreds of layers and millions or billions of weights connecting them. They are then very large and complex to work with, and the larger the model is, the more computation time, memory, and energy is consumed. Depending on the hardware the model is running, this could cause a delay of the response time (also called *latency*) from inputting data to the output. It could be useful to tune the parameters and weights during training, but when performing inference it might be required for the model to run at low power or on another type of hardware. Then it could instead be useful to simplify the model in order to decrease the number of calculations and reduce the latency, although it might cause a reduction in estimation performance [45]. In industrial applications, the software is often running on industrial PCs that can have limitations in the hardware and power supply available. This means that optimization might be needed to get a model to run on a sufficient frequency to be able to perform its intended task.

There are different ways to optimize machine learning models to maximize their performance in relation to power consumption and latency. *Pruning* is a method where each artificial neurons individual contribution to the network is evaluated. If a neuron is rarely or never used, it could be removed from the network without any significant difference in the output. This way the size and complexity of the model could be reduced and the latency improved. *Quantization* is another method that reduces the numerical precision from the weights from for example a 32-bit floating-point to 8-bit, which also reduces the size of the model and speeds up the

computation [45].

2.5 Human Pose Estimation

Human Pose Estimation (HPE) is the CV task of deducing the pose of a human from sensors, such as cameras [46]. HPE can be applied to topics ranging from video games to Human-Robot Interaction (HRI). The task of HPE for a clearly visible pose has several problems partly due to the many poses a human body can have. This can cause self-occlusion when a body part of one person blocks the view of another body part for the same person, as well as a vast amount of possible poses to estimate. The introduction of obstacles blocking the view of a pose leads to the risk of one or more joints being occluded, which puts more demand on the HPE task. Also, the human body can come in various shapes and forms and a general pose estimation method needs to handle this as well as the different types of clothes a person can wear [46].

2.5.1 Methods for HPE

In the earlier works of HPE, traditional machine learning on simple image-features was used to estimate a human pose [47]. These features were hand-crafted, meaning that the features were defined manually. However, these methods were not sufficient at estimating the human pose accurately [47]. According to [47], more recent work utilizing deep learning models has shown to be capable of extracting better features leading to better performance by a large margin.

The human body has many attributes ranging from how its joints are positioned to the texture and color of the body surface. The model of a human does not need to carry all available information, only the information needed for the specific task [46]. Three common models are the skeleton-based model [46], the contour model [46] as well as the volume-based model [46]. If only the pose of the body is of interest, then the skeleton-based model is simple and flexible [46]. In the case of the skeleton model, the HPE task is accomplished by breaking down the full human body into distinct parts, called keypoints, such as the joints of the arms and legs, and then trying to estimate their position in the frame.

The keypoints position can be estimated directly through a regression approach [46], or indirectly through a detection-based approach [46]. In the detection-based approach the keypoint is estimated as a heat-map, sometimes called a confidence-map. Using a heat-map, the keypoint position is expressed as a distribution where the peak of the heatmap can be taken as the position of the keypoint [48]. In addition to the position, the value of the peak carries information about the confidence of the estimation, which can be used to determine the quality of an estimate. The regression approach in turn is easier to use but more difficult for the network to learn [47] and is, therefore, a less common approach.

2.5.2 Distinctions between current approaches

There is no guarantee that only one person is being in front of a camera. The HPE algorithms, therefore, need to handle multiple people being located in the image. There are two main ways of tackling this problem, the top-down and the bottom-up approach [47].

The top-down approach generally starts by estimating a bounding box around each present person [47]. After this, a single-person pose estimator on each person is used. The run-time of this approach tends to increase linearly per added person since a pose estimation algorithm needs to be run once on each found person. This is problematic when running in real-time. Another problem with the top-down method is due to its early commitment [48]. This means that if no bounding box of a person can be found despite some or all of the pose is in the frame, the next step of estimating the key points within that box can not be performed either [48].

The bottom-up method on the other hand starts by estimating the position of every keypoint present in the frame [48]. After this is done, the algorithm links the key points together appropriately to form each pose. This solves the early commitment problem of the top-down approach while introducing new challenges in knowing which keypoints correspond to which pose [48].

HPE is most commonly performed in 2D, however, there are also methods for 3D-HPE. This additional information is useful for 3D-based interactions as well as to get more useful data for analyzing an estimated pose and its feasibility. Previous methods have done this by using 2D pose estimates and from this regressing the depth data using DL methods [47], [49], [50]. In [50] for example, the model is trained for a monocular RGB-camera with ground-truth data for keypoints in 3D and is robust against occlusions. Another possible method is to take the RGB 2D-HPE and fuse it with depth data to transform it to 3D-HPE.

2.5.3 Training and evaluating HPE

Each solution is often trained on publicly available datasets, and there are competitions each year for the most accurate solution. One difference is which keypoints are included, and therefore which keypoints can be estimated. Two commonly used datasets are the Microsoft COCO dataset and the MPII dataset, where one difference is that the Microsoft COCO dataset has keypoints for eyes and ears while MPII only has one keypoint representing the entire head [47], [51]. These additional keypoints can be useful when for example head-pose or eye position is of interest. There are many metrics for evaluating the performance of a dataset. One common method is the Average Precision (AP), which defines a true positive as being within a certain threshold around the ground truth [47].

2.6 Human Robot Interaction

The study of Human-Robot Interaction is a vast field of research. One area of HRI is how an interface between AMRs and humans located in the same environment could be designed. The interactions can range from simple interactions such as a person stepping in front of an AMR causing it to halt abruptly, to the AMR perceiving a person and changing its behavior according to certain actions of the person.

2.6.1 Human Aware Navigation

The field of Human Aware Navigation (HAN) can be described as the intersection between HRI and robot motion planning [3]. The field has the following goals relevant to this thesis:

- To increase the comfort for both active users and passive pedestrians
- To make the robot move in a similar way as humans do.

The usability of AMRs relies on their ability to navigate around humans. Certain safety elements are used, making the robots safe enough to interact in environments where humans are present. However, no matter how safe the AMRs are, the perceived safety from the perspective of a person is of high importance where actions such as sudden stops of the robot can be seen as unsafe [6]. If the AMR is perceived as unsafe, the human could lose trust in the technology [3] which can, in turn, make the incorporation of an AMR into the workflow more difficult.

Not only perceived safety is of importance, but for the AMR to be integrated into an environment populated by people there is a need for Socially Compliant navigation. In a very basic scenario, the human is not distinguished from other objects and the navigation of the robot is simply optimized around this human, if possible. However, when people navigate around each other the interactions adhere to some sort of cooperation which enables smooth avoidance of collisions [52]. An interface for humans to communicate with an AMR would allow for easier cooperation which in turn would allow for more socially compliant behavior.

Human-Robot Interactions will in this thesis be divided into two main categories: *Active* interactions when a user actively gives commands to the AMR and *passive* interactions when the AMR perceives a person and acts accordingly. Further, a similar distinction will be made between pedestrians and users: Pedestrians are not actively trying to interact with the AMR while users are actively trying to engage the AMR.

The most common approach when researching comfort is by identifying uncomfortable behaviors and then correcting them [3]. *Proxemics* is the study of distances in interactions, and one goal is to determine what robot-human distance is perceived as comfortable. If for example a robot is set to approach a human, studies show that even with a human-sized robot approaching, users find it comfortable when approaching as close as to $0.5m$ [53].

When making a robot behaves like a human, the approach is to identify the pa-

rameters of the desired behavior [3]. These parameters could be the movement speed and time between action and reaction. There are higher-level decisions as well, such as which side of a person to pass on that also should adhere to human convention.

2.7 Inertial Measurement Units

An Inertial Measurement Unit (IMU) is a device that has a three-axis accelerometer and a three-axis gyroscope. These two components can be used to measure the rotation of an object. The accelerometer measures the linear accelerations, and the gyroscope measures the rotational velocities. At rest, the three-axis measurement of the accelerometer describes a vector along the direction of gravity [54]. To get the rotation angle from the gyroscope, the output rotational velocity needs to be integrated. There is a risk that there are non-zero bias and noise in the output of this sensor [54]. Integrating including this error leads to *integration drift*. Consequently, a fusion of both sensors is needed to accurately estimate the rotation of the camera [54]. The three rotations are the roll, pitch, and yaw angles where the roll is a measure of the inclination in the front direction and the pitch is a measure of the inclination in the lateral direction. The yaw angle is the rotation along the floor.

3
User study

When determining functions that intend to support the usage of technology for humans, it is the end user's perspective that one has to start from. Therefore, there must be an understanding of what problems and needs there are for end-users today in order to develop as useful functionality as possible. This was initially retrieved by a market analysis in collaboration with Kollmorgen to understand what similar functionalities there are on the market today. Given an initial understanding of what interactions are feasible to perform with the used technology, a first set of conceptual interaction ideas was then put together:

1. The possibility to tell the AMR to stop with a gesture.
2. The possibility to direct the robot to take a detour when going to a position.
3. The possibility to direct an idle robot to go to a new position.
4. The possibility for the robot to detect if it is moving close to a pedestrian that has its back turned to the robot and is therefore unaware of the robot. When the robot perceives this, it will reduce its speed.

These interactions were used as initial candidates of interaction concepts to implement. They were extracted based on inspiration from the market analysis, inputs are given from people with experience in the business, and also from creativity and personal experience from the authors.

3.1 Interviews

To get a better insight into what new functions are wanted in the industry and what makes for a good HMI, three interviews were conducted. One of the interviewees is a user experience designer for AMRs, and two are working on sites that utilize AGVs. In preparation for the interviews, a video with demonstrations of the initial interactions was prepared. The reason for this was to give the interviewees a rough idea of what concepts were realistic to perceive the current technology used. In the video, an AMR was manually controlled to simulate a desired behavior when interacting with a user or a pedestrian.

The interview was divided into a number of questions:

1. The purpose of the interview was introduced.
2. The interviewee was asked about how they work with AGVs at their facility if they do work with AGVs.

3. The interviewee was asked about actually dangerous and perceived as dangerous scenarios in working with AGVs.
4. The interviewee was asked about annoyances in working with AGVs.
5. The interviewee was asked about what new functionality he or she would like to see in an AMR.
6. The interviewee was then shown a video of a user interacting with an AMR in a set of scenarios consisting of what the authors have come up with as well as scenarios inspired by previous interviews.
7. During the demonstration the interviewee was asked if he or she would find the functions demonstrated in the interactions useful and for possible improvements of it.
8. With the hypothesis of the interviewee getting inspiration from the proposed interactions he or she was once again asked about new interactions that would be useful.

The results of the interviews, together with a feasibility analysis resulted in a set of functions and concepts to be implemented.

3.2 Interview Results

Below a summary of the most significant insights from each conducted interview is presented. The interviews were made with people that in different ways are working in close contact with either AGVs or AMRs. Further detailed notes and details can be seen in Appendix A.

3.2.1 Interview 1

Based on the answer to question 7, it emerged that users might want to minimize the amount of work and knowledge needed when handling an AMR. For this reason, the possibility to redirect an AMR to take a detour can be seen as adding a feature that is not worth the extra effort for a user to learn and might not be used in a real application. It could therefore be an unnecessary feature in comparison to the potentially beneficial aspects it could give. This feature was not included in the set of demos in the subsequent interviews. By similar reasoning, more passive features such as slowing down when behind a person's back could be considered rewarding since it does not take any effort from the user or pedestrian but could still improve the interpreted behavior. A source of inspiration for what intuitive behavior is could be the normal car traffic where the vehicles are driven by humans.

3.2.2 Interview 2

After the interview in section A.2, the need for passive human-aware navigation was reinforced. AGVs could sometimes drive too close to a human at too high speed for it to feel comfortable. The desired functionality for the AGV to acknowledge that it has perceived pedestrians was also brought up. Additionally, new functionality was mentioned. When a person is performing kitting, where a user is gathering items from multiple places, an AMR should be able to follow the user from bin to

bin so that the user can put the items on the AMR. This new function was also incorporated into the video for the next interview.

3.2.3 Interview 3

From the interview in section A.3, it was understood that the speed of the AGV should be moderated according to the pedestrians in its surroundings. It happens that AGVs suddenly show up behind the back, which is interpreted as uncomfortable behavior. There are times where there is a lot of traffic with both AGVs and manual trucks, where humans need to be able to know what AGV is about to do. There are occasions where AGVs are getting in the way and trap manual trucks, where people have to interrupt their work and manually move the AGVs to solve the situation. The concept of stopping an AGV by a hand gesture would be useful if it would work reliably. This would be especially useful in a case where a person for example is driving a manual truck and sees that an AGV is about to get in the way or trap the manual truck.

3.3 Final interaction concepts

The interviews with people that are experienced in the field gave good inputs and new perspectives of working with the application of an AMR or AGV system. Based on them an improved understanding of what interactions could be useful for the end-users was retrieved. Below, the resulting interaction concepts that were decided to be examined further are described.

3.3.1 Stopping

The ability to stop the AMR by a hand gesture when it is running could be useful in multiple aspects. Since it will only make the AMR stay in its current position, it will not affect anything else around it or have any specific prerequisites to perform the action. As mentioned in A.3 this could give a human the possibility to stop a running AMR if it is predicted that there will be an issue or clash if it would continue. This way, the manual worker also does not need to go to the controller of the AMR to stop it, but could instead in a simpler way continue with the work it is currently doing.

3.3.2 Move idle AMR by pointing

If an AMR is idle it might be in a situation where it is located in the way of something or someone else. The idea is then that a human should have the opportunity to simply point it to a location where it does not hinder ongoing work around it. This could be compared with the option of manually drive the AMR on the controller using for example a joystick, but where the user should not have to go to the controller in order to move it to the desired position, which could be annoying for a user. Instead, it could stay on its manual truck and simply point the AMR away to a suitable location.

3.3.3 Reduce speed around humans

A common problem that arises is that people feel uncomfortable when AMRs are driving fast close to them. Even though people know that the vehicle will stop, it does not give a safe impression if it drives at full speed until it abruptly stops at a close distance. Therefore the AMR should slow down more smoothly when it is detecting that it is a human obstacle it is driving towards, and stop at a more convenient distance.

This behavior could be particularly useful if the AMR notices that the human has its back turned towards it, and it is, therefore, less likely that the person is aware that the AMR is heading towards it. If the AMR instead detects the eyes of a human obstacle, it is more likely aware of the vehicle approaching. This could reduce the number of uncomfortable situations of surprise.

The behavior could also be different whether it is one person or a group of people that are in the way of the AMR. If it is one person that is believed to be aware of the vehicle, it could be considered more likely for the person to move out of the way. Then one could give the AMR a bit more speed and smaller distance to the human before it has to stop. If it instead is a group of people blocking the way, where it could be considered less likely that they will move out of the way since they could have a conversation or are working with something and might not want to be disturbed. Then the AMR could instead give a bit more safety distance and a slower pace when approaching the group.

3.3.4 Follow person

If a person is going to collect several objects at various locations it could be useful to have an AMR that follows and supports by giving the opportunity to load on it. This functionality could then replace the need to have a manual trolley brought along. It also eliminates the need for the person to deliver a fully loaded trolley to its intended location, but could instead send the AMR there automatically. This could make the work significantly more efficient since it would eliminate the time it takes to manually deliver the objects, and could instead instantly begin on a new badge with a new AMR.

3.4 Discussion of interaction concepts

Working with new technology that aims to simplify everyday work for humans is a difficult task since all people think and interpret things differently. This means that even if a solution could be interpreted as helpful for one person, it could still be interpreted as disturbing for others. Working sites are also usually different, which means that there could be entirely different problems and needs depending on how the situation is. In order to design a system that attempts to help people at as many different working sites as possible, generalizations on how people think and interprets situations have to be made. It is therefore useful to take inputs from

people who have experience with working with AGVs or AMRs in different ways. This gives a good insight into what problems could occur and helps when trying to identify what specific problems that feature using HPE could improve or solve.

An important note is that the interviewees were found through the network of Kollmorgen. Since AMRs are a relatively new field for them, their main customer base is in the AGV business. The contact with the second and third interviewees was, therefore, made through this network. The experiences and thoughts that were given in those interviews were therefore based on the usage of AGVs and not AMRs. It was however considered as experiences similar enough to be used also for the AMRs since the daily usage and many of the issues are believed to be similar between the two. Since the interviewees were found through Kollmorgen there could potentially be experiences that only apply to their particular products, and not representing general usage of AGVs made by other manufacturers.

The given interaction concepts were intentionally a slightly scattered mix of some different use cases of HPE where the behavior of the AMR could be improved. The reason for this was to be able to return a general assessment of what kind of interactions were interpreted as the most valuable improvements by end-users. An important distinction between interactions is those that are considered active and those who are passive from a humans perspective, as described in subsection 2.6.1. In this case, the interactions that stop the AMR and moves an idle AMR by pointing are considered active interactions since they require an input performed by a human where the AMR is supposed to respond in a certain way. The interactions where the AMR reduces its speed and where it follows a person are instead considered as passive interactions since they could be performed continuously without any specific action by a human. The follow person interaction would however probably need some kind of initial action by a user to be activated, since this kind of function normally is not the primary task of an AMR.

To be able to point an AMR to a certain location, the calculations have to be in 3D space to return a somewhat precise location compared to what the human intended. An important note is therefore that additional information of what the distance from the camera mounted on the AMR to each keypoint in a human pose is. This information is also very useful information when determining the speed around humans since it then would be possible to adjust the speed depending on the distance. This could give the AMR the potential of a much smoother behavior that is likely to be interpreted as more intuitive by users. A solution for retrieving information about each keypoint in 3D is therefore considered very valuable for this application.

The concepts presented were extracted by a basic understanding of the needs of the industry. Due to an ongoing pandemic of Covid-19, it was not possible to visit any sites where the products were used, which potentially could have given a much better understanding of how the systems work in end applications. The interviews were made through online calls on a computer, which gave limited abilities for the interviewees to show and explain how the systems work at each site. It would also

be beneficial to do interviews with more people at different workplaces to get a more broad and general understanding of how the problems could differ overall.

4
Evaluation of HPE methods

To achieve the desired improvements of the AMRs behavior, a key component is to have a DL model that can perceive the local environment with sufficient frequency and robustness to run in a real-time application. It is then important to know what different approaches and methods there are for that kind of application. Therefore a study on different alternatives has been conducted to understand the pros and cons with different methods. Based on that, a set of models has been selected that is supposed to reflect some of the varieties of HPE methods there are available online. The models are then tested on the AMR system to get a good understanding of how the models perform in this particular setup. Based on the tests a comparison has been conducted of what pros and cons there are for each model, and also an assessment of which models that are suitable to use for the application.

4.1 Metrics

Each model is evaluated based on the following metrics:

- Frequency when the model is running on a CPU and/or a GPU, in frames per second (fps)
- Latency when the model is running on a CPU and/or a GPU in milliseconds (ms)
- Precision of the model according to their respective research papers given in Average Precision (AP)
- Occlusion handling of each model, which is an observed estimation on how good the model estimates a pose when parts of the body are missing
- False positive detection, which refers to what extent poses that are observed as falsely detected by the model even though it is not an actual human pose it returns

To be able to deploy HPE on a real-time system the system must be able to estimate poses with a sufficient frequency to be able to interpret the poses intended by a human. It is also important to not have a too large latency such that an interpreted pose is performed within a sufficient time from when it was executed. Good precision of a model increases the certainty that an estimated pose is correct, but it could also be an indication of a model that is computationally heavy to run on light systems and might therefore require a GPU to run sufficiently.

Since the camera has a limited Field of View (FoV) from where it is placed on the AMR, only parts of a body are likely visible in an image. Therefore, it would be

good if the model is able to estimate human poses even if an entire body is not visible. This is what the metric *occlusion handling* refers to. It is also crucial that the poses that are given by the model are reliable as actual human poses, and not poses found in other features of the input image. This is what *false positive detection* refers to, which is an observed estimation of how many false poses are given by a model.

All these metrics are factors that have to be taken into account to determine which model suits the application the best. It is challenging to find a good balance between sufficient precision and reliability of the poses. At the same time, it can not be too computationally heavy for the system to run in real-time.

4.2 Setup for evaluation

The setup used when evaluating the DL models is illustrated in Figure 4.1 where the estimated poses were displayed over the original image for each frame. The resulting performance of the models is taken when they were running in a simulation environment of the AMR system, which is as close to the real system as possible. The simulation environment is a ROS environment where all the software that runs in the real robot is running but where the robot hardware is switched to models that simulate its behavior. The models are running in a ROS node where they subscribe to an image feed published by the camera. It then estimates and extracts human poses in the given image and publishes them in a new topic, which is sent to the node that executes robot behavior based on the poses.

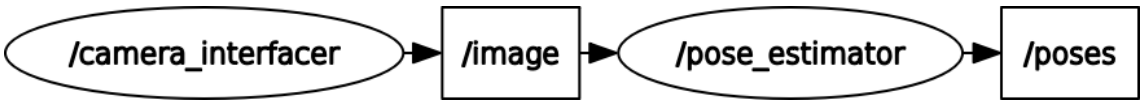


Figure 4.1: Architecture of the node structure for evaluation

The RGB-D camera that has been used during the tests is an Intel Realsense D435i. The performance of the DL models that run on CPU has been retrieved with a Dell Latitude 7280 PC with an Intel i7 7th generation CPU processor running a Linux Ubuntu 18.04 LTS operating system. The performance when running on a GPU has been retrieved with an NVIDIA Jetson AGX Xavier industrial PC with a 512-Core Volta GPU running also running a Linux Ubuntu 18.04 LTS operating system.

4.3 Human Pose Estimation method performance

A study on what different approaches there are for performing human pose estimation based on RGB images has been conducted. Due to the fast development of DL that has been made during the last decade described in subsection 2.4.3, most of the models found are only one or a couple of years old. It is also a very active field of research, where improved models and new approaches come up every year. A

set of models is presented, which is intended to give a good representation of what different options there are available online as open-source software.

4.3.1 OpenPose

With this method, a bottom-up approach is taken where each keypoint is estimated as a confidence-map while in parallel a Part Affinity Fields (PAF) is estimated. The PAF estimates the direction from each joint to the next [48]. For example, all left shoulders and elbows are estimated in terms of a heat-map. Simultaneously, a PAF is estimated linking the correct left shoulder to the correct left elbow, associating the key points of each person together.

The implementation used is a package called *trt_pose*, which is given out by NVIDIA to run real-time pose estimation on the NVIDIA Jetson modules [55]. This is also the model used for HPE in the NVIDIA ISAAC SDK, which is a toolkit for the deployment of robots powered by AI [56]. It is therefore assumed that this model is optimized for running inference in real-time, which is positive for this usage case.

When this OpenPose implementation is run on the system using a GPU it runs with a frequency of 12 fps and with a latency of 1000 ms, which is a really good performance when run in real-time applications. The model has an average precision of 63.5% according to the research report [48], which is a good number. It is also very good at detecting humans even if an entire body is visible and it does not give many false estimations, which makes it seem like a very reliable model to use.

4.3.2 Lightweight OpenPose

Lightweight OpenPose is based on a study that analyzed the OpenPose solution and heavily optimized it in terms of speed while sacrificing some accuracy granting an increase in the frequency of 27 times compared to the original solution in a complex test case [57]. The main change introduced is that the backbone of the network was changed to a more lightweight version, while the depth of the network was decreased. Further, the initial layers of the two parallel estimations (the keypoint and the PAF estimations) are shared in this implementation.

The implementation of Lightweight OpenPose used is a model given out by the Intel Open Visual Inference and Neural network Optimization (OpenVINO) toolkit, which is a DL toolkit for CPU usage that supports quick development of many different types of applications for example human vision, speech recognition, natural language processing, and recommendation systems [58]. It includes scripts for extracting and interpreting the data given by the model and also plotting the estimated keypoints on the original image. Since Lightweight OpenPose, similar to OpenPose described above, is given out by a toolkit for applications of DL it is assumed that this model is optimized for running inference in real-time.

The model has been benchmarked with a frequency of 13 fps and a latency of

160 ms when run in the simulation environment on a CPU, which is a really good performance. The model has a precision of 42.8% when run according to its research paper [57]. Similar to the OpenPose model running on the GPU this model is also very good at occlusion handling and understands when there are only parts of a body present. It has been noted to give some false pose estimations when run on the real robot, which is something that has to be dealt with to use in a real-time application. Since the model is created for performing inference on CPU hardware, which requires lighter models and fewer calculations, the performance is considered good due to its high frequency, low latency, and perceived reliability despite having a relatively bad precision.

4.3.3 FastPose

The model called FastPose is a solution given by an open-source library that performs 2D pose detection on input images. It is a top-down method where it uses the model called *You Only Look Once* (YOLO) [59] for object detection. Based on the bounding boxes of humans found by the YOLO model, a custom Mobilenet v1 using 50% of the parameters is used as a feature extractor [60]. On top of this, a basic bounding box tracking algorithm is used to keep track of a person as long as it stays in the image [61].

The FastPose package is designed to be easy to use and implement in Python. The package includes scripts that simplify the use such that the extracted keypoints are given directly, and there are support scripts that plot the keypoints on the original image. When benchmarking the model it gives a performance of 25 fps and a latency of 90 ms when it is run on a CPU, which is extraordinarily good. There is no precision value found for this model, but it is seemingly a bit unstable since it gives quite inconsistent estimations that indicate a lower rate of precision. It is also noted that the model seems to always expect an entire body when given a bounding box interpreted as a human by the classifier, which makes it squeeze in legs quite randomly of a body that is only visible from the waist and upwards even if it detects the upper body correctly. It is however giving good estimations when a full body is visible, but considering the limited FoV that the camera has, it is common for only a part of a body to be visible. False pose estimations occur with a relatively high frequency when a human is not present in the image, which makes it slightly difficult to work with and requires post-processing to filter out false pose estimations.

4.3.4 HRNet

HRNet is the official implementation of the research paper *Deep High-Resolution Representation Learning for Human Pose Estimation* [62]. The novelty with this approach is to maintain a high-resolution representation of the image and combine that with running high to low-resolution sub-networks while keeping efficient computation complexity and number of parameters. The model is run with the top-down method, where it uses bounding boxes of persons produced by the *FasterRCNN* network [63]. The network is a top competitor in many pose estimation challenges,

and has been further used as a backbone for other high performing models [64].

To implement the HRNet model, an open-source module called *simple-HRNet* is used which is a simplified version of the implementation with ease of use in mind [65]. When run on a CPU the model estimates keypoints with a frequency of 1.85 fps with 1100ms latency. This latency is hard to work with when trying to perform real-time functionality based on it. When running the model on a GPU instead the frequency increases to 2.5 fps with a latency of 1350 ms, which is a frequency that could be sufficient to use but where the latency still is slightly too high. According to the research report, [62] the model has an average precision of 76.3%, which is good. With that, it is among the best models on the top lists of performing keypoint detection [66]. The model does not seem to give false pose estimations from the tests. It has a slight problem to detect a human when a part of a body is visible, but it is still considered sufficient enough to use. Altogether it is a model that could be used for a real-time application if precision and reliability are of primary importance. However, it requires a relatively high computational power to run at a sufficient frequency and latency.

4.3.5 Higher-HRNet

The model *Higher-HRNet* is a novel bottom-up approach for HPE. As well as with previously presented HRNet, Higher-HRNet uses high-resolution representations of the images within its network [67]. This approach is however using a bottom-up approach, which makes it fundamentally different from the previous HRNet. The idea with this is to be able to execute the HPE task also on small objects in an image.

Along with the previously presented HRNet model, there is a similar simplified open-source package called *simple-HigherHRNet*, which contains support scripts that simplify the implementation of the model [68]. When run on a CPU the model estimates keypoints with a frequency of 0.51 fps and a latency of 2300 ms, which is not usable at all. When running the model on a GPU instead the performance increases to 1.8 fps and 2200ms latency, which is better, but not quite good enough to use in a real-time application. According to the authors of the model it has a precision of 70,5%, which is very good [67]. It is also very good at occlusion handling, and no false pose estimations have been observed when testing the model.

4.3.6 PoseNet

PoseNet is a bottom-up approach for pose estimation and instance segmentation of multiple persons in an image using an efficient single-shot model [69]. The model is used as a demo example for pose estimation by TensorFlow, which is a commonly used machine learning library in Python that is developed and maintained by Google.

The PoseNet model is tested with a Python implementation based on an open-source package with support for interpreting the output of the model [70]. When running the model on a CPU it runs with a frequency of 8 fps and a latency of 220ms. This is

a very good performance for running on a CPU only, and well suitable for running on a real-time application. When running the model on a GPU instead the frequency increases to 13 fps and the latency to 140 ms, which is even better. The model has a precision of 68.7% according to the authors of the model, which is considered good [69]. It is also good at occlusion handling and no false pose estimations have been observed, which gives a robust impression.

4.3.7 Cubemos

There is a Skeleton Tracking SDK for Intel Realsense Cameras given out by Cubemos, which is a paid-for package for real-time estimation on CPU hardware [71]. The package is designed to be simple to implement and to use in applications. Since this package is not open source, it requires a valid license to run. This package is included in the study since it is considered a good reference compared to the other solutions that have been considered in this project.

A test version of the package has been tried out. It should be noted that it was not fully incorporated in the simulation environment like the rest of the models when tested, so the retrieved performances might be slightly misleading. When run in a real-time demo on a CPU the model had a frequency of 10 fps and a very small latency. It is good at detecting poses even if parts of the body are missing. Some false pose estimation occurs, but overall it has a very good performance and reliability.

4.3.8 ML model summary

Below, summarizing tables of the model options described above can be seen. In Table 4.1 there are objective results retrieved from test measurements and reports from the authors of each model. In Table 4.2 metrics that are subjectively observed are presented.

	Objective metrics				
Models	**FPS CPU**	**Latency CPU (ms)**	**FPS GPU**	**Latency GPU (ms)**	**Precision AP (%)**
OpenPose	-	-	12	100	65.3
FastPose	25	90	-	-	-
Lightweight OpenPose	13	160	-	-	42.8
HRNet	1.85	1100	2.5	1350	76.3
Higher-HRNet	0.51	2300	1.8	2200	70.5
PoseNet	8	220	13	140	68.7
Cubemos*	10	Very small	-	-	-

Table 4.1: Table summarizing the objective results of the HPE model comparison

	Observational metrics			
Models	**Occlusion handling**	**False positive detection**	**Implementation language**	**Implementation simplicity**
Openpose	Good	Never	Python	Slightly tricky
FastPose	Bad	Often	Python	Straight forward
Lightweight OpenPose	Good	Sometimes	C++	Slightly tricky
HRNet	Sufficient	Never	Python	Straight forward
Higher-HRNet	Good	Never	Python	Straight forward
PoseNet	Good	Never	Python	Slightly tricky
Cubemos*	Good	Sometimes	Python or C++	Simple

Table 4.2: Table summarizing the observational results of the HPE model comparison.

Here it should be noted that the model called *Cubemos* is not tested on the same simulation environment as the others, which is why it has a * behind it. If a model has not been run on certain hardware or if certain information has not been found, it is represented with a - in the table.

4.4 Discussion of HPE method results

The main purpose of the retrieved set of models is to get a basic understanding of what potential improvements the use of HPE could give for the final behavior of the AMR on a conceptual level. It is therefore not intended to be an entirely finalized functionality that is ready to be implemented into the product immediately. Since AI and DL are relatively new concepts that have come up during the last decade and are continuously being improved in research, the potential improvements they can give when deploying them in final products are still something that remains to be explored in many cases.

The set of DL models that has been retrieved and tested on the system is only a small pick of available models online as open-source software. It does however contain models with significant differences compared to each other, where there are both top-down and bottom-up models represented as well as models that are run on both CPU and GPU. A key factor in this application is that the AMR system is currently being run on a CPU only, which makes models that run on CPUs particularly interesting to explore. As described in subsection 2.4.3, the main reason why DL has made such a breakthrough during the last decade is the increased computational power that is offered by modern GPUs, which makes it possible to train deep networks. Therefore most models are developed on GPUs, and there is a significantly lower number of alternatives that run on CPUs.

In the result shown in subsection 4.3.8, one can see that there is a clear increase of frequency and decrease of latency for the models that have been run on both types of hardware. It is therefore a clear advantage to have a GPU available in the AMR system if possible. But since industrial GPUs are relatively expensive compared to

the current hardware, a judgment that has to be made is whether it is possible to achieve a sufficient interpretation when running on a CPU only.

However, an important note is that the methods have been tested in a simulation environment running on a PC for CPU tests and an NVIDIA Jetson AGX Xavier for GPU tests. This is not the same hardware setup that would be used if it would be run on the real AMR system, and it is therefore likely that the retrieved frequency and latency could be slightly decreased due to differences in computational power between CPUs. The result should therefore be interpreted as more of general guidance of how each model could potentially perform rather than giving an expectation of the same particular performance as retrieved in these tests.

From what the tests have shown in this thesis project, it is considered enough to use a model that only requires a CPU to run. From running tests on the system, it is estimated that a frequency of at least 2-3 fps is required to be able to interpret possible intended interactions from a person in a stable way. A latency of more than 1000-1500 ms is also difficult to work with from a humans perspective since it causes slow response time that might lead to confusion whether the AMR has interpreted an interaction or not.

When running real-time inference on a CPU, relatively light models are required to be able to run them at a sufficient frequency. This is because of the reduced computational power available in CPUs compared to GPUs as described in subsection 2.4.1. As subsection 2.4.5 explains, the drawback with this is that the pose estimation performance of the models is likely to be reduced. This is therefore a trade-off that has to be made when choosing which model to use in the application.

This drawback is also something that has been noted from the results when testing the models. The FastPose method is a good example of when a model is too light, which affects the pose estimation performance significantly. The model runs on the system with a very high frequency and gives sufficient estimations when there are full human bodies in the image. The big problem is when there is only half a body visible as described in subsection 4.3.3, which is a common case on the AMR system due to the limited FoV of the camera. It is also giving some false estimations of poses when people are not visible, which makes the model too unreliable to use in the intended application.

On the other end of this spectra are the HRNet and Higher-HRNet models, which are both giving reliable estimations with no false pose estimations, but runs with a much slower frequency and much higher latency when they are run both on a CPU as well as on a GPU compared to FastPose. When running them on the AMR system they, therefore, do not quite interpret the local environment in a sufficient way for a real-time application, mainly due to the delay that the latency gives.

The models that have given the best performance when run on a CPU are the Lightweight OpenPose and PoseNet models, which can run at good frequencies and

do not have a latency that affects the behavior of the system. They are also good at producing good pose estimations even if only parts of a body are visible and do not give too many false estimations, which are very good features for this kind of application. They are therefore considered as good options to use if only a CPU is available. On the GPU the OpenPose, as well as the PoseNet model, are run with a convincing performance, both by frequency and reliability.

The retrieved performance of the open-source models are also performing in a similar, if not better, way than the paid-for Cubemos model described in subsection 4.3.7. The implementation of the open-source models is also relatively simple, so there is no big reason to use the paid-for model from that aspect either. It is therefore considered unnecessary to pay for a licensed model such as Cubemos when there are open source options with similar performance and simplicity.

Another note is that the bottom-up methods seem to be better at occlusion handling than the top-down methods. The top-down methods FastPose and HRNet have shown slightly worse flexibility when only a part of a body is visible. The HRNet does not give any false-positive estimations, but it seems to require more parts of a body to be visible than the bottom-up methods to detect the pose. A reason could be that the bottom-up methods as described in subsection 2.5.2 detect body parts individually before connecting them to a full-body context, which might make it more suitable for interpreting humans located in the local environment when the camera range is limited.

The resulting models that have been examined during this thesis project are considered to show a good overall representation of some different general approaches there are available as open-source for performing HPE. An important note is however that the tested models are not necessarily representing the best options available, even though some of them are considered sufficient models to run in this kind of application. As mentioned in section 1.4 the model packages used are selected since they include support scripts for interpreting the output, which is a limiting factor when choosing which models to examine. The reason for this is that it simplifies the implementation of the models significantly since it is common for different models to have a different kind of raw output depending on the HPE approach. It then requires a more extensive theoretical understanding of each method individually to be able to interpret the output, which is time-consuming and was considered unnecessary for a more comprehensive study. It could however be something worth looking into if one would decide to proceed with the concept of HPE to develop a final functionality for the AMR as a product since there are methods with good performance that are not as well documented as the ones tested in this project.

The observational metrics that have been retrieved should be interpreted as general guidelines since they are subjectively observed on how well the model seems to perform according to each metric. An important note here is that there has been an ongoing pandemic of Covid-19 during this thesis project, which has constrained the tests and the collection of results to be carried out in an isolated home environment

only. Due to this fact, the results have been taken from a simulation environment only and not on the real AMR system, which would be preferred otherwise. It also means that the models have not been tested in a warehouse- or factory-like environment, which potentially could cause other performances or issues with the models that have not been observed during these circumstances.

5

Design and Implementation

The implementation of each interaction from chapter 3 has two main parts. Firstly, the poses need to be interpreted, and secondly, the AMR needs to respond in a certain way. Additionally, both the interpretation and the response need to be robust and intuitive. Human Pose Estimation methods can perceive keypoints as detailed as joints, eyes, ears, and noses. This limits the level of detail of how a human pose can be perceived. For an interaction to be intuitive, it means that the gesture used by the user should come naturally when a corresponding response of the robot is desired. The AMR should in turn respond expectedly from the user's point of view.

5.1 Hardware setup

The robot that is used is a lightweight differential wheeled robot with its pivot point in the middle of its driving axis, illustrated below in Figure 5.1

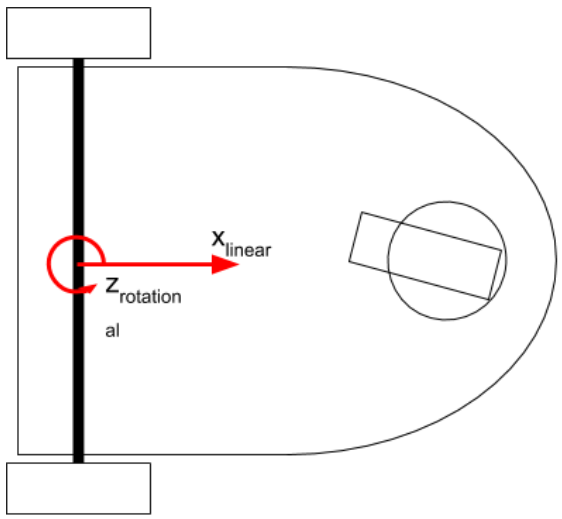


Figure 5.1: Schematic of the AMR

Due to this design, the robot can rotate around the z-axis and move in the x-direction. The robot can therefore be controlled by sending commands for velocity and angular velocity independently and the transformation from this to the actual actuation of the robot is handled automatically.

5.1.1 Camera setup and field of view

Among the robot's sensors, it has a camera mounted on the front at a height of $18cm$. The camera used is an Intel Realsense D435i RGB-D camera. The camera's depth sensor has a 87 ± 3 horizontal FoV and a 58 ± 1 vertical FoV, the camera's color sensor has a 69 ± 1 horizontal FoV and a 42 ± 1 vertical FoV. Further, the camera has a 6 Degrees of Freedom (DoF) IMU [72]. This data is presented in Table 5.1.

Camera name	RGB sensor		Depth sensor		IMU
	Horizontal FoV (deg)	Vertical FoV (deg)	Horizontal FoV (deg)	Vertical FoV (deg)	DoF
Intel Realsense D435i	69 ± 1	42 ± 1	87 ± 3	58 ± 1	6

Table 5.1: Relevant data of the camera used

Since the color sensor's FoV is smaller than that of the depth sensor, the FoV of any combined image is that of the color sensor.

The camera is also used for collision avoidance so the horizontal FoV needs to be maximized. Because of this, the camera in *landscape mode*, with the horizontal FoV as the widest. Also, the camera is tilted upwards so that the field of view is along the floor, while still keeping part of the floor in its field of view. Then, there is a certain distance needed to perceive the full pose of a $1.8m$ tall human. As mentioned, the camera is mounted at a height of $0.18m$ which puts the theoretical distance needed at slightly more than $2m$. This is illustrated in Figure 5.2.

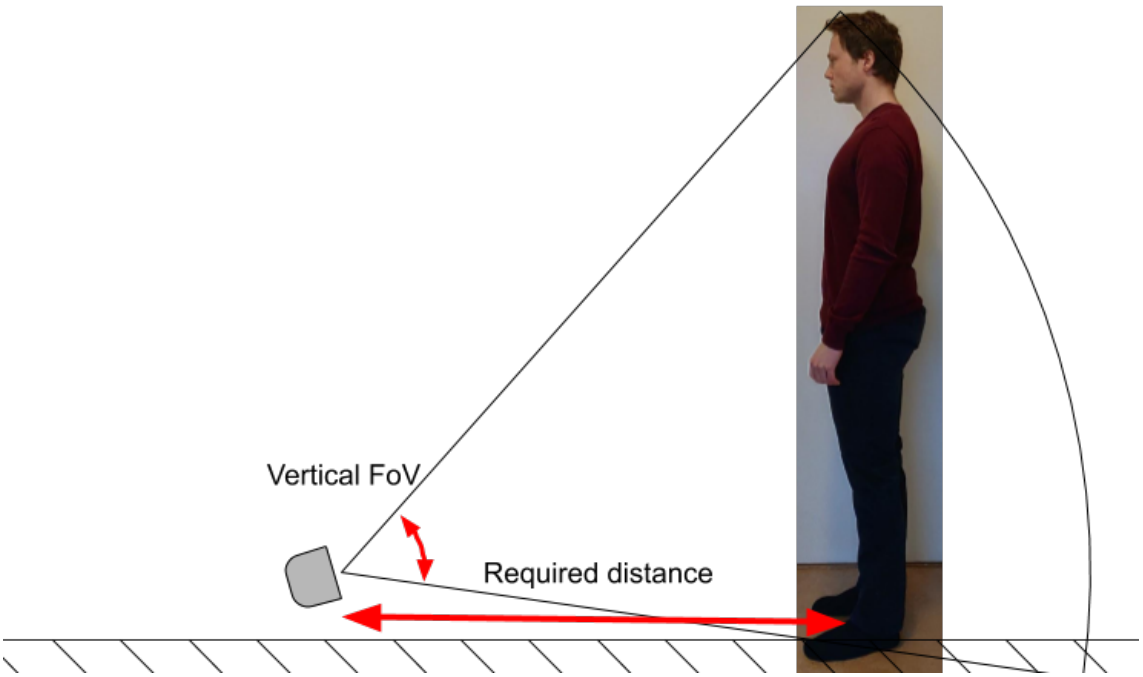
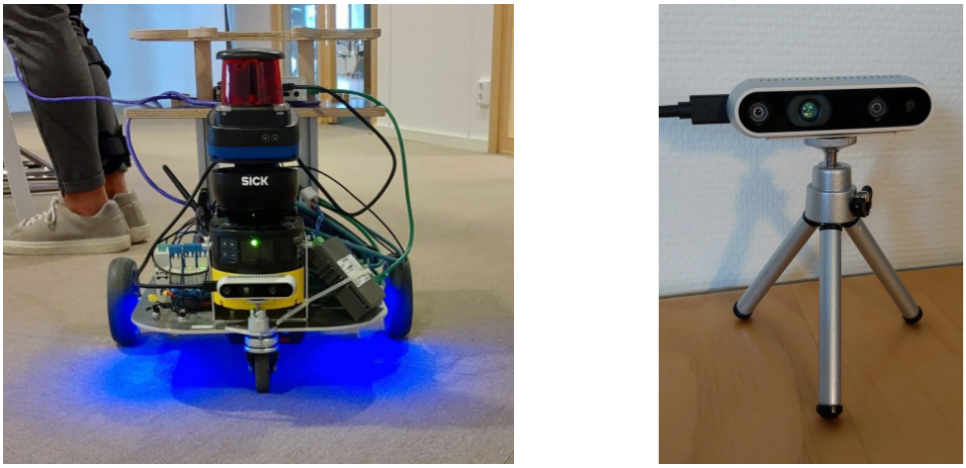


Figure 5.2: Camera placement and its effect on the necessary distance to human to perceive its full pose.

Additionally, the further away from the camera that the human pose is, the less accurate it is perceived. The interactions that will be implemented are all relevant

when the human is in close range of the robot, so not much useful information can be derived from poses that are more than $5m$ away from the camera. Therefore, the range of view is cut off at $5m$.



(a) Front view of the robot

(b) The camera used

Figure 5.3: Images of the robot and camera used in this thesis

5.2 3D poses

As described in chapter 3 many of the interactions require estimating the human pose in 3D space. The 2D poses were transformed to 3D by a simple sensor fusion method. To enable 3D HPE from 2D HPE and depth data from an RGB-D camera, the depth image is first aligned with the color image. This process enables the pixel-by-pixel match between color and depth images. The 2D pose estimation can then be fused with the depth data. Using the focal length as well as the principal point of the frame results in a pose estimation in 3D space. The rotation of the camera is unknown due to uncertainty in mounting and needs to be rotated so that it is aligned with the floor. How this is done is described in section 5.3.

Not only data of the individual keypoints of each pose is of interest for the interactions, but some data related to the full pose is also of interest. When building the 3D poses, four measurements are calculated by averaging each keypoint:

- The angle that the pose deviates from the frame-center.
- The distance from the camera.
- The x distance from the camera.
- The y distance from the camera.

This is further illustrated in Figure 5.4.

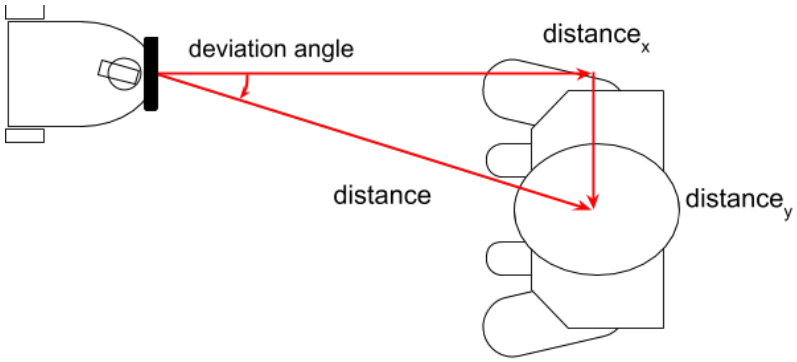


Figure 5.4: Additional data of each pose

5.3 Rotating in 3D poses using IMU

The camera is mounted with a certain error in its rotation compared to the exact orientation it should have on the AMR. However, there is an embedded IMU that is aligned with the camera and that can be used to rotate the perceived 3D poses to counteract this rotation error. The IMU has both an accelerometer and a gyroscope. The accelerometer is used to get an initial understanding of the direction of gravity, which corresponds to the vertical direction. From this, an initial guess of the roll and pitch is known.

For the next estimations, both the gyroscope and the accelerometer are used to update the rotation which improves the initial estimate. First, the change in rotation as measured by the gyroscope is added to the rotation, and then the rotation as measured by the accelerometer is filtered into this estimate. The output of this is a rotation matrix that is used to rotate each pose in 3D space.

5.4 Removing infeasible pose estimates and determining the user pose

Some of the estimated poses are *false positives*, meaning estimated poses that do not correspond to an actual pose and are often of unrealistic proportions. To minimize the risk of these false positives affecting the robot, they need to be filtered out. For many models, each estimated 2D pose can return a value that describes the strength of the estimate and can be used to sort out poor estimates. Further, pose estimations in 2D are expressed in terms of pixels of an image, and in 3D they are instead expressed in terms of distances. This enables feasibility checking based on realistic proportions and lengths of human bodies and limbs. Using this, the 3D poses can then be filtered based on their feasibility, and poses deemed as infeasible based on limb lengths and proportions can be discarded.

One of the remaining 3D poses is then chosen as the main user. The main user is the one that is actively communicating with the AMR and the choice of the user

is not trivial. In this thesis, the user choice was set to be partially based on how centered it is in the image, and partially on its distance in meters from the camera. The centredness of a pose is expressed in radians away from being in the center of view of the camera. The ratio of importance between how centered the pose is and its distance from the robot is set to be $1 : 10$.

5.5 System Architecture

The resulting data flow to accomplish this is described in Figure 5.5

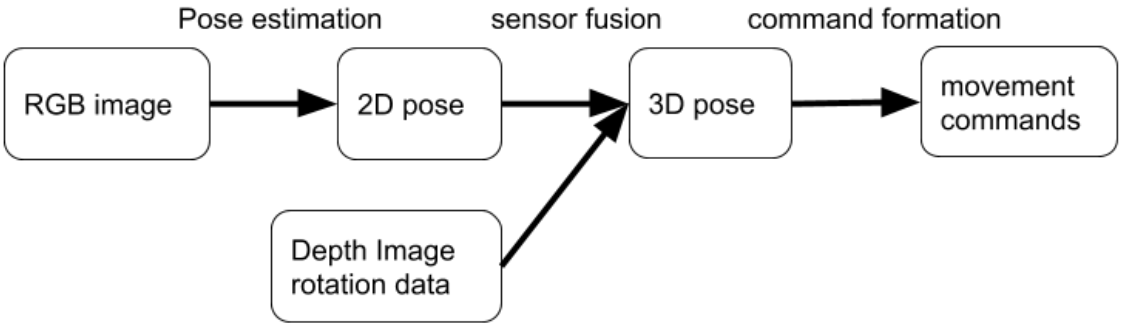


Figure 5.5: Data flow

The flow of data was first implemented within one node, where all flow happened sequentially in one process. To increase the modularity the node was instead split into three parts:

- **A camera interfacer node:**
  A node that is responsible for opening up a camera pipeline and for publishing depth and RGB data. This data also includes the estimated rotation of the camera derived from IMU-data.
- **A pose estimator node:**
  A node that subscribes to images, estimates the human poses, and publishes the estimated keypoints and score.
- **A robot interactor node:**
  A node for interpreting poses and for sending movement commands to the robot. The commands are either sent through the Navigation Stack or as self-derived movement commands through a simple P-controller.

The resulting first architecture of the nodes is illustrated in Figure 5.6, where the robot interactor node is responsible for 2D- to 3D pose fusion.

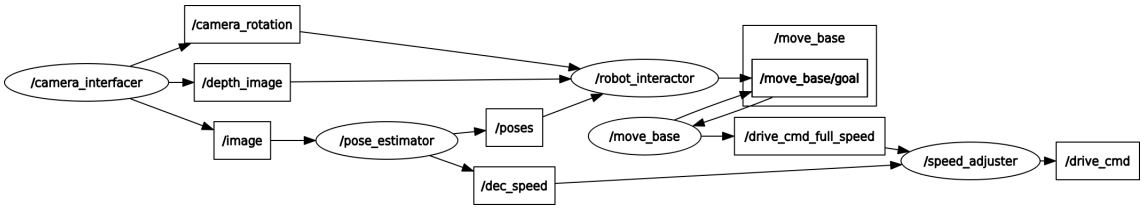


Figure 5.6: Architecture of the node structure with 2D- to 3D pose fusion in the interaction node.

The reason for this architecture choice was to put the camera interfacer- and robot-interactor node on the robot and the pose estimator node on separate hardware. This, therefore, minimizes the data sent between hardware since it could lead to latency.

However, the inference time for the pose estimator varies. Since the 2D pose needs to be fused with the depth data captured at the same time, the time-synchronization of these two ROS messages was therefore unstable. If the pose estimator node is instead responsible for 3D pose estimates and the proper rotation of the estimates, the messages are sent more sequentially and the time synchronization is simplified. Additionally, in the second architecture, all interactions are done in 3D, so that the backs are detected by the robot interactor node. This second architecture is illustrated in Figure 5.7.

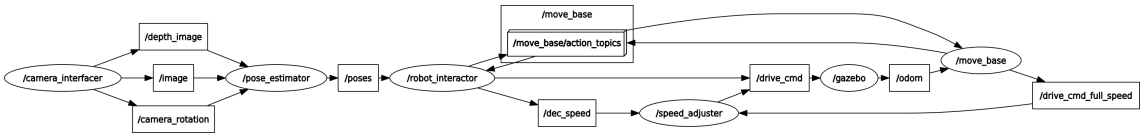


Figure 5.7: Architecture of the node structure with 2D- to 3D pose fusion in the pose estimation node.

Since the pose estimator node is the bottleneck in this sequence, a camera interfacer node used for only this purpose should have its publishing rate set to be similar to the pose-estimation rate. The reason for this is that a rate higher than necessary would lead to unnecessary computational costs when publishing unused images.

5.6 Implementing the interactions

All of the implementations were written for 3D poses, and they were using the architecture described above in Figure 5.7.

5.6.1 Robot in follower-mode

The follower mode uses the full-body 2D coordinates on the floor and it follows this point using a controller that sends movement commands to the robot. The controller is designed to keep a distance of $0.5m$ to the user. The distance is picked based on what was described in subsection 2.6.1 to be a comfortable distance.

The ability to follow a user requires the user to stay in the field of view of the camera. Therefore a controller was implemented as two P-controllers, one that controls the rotation to keep the user in the center of the image and one that keeps a distance of $0.5m$ from the user. Due to the schematic of the robot described in 5.1, the rotation and linear velocity can be controlled separately. As mentioned in chapter 3, the speed needs to be within a comfortable range, and the speed should also not be large enough to cause a danger to the motors. Therefore, the output linear- and angular velocities have the same thresholds at the maximum velocity as that it has when using the Navigation Stack: $0.5m/s$ and $0.5rad/s$. If no pose is detected, the linear- and rotational velocities are instead set to zero. An illustration of the controller can be seen in Figure 5.8.

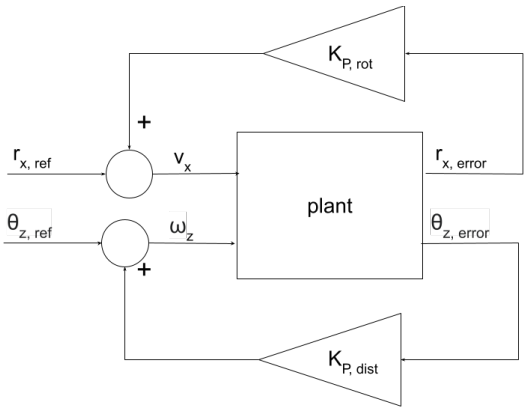


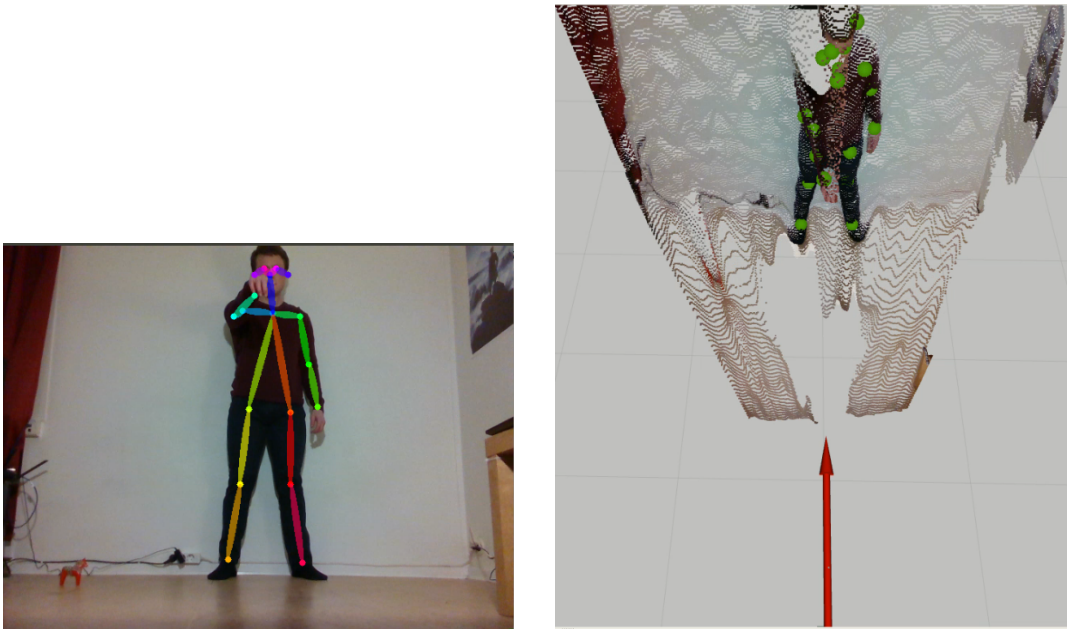
Figure 5.8: The P-controller for follower mode where $r_{x,ref}$ and $\Theta_{z,ref}$ are reference angle deviation from having the pose in the frame center and reference distance to the perceived main pose. v_x and ω_z are the linear and rotational movement commands. $r_{x,error}$ and $\Theta_{z,error}$ are the angle deviation from having the pose in the frame center and distance to the perceived main pose.

5.6.2 Stopping the robot by gesture

The gesture to stop the robot was chosen as the right arm pointing at the robot. The definition of this is that the angle between the hand, eye, and robot does not exceed a threshold angle. Additionally, the arm needs to be straight. The definition of a straight arm, in this case, is that the distance of the hand from the eye is more than a certain threshold. An illustration of this gesture can be seen in Figure 5.9. Detecting this gesture results in all current navigation goals for the Navigation Stack to be canceled. Additionally, no new movement commands are formed from the current frame by other interactions.

5.6.3 Decreased robot speed when behind the back of a pedestrian

To implement this, all detected poses within a range of $5m$, as mentioned in section 5.1 are evaluated to determine whether it is having its back turned to the robot



(a) Stopping gesture in 2D

(b) Stopping gesture in 3D

Figure 5.9: The gesture for stopping the robot

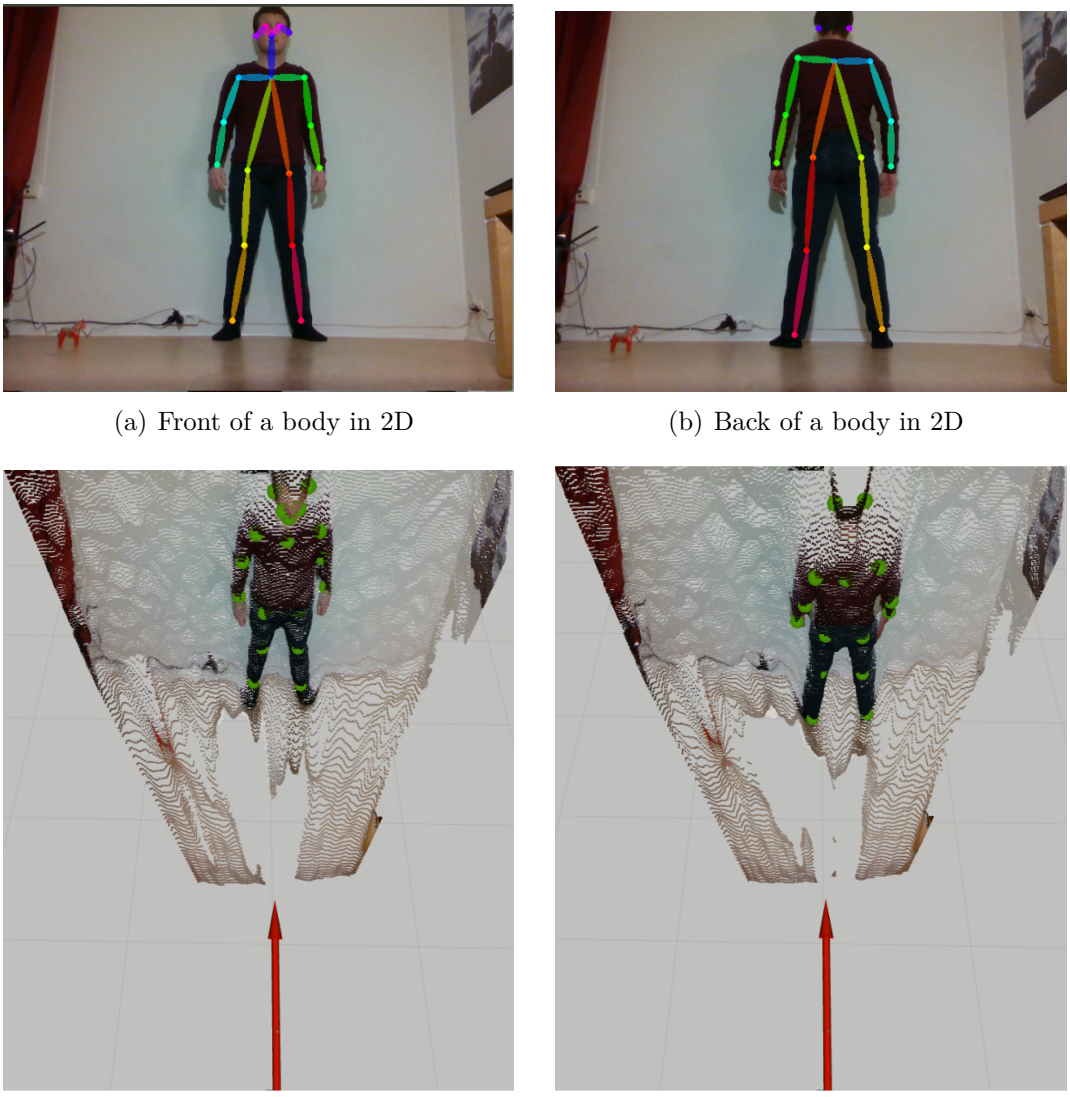
or not. Having the back turned towards the robot is defined by the shoulder-, hip-, elbow- or feet pair "flipped". This means that for example if a person's anatomical left shoulder was to the left of the right shoulder from the robot's point of view, the back was turned towards the robot. This is illustrated in Figure 5.10. As can be seen in Figure 5.10, the Human Pose Estimator can distinguish between left and right limb pairs. When a back is detected, the speed is scaled down to a set threshold velocity.

The ROS Navigation Stack sends velocity commands to the AMR that controls its movement towards a given destination. When the ROS Navigation Stack is used together with this feature, the calculated velocity commands are filtered and altered before being sent to the robot.

This is achieved by remapping the output control command message from the Navigation Stack to an intermediate message. An intermediate node then listens to this message, scales it, and sends the altered movement message to the robot.

5.6.4 Giving the robot a new command by pointing

Pointing, in this case, is defined as having a straight arm where the hand is below the shoulder. This means that the arm is pointing somewhere towards the floor. The definition of a straight arm is that the angle between the shoulder, elbow, and hand does not exceed a threshold value. By this definition, there is a high probability of false positives since every straight arm triggers the interaction. To solve this problem, an additional constraint was set for the non-pointing arm to have its hand



(a) Front of a body in 2D

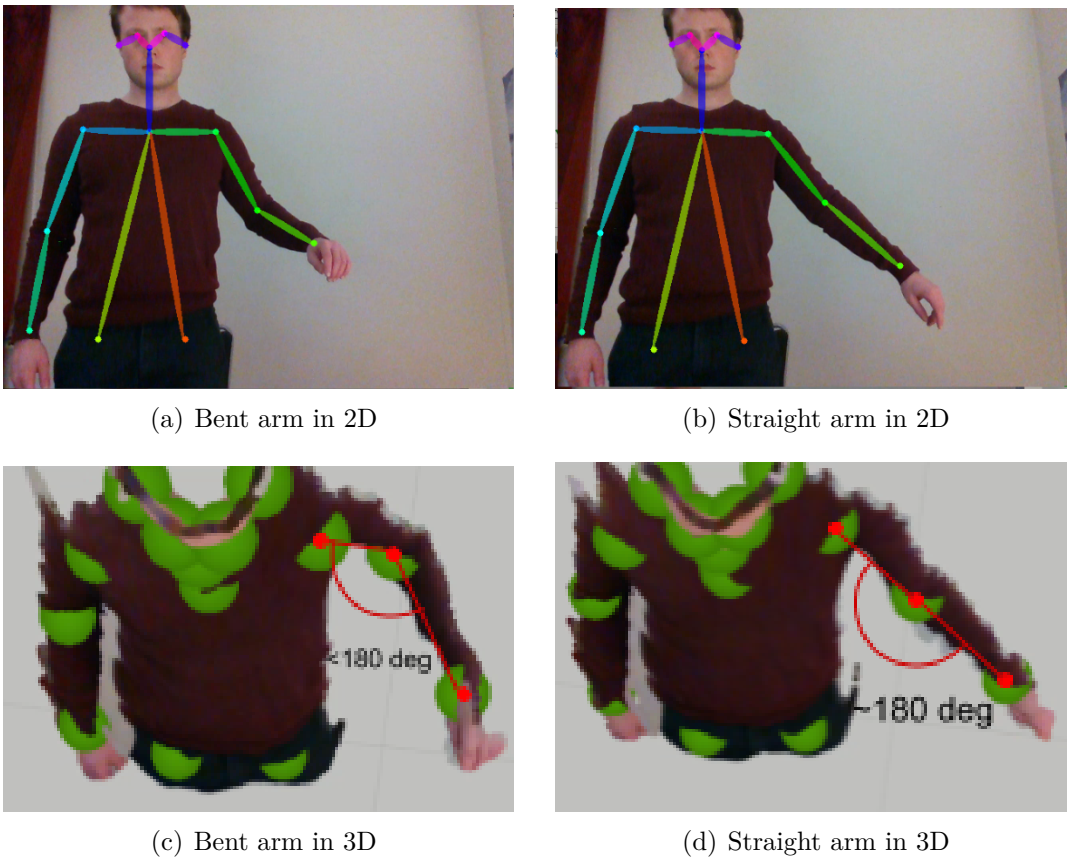
(b) Back of a body in 2D

(c) Front of a body in 3D

(d) Back of a body in 3D

Figure 5.10: Illustration of the perceived front and back of a human body and its corresponding estimated pose. The turquoise limb is the anatomical right arm of a person, the green limb is the left arm. The red limb corresponds to the right leg and the yellow limb corresponds to the left leg.

above its elbow. The desired point is then the projection of the line between the eye and the hand that intersects the floor plane. The floor-plane is set as being a horizontal plane at a set distance below the camera. This distance is the height at which the camera is mounted which could vary between different AMR setups, and is therefore given in the specifications of each individual AMR system. The can be done since the 3D pose is accurately rotated according to IMU-data. The desired yaw-angle of the robot is such that it directs its front towards the user. These definitions are illustrated in Figure 5.12.



(a) Bent arm in 2D

(b) Straight arm in 2D

(c) Bent arm in 3D

(d) Straight arm in 3D

Figure 5.11: Illustration of what defines a bent arm.

5.7 Results

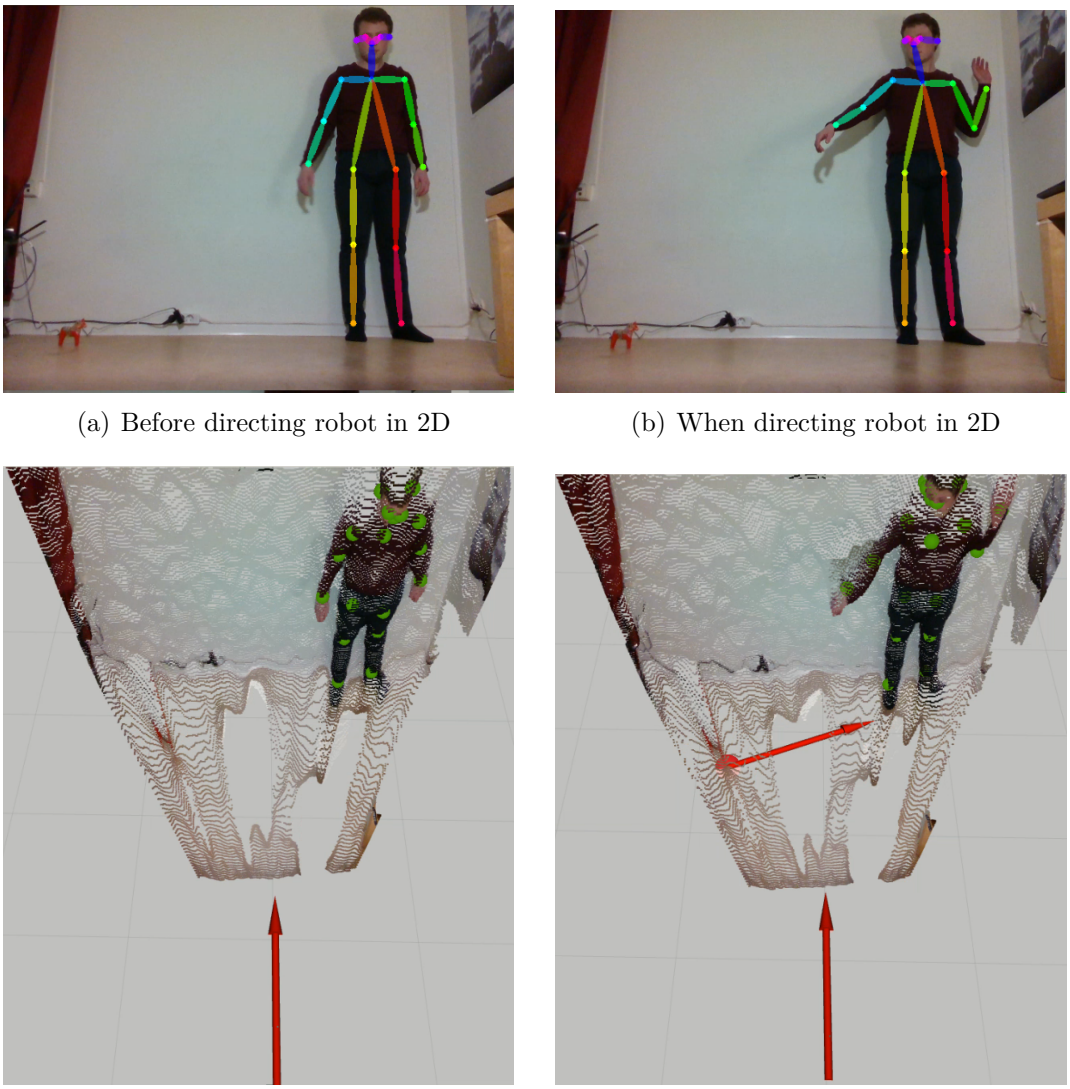
The performance of each interaction was analyzed with regard to stability and usability for the usage scenario. The main method used when evaluating the performance was OpenVINO's implementation of Lightweight OpenPose.

5.7.1 Robot in follower mode

In many cases, the user is not visible enough for its pose to be detected, which left the controller with no point to follow. However, using P-controllers led to smooth behavior in simulations. The P-controllers react instantly to a movement and accurately follow a person as long as it is within the field of view. Therefore this implemented interaction is considered as successfully achieved.

5.7.2 Stopping the robot by gesture

When the user intends to stop the robot, the robot behaves as expected. However, the choice of interpretation is implemented in a way that makes the user block the view of itself. This interaction had consequently many false negatives, but few false positives.



(a) Before directing robot in 2D

(b) When directing robot in 2D

(c) Before directing robot in 3D

(d) When directing robot in 3D

Figure 5.12: Illustration of the directing interaction.

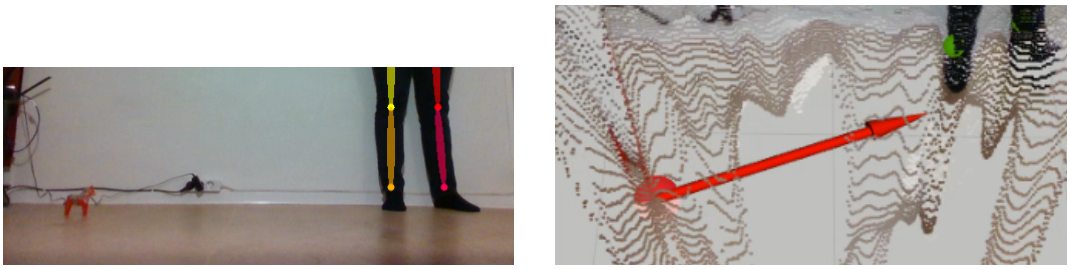
5.7.3 Decreased robot speed when behind the back of a pedestrian

The performance of this implementation was good. The value set to lower the robot velocity was not tested in the actual robot, so no testing was done on the comfortability of it. However, there were not many false positives when the poses were accurately sorted and not many false negatives were detected if at least one joint pair was perceived by the robot. Therefore, the implementation was successful.

5.7.4 Giving the robot a new command by pointing

When the pose of the actual user is chosen as the one that the robot is listening to, the projected point is within $5cm$ when not pointing across the body and 1m away from the body when the right hand is pointing towards the right. When the

right hand is pointing across the body, the projected point is less accurate and within around 20cm from the point pointed at. An illustration of the performance is illustrated in Figure 5.13 where the user is directing the robot to the position of a small wooden horse.



(a) Illustration of performance in 2D

(b) Illustration of performance in 3D

Figure 5.13: Illustration of the performance when directing the robot by a gesture. The user is pointing at the position of a small wooden horse, more clearly illustrated in Figure 5.13(a). In Figure 5.13(b) the small wooden horse can be seen as a red blur at the left. The point pointed at as perceived by the robot is illustrated as a red point, and the resulting vector describing the position and rotation of the movement command is illustrated as a red arrow.

The navigation to the point pointed at is poor when using the Navigation Stack, especially if it is close to the user. The Navigation Stack is sometimes slow to navigate to the point, and in some cases, it completely fails at finding a path. This means that the implementation loses some robustness due to the problems of the Navigation Stack.

5.8 Discussion of the implementation

The purpose of the final interactions that were carried out and implemented in this thesis project are basic concepts that are considered as a sufficient base to be able to determine whether the use of HPE in the AMR system is something to proceed with or not. Based on the interviews it seems like the systems that are running in the industry today require a lot of adjustments made from humans who are working in the same environment as the robots. The interviewees also pointed at some different problematic situations that occur relatively often, which causes disturbances in everyday work both for humans and AGVs. This is indicating that the concepts that are proposed in this thesis could add significant value to the user experience of the AMR system. It was also noted in the market analysis made that this kind of functionality is something that is slowly finding its way into the automation industry.

As the results show, the interactions from the user study in chapter 3 were successfully implemented. The intended use of each interaction is potentially useful according to the conducted interviews, however, this was not verified by actually testing the final result on the interviewees. The resulting implementation also uses separate nodes to make it fit better in the context of other uses of the camera and

to enable parallel processing for better performance.

The problem of discarding poor poses and choosing the main pose to listen to is a large source of instability. For many pose detection methods, some sort of score that describes the estimated accuracy of the pose can be extracted. However, the score of each detected pose is determined as not a reliable metric as the sole reason to discard detected poses. Sorting based on feasibility is better, and the added data can be used to sort out bad poses. Not only the discarding of poor poses is improved in 3D, but due to the instability of simple 2D poses the sorting of what pose is the main user was improved as well. This could still lead to instability of main-user choice, so some object tracking could be implemented as well based on the midpoint of each pose. This would lead to more accuracy in a more complex scenario, but the need for it was not tested for.

Similarly to the problem of poses not being tracked, the implementation interprets poses for each frame separately and not in sequence. This limits the interpretations and is the reason why designing the "point by hand gesture" is difficult to implement in an intuitive way that avoids false-positive readings. A more intricate state machine would solve this problem and enable a more intuitive interaction.

The camera needs to be placed at a reasonable height, meaning not higher than the robot and not lower than the floor. Then, the placement that minimizes the distance needed between a robot and human is at floor level and tilted upward so that the limit of its field of view is parallel to the floor. This is still not very realistic so the mounting height of $0.18m$ is good enough and for the floor to be visible the camera needs to be slightly tilted down for the floor to be in its field of view. On the other hand, if the camera would be rotated 90 deg, then in a similar scenario the distances needed would be only $0.7m$. This latter scenario would however limit the horizontal FoV which would have negative implications for interactions such as the follower mode, as well as for possible other parallel usages of the camera such as obstacle avoidance.

The implementation is designed as separate ROS nodes which opens up the possibility to use a pose estimator written in a different language than the other functions, such as Python, in which many of the HPE methods are implemented in. The publishing rate of the camera-interfacer was set to not be faster than the pose-estimator and this publishing rate depends on which pose estimator is used. The camera would however fill multiple purposes when used on the full AMR system, so the publishing rate simply shouldn't be less than the pose-estimation rate to maximize the rate of the full solution.

5.8.1 Follower mode

The interaction is a simple one, and the interaction is considered intuitive to use. When in follower mode, the performance was limited mainly due to the camera's field of view which makes a case for keeping the camera in landscape mode. During

a first iteration, the Navigation Stack, described in subsection 2.3.3, was used. This resulted in no control for keeping the user in the horizontal field of view, leading to poor behavior in a real usage scenario. When a P-controller was used instead this problem was solved, with the consequence that the user is responsible for not steering the robot into an obstacle. P-controllers were chosen over more advanced ones because they are simple to implement and they worked adequately for the purpose.

Further, from the interviews, it became apparent that the robot should not appear too quick to react when around a pedestrian or user. The problem of the instantly reacting controller could be improved by moving after the user has changed its position outside of a certain threshold.

In the Lightweight OpenPose implementation, the legs get correctly linked together only when the hip is detected. This demands that the robot perceives almost half of the body to function, and within a range of $0.5m$ this is not possible. Using the Navigation Stack the robot could continue navigating to a point as close as $0.5m$ to the user, but there are limitations to this as well. In the final implementation, the robot stops as soon as it cannot perceive the user. There is a theoretical risk of it picking up a new user, the previous second-best candidate in the user-choice in this solution, but it has not been seen in the simulations yet. One alternative is to use the Navigation Stack to autonomously continue after losing track of the user but assumes things about a user no longer in the field of view. Another possible solution is to use LiDAR tracking for this application instead, or at least use LiDAR when the robot is too close to the user. This latter solution was not tested, but there are previous studies tracking users with only a LiDAR which shows that this could be done.

5.8.2 Stopping the robot by gesture

The design of this gesture is considered to be intuitive and the implementation is relatively simple, but it needs to be robust since one usage scenario of it is for safety. The main limitation of this implementation is that the pointing hand easily blocks the view of the rest of the arm as well as of the eye, which is needed to define the gesture. This leads to some missed interactions, so this can not be seen as a very reliable interaction with the current implementation. Further, a full-body pose needs to be far away from the robot to be fully perceived, but it is hard to imagine an intuitive gesture for stopping the robot that does not involve hand gestures. A gesture that does not involve the head could be an improvement in this aspect but that was something that was not looked into further.

5.8.3 Decreased robot speed when behind the back of a pedestrian

This implementation leads to the robot moving more intuitively in the presence of pedestrians. The reason for this is that it behaves similarly to how a car driver acknowledges that it has perceived a pedestrian in traffic. The performance of this

implementation depends on how well poses are detected. If the poses are filtered correctly, the current implementation works well. Because of limitations in the field of view, the legs of one person might not be linked to each other which leads to the corresponding person not being labeled as having its back turned to the robot, with the consequence that a turned back is not detected and the robot continues to move at full speed.

5.8.4 Giving the robot a new command by pointing

The interaction was in a first iteration initiated when the robot perceived a person pointing with a straight arm at the floor. This was considered an intuitive interaction, but it proved to cause too many false positives. The problem was mitigated by adding the requirement that the other arm should have its hand above its elbow. This, however, made the interaction less intuitive. Separating the interaction into an initiation followed by the pointing gesture could be more intuitive. This would however require a more intricate state machine.

5.8.5 Summary

The implementation of the final interaction concepts has been tested and run in a simulation environment where the AMR system successfully responds with the desired behavioral changes. This shows that the concepts seem to work as proof of concept. It would however be preferable to test them on a real AMR system in a realistic environment, which has not been possible due to the pandemic. This would give a much better overall feedback on how the system responds and how well the behavior has improved. To insert the functionality in the product, it would be preferable to have an iterative process with end-user tests who could give feedback on if/how it improves their work environment and what additional changes should be made. This kind of contact would be really valuable in order to design useful improvements as possible.

6
Conclusion

In section 1.2 three research questions were presented which have been the basis of what has been analyzed throughout the thesis. In this chapter, these questions are answered and briefly discussed. The first question to answer was:

Q1: *From the perspective of a person working in the same area as AMRs, what features could improve the behavior of the AMR system using HPE?*

As presented in chapter 3, four interaction concepts have been identified which are believed to have the potential of improving the interpreted behavior of the AMR system. There is e.g. the concept of reducing the speed of the AMR when a human is identified nearby and the function where the AMR follows a person to assist when picking material in a warehouse. These two concepts are mainly considered as passive interactions since they do not require any particular actions from a person in order to be executed. There is also the concept of pointing an idle AMR to a certain location, and the function of stopping and AMR with a gesture. These interactions are instead considered active since they involve an active gesture from a human to be executed. In order to know to what extent these four interaction concepts could be helpful, they would have to be tested and evaluated on multiple worksites using the AMR systems which have not been possible during this thesis project.

The second research question to be answered was:

Q2: *Among available Deep Learning alternatives of online Human Pose Estimation, what are the strength and weaknesses of the different approaches? What method is suitable to be used in real-time on an AMR system?*

To get a good understanding of how the use of different HPE methods could affect the performance of the implemented interaction concepts differently, a study on a set of alternative methods has been presented in chapter 4. There are methods using both the top-down and bottom-up approaches of the HPE task which have shown to perform slightly differently. In this particular application, the bottom-up approach has shown the most promising results, mainly due to the limited field of view given by the RGB-D camera used.

Another part of the study was to get an idea of how the performance of the interaction concepts could improve if there would be a GPU available in the hardware of the AMR system. This difference was retrieved by testing some methods on both CPU and GPU, which unsurprisingly showed a clear improvement in frequency. However, methods run on CPU only has also shown promising results which indicates that a GPU might not be necessary in order to add Human-Robot Interaction functionality to the AMR system.

In order to run the intended functionality on CPU hardware only, the most important trade-off is to have a model that is sufficiently robust at the same time as it can not be too computationally expensive to run on a sufficient frequency. In the set of HPE methods that are examined, there are examples at both ends of this spectra. Based on the current hardware on the AMR system where there is only CPU available, the methods called *Lightweight OpenPose* and *PoseNet* are the ones that have shown the best performance overall and are considered good enough to run in this kind of real-time application.

The third and last question to be answered was:

Q3: *Given a functional HPE system, how should the interactions according to Q1 be designed, and is the AMR response intuitive from a human's perspective?*

A proposed design and implementation of Human-Robot Interaction functionality for the AMR system has been presented in chapter 5. The functionality of each interaction concept retrieved from chapter 3 has been implemented in the AMR system, which during tests are all giving desired responses in the behavior. A proposed design of the active interactions has been presented which based on tests is considered sufficiently robust to use as a proof of concept. The implementation has a flexible architecture that supports e.g. changes between different HPE methods or the addition of more interaction concepts between humans and the AMR.

Overall, the implementation shows that there definitely is a potential of adding Human-Robot Interaction to the current AMR system. Even though it would require further tests with end-users, it is believed to be a functionality that improves the perceived general behavior of the significantly. It is therefore recommended to continue working with this, where the proposed design and implementation is considered a good base for further development.

6.1 Future work

Based on the results and the conclusions that have been obtained in this thesis, some potential further improvements and continuations of the work have been found. These are presented in the following list.

- Based on the initial study on HPE methods conducted, it could be decided what type of approach is the most suitable, such as the bottom-up approach. From that, additional tests with more methods using that specific approach could be conducted to ensure that the best available alternative is used.
- To better verify the usability of the proposed system, end-user tests could be conducted in a suitable environment. This would give good inputs on how the behavior of the AMR is perceived by users, and what additional improvements could be made.
- Due to the limited field of view with the camera used, a full human pose is only seen if far away enough. By rotating the camera such that the long side of its FoV is vertical, a human can stand closer. However, this reduces other concurrent uses of the camera such as obstacle detection. To counteract this,

a camera with a wider field of view, or a fusion of multiple cameras can be used. Alternatively, a dedicated camera with a moving mount can be used for pose estimation.

- Tracking could be implemented for choosing the main user pose, making the choice more stable over time. This could also help when the user is no longer detected when in follower mode.
- The current implementation uses a very simple state machine. Because of this, for example, the interaction of directing the robot by the gesture is defined in a non-intuitive way. To improve on this, or to enable more intricate interactions, a more intricate state machine to handle this should be implemented.
- Link two corresponding legs together, even if the upper body is out of frame. This would improve the performance when the upper body is out of view.

Bibliography

[1] G. V. Research, "Automated guided vehicle market size, share trends analysis report by vehicle type, by navigation technology, by application, by end-use industry, by component, by battery type, by region, and segment forecasts, 2020 - 2027," February 2020. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/automated-guided-vehicle-agv-market

[2] "Kollmorgen automation ab," https://www.kollmorgen.com/en-us/solutions/automated-material-handling/automated-guided-vehicles/.

[3] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: A survey," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1726–1743, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2013.05.007

[4] M. Svenstrup, S. Tranberg, H. J. Andersen, and T. Bak, "Pose estimation and adaptive robot behaviour for human-robot interaction," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3571–3576, 2009. [Online]. Available: https://vbn.aau.dk/ws/portalfiles/portal/36563049/Adaptive-Pose-Estimation.pdf

[5] R. T. Chadalavada, H. Andreasson, M. Schindler, and R. Palm, "Accessing your navigation plans ! Human - Robot Intention Transfer using Eye - Tracking Glasses." [Online]. Available: http://iliad-project.eu/wp-content/uploads/2019/08/Chadalavada_etal-ICMR_2018-Implicit_Intention_Tranference-Data_Analysis-SUBMITTED.pdf

[6] M. Fernandez Carmona, T. Parekh, and M. Hanheide, "Making the case for human-aware navigation in warehouses," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11650 LNAI, no. 732737, pp. 449–453, 2019. [Online]. Available: http://iliad-project.eu/wp-content/uploads/papers/ILIAD-TAROS19-short-paper.pdf

[7] D. Tezza and M. Andujar, "The State-of-the-Art of Human-Drone Interaction: A Survey," *IEEE Access*, vol. 7, no. November, pp. 167 438–167 454, 2019. [Online]. Available: https://www.researchgate.net/publication/337500595_The_State-of-the-Art_of_Human-Drone_Interaction_A_Survey

[8] M. Van Den Bergh, D. Carton, R. De Nijs, N. Mitsou, C. Landsiedel, K. Kuehnlenz, D. Wollherr, L. Van Gool, and M. Buss, "Real-time 3D hand gesture interaction with a robot for understanding directions from humans," *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, no. May 2014, pp. 357–362, 2011. [Online]. Available: https://www.researchgate.net/publication/224256248_Real-

time_3D_hand_gesture_interaction_with_a_robot_for_understanding_directions_from_humans

[9] A. C. Medeiros, P. Ratsamee, Y. Uranishi, T. Mashita, and H. Takemura, "Human-Drone Interaction: Using Pointing Gesture to Define a Target Object," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12182 LNCS, no. July, pp. 688–705, 2020. [Online]. Available: https://www.researchgate.net/publication/342835810_Human-Drone_Interaction_Using_Pointing_Gesture_to_Define_a_Target_Object/link/5f226c09458515b729f33336/download

[10] L. Takayama, W. Ju, and C. Nass, "Beyond dirty, dangerous and dull: What everyday people think robots should do," *HRI 2008 - Proceedings of the 3rd ACM/IEEE International Conference on Human-Robot Interaction: Living with Robots*, no. January, pp. 25–32, 2008.

[11] UN General Assembly, "Transforming our world : the 2030 Agenda for Sustainable Development," *UN General Assembly*, no. A/RES/70/1, pp. 1–35, 2015. [Online]. Available: https://www.refworld.org/docid/57b6e3e44.html

[12] D. M, September 2013. [Online]. Available: http://www.reconditionedforklifts.com/blog/forklift-history-2/history-automated-guided-vehicles

[13] G. Ullrich, *Automated Guided Vehicle Systems*. Springer-Verlag Berlin Heidelberg, 2015.

[14] S. Alexandersson, "Meet a real pioneer in laser-guided agvs," September 2015. [Online]. Available: https://www.kollmorgen.com/en-us/blogs/_blog-in-motion/articles/samuel-alexandersson/meet-a-real-pioneer-in-laser-guided-agvs/

[15] Wikipedia, "Ndc netzler and dahlgren co ab," 2020. [Online]. Available: https://en.wikipedia.org/wiki/NDC_Netzler_%26_Dahlgren_Co_AB

[16] C. Benevides, October 2020. [Online]. Available: https://www.conveyco.com/advantages-disadvantages-automated-guided-vehicles-agvs/

[17] E. Romaine, "Types and applications of autonomous mobile robots (amrs)," August 2020. [Online]. Available: https://www.conveyco.com/types-and-applications-of-amrs/

[18] "One look is worth a thousand words," *Piqua Leader-Dispatch*, August 1913.

[19] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer Science Business Media, 2010.

[20] S. A. Papert, "The summer vision project," 1966. [Online]. Available: http://hdl.handle.net/1721.1/6125

[21] R. Klette, *Concise Computer Vision: An Introduction into Theory and Algorithms*. University of Auckland: Springer London Heidelberg New York Dordrecht.

[22] F. Luongo, R. Hakim, J. H. Nguyen, A. Anandkumar, and A. J. Hung, "Deep learning-based computer vision to recognize and classify suturing gestures inrobot-assisted surgery," August 2020. [Online]. Available: https://arxiv.org/abs/2008.11833?

[23] "Depth camera d435," https://www.intelrealsense.com/depth-camera-d435/.

[24] "About ros." [Online]. Available: https://www.ros.org/about-ros/

[25] A. Koubaa, *Robot Operating System (ROS)*. Springer International Publishing AG, 2019.

[26] "Ros communication." [Online]. Available: http://wiki.ros.org/ROS/Patterns/Communication

[27] "Ros publishers and subscribers." [Online]. Available: http://wiki.ros.org/roscpp/Overview/Publishers%20and%20Subscribers

[28] Y.-Y. Chen, Y.-H. Lin, C.-C. Kung, M.-H. Chung, and L.-H. Yen, "Design and implementation of cloud analytics-assisted smart power meters considering advanced artificial intelligence as edge analytics in demand-side management for smart homes," May 2019.

[29] T. Balodi, "Introduction to machine learning: Supervised, unsupervised and reinforcement learning," 2019. [Online]. Available: https://www.analyticssteps.com/blogs/introduction-machine-learning-supervised-and-unsupervised-learning

[30] J. Schmidhuber, "Deep Learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015. [Online]. Available: https://arxiv.org/pdf/1404.7828.pdf

[31] A. Ye, "Supervised learning, but a lot better: Semi-supervised learning," 2020. [Online]. Available: https://towardsdatascience.com/supervised-learning-but-a-lot-better-semi-supervised-learning-a42dff534781

[32] O. Sci, "Cpu vs gpu," 2020. [Online]. Available: https://www.omnisci.com/technical-glossary/cpu-vs-gpu

[33] M. V. Valueva, N. N. Nagornov, P. A. Lyakhov, G. V. Valuev, and N. I. Chervyakov, "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation," *Mathematics and Computers in Simulation*, vol. 177, pp. 232–243, 2020. [Online]. Available: https://doi.org/10.1016/j.matcom.2020.04.031

[34] Deeplizard, "Convolutional neural networks (cnns) explained," December 2017. [Online]. Available: https://www.youtube.com/watch?v=YRhxdVk_sIs

[35] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, no. February, pp. 3642–3649, 2012.

[36] V. Bansal, "The evolution of deep learning," 2020. [Online]. Available: https://towardsdatascience.com/the-deep-history-of-deep-learning-3bebeb810fb2

[37] A. L. Chandra, "Mcculloch-pitts neuron — mankind's first mathematical model of a biological neuron," 2018. [Online]. Available: https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1

[38] T. J. Sejnowski, *The Deep Learning Revolution*. The MIT Press, 2018.

[39] "Connectionism," 2015. [Online]. Available: https://plato.stanford.edu/archives/fall2018/entries/connectionism/

[40] G. E. Hinton, "Learning multiple layers of representation." Trends in Cognitive Sciences. 11 (10), October 2007.

[41] K. D. Foote, "A brief history of deep learning," February 2017. [Online]. Available: https://www.dataversity.net/brief-history-deep-learning/

[42] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and D. . Jurgen Schmidhuber, "Flexible, high performance convolutionalneural networks for image classification."

[43] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification." 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012.

[44] Hazelcast, "What is machine learning inference?" 2020. [Online]. Available: https://hazelcast.com/glossary/machine-learning-inference/

[45] M. Robins, "The difference between deep learning training and inference." [Online]. Available: https://www.intel.com/content/www/us/en/artificial-intelligence/posts/deep-learning-training-and-inference.html

[46] Y. Chen, Y. Tian, and M. He, "Monocular human pose estimation: A survey of deep learning-based methods," *Computer Vision and Image Understanding*, vol. 192, p. 102897, 03 2020.

[47] Q. Dang, J. Yin, B. Wang, and W. Zheng, "Deep learning based 2D human pose estimation: A survey," *Tsinghua Science and Technology*, vol. 24, no. 6, pp. 663–676, 2019.

[48] Z. Cao, T. Simon, S. E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 1302–1310, 2017. [Online]. Available: https://ieeexplore.ieee.org/document/8099626

[49] W. Gong, X. Zhang, J. Gonzàlez, A. Sobral, T. Bouwmans, C. Tu, and E. H. Zahzah, "Human pose estimation from monocular images: A comprehensive survey," *Sensors (Switzerland)*, vol. 16, no. 12, pp. 1–39, 2016.

[50] D. Mehta, O. Sotnychenko, F. Mueller, W. Xu, S. Sridhar, G. Pons-Moll, and C. Theobalt, "Single-shot multi-person 3D pose estimation from monocular RGB," *Proceedings - 2018 International Conference on 3D Vision, 3DV 2018*, pp. 120–130, 2018.

[51] V. Gupta. (2018) Deep learning based human pose estimation using opencv (c++ / python). [Online]. Available: https://www.learnopencv.com/deep-learning-based-human-pose-estimation-using-opencv-cpp-python/

[52] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *International Journal of Robotics Research*, vol. 35, no. 11, pp. 1352–1370, 2016. [Online]. Available: https://journals.sagepub.com/doi/pdf/10.1177/0278364915619772

[53] K. L. Koay, D. S. Syrdal, M. Ashgari-Oskoei, M. L. Walters, and K. Dautenhahn, "Social Roles and Baseline Proxemic Preferences for a Domestic Service Robot," *International Journal of Social Robotics*, vol. 6, no. 4, pp. 469–488, 2014.

[54] M. Kok, J. D. Hol, and T. B. Schön, "Using inertial sensors for position and orientation estimation," *Foundations and Trends in Signal Processing*, vol. 11, no. 1-2, pp. 1–153, 2017.

[55] NVIDIA-AI-IOT, "trt_pose." [Online]. Available: https://github.com/NVIDIA-AI-IOT/trt_pose

[56] NVIDIA, "Nvidia isaac sdk." [Online]. Available: https://developer.nvidia.com/isaac-sdk

[57] D. Osokin, "Real-time 2d multi-person pose estimation on cpu: Lightweight openpose," https://arxiv.org/pdf/1811.12004.pdf, 2019.

[58] Intel, "Openvino™ toolkit overview," 2020. [Online]. Available: https://docs.openvinotoolkit.org/latest/index.html

[59] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once:unified, real-time object detection," 2015. [Online]. Available: https://arxiv.org/abs/1506.02640

[60] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," April 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[61] École Pylytechnique Fédérale de Lausanne, "Fastpose," 2018. [Online]. Available: https://drnoodle.github.io/fastpose_html/

[62] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *CVPR*, 2019. [Online]. Available: https://arxiv.org/abs/1902.09212

[63] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *CVPR*, 2015.

[64] P. Naf, "Overview of human pose estimation neural networks — hrnet + higherhrnet, architectures and faq — 2d3d.ai," June 2020. [Online]. Available: https://towardsdatascience.com/overview-of-human-pose-estimation-neural-networks-hrnet-higherhrnet-architectures-and-faq-1954b2f8b249

[65] S. Pini, "Multi-person human pose estimation with hrnet in pytorch." [Online]. Available: https://github.com/stefanopini/simple-HRNet

[66] "Keypoint detection on coco." [Online]. Available: https://paperswithcode.com/sota/keypoint-detection-on-coco

[67] 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 *CVPR*, 2020.

[68] S. Pini, "Multi-person human pose estimation with higherhrnet in pytorch." [Online]. Available: https://github.com/stefanopini/simple-HigherHRNet

[69] G. Papandreou, T. Zhu, L.-C. Chen, S. Gidaris, J. Tompson, and K. Murphy, "Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model," March 2018. [Online]. Available: https://arxiv.org/abs/1803.08225

[70] R. Wightman, "Posenet python," 2019. [Online]. Available: https://github.com/rwightman/posenet-python

[71] Cubemos, "Skeleton tracking sdk for intel® realsense™ depth cameras." [Online]. Available: https://www.intelrealsense.com/skeleton-tracking/

[72] *Intel RealSense D400 Series Product Family*, 005th ed., https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/Intel-RealSense-D400-Series-Datasheet.pdf, Intel, 2019.

A

Interviews

A.1 Interview 1

Following are the main points from an interview with a user experience (UX) designer that is working with the usability and design of an AMR system. It has no practical experience in working with the application of AMRs, but is experienced in conducting user tests and what typical inputs users could give that developers generally do not think about.

- Users do not want to use new functions if it is too difficult to use, they want to have it as easy as possible.
- Extended functionality needs to be reliable, otherwise it will not be used.
- A robot should not be "annoying", if it is coming up from behind it should not give the user a signal to move. Instead it should wait or take another route.
- A user would want to get acknowledgement from a robot that it has been seen.
- The behaviour of an AMR is more intuitive if it acts more human-like.
- An intuitive behaviour of vehicles could be found in normal car traffic, since they are driven by humans and are something that people are used to.
- The size and perceived threat of an AMR decides what speeds are interpreted as comfortable and suitable for a pedestrian.

A.2 Interview 2

Following are the main points from an interview with a worker at a site with 8-10 AGVs that handle pallets. The AGVs are mostly in contact with warehouse workers.

- The robots can sometimes get uncomfortably close to people.
- When they do get close they give off an audio signal
- Accidents have happened where AGVs have driven in to manual trucks.
- The AGV does not wait for you when you leave, instead it drives directly after you.
- It should be clearer where the AGV is going
- The AGV should not start moving as soon as it can after a person has moved.
- One idea is that when a person is doing *kitting* the AGV could follow the person performing the kitting task.
- If there is a problem during kitting, the person should be able to point the robot in another direction to make it leave.
- The AGV cannot stack pallets.
- A set of AGVs should be able to move like a caravan.

- There should be a more convenient way to load pallets.
- An AMR should be able to be taken out of the kitting process, and then taken in again, and an other AMRs should make room for it if it is to be taken in again.
- A user can get scared if the user is not prepared for the robot to appear.
- The robot should not warn by a beep, instead a voice would be better.
- A user should get informed if a robot has perceived the user.
- If a user is looking at a robot, the robot should respond some way.

A.3 Interview 3

Following are the main points from an interview with a worker at a site with five AGVs and also a set of manual trucks. The facility where the AGVs operate is described as having narrow passages such that it sometimes could be difficult for a pedestrian to pass an AGV without being very close to it.

- AGVs are sometimes hit by manual trucks
- The AGV is often in the way, however you learn its route.
- The speed of the AGV has been reduced in production areas, while a higher speed is maintained in the warehouse.
- No reported serious accidents
- It has been reported that that an AGV has hit a pedestrian
- The AGV uses sound- and visual signaling to communicate with pedestrians.
- The sound signals are perceived as irritating by the interviewee.
- The AGV is slowing down when an obstacle in the way is detected.
- The presence AGVs could sometimes give a feeling of insecurity, e.g. when they suddenly show up directly behind the back.
- When shown the demo of stopping the AMR by hand gesture, the response was positive. It is possible that a manual truck can get trapped by an AGV, and it is necessary to manually turn off and move the AGV to get out. This could solve the issue.
- It has happened that an AGV has not detected the forks of a manual truck and has driven in to them resulting in damages.
- When shown the demo of directing an AMR by hand-gesture the interviewee was worried that the AMR would lose its positioning.
- Using an AMR should be as simple as possible if many people are using it. For example, how do you get it working if it gets stuck.
- An AGV should be able to find a pallet itself within a certain area, and not only rely that it is placed on an exact position.