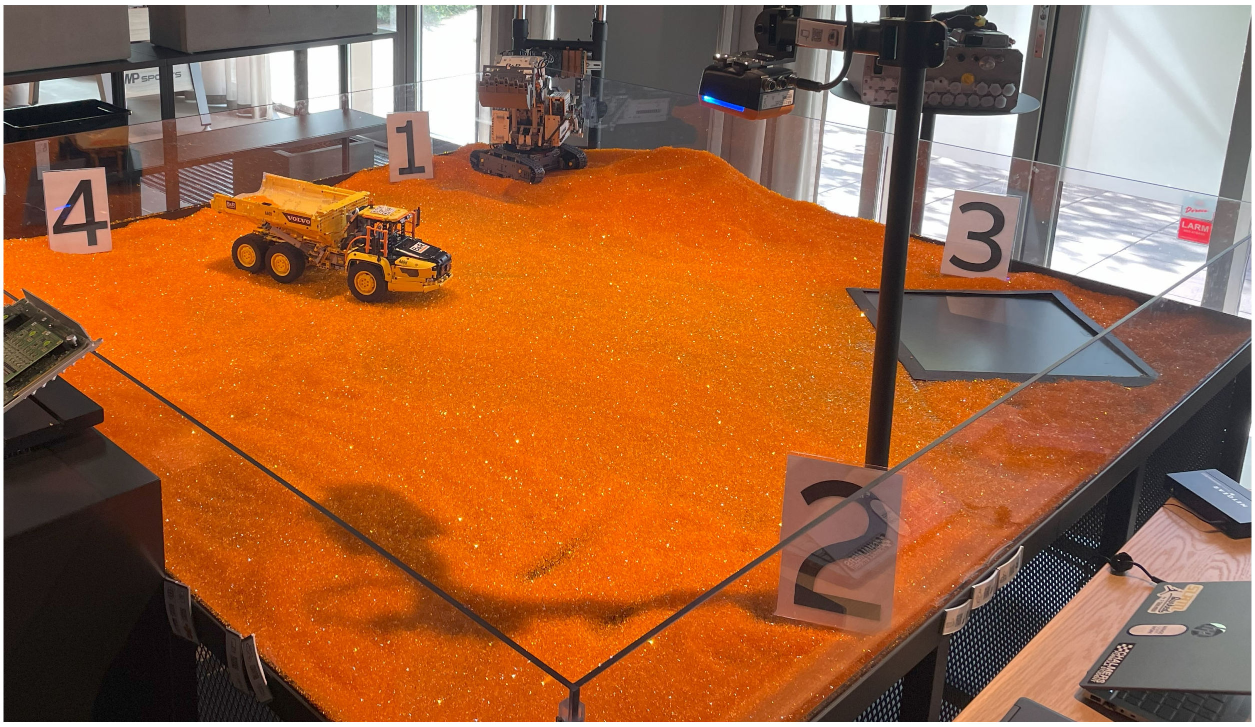


Camera-based State Estimation and Autonomous Motion Control

Perception and Control of a Hauler Truck in a Demo Site

Master's thesis in Systems, Control and Mechatronics

Kevin Bielecki
Rasmus Ekedahl

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

Camera-based State Estimation and Autonomous Motion Control

Perception and Control of a Hauler Truck in a Demo Site

KEVIN BIELECKI
RASMUS EKEDAHL

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2024

Camera-based State Estimation and Autonomous Motion Control
Perception and Control of a Hauler Truck in a Demo Site
KEVIN BIELECKI
RASMUS EKEDAHL

Supervisor: Hanna Hermansson, B&R Industrial Automation
Examiner: Martin Fabian, Electrical Engineering

Master's Thesis 2024
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The demonstration area where the system is deployed.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Camera-based State Estimation and Autonomous Motion Control
Perception and Control of a Hauler Truck in a Demo Site

Kevin Bielecki
Rasmus Ekedahl

Department of Electrical Engineering
Chalmers University of Technology

Abstract

This thesis explores the development of an autonomous system, designed for B&R Industrial Automation to demonstrate autonomous solutions on their products without any operator input. The goal of this thesis is to develop an autonomous system that can manoeuvre a mobile unit between different stations, in a collision-free and smooth manner primarily to enhance sales demonstrations. With a single camera mounted in the ceiling, the system can make well-informed decisions using perception, motion planning and motion control. Key components include a machine-learning model for perceiving the environment, a path- and trajectory planner, and a linear Model Predictive Control (MPC) system. The project resulted in a fully functional autonomous system that could execute demonstration runs, offering opportunities for further development.

Keywords: Autonomous systems, Computer vision, Object Detection, YOLO, Path-planning, Motion planning, Motion control, MPC, Machine learning.

Acknowledgements

This master's thesis was carried out at B&R Industrial Automation during the spring of 2024. We wish to thank our academic examiner and supervisor from Chalmers, Professor Martin Fabian, for his continuous support and feedback throughout this project.

We also want to give special thanks to our supervisor, Hanna Hermansson, and everyone at B&R Industrial Automation who assisted us during this project. Your dedication, support, and the opportunity to work on this thesis have been irreplaceable.

Kevin Bielecki, Rasmus Ekedahl, Gothenburg, June 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AMR	Autonomous Mobile Robot
APC	Automation Personal Computer
AVX2	Advanced Vector Extensions 2
CNN	Convolutional Neural Networks
COCO	Common Objects in Context
CPU	Central Processing Unit
GPU	Graphics Processing Unit
HMI	Human-Machine Interface
IoT	Internet of Things
IoU	Intersection over Union
mAP	Mean Average Precision
MPC	Model Predictive Control
PID	Proportional – Integral – Derivative
SSD	Single-shot Detector
TPU	Tensor Processing Unit
YOLO	You Only Look Once
ZOH	Zero-Order Hold

Nomenclature

Below the nomenclature that has been used throughout this thesis is presented.

Indices

i	Index for iterations
k	Index for discrete time step

Parameters

Δt	Time discretization step (time interval) [ms]
t	Time [ms]
L	Wheel base [m]
L_t	Trailer length [m]
n	Number of waypoints
N	Control horizon

Variables

δ	Steering angle [rad]
a	Longitudinal acceleration $[m/s^2]$
x	x-coordinate [m]
y	y-coordinate [m]
θ	Heading angle [rad]
v	Longitudinal velocity [m/s]
ψ	Relative angle of trailer [rad]
$\mathbf{x}$	State vector

P	Point including x and y coordinate [m]

Contents

List of Figures

List of Tables

1
Introduction

As technology evolves, the pursuit of automation expands in several areas [1]. Within the field of autonomous systems, perception and control are two fundamental challenges that play an important role in enabling systems to make well-informed decisions based on their surroundings. By continuously monitoring and gathering information from the surrounding environment, the system can update its internal representation of the world. This technique is often referred to as state estimation and its data can be used in decision-making systems such as a motion planner to determine feasible or optimal routes from a starting point to a goal. Further, motion controllers are used to ensure that the planned path is maintained. By combining these components, autonomous systems could navigate through complex and dynamically changing environments to reach desired locations on time, efficiently and safely.

1.1 Background

This master thesis is a collaborative project with B&R Industrial Automation. To display pioneering technology and digital innovation with B&Rs products, a showroom called OrangePoint is facilitated at the main office in Malmö. Within OrangePoint, one of the demonstrations showcased is a small-scale site containing a miniature hauler truck referred to as a mobile unit, displayed in Figure 1.1. The mobile unit can be controlled remotely by an operator via Bluetooth communication, to drive the truck between different stations using a computer from B&R. To further improve this showroom, B&R wants to implement a fully automated system that can replace the operator, navigating the area and driving the mobile unit between multiple stations. The goal is to complete various tasks and interact with different products from B&R and external suppliers.

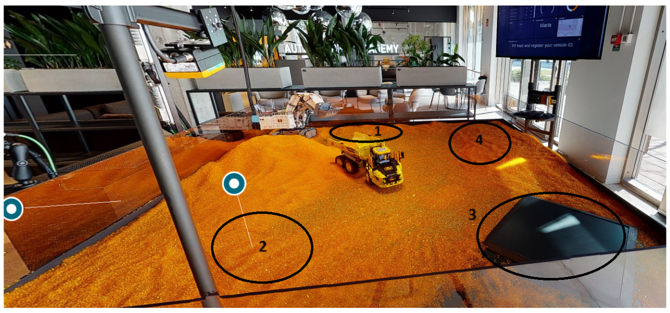


Figure 1.1: Demo site with marked stations at B&R industrial automation in Malmö.

The demonstration site displays a fusion of Internet of Things (IoT) and Cloud services, incorporating third-party solutions and enabling remote connectivity. It showcases how various industries can develop a resilient, robust and future-ready platform with B&Rs hardware and software solutions [2]. The proposed addition of an autonomous feature aims to enable the mobile unit to autonomously navigate a complex environment by leveraging real-time data analysis, machine learning algorithms, and computer vision, deployed on a hardware control unit from B&R. This feature is anticipated to ensure safe and efficient operation of the mobile unit while travelling between different stations within the demonstration area. Its integration is seen as a crucial step in enhancing the platform's capabilities and displaying how B&Rs technologies can be utilized at the forefront of mobile automation.

The demonstration area is a compact 2×2 meter square, designed to simulate a sand-covered environment using a base layer and larger piles of orange plastic. This area also contains a miniature mobile digger and various equipment with the potential of expanding with more features and products in the future. The layout features static obstacles, sharp turns and areas where a combination of reversing and going forward is necessary to navigate and effectively reach the desired positions, thus posing several challenges when designing an autonomous system.

For this project, Figure 1.1 illustrates four pre-determined stations. Loading material at station 1, camera and QR-code identification at station 2, weighing of the loaded vehicle at station 3, and finally unloading at station 4. The objective is for the mobile unit to autonomously navigate between these stations in a safe and robust manner. At each station, the mobile unit will pause and wait for a "go-ahead" signal, which indicates the completion of the required process before moving on to the next station.

1.2 Aim

The primary goal of this thesis is to investigate different approaches, and develop a system for the perception and control of an autonomous mobile unit with hardware from B&R. The results should be presented on a mobile unit that will travel between different stations inside a demonstration area without any operator input.

1.3 Limitations

To define the scope of the project, the following list of limitations and aspects are considered within this thesis.

- The system is only intended for the demo site and therefore not general purpose, meaning modifications of setup or different locations will require adjustments and additional work to ensure the same performance.
- The primary goal of this project is to achieve smooth operation of the mobile unit rather than optimal computational efficiency.
- The system is limited to only one mobile unit, thus not considering other mobile units within the area.
- The perception system will not consider the location of anything but the mobile unit. All other obstacles are considered static, and therefore their locations are predefined.

1.4 Research Questions

This project aims to implement an autonomous system that enables the mobile unit to navigate between different stations. To assess the system and guide development, the following research questions will be explored.

1. To what extent can camera-based state estimation of a mobile unit be used to determine different kinematic properties, including position, speed, and relative angle of joints?
2. What type of control strategy will ensure sufficient motion planning and control of the autonomous mobile unit within the predefined area to ensure safe and efficient navigation without operator input?
3. What are the key factors affecting the reliability and accuracy of the autonomous system controlling the mobile unit, and how can these factors be managed?

1.5 Ethics and Sustainability

This thesis aims to explore the field of perception and control of autonomous mobile units, a technology that has many upsides when implemented on a larger scale. This project is a proof of concept on a small scale within a secure area. However, the same technologies and principles can be applied within other industries and on a larger scale. Therefore it is important to consider the sustainability and ethical implications of the project, as well as the potential consequences this technology could have for the future. Below some important ethical and sustainability aspects are further addressed primarily focusing on autonomous transportation vehicles.

The perception and control within the autonomous system must be reliable and robust to ensure safety for the vehicle and its surroundings. However, in the case of failure, the autonomous system must be able to hand over control or go into a fail-safe mode in order to come to a collision-free and safe stop [3]. At the same time, removing the "human factor" from critical operations could lead to a reduced risk of errors and mitigate the risk of human injuries.

The perception functionality of an autonomous system generally contains sensitive data about its environment. Whether the data is collected with cameras, lidars or GPS, the contents must be protected from external operators with ill intent. It is also important to ensure the integrity of other people is upheld. To mitigate this problem it is critical that this data is properly protected and defining data destruction as a continuous process [3].

While autonomous transportation could improve the social working conditions within several industries, by removing monotonous work and heavy lifting, it is important to consider the aspect of a increased lack of jobs for humans. Truck- and forklift drivers and many more occupations risk replacement in the future as autonomous driving technology evolves [3]. Therefore the impact of deploying autonomous technology should be assessed in each separate industry and a plan for relocation the workforce should be made.

From an environmental perspective, autonomous mobile units' capability to optimize routes and driving behaviours can also result in enhanced energy efficiency compared to manual operations. This efficiency could lead to decreased emissions, contributing positively to environmental sustainability, while also enabling cost savings for companies applying these technologies [3].

2

Preliminaries

This section establishes the theoretical foundation for the report, providing the necessary background on concepts and methods essential to this thesis.

2.1 Computer Vision and Machine Learning

Enabling machines to interpret and understand visual inputs, could often involve identifying and locating objects within an image [4]. Object detection can be achieved by classifying different parts of each image into various categories and using techniques like deep learning and convolutional neural networks (CNNs). There are several different approaches within this field, where algorithms such as Region-based Convolutional Neural Networks (R-CNN) or You Only Look Once (YOLO) algorithms are commonly used. These models improve the speed and accuracy of detection by focusing on specific regions of interest within the image and executing classification in a single pass. These networks learn to recognize patterns and features from larger datasets and labelled images and are widely applied within different industries, where real-time accuracy is crucial. Ongoing research and development in object detection focus on increasing the robustness and efficiency of these models. Efforts include improving the training datasets to cover more diverse scenarios and conditions, optimizing algorithms to reduce computational demands, and refining accuracy to distinguish between closely similar objects [5].

2.1.1 You Only Look Once

You Only Look Once (YOLO) is a deep learning-based algorithm mainly used for object detection claiming to provide real-time performance, high accuracy, and is open source [6]. The YOLO model can estimate both bounding boxes and predicts object classes simultaneously, while still maintaining high accuracy. The object detector used in YOLO is a single-shot method, meaning that the entire frame of the image is analyzed and made predictions, all at the same time. This approach differs from many other methods, such as RCNN or Fast RCNN, which first detect possible regions of interest and then perform image recognition. Figure 2.1 classifies different algorithms, where the main distinction between the different approaches suggests that single-shot methods result in improved real-time performance while two-shot detection yields a higher accuracy.

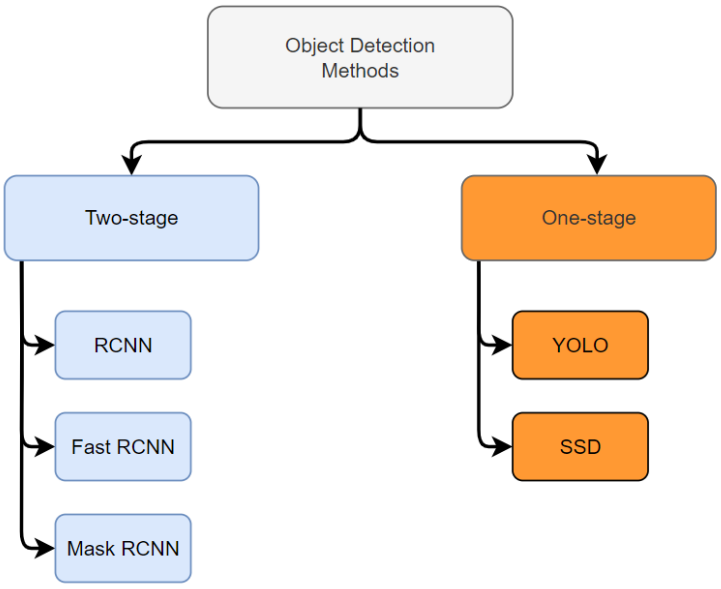


Figure 2.1: Different methods of object detection classification

Figure 2.1 describes the branching of the different object detector methods and whether they are of Two-stage or One-stage classification. The YOLO-model is a CNN that can be used to recognize and identify items with high speed and accuracy [6]. The detection model consists of 24 convolution layers where 20 of these are pre-trained. These are then followed by 2 fully connected layers, which in the end yield a $7 \times 7 \times 30$ tensor of predictions. This direct prediction mechanism is what enables YOLO to achieve its high speed, differentiating it from other detection systems that often employ separate steps for feature extraction, proposal generation, and object classification. The YOLO architecture is shown in Figure 2.2 with the $7 \times 7 \times 30$ tensor as the final output.

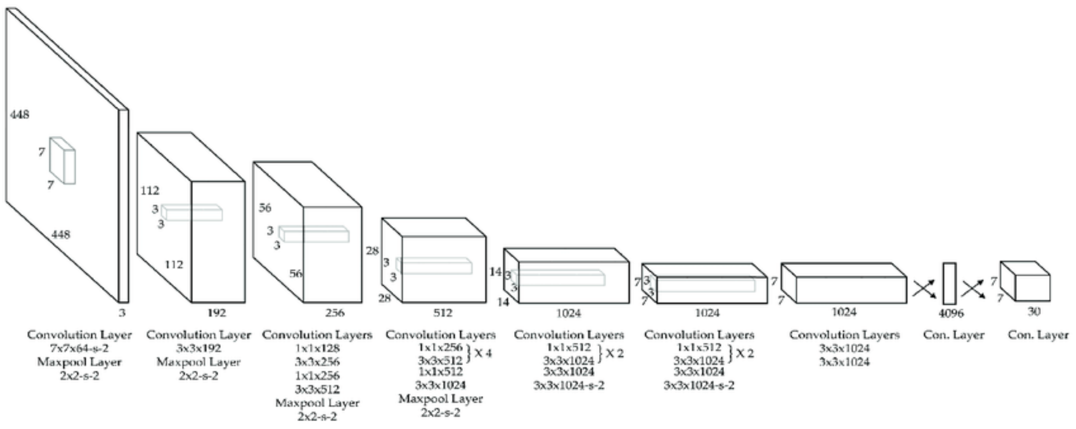


Figure 2.2: YOLO architecture [6].

Since the first version of YOLO was published in [6], different versions of the algorithm including YOLOv3, YOLOv4, all the way up to YOLOv9 have been developed. Each iteration aims to improve its detection accuracy and speed while keeping a low computational complexity to achieve real-time performance. Further information and comparison of different versions of the YOLO algorithm can be found in [7, 8].

2.1.2 Model Quantization and Pruning

One main goal when designing new deep-learning models is improving the accuracy. However, this commonly also results in larger model sizes. Consequently, with larger models comes the need for more computational resources. Simultaneously, there is a demand for the deployment of high-precision models on less powerful hardware, both in terms of cost and scalability. Two effective strategies for achieving these objectives are *pruning* and *quantization*. These techniques not only help in scaling down the models but also ensure that their precision remains as close to the more dense models as possible [9].

Large sets of weights in deep-learning models are commonly replicated more than once, lack content, and are together with different pathways not important after training the model [10]. Pruning involves removing parts of the model that are considered less important or redundant for deployment. The primary goal is to reduce the complexity of the model without significantly impacting its performance or accuracy. Reducing the model parameters can result in a model with improved runtime performance and reduced computational complexity.

Additionally, most weights in a model are typically too precise for runtime applications and this type of precision is generally not needed after the model has been trained [10]. Quantization refers to the method of reducing the precision of the numbers used to represent the model weights. With quantization, the computational power and memory needed to run the model can be reduced using lower precision datatypes such as 8-bit integers instead of the standard high precision 32-bit floating points.

2.2 Optimization problems

An optimization problem seeks the optimal solution from a set of possible choices. The primary objective of such problems is to find the minimum or maximum value of a function, known as the objective function, under a set of constraints. These constraints are typically expressed as inequalities and equalities that the solution must satisfy [11]. An example of a mathematical formulation of an optimization problem is presented in (2.1).

$$\begin{aligned} \min_{x} \quad & f_0(x) \\ \text{subject to} \quad & g_i(x) \leq b_i, \quad i = 1, \dots, m \\ & h_j(x) = 0, \quad j = 1, \dots, p \end{aligned} \tag{2.1}$$

In (2.1), the vector x is the decision variable, and x^* is the optimal solution to the problem. The function f_0 represents the objective function or cost, that is to be minimized, while $g_i(x)$ and $h_j(x)$ represent inequality- and equality constraints that the solution must respect. The constraints ensure that the solution not only optimizes the objective function but also remains within a feasible region defined by the set limits. A solution to the optimization problem corresponds to a choice that has a minimum cost, among all choices that meet the constraints [11].

2.2.1 Convex vs Non-convex optimization problems

The nature of the objective function and the set over which the optimization is performed can be divided into two different types of optimization problems: convex and non-convex, with their differences illustrated in Figure 2.3.

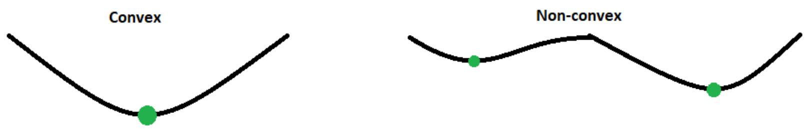


Figure 2.3: Distinction between a convex and a non-convex function.

A convex optimization problem is generally characterized by every local minima also being a global minimum within the feasible set. This property significantly simplifies the search for an optimal solution as convex problems only have a unique solution or multiple solutions forming a convex set. In comparison, non-convex functions contain both local and global minima, posing a larger challenge to find the optimal solution as the problem becomes more complex and computationally heavy [12]. The general trade-off between these types of optimization problems is the gain in accuracy compared to the added computational complexity. Non-convex problems allow for more complex problem formulations that could yield a higher accuracy than a convex function, however, solving these types of problems generally requires more computational power to converge to the optimal solution.

3
Technical Concept

The main purpose of this thesis is to research and develop methodologies that enable a mobile unit to navigate fully autonomously within a pre-defined area. This can be divided into three partial problems, state estimation, motion planning, and motion control, which are open-ended problems and can be solved in multiple different ways. Thus, to highlight the aspects that form the basis of the chosen technical concept, the following sections focus on how requirements and limitations affect the choice of technical solutions for the project.

3.1 Company Requirements

Ensuring that the technical solution is adapted to its use case is an important aspect of the autonomous system which will primarily be used in sales demonstrations, showcasing the capabilities of B&R's hardware and software in autonomous applications. To achieve this alignment, a set of system requirements was defined together with the company, ensuring that the solution covers all desired aspects. Table 3.1 presents the system requirements for the project.

Requirement	**Priority**	**Details**
Performance analysis of hardware	Must have	Explore the potential applications of B&R's hardware technology with software that requires significant computational resources.
Modularity	Must have	All parameters that affect the system behaviour should be simple to change to display different scenarios.
State estimation with a camera	Must have	Estimate the position of the mobile unit, relative angle, and velocity.
Cropping of image	Must have	Used to specify what parts of the camera frame objects should be detected within. Should be easy to adjust based on the location of the demo area.
Go through production cycles	Must have	Go between different stations within the site, where the location of each station can be adjusted.
Defining restricted areas	Must have	A configurable area where the vehicle is allowed to move.
Efficient and smooth navigation	Must have	Opt for a visually appealing and smooth route to enhance the audience's perception.
Automatic adjustment of cropping parameters	Nice to have	Automatic detection of demo location and orientation to adjust cropping parameters.
Adaptive visual mapping	Nice to have	Mapping with the camera obstacles and areas that are difficult to traverse within.

Table 3.1: System requirements for the project.

In Table 3.1, the tasks listed as *must have* are critical for the project and must be implemented. The topics that are listed as *nice to have* are to be implemented if time permits.

From the company's perspective, the autonomous system actuating the mobile unit should be appealing to customers, displaying a complex technical solution with intuitive and smooth movements and decisions. The project therefore prioritises a final product with the above-mentioned features rather than finding the optimal solution in terms of shortest path, energy efficiency etc.

3.2 Hardware Specifications

Based on the requirements above, a technical analysis was performed on the given hardware to further evaluate its capabilities. This was mainly done for benchmarking purposes and to eliminate any potential bottlenecks later on in the project, concerning hardware specifications. Below, a description of each hardware component used in the project is specified.

3.2.1 Automation PC (APC)

The processing unit used in this project is the APC Mobile 3100, a product from B&R Industrial Automation displayed in Figure 3.1. The computer houses a PLC and a Linux operating system, both using an Intel processor for computations. All work done within this project is done on the Linux side of the APC. The objective is to integrate the entire system within a single B&R hardware unit, along with components from other suppliers. If necessary, the selected APC can be upgraded to an APC Mobile 3100 with an Intel i7 central processing unit (CPU) and increased RAM.

Parameters	Specifications
Material number	5MPC3100.K038-000
CPU model	Intel Celeron 3965U
CPU speed	2.2 GHz
RAM	8 GB

Figure 3.1: Mobile Automation PC 3100 [13].

3.2.2 Camera

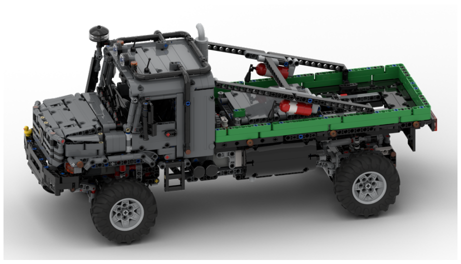
The camera selected for this project is an ArkCam image sensor, specifically designed for monitoring both mobile and stationary industrial environments [14]. The sensor choice by the company is strategic, primarily because of its widespread use in mobile applications. The goal is therefore to incorporate it in the demonstration area to show its capability to be integrated seamlessly with B&R's hardware alongside other components from various suppliers, aiming to deliver a complete system solution. Figure 3.2 provides sensor specifications and shows the ArkCam sensor.

Parameters	Specifications
Max videostream	1280x720@60fps
Latency	<100 ms
Power consumption	<3 W
Viewing angle	130°

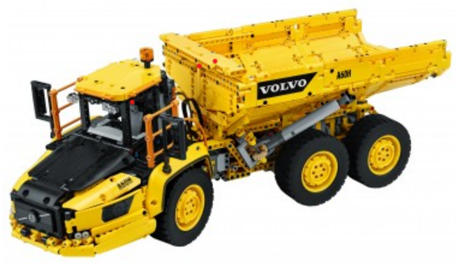
Figure 3.2: ArkCam basic+ Mini 130 and table of specifications.

3.2.3 The Mobile Unit

In this project, the used mobile units are miniature Lego trucks designated to demonstrate the autonomous functionality of the system. However, the result of this thesis work is meant to be applicable across a range of sectors, including the autonomous mobile robot (AMR) industry, automotive- and construction industries, and other fields that employ similar technologies and the Lego trucks are utilized for proof of concept. The two mobile units employed in this project are shown in Figure 3.3.



(a) Rigid truck for testing during development.



(b) Articulated truck used in the demonstration area.

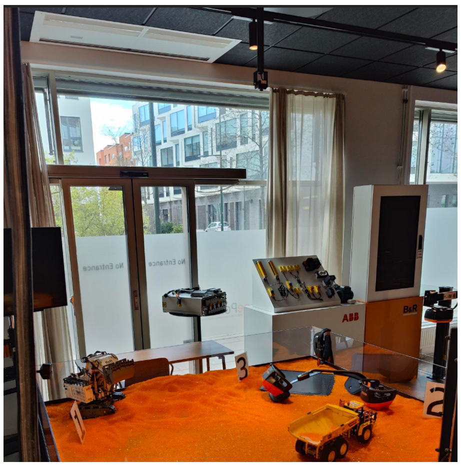
Figure 3.3: The two mobile units used in the project.

Figure 3.3a, shows a truck with rigid dynamics i.e. the steering capabilities are directly influenced by its wheelbase. In comparison, Figure 3.3b has another degree of rotational freedom around the joint between the head of the truck and the trailer. The latter truck is the main unit used in the demonstration area. However, since it is commonly used for sales purposes, the rigid truck it has been utilized for testing the system during development.

Both mobile units are controlled with longitudinal drive motors and a motor controlling the requested steering angle, where the steering geometry is Ackerman. The main implication of using two different mechanical setups is the manoeuvrability of the vehicle. Simply put, a rigid body provides simpler dynamics when performing more complex actions such as reversing etc. However, when comparing turning capabilities the articulated mobile unit allows for better manoeuvrability as the pivot point allows for sharper turns, making it more suitable when manoeuvring tight spaces such as the demonstration area [15]. The trade-off between complexity and manoeuvrability is further analyzed mathematically in Chapter 6.1.

3.3 Conceptual overview

With the specified hardware and desired functionality of the system, a conceptual overview could be determined, including operational behaviour and technical solutions. With the current functionalities considered and the fact that the mobile unit does not hold an adequate internal processing unit, a centralized approach was taken where the APC estimates the position, plans the desired path and actuates the mobile unit based on a single sensor, a camera mounted above the demonstration area as shown in Figure 3.4.


(a) Camera mounted in the ceiling.

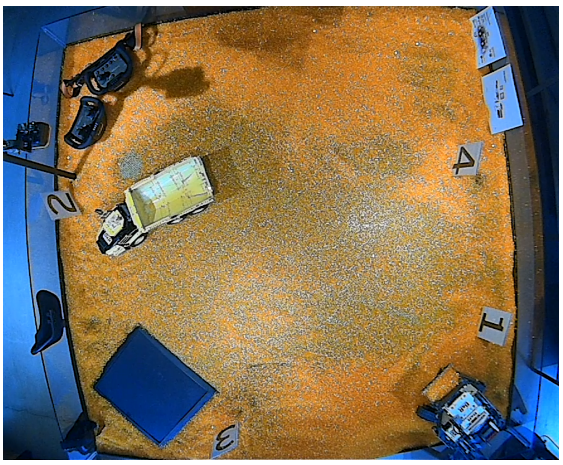

(b) Field of view from the camera.

Figure 3.4: Camera setup in the demonstration area.

A simplified overview of the system architecture is presented in Figure 3.5, where the system is divided into 3 larger sections, *Perception*, *Motion Planner*, and *Motion control*.

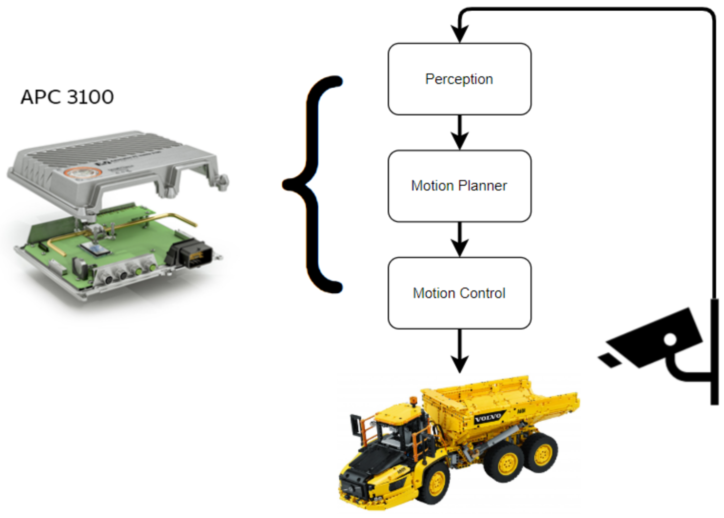


Figure 3.5: Full system overview.

With a centralized approach, all subsystems must run on a single CPU, limiting the computational complexity and thereby what can be achieved in terms of real-time performance. Therefore, this aspect must be considered throughout the whole project.

With a single camera as the available sensor and providing the system with real-time updates, the desired states of the mobile unit can be estimated. This will be achieved using a machine-learning model for object detection within the perception system. Simultaneously, the desired goal state is given as an external input to the system. The motion planner then combines this with the current state from the perception system to plan the desired path and define a set of desired states at each iteration. Finally, a motion controller utilizes all these inputs combined with a pre-defined feasible area to decide how to manoeuvre the vehicle to reach the desired goal state collision-free and smoothly. The upcoming chapters will delve deeper into each of the subsystems discussed, providing detailed explanations of the reasoning behind the decisions, methodologies, and implementation processes.

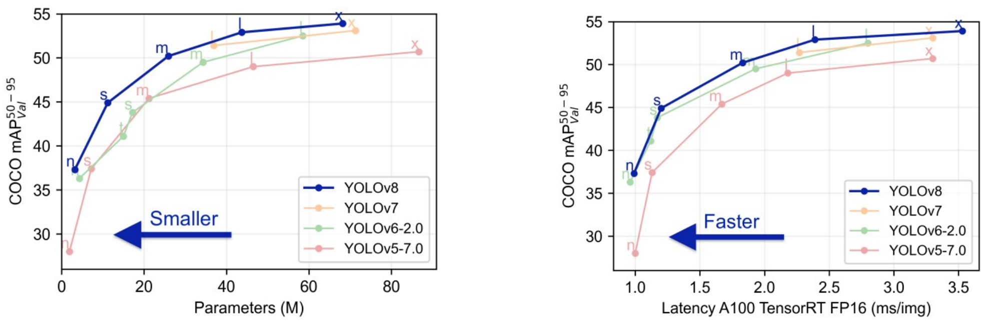
4 Perception

In robotics and autonomous systems, accurately perceiving the environment is important for effective and accurate operations. Perception involves the use of various sensors to gather data, which is then processed to estimate the state of the system. These states can be essential as they provide the system with an object's position, orientation, and other desired dynamic attributes critical to its decision-making processes. Among the sensors available, cameras are particularly valuable due to their rich data capture. To leverage this data effectively, object detection frameworks can be employed, which can enable recognition and tracking of objects within the camera's field of view. There are several frameworks available for object detection, each with its unique strengths and applications. Some of the most recognized are R-CNN, SSD, and YOLO models [16].

4.1 Choosing the Perception Framework

In this project, a camera serves as the primary input sensor, capturing visuals of the system's environment. Utilizing computer vision techniques, the developed system is designed to detect, recognize, and track the mobile unit, leveraging this data to estimate the desired states of the mobile unit. The challenge in accurately estimating the state from visual inputs lies in dealing with varying visual conditions, potential obstacles, undetectable objects and the necessity for processing the data at real-time speed. A critical aspect of this project is identifying and consistently tracking the targeted mobile unit in real time, which is essential for a reliable system. With these aspects in mind, a one-stage method was desired to minimize the model complexity, narrowing the selection down to YOLO or SSD. Based on prior comparisons the YOLO framework was selected due to its smaller model size, comparable real-time accuracy and its compatibility with other tools [17].

The trade-off between speed and accuracy became more nuanced as the YOLO framework evolved. Each YOLO model is aimed at increasing either speed, accuracy, or a balance of both. YOLOv4, for instance, is noted for its robustness and efficiency in real-time settings. YOLOv5 for its new innovations, such as new network backbones, improved data augmentation techniques, and optimized training strategies. YOLOv7 and YOLOv8 further push the boundaries in terms of accuracy, integrating techniques from the latest research to improve detection performance [18]. Figure 4.1 shows a comparison of various YOLO models, illustrating the correlation in model size, speed, and accuracy for the different models.



(a) The relationship between model complexity and detection accuracy.

(b) The tradeoff between inference speed and accuracy for the same models.

Figure 4.1: Performance comparison of different YOLO models [18].

From Figure 4.1 it can be observed that YOLOv8 maintains a leading accuracy rate while it has a slightly larger number of parameters compared to YOLOv6-2.0. This indicates a higher efficiency in parameter utilization since it achieves higher accuracy with a comparable number of parameters. In addition to this, YOLOv8 displays the lowest latency. For a real-time application, this is critical as it means the model can process and analyze frames more efficiently. With all this considered, YOLOv8 is the most balanced option for the project's real-time application. It achieves the highest accuracy and does so at the least latency.

During the course of the project, a new iteration of the YOLO algorithm, YOLOv9, was released. Comparisons between the previous version, YOLOv8, and YOLOv9 suggest that the latter offers reduced complexity and enhanced performance for object detection [19]. However, as YOLOv9 is in the initial stages of deployment, and compatibility with additional tools such as trackers, visual aids, and other functionalities remains limited, this poses challenges for its integration. Therefore the choice of using YOLOv8 for this project remains.

The perception system in this thesis is based on the YOLOv8 model, with a custom-trained model dataset. The YOLOv8 model handles object detection, classification, and segmentation tasks. YOLOv8 introduces improvements over previous YOLO versions, such as better feature extraction, more sophisticated backbones and features that make it easier to use and tailor for the project's specific application. This results in enhanced accuracy and lower latency, especially in challenging scenarios like small object detection or in conditions with poor lighting or occlusions. In addition to this, YOLOv8's architecture allows for efficient custom training with new datasets. In comparison to other models like Faster R-CNN, SSD, or Mask R-CNN, YOLOv8 offers a superior balance of speed, accuracy, and flexibility [18].

4.2 Perception System Overview

The perception system works on a three-phase principle: process video, process frame, and annotate frame. Figure 4.2 shows a flowchart of the working principle. Each principle of the system is structured as a distinct task within the 'State Estimation' group. This approach organizes the different parts of the program into specific, manageable sections, each responsible for a particular aspect of the overall process.

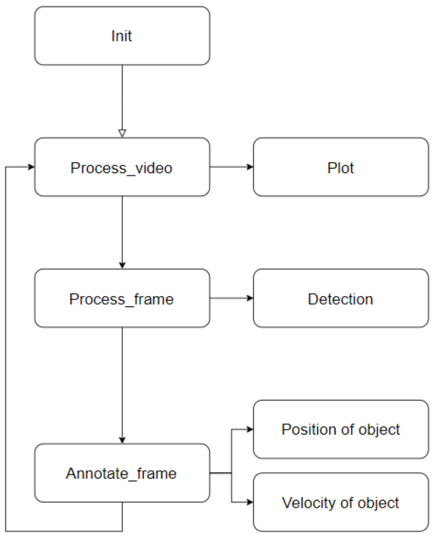


Figure 4.2: Flowchart of the three-phase principle of the perception system.

Figure 4.2 presents a visualization of the three-phase principle and how the principles of the perception system communicate. The perception system features a dynamic configuration script which allows the configuration of settings to be adjusted. The process video principle captures raw footage from the chosen video source. Then, the process frame principle uses the YOLOv8 model for executing computer vision tasks, enabling to detect and track objects. The model continuously updates for each frame to maintain accurate detection and tracking. Furthermore, the annotate frame principle ensures that all detections are confined within the pre-set detection zone. The system operates in a loop, consistently refreshing the visual output in sync with the system's frequency.

In this thesis, the detection zones have been specifically set to suit the sandbox area at OrangePoint in Malmö. This configuration ensures that detections are limited to objects within the sandbox, effectively filtering out irrelevant targets. The video input for the system is streamed from a live camera feed, strategically placed in the ceiling above the sandbox. Both the detection zone and video input source are customizable and can be tailored to desired specifications through a configuration file. The perception system is designed to accept inputs into the detection zone that are proportionate to the camera's field of view. This configuration assumes that the sandbox at OrangePoint in Malmö remains unrotated. Any rotation would introduce inaccuracies in the position calculations within the state estimation process. The location of the unrotated sandbox can be configured depending on the desired position.

4.3 State Estimations

State estimation is a process aimed at deducing the state of a system, such as position, using observed data. It uses a mathematical model, later described in Chapter 6.1, that describes how the system's state changes over time and how this state correlates with the observed data. In the real world, measurements often come with noise and may not be complete. When implementing state estimation with a camera and computer vision, the approach involves using a camera to capture images or video frames, which act as observational data. These images may include objects whose states (like position or orientation) are to be estimated. Computer vision methods are then applied to identify and track features of objects across successive frames. Techniques such as optical flow or object detection algorithms, for instance, YOLO, are commonly used [7]. The detected features allow the system to estimate movements or positional changes of the objects. Many applications require these estimations to be performed in real-time, necessitating the use of efficient algorithms and, in some cases, the support of hardware acceleration.

4.3.1 Position and Velocity

In this thesis, the YOLOv8 computer vision model is used together with a camera to enable real-time tracking of a mobile unit. Object detection, the core task of this model, involves specifying the location and categorizing objects within an image or video stream. In this thesis, a video stream acts as the visual input for the model.

The detector's output consists of bounding boxes that encompass the identified objects in each frame, together with class labels and confidence scores. To track the mobile unit, separate tracking algorithms like BYTE, SORT (Simple Online and Real-time Tracking) or DeepSORT can be used [20]. These algorithms take the detections from YOLOv8 and apply a series of steps to achieve continuous tracking of objects as they move across the video frames. The primary challenge is to maintain the identity of each object from frame to frame, despite changes in position, orientation, scale, or in interaction with other objects. In this thesis, BYTE is used to track the mobile unit, because of its robust and accurate detection performance. BYTE uses the help of associating multiple low-score detection boxes as it can indicate the existence of objects. It also highlights the method for using detection outcomes to improve multi-object tracking [20].

The position is gathered by extracting the centre point of the detection box for the tracked object, irrespective of its orientation. The x and y coordinates for the detected objects are updated with each updated frame, in line with the frequency Δt. The x and y coordinates are plotted on a 2D plane, as the camera is oriented directly over the demo area. Initially, the origin is placed at the top left corner of the frame. However, upon configuring specific detection zones, the origin shifts to the top left corner of the selected detection zone, as illustrated in Figure 4.3.

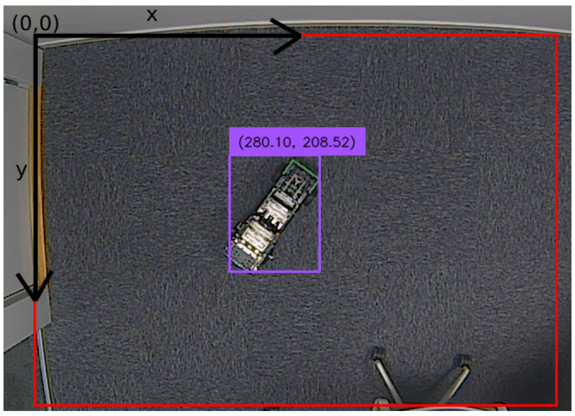


Figure 4.3: A frame of the Mercedes Lego truck with active perception system, displaying the coordinate system and the detection area

Figure 4.3 shows a snapshot from the perception system's output. It features the detected object outlined in purple, along with its x and y centre coordinates. Additionally, there's a red box that represents the pre-set detection zone. The origin of the coordinate system is indicated in the top left corner of the detection area.

To calculate the velocity of a moving vehicle using a camera, it is essential to track how specific points on the vehicle shift over time. These points, which remain fixed to the bounding box centre point, move at the same velocity and direction as the vehicle when it is in motion relative to the camera. In this project, the camera is strapped to the ceiling, so the vehicle's speed is measured in relation to the camera, which corresponds to the speed relative to a stationary plane in the camera's view.

To determine the vehicle's speed, frames captured by the camera are analyzed. This process allows for the measurement of the vehicle's momentary speed. The calculation of this velocity is based on the change in the position of the reference points across the current frames according to:

$$v = \frac{\Delta P}{\Delta t}. \tag{4.1}$$

Here v is equal to the velocity of the truck, and ΔP corresponds to the Euclidean distance between two points, displayed in (4.2). Δt corresponds to the measured time difference it takes for the vehicle to be transported the distance ΔP, i.e $\Delta t = t_k - t_{k-1}$.

$$\Delta P = \sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2} \tag{4.2}$$

The velocity, v, is a velocity vector of a point where $v \in \mathcal{R}^2$ i.e in 2D space since only one camera is used in this thesis. The measured time, Δt, is equal to the time which passes between two processed video frames and is equal to the update frequency. To find the velocity of the vehicle, one point is not enough. For this reason, the estimate of the velocity of the mobile unit can only be used after two time periods, $t > 2\Delta t$, where at least two positions have been registered. Figure 4.4 illustrates the points and timestamps necessary for calculating the velocity.

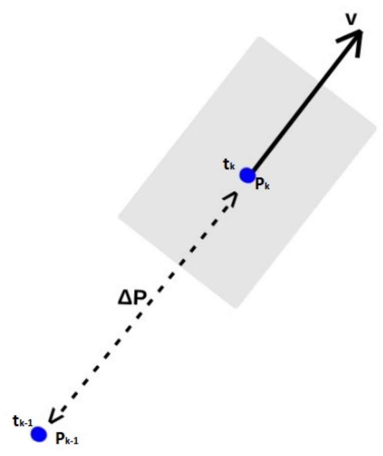


Figure 4.4: Illustration of the position, velocity and timestamps for the mobile unit

When dealing with image processing and tracking, the orientation and placement of the camera play a crucial role in how to interpret and manipulate the captured data. If a camera is not positioned in a bird's-eye view, that is, directly overhead, the resulting images can exhibit perspective distortion. This distortion will then skew the perceived dimensions and positions of objects within the frame, calculating the kinetic properties like velocity or distance impractical.

One common approach to this problem is to apply a coordinate transformation [21]. This process involves adjusting the image coordinates to reflect the true layout of the frame. It corrects for the perspective-induced distortions, aligning the image closer to what would be seen from a top-down view. The transformation is essential for precise tracking and measurement, as it ensures that the calculations are based on the actual arrangement of objects, rather than their distorted image representations.

This thesis is however fortunate to avoid these complexities. The camera setup is strategically placed directly above the surface that is being tracked. This positioning provides a bird's-eye perspective, thus naturally eliminating significant distortion that would otherwise be present. As a result, the images that are captured are already in a desirable format for analysis, decreasing the complexity of the perception system. No camera calibration has been performed in this project but could yield improved accuracy as distortion due to the camera lens is still present.

4.4 Model Performance and Training

The YOLOv8 architecture provides a range of different-sized models. Some of these models are presented in Table 4.1. When comparing each model at a set pixel size, the mean average precision (mAP) and the latency can be evaluated. The mAP measures the average precision of an object detection model over a range of intersection over union (IoU) thresholds, in this case between 50% – 95% [22].

Table 4.1: Performance metrics for different YOLOv8 models.

Model	Pixel Size	$\text{mAP}^{\text{val}}_{50-95}$	Speed CPU ONNX [ms]
YOLOv8n	640	37.3	80.4
YOLOv8s	640	44.9	128.4
YOLOv8m	640	50.2	234.7
YOLOv8l	640	52.9	375.2

Within object detection with YOLOv8, Table 4.1 displays a trade-off between the model's size and its performance characteristics on a CPU. A larger model typically yields increased accuracy and precision but comes with a cost of decreased processing speed and a higher demand on computational resources. To choose a suitable model for the application one must achieve a balance between accuracy and speed, taking into consideration the computational capacity available for the task.

Based on the data presented in Table 4.1 and considering the hardware specifications outlined in Chapter 3.2, the YOLOv8n "nano" model was selected for the project. The choice was made to keep inference time as low as possible, accepting a certain trade-off in accuracy to ensure real-time performance.

Utilizing a CNN for the detection and tracking of an object is a crucial part of the overall system as the perception lays the foundation for the other systems to make well-informed decisions. Thus, the state estimation system must be robust, ensuring that the tracking of the mobile unit is never lost. This was not the case when using the pre-trained model provided with YOLOv8, resulting in the detection of multiple undesired objects, tracking loss, and false negatives as shown in Figure 4.5. Therefore the model had to be trained on a custom dataset suitable for the application.

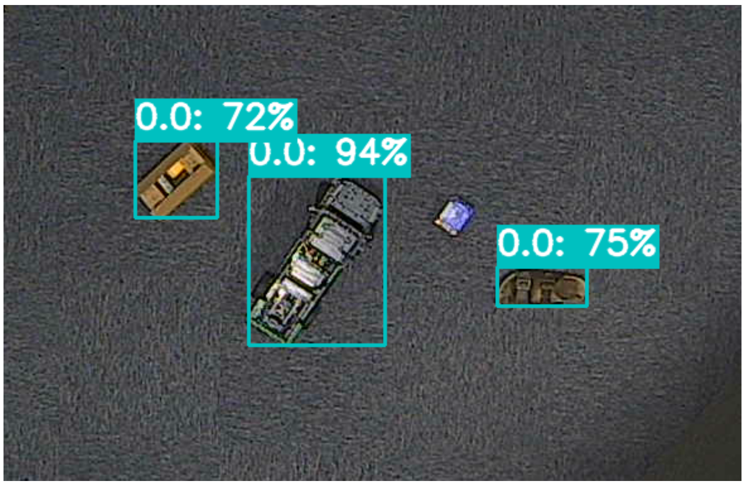


Figure 4.5: Undesired detections and false negatives with a pre-trained YOLO model on the COCO dataset.

To achieve desirable results, the model must be modified and trained on data such that the mobile unit can be recognized and tracked at all positions within the demo area. For the chosen YOLOv8-n model the following steps were taken to obtain a robust model that could complete the tasks specified in Chapter 4.3.

4.4.1 Data Acquisition

The performance and efficiency of a YOLO model are highly dependent on the data that it is trained on. Therefore, the pre-trained YOLOv8-n model was extended and trained with a dataset including images of the mobile unit in various contexts within the demo area. To save time and ensure a varied dataset, multiple videos of the mobile unit were taken, covering different production cycles in various conditions and placements around the demo area. The objective was to form a broad dataset to enhance the model's ability to generalize effectively and mitigate the risk of overfitting. The gathered videos were then converted into images, with a selected number of captured frames from each video to compile a large dataset. The final dataset consisted of 1450 images, where a sample of the labelled dataset is shown in Figure 4.6.

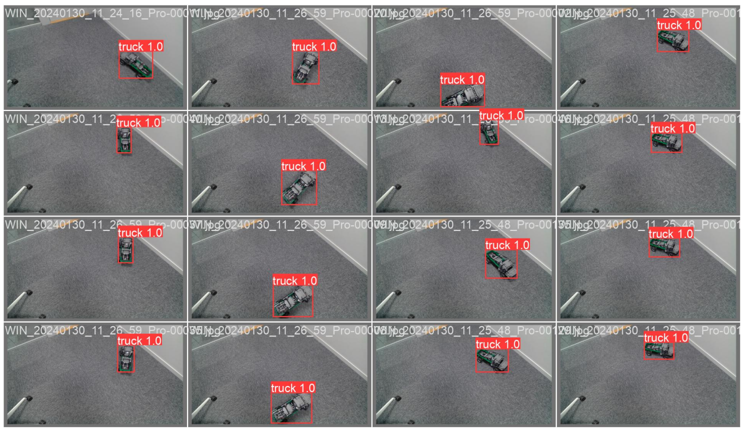


Figure 4.6: Sample of the labelled dataset used for the perception system for testing on the Mercedes truck.

4.4.2 Image Annotation

Image annotation is the process of adding metadata to a set of images, i.e. annotating the desired objects within each frame with bounding boxes and labels. This is done to guide the algorithm to learn from the provided data and emphasise certain points.

The process of image annotation can be time and resource-consuming for a large dataset. Therefore, to avoid manually annotating each image in the new dataset a base model was employed to automate the process. A base model is a large foundation model that can be applied for multiple purposes, trained on large datasets [23]. Within this project, the Grounded Segment Anything Model (SAM) is used, a model that can segment out individual objects from an image [24]. The base model is trained on over 11 million images and 1.1 billion masks, and when given prompts of desired objects it can annotate a large dataset with bounding boxes and labels quickly and without any other external inputs. A result of this is shown in Figure 4.6.

4.4.3 Model Training

Training a YOLOv8 computer vision model for real-time applications, particularly for consistent detection and tracking of a specific object, is critical. When utilizing transfer learning to train a dataset precisely tailored to the trait of a particular object, the model significantly improves in detecting that object accurately. It learns to identify unique features and variations of the object, effectively distinguishing it from similar items or background interference. To achieve the distinction between the target and other objects, a set of the early layers in the model are frozen, meaning that they are not updated during the training process. Instead, only the deeper layers are fine-tuned with the new data. This is an important aspect as it leverages the generic features learned from the standard dataset and adapts more specific features in the deeper layers. With this method the number of false positives and negatives can be reduced, thereby improving the accuracy.

Moreover, the effectiveness of the model in real-time scenarios depends on its ability to swiftly and reliably re-identify the object in successive frames, adapting to movement and partial obscurations. This training is key to maintaining consistent tracking, regardless of changing conditions. Additionally, by refining the model's focus on a specific object, it becomes operationally more efficient. This efficiency translates to reduced computational complexity, making the model a better fit for systems with limited processing capabilities, such as the system developed in this project. This targeted training approach not only elevates the model's performance in its primary task but also enhances its applicability and reliability, for identifying and tracking the desired object [25].

The effectiveness of the detection model is significantly impacted by how well the training data is balanced to avoid underfitting and overfitting, particularly when the model is trained for a singular objective. Underfitting occurs when the trained model is too simplistic, failing to capture the complexity and variability in the data. This can lead to poor performance as the model cannot generalize well to new, unseen scenarios. On the other hand, overfitting occurs when the model is excessively tailored to the training data, capturing noise and anomalies as if they were significant patterns. This may result in a model that performs well on training data but poorly on new, real-world data, as it becomes too specialized [26]. To avoid this, a varied and large dataset is used for the model training, allocating 80% of the dataset to training and 20% for validation. To avoid overfitting, an “early stopping” algorithm is implemented when training the model. This entails continuously monitoring the validation metrics and stopping the training of the model if the metrics indicate a performance plateau over a set amount of epochs, i.e. the model does not display improved performance over time with more training.

4.4.4 Process Acceleration on CPU

With the trained model implemented in the preception system, the maximum allowed throughput was deemed to be 100 [ms] to ensure that each iteration for the full system could be completed within 200 [ms]. However, without accelerating the process on the target hardware, the time for a single computational iteration (preprocessing, inference, and post-processing) took approximately 600 [ms].

To mitigate the problem of insufficient inference rates in real-time applications, machine learning models are typically deployed on GPUs (Graphics Processing Units), or TPUs (Tensor Processing Units), which are capable of conducting numerous parallel operations. Alternatively, strategies such as model acceleration or sparsification, including pruning and quantization, could be employed to speed up the inference rate [10].

As stated in Section 3.2, the target hardware unit within this project is an APC containing only a CPU. Thus, common hardware acceleration techniques such as the use of GPUs are not available. Instead, the focus is shifted towards model acceleration and sparsification to reduce the complexity of the model and in return increase the model throughput. To achieve this without compromising accuracy to a significant extent, different tools can be applied. Within this project, a few methods were investigated and are briefly highlighted below.

4.4.4.1 ONNX Runtime

ONNX runtime [27] is a machine-learning engine aimed at executing inference on a wide range of platforms and hardware to accelerate the throughput. To obtain this, the engine analyzes the model's graph and determines how it can be optimized for execution. Then the model is partitioned and the engine can thereafter dynamically assign computational tasks, thus ensuring efficient execution of individual tasks and a holistic optimization of the entire model.

4.4.4.2 OpenVino

For an Intel-based system OpenVino [28], short for Open Visual Inference & Neural Network Optimization, can be applied to optimize and improve inference on a target hardware application. Developed by Intel, the tool compresses the deep learning models and supports deployment- and hardware optimization for a large number of Intel CPUs, taking advantage of the specific hardware capabilities of each supported device.

4.4.4.3 DeepSparse

DeepSparse [29] is an engine that utilizes sparsity to accelerate inference within neural networks on CPUs. By utilizing structured and unstructured sparsity, weights with no impact on the system during runtime are known and can thereby be avoided during runtime. To further optimize for CPU architectures, the runtime computations are organized into "Tensor-columns", allowing effective cache utilization. This is done by reducing the amount of data transportation in and out of the larger cache memories, which usually is a large bottleneck for memory-bound systems [29]. The DeepSparse tool facilitates acceleration for both dense models and models sparsefied through quantization and pruning. In this project, both model types are evaluated to investigate the performance enhancements of a reduction in model complexity. The evaluation and selection of the acceleration method is displayed in Chapter 7.1.3.

5 Motion Planning

The primary objective of a motion planner is to determine how a mobile unit should navigate through a specified environment. This includes deciding the desired path that the mobile unit should take, as well as its associated states such as position, velocity, and pose at each point in time. To achieve this, the motion planning problem is divided into two parts, a global path planner and a trajectory planner. The global path planner effectively links the mobile unit's initial state to a set of specified goal states and the trajectory planner then locally plans the desired states along the path, taking the physical constraints into consideration, similar to [30]. Within this project, the environment is considered static and all objects are mapped beforehand within the 2D space.

5.1 Path Planner

The path planner aims at finding a path between the start and goal states. By assuming no dynamic obstacles except for the mobile unit itself affecting the environment, a predetermined map can be used to determine the desired path. This involves the creation of a grid map of the demonstration area, where the environment is discretized into a series of nodes. These nodes serve as stations in defining and facilitating the navigation of the mobile unit's path.

With a grid map defined, multiple approaches can be taken to find an optimal path. Conventional trajectory optimization techniques such as search-based algorithms like A*, or sampling-based algorithms such as RRT are commonly used [31,32]. However, within this project, the focus is not on finding the optimal path in terms of distance, energy minimization or time, but rather on ensuring that the path is aesthetically pleasing, easy to modify and smooth. Therefore a graph-based approach is used, where the shortest Euclidean distance between each node is interpolated and used as the desired path for the mobile unit. This was done to reduce computational complexity and rely on the motion controller to maintain a smooth and collision-free path. A simplified example of such a grid map with a path is shown in Figure 5.1 to visualize how the system internally interprets the environment.

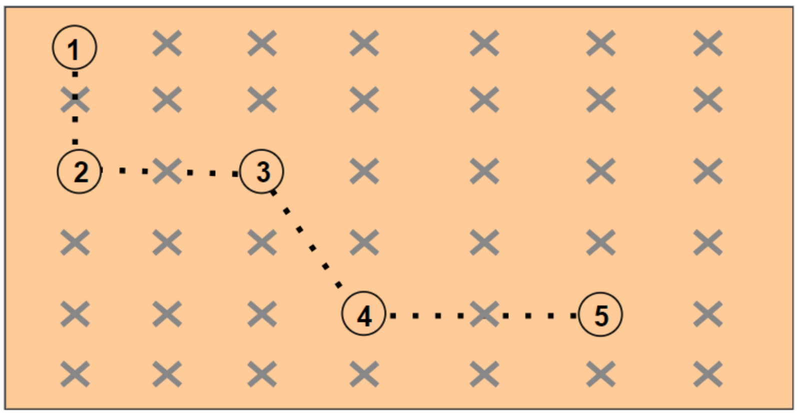
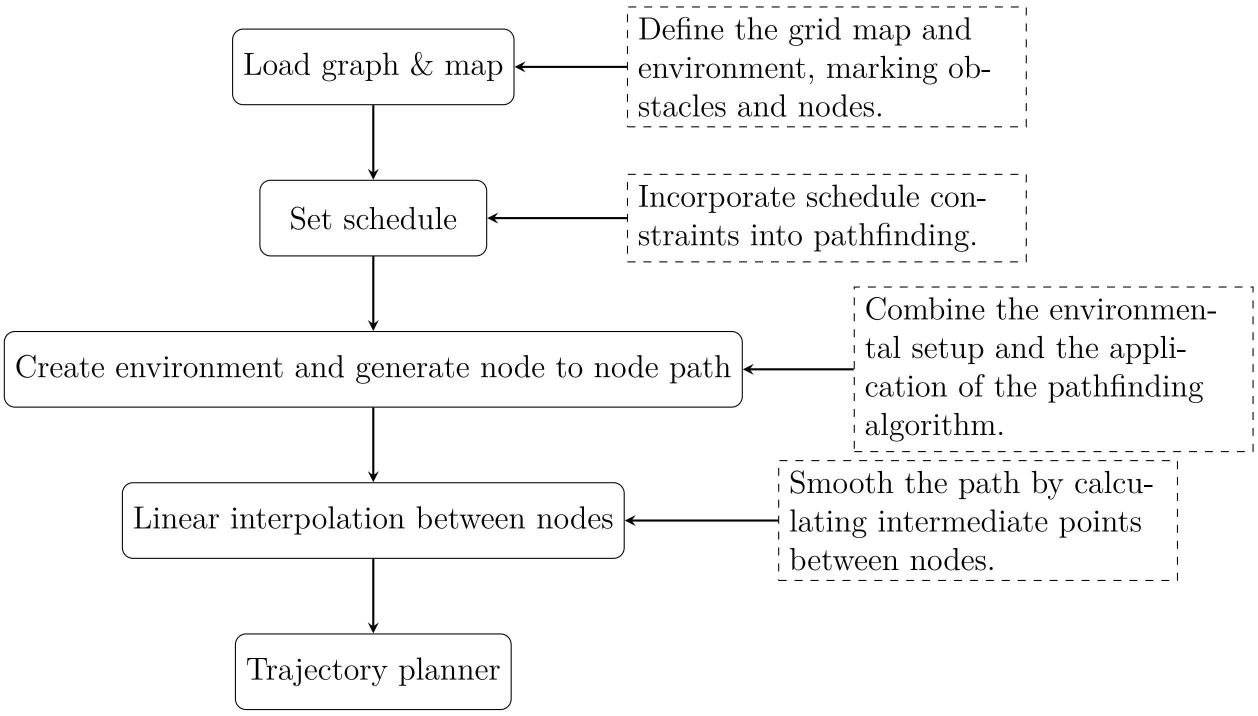


Figure 5.1: Visual representation of the grid map with defined station nodes.

To define a desired velocity and behavior when approaching the different stations a schedule is also provided to the path planner. This entails a specification of when the mobile unit should arrive at each station, allowing the system to incorporate more aspects of the desired behaviour into the final motion plan. An overview of the path planner is presented in Figure 5.2.


Load graph & map
Set schedule
Create environment and generate node to node path
Linear interpolation between nodes
Trajectory planner

Define the grid map and environment, marking obstacles and nodes.

Incorporate schedule constraints into pathfinding.

Combine the environmental setup and the application of the pathfinding algorithm.

Smooth the path by calculating intermediate points between nodes.

Figure 5.2: Flowchart of path generation in a grid map environment with a description for each process.

The purpose of this system is to generate a navigable path within a predefined environment using the Cartesian coordinates x, y. The algorithm comprises processing a schedule, generating a node-to-node path, and performing linear interpolation between nodes to smooth out the path.

5.2 Trajectory Planner

The purpose of a trajectory planner is to enable a robot to navigate its desired path in a way that respects its physical limitations [33]. This involves determining a set of reference states along the desired path that the mobile unit should adhere to. A trajectory planner is designed as a subsystem within the greater motion planning system, subsequently providing the reference states over the control horizon at each time step, k. The trajectory planner translates the path information such that the control system can manage detailed motor instructions for controlling its movement, taking the mobile unit's current state into account, the planned node-to-node path, and the dynamic constraints of the environment.

With the information from the path planner, a set of references can be defined for the mobile unit based on the given path. The states of the model include the x and y coordinates and the heading angle θ. Additionally, a longitudinal reference velocity, v, is incorporated into the state vector, where the velocity reference is based on the distance to the next station and the desired arrival time. To ensure that the set references adhere to what is physically possible to achieve, physical constraints are incorporated to saturate the references including the linear velocity and heading angle. To ensure obtainable reference states, the x and y coordinates are also limited to only within the boundary of the predefined demonstration area.

The reference generation's main goal is to discretize a continuous reference path into a series of states over the control horizon, N. The framework for generating a linear reference trajectory uses two principal functions: global path sampling, and segment-wise interpolation. The global path sampling function systematically invokes the segment-wise interpolation to create a path of uniformly spaced points from a given set of waypoints.

The global path sampling function operates on a set of n waypoints, $\mathcal{W} = \{W_1, W_2, \ldots, W_n\}$, that define the trajectory. The objective is to construct a sequence of points $\mathcal{P}$ that captures the essence of the path with a desired resolution.
The step size, Δs, is calculated as the product of the vehicle's velocity v and the control system's sampling time interval Δt. The function iteratively samples each segment of the trajectory, where the distance Δs determines the gap to the next node on the generated path. Within each segment between consecutive waypoints, W_i and W_{i+1}, segment-wise interpolation is executed. For a segment of length L where $L = \|W_{i+1} - W_i\|$, a series of intermediate points are computed based on the linear interpolation principle:

$$P(\lambda) = W_i + \lambda(W_{i+1} - W_i), \tag{5.1}$$

where λ is a parameter that increments in steps sized to maintain the spacing Δs, terminating once the segment is fully sampled.

The segment's interpolated points are:

$$P_k = W_i + \left(\frac{k \cdot \Delta s + R}{L} \right) (W_{i+1} - W_i) \tag{5.2}$$

for $k = 1, 2, \ldots$ such that $k \cdot \Delta s \leq L$, and R is the remainder from the previous segment's interpolation, ensuring that the spacing between points remains consistent across the segment boundaries. Each segment-wise interpolation yields a set of points and a new remainder, which is carried forward to the subsequent segment, preserving the geometric resolution.

The linear reference generation transforms a continuous trajectory into a series of discrete, equally split waypoints, and a heading angle. The waypoints, together with the heading angle serve as a reference for the control systems to dictate the movement of the mobile unit along the predefined path. The algorithm uses linear interpolation to ensure a predictable outcome. Given two known points, the interpolated points will always lie directly between them in a straight line, allowing for a path that is both smooth and efficient.

In this project, several nodes are established to manage the operational location of the mobile unit. Upon arriving at a designated node, the mobile unit stops its movement for a predetermined duration to execute specific tasks at that station. As an example of a station task, the unit could pause at a station equipped with QR code identification technology until it is recognized, after which it will proceed to the next designated node. The system used at OrangePoint has four such stations which are predefined by entering the x and y coordinates of each station, along with a priority level that determines the sequence in which the stations are visited. After completing the sequence, the unit either resets and begins the cycle from the beginning or terminates its operation at the final station, depending on the user input in the configuration file.

The trajectory planner is horizon-based, which allows for planning over a predefined number of steps or time intervals into the future on the given path. This will be referred to as the reference horizon. This approach enables the trajectory planner to respond to future conditions and objectives, allowing for adjustments to the set of references as new information becomes available [34]. The length of the reference horizon is directly dependent on the control horizon in the motion control system, balancing the benefits of foresight against the need for timely decision-making.

6
Motion Control

To track the desired trajectory, a motion control system is developed. The system can be further divided into two main segments, a high-level controller and a low-level controller. The high-level controller is tasked with the core computations, aiming to minimize the deviation between the reference and the estimated position. The discrepancies from the intended path are then converted into actuation requests, such as steering angle and longitudinal velocity to reduce the deviation over time. The low-level controller then acts as an allocator, converting the requested actions into motor commands and facilitating communication between the APC and the remote mobile unit. A top-level overview of the motion control system, including the flow and interaction of signals is presented in Figure 6.1.

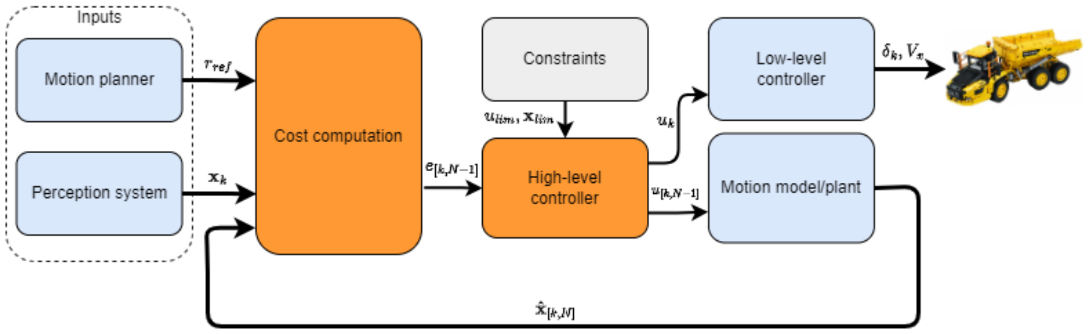


Figure 6.1: Simplified motion control overview.

The motion control architecture of Figure 6.1 shows multiple interconnected subsystems, together resulting in the actuation of the mobile unit. Subsequent sections will provide a more detailed description of each subsystem within this figure.

6.1 Motion Models

A motion model is derived to capture the dynamic behaviour of the mobile unit used in this project and how current actions affect the system's future states. The motion model is crucial for system validation because it is integrated into the simulator. Additionally, it can be used in more complex controllers, enabling the prediction of future states based on current states and actions.

Based on the design of the mobile units detailed in Chapter 3.2, two motion models are derived. One with a rigid body and another featuring an additional degree of freedom between the head and the trailer, known as an articulated body. Each model is a simplification of reality and some approximations have been made. For example, the Ackerman steering is considered parallel and the power distribution between the front and rear axis is neglected.

6.1.1 Rigid Motion Model

Within this project, a simplified car model [35] is used to derive the dynamics of the rigid mobile unit. Due to the wheel alignment and steering configuration, certain constraints are imposed on the car, limiting the rotation around its z-axis proportional to its wheelbase. Figure 6.2 shows the rigid motion model with its position in two-dimensional space and heading orientation.

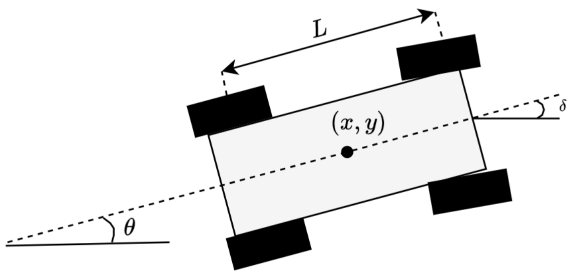


Figure 6.2: Simplified model of the rigid mobile unit.

From Figure 6.2, the discrete state vector can be represented by $\mathbf{x}_k = [x_k, y_k, v_k, \theta_k]^T$. The control inputs coupled to the states are denoted by $u_k = [a_k, \delta_k]$, where a_k is the longitudinal acceleration and δ_k is the steering angle deviation from its zero position. To further model the mobile units dynamics, a non-linear motion model in continuous time is derived in (6.1) with the mentioned states and control actions.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{v} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ a \\ \frac{v}{L}\tan(\delta) \end{bmatrix} \tag{6.1}$$

To find numerical solutions to the differential equations of (6.1) in discrete time, the system is discretized using forward Euler discretization:

$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + f(\hat{\mathbf{x}}_k, u_k)\Delta t. \tag{6.2}$$

The non-linear motion model is thereafter implemented into the simulator.

To further simplify the model for linear control system applications, the motion model is linearized. Restricted by its physical limitations, larger changes in the states are limited within a small time frame. Thus, a first-order Taylor expansion is used to improve the accuracy of the linearized model at smaller state deviations from the nominal state. This means that the system is linearized around a nominal state $(\bar{\mathbf{x}}, \bar{u})$ to mitigate deviations between the linear and non-linear model around a given operating point. The final linearized motion model becomes:

$$\hat{\mathbf{x}}_{k+1} = A\mathbf{x}_k + Bu_k + C. \tag{6.3}$$

When using the linear motion model in control algorithms, the operating point is continuously updated to ensure the accuracy of the motion model. Below are the A and B matrices, together with the correction matrix C presented at an arbitrary operating point $(\bar{v_k}, \bar{\theta_k}, \bar{\delta_k})$.

$$A = (I + A'\Delta t) = \begin{bmatrix} 1 & 0 & \cos(\bar{\theta_k})\Delta t & -\bar{v_k} \cdot \sin(\bar{\theta_k})\Delta t \\ 0 & 1 & \sin(\bar{\theta_k})\Delta t & \bar{v_k} \cdot \cos(\bar{\theta_k})\Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{\tan(\bar{\delta_k})}{L}\Delta t & 1 \end{bmatrix} \tag{6.4}$$

$$B = (B'\Delta t) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \Delta t & 0 \\ 0 & \frac{\bar{v_k}}{L \cdot \cos^2(\bar{\delta_k})}\Delta t \end{bmatrix} \tag{6.5}$$

$$C = \begin{bmatrix} \bar{v_k} \cdot \sin(\bar{\theta_k})\bar{\theta_k}\Delta t \\ -\bar{v_k} \cdot \cos(\bar{\theta_k})\bar{\theta_k}\Delta t \\ \frac{\bar{v_k} \cdot \bar{\delta_k}}{L \cdot \cos^2(\bar{\delta_k})}\Delta t \end{bmatrix} \tag{6.6}$$

The C-matrix of (6.6) can be described as a correction term that accounts for differences between the predicted and actual dynamics of the vehicle model. The correction term calculates the difference between the actual system dynamics and its linear approximation:

$$C = f(\bar{\mathbf{x}}, \bar{u}) - A'\bar{\mathbf{x}} - B'\bar{u}. \tag{6.7}$$

6.1.2 Articulated Motion Model

To account for the additional degree of freedom between the head and trailer of the mobile unit shown in Figure 3.3b, the motion model (6.3) is augmented with an additional state ψ as shown in Figure 6.3.

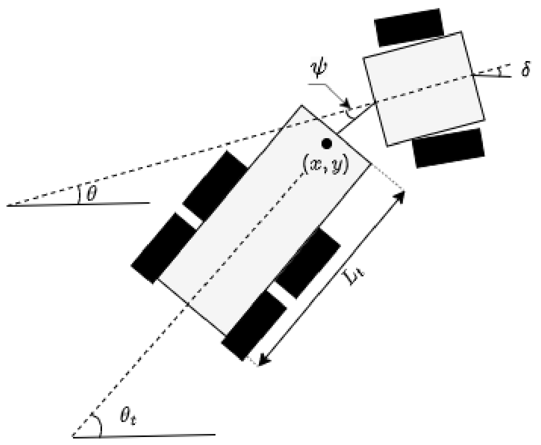


Figure 6.3: Simplified model of the articulated mobile unit.

Figure 6.3 shows the articulated mobile unit, highlighting the variables affecting the relative joint angle. Based on the model given in Figure 6.3 and [36], the relative joint angle, ψ, can be derived as the deviation between the heading of the trailer and the heading of the mobile unit. The rate of change of the relative joint angle is:

$$\dot{\psi} = \dot{\theta}_t - \dot{\theta} = \frac{v}{L_t} \cdot \sin(\theta - \theta_t) - \frac{v}{L} \cdot \tan(\delta). \tag{6.8}$$

As there is no wheelbase affecting the steering capabilities for the head of the mobile unit and assuming small joint angle deviations, (6.8) can be approximated as:

$$\dot{\psi} \approx \frac{v}{L_t} \sin(\psi - \delta). \tag{6.9}$$

The rate of change in the relative joint angle can then be added to the current angle and implemented in the augmented non-linear motion model as an additional state:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ v_{k+1} \\ \theta_{k+1} \\ \psi_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + \Delta t \cdot v_x \cos(\theta_k) \\ y_k + \Delta t \cdot v_x \sin(\theta_k) \\ v_k + \Delta t \cdot a \\ \theta_k + \Delta t \cdot \frac{v_k}{L} \tan(\delta_k) \\ \psi_k + \Delta t \cdot \frac{v_k}{l_t} \sin(\psi_k - \delta_k) \end{bmatrix} \tag{6.10}$$

The motion model is linearized as for the rigid motion model above.

6.2 High-level Controller

The primary objective of the high-level controller is to minimize the trajectory deviation from the setpoints given by the motion planner. Achieving this objective involves a recurrent process of identifying a sequence of viable control signals that ensure that the mobile unit adheres to the intended trajectory. The complexity of this type of system can vary considerably, where the constraints and cost minimization can be handled as two separate entities or incorporated into a controller that can handle both. This section presents the implementation of two different control strategies, a classical PID controller, and a model-predictive controller (MPC) with the intent of evaluating what different levels of complexity yield in terms of performance.

6.2.1 PID Control

To ensure a fully working system and to set a baseline for trajectory tracking, a simple Proportional-Integral-Derivative (PID) controller was implemented. To do so, the control objective was defined as a single input single output (SISO) system with the sole objective of tracking the current desired position and correcting the steering angle to minimize the deviation. The discrete expression for the PID controller is:

$$u_\delta[k] = K_p e[k] + K_i \Delta t \sum_{i=0}^{k} e[k] + K_d \frac{e[k] - e[k-1]}{\Delta t} \tag{6.11}$$

Here K_p, K_i and K_d are weights that were manually tuned to improve the performance during testing.

The control action, denoted as u_δ, for each iteration, k, consists of the accumulated error from the current $e[k]$ and previous $e[k-1]$ states, combined with the predefined weights. The accumulated sum yields a control action that affects the change in steering angle to mitigate the observed error. The error term, $e[k]$, represents the current deviation between the mobile unit's estimated centre point and the reference position. The calculated error at each sample is given by:

$$e[k] = \|\mathbf{x}_{ref,k} - \hat{\mathbf{x}}_k\|^2, \tag{6.12}$$

which represents the mobile units x and y coordinates as $\hat{\mathbf{x}}_k$ and the corresponding reference point as $\mathbf{x}_{ref,k}$. The error calculation serves as the foundation for determining the steering angle in the next iteration. To ensure that the resulting control action is within the mobile unit's feasible operational bounds the control action is saturated, and an anti-windup solution is incorporated.

Since the mobile unit operates at low speeds and efficiently reaches the desired longitudinal velocity within a reasonable time on all surfaces in the demonstration area, there is no necessity for a dedicated controller for the longitudinal velocity input. The velocity command is instead directly based on the reference from the trajectory planner.

6.2.2 Model Predictive Control

To further improve the motion control system, more advanced controllers were investigated, allowing the mobile unit to handle complex tasks and solve difficult manoeuvres in tight environments. This could be achieved in multiple ways but a method with forward-looking capabilities that can incorporate constraints and limitations into the problem formulation was desired. A full-state feedback design and the ability to model constraints could be more effective in navigating tight environments than previous controllers as the end goal is to achieve a smoothly controlled mobile unit [37]. For these reasons, an MPC approach was chosen.

MPC is a control strategy that explicitly accounts for future events to make current decisions. Unlike PID controllers, which react to present errors, MPC formulates an optimization problem that predicts future system behaviours over a given prediction horizon, solving for the optimal control inputs at each step [38]. This forward-looking capability allows this type of controller to manage constraints and multiple input, multiple output (MIMO) systems more effectively, which was desired in this project.

An MPC formulation can contain both linear and non-linear dynamics. Non-linear problems, while potentially more precise, are also more computationally demanding [39]. In contrast, linear problems, though approximations, can be kept convex, requiring less computational power to solve. Based on the available computational resources for this project and the fact that higher precision is deemed unnecessary for this application, a linear quadratic control problem is formulated with the linear motion models presented in Chapter 6.1. The motion model is chosen depending on what mobile unit is used. The final problem formulation and the objective function are further described below.

6.2.2.1 Cost and Constraints

To ensure that the system maintains the desired reference trajectories with smooth behaviour, an objective function is formulated that incorporates a set of costs designed to penalize undesired behaviours. Each cost component is treated as a soft constraint. This approach requires less computational resources than methods with hard constraints and inequalities, even if a harder constrained problem yields a smaller feasible set [40]. However, it presents a trade-off between accuracy and computational complexity. The problem formulation in this project aims to minimize the overall cost, requiring more tuning to achieve the desired results while a harder-constrained problem is less dependent on the tuning but instead is more computationally expensive. The formulation of each cost and constraint is further described below, inspired by [41,42].

Similar to the PID controller a state deviation cost is derived, in this case providing full state feedback where the deviation between the references states $\mathbf{x}_{ref} = [x_{ref}, y_{ref}, v_{ref}, \theta_{ref}]^T$ and the current state vector $\hat{\mathbf{x}}$ is found at each step k. Each state deviation is penalized with a cost-matrix Q, and computed over the entire prediction horizon, N. Additionally, a terminal cost is added to the final state deviation at the horizon N, together accumulating the total cost for reference path deviation over the entire horizon:

$$J_{\mathbf{x}} = \|\mathbf{x}_{ref,k} - \hat{\mathbf{x}}_k\|_Q^2, \quad k \in \mathbb{N}_{[0,N]} \tag{6.13}$$

$$J_{\tau} = \|\mathbf{x}_{ref,N} - \hat{\mathbf{x}}_N\|_{Qt}^2 \tag{6.14}$$

Additionally, to maintain the desired velocity given by the motion planner and minimize steering effort, an actuation cost is implemented (6.15), weighed with a matrix denoted R. To avoid oscillations and fast action changes, acceleration and steering jerk are also penalized as an additional cost (6.16).

$$J_{\mathbf{u}} = \|u_k\|_R^2, \quad k \in \mathbb{N}_{[0,N-1]} \tag{6.15}$$

$$J_{\mathbf{u}'} = \|u_{k+1} - u_k\|_{Rd}^2, \quad k \in \mathbb{N}_{[0,N-1]} \tag{6.16}$$

To define the feasible region of the solution and determine the bounds, a set of inequality constraints is defined. These constraints limit the feasible region, helping the solver to find an optimum within these bounds. Specifically, the control inputs u_k are constrained by minimum and maximum allowable values:

$$u_{\min} \leq u_k \leq u_{\max}, \tag{6.17}$$

ensuring that the vehicle's actuators operate within safe and efficient limits.

To constrain the feasible solution to be within the bounding box of the demonstration area, four additional inequalities were added as an upper and lower bound, denoted x_b and y_b:

$$\begin{aligned} x_{b,\min} \leq x_k \leq x_{b,\max} \\ y_{b,\min} \leq y_k \leq y_{b,\max}. \end{aligned} \tag{6.18}$$

These constraints limit the controller's horizon to be inside the area, thus avoiding a series of actions that could lead to collisions with the walls.

In addition to the inequalities, an equality constraint is defined to ensure that the solution strictly adheres to the modelled system behaviour, ensuring feasible physical solutions:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, u_k). \tag{6.19}$$

Together, these constraints ensure that the solution derived from minimizing the cost of the objective function adheres to the physical- and operational limits of the mobile units. Where the main intent is to estimate and achieve feasible actions that the mobile unit can perform.

6.2.2.2 Problem Formulation

With all terms in the objective function being quadratic and the constraints being linear, the final optimization problem, given in (6.20) and (6.21), is formulated as a quadratic minimization problem over the horizon N. With a quadratic problem, a convex solution can be guaranteed, meaning that any local minimum is also a global minimum, ensuring the optimal solution at each iteration.

$$\min_{u} \sum_{k=0}^{N-1} [J_{\mathbf{x}} + J_{\mathbf{u}} + J_{\mathbf{u}'}] + J_{\tau} \tag{6.20}$$

$$\begin{aligned} \text{s.t.} \quad \forall k &\in \mathbb{N}_{[0,N-1]} \\ u_{\min} \leq u_k &\leq u_{\max} \\ x_{b,\min} \leq x_k &\leq x_{b,\max} \\ y_{b,\min} \leq y_k &\leq y_{b,\max} \\ x_{k+1} &= f(x_k, u_k) \end{aligned} \tag{6.21}$$

The quadratic problem formulation above entails a predictable and low-cost solution in terms of computational complexity, as iterating through several local minima can be avoided if the problem is convex. With all terms in the objective function being quadratic and using only linear constraints, convexity can be guaranteed by ensuring that the objective function is positive and semi-definite.

There is a wide range of available solvers that can efficiently solve convex problems, in this project, an interior point solver called *ECOS* is chosen from the *CVXPY*-library [43]. The interior point method transforms the original problem into a sequence of approximate problems, which become progressively closer to the original problem. Rather than handling the constraints directly, this method uses barrier functions that make the cost of approaching the boundary of the feasible region tend towards infinity [44]. Thus, ensuring that the solution is within the feasible region without handling constraints in a way that would increase computational complexity.

6.3 Low-level Control

As previously stated, the actuation of the mobile unit is facilitated by the implementation of a low-level controller. The subsystem serves an intermediate role by processing the high-level motion requests into specific, executable motor commands on the mobile unit. The available motor commands may vary based on the motor configuration of the mobile unit. In this thesis, the mobile unit is controlled through steering and velocity commands.

To maintain centralized control of the entire system and because the internal processing unit of the mobile device is not directly accessible or adequate, the low-level controller is located on the APC. From this setup, motor commands are transmitted to the mobile unit via Bluetooth. This arrangement ensures reliable delivery of actuation commands to the mobile unit once a Bluetooth connection is successfully established, eliminating the need for physical access to its internal components. Each hub on a Lego Truck possesses a unique Bluetooth ID, therefore successful connection to the intended mobile unit can be established. This centralized approach also opens up the possibility of controlling multiple mobile units from a single APC in the future.

Discretized with zero-order hold (ZOH), the mobile units will hold the previous actuation commands between samples. The physical limitations within the mobile unit also resulted in additional latency between the requested and fully completed actuation. To mitigate a buffer build-up, resulting in increasing latency over time and race conditions, the low-level controller cannot send commands without the previous request being completed. To ensure this, a mutex lock is implemented to verify that the mobile unit has processed and completed the previously requested command before allowing another request to be sent, simultaneously taking care of the buffer by choosing the most recent actuation requests.

7
Results

The performance of the system and its subsystems was evaluated by a series of tests, both in simulation and on the physical hardware. This chapter presents the results, covering both a system overview and the test of each subsystem to ensure that it meets the requirements presented in Table 3.1. All tests were performed on an Intel i5-8350U CPU with 4 cores and a clock speed of 1.7 GHz, together with 16GB of RAM. The specifications are similar to the upgraded version of the APC presented in Chapter 3.2.

7.1 Perception

In this project, the perception system is designed to process and interpret data from a single sensor input, the camera, for use by the rest of the system. Thereby, the precision and latency of the system largely depend on the performance of the perception system. The perception subsystem is evaluated first separately and later together with the entire system. The main applications of the subsystem are further evaluated and presented below.

7.1.1 Dataset and Model Training Evaluation

To ensure consistent detection and tracking of the mobile unit, a dataset from the demonstration area was used and the training results are shown in Figure 7.1.

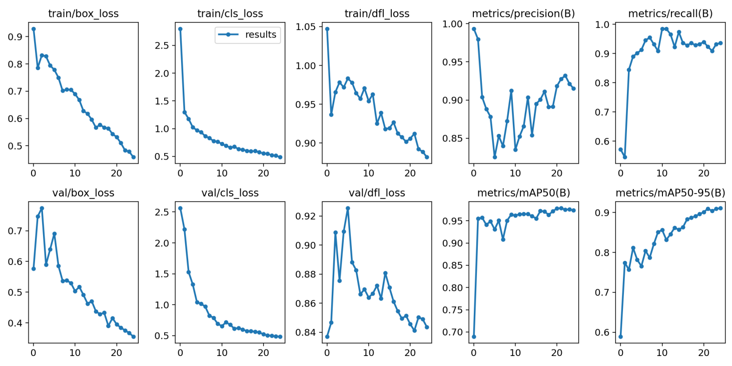


Figure 7.1: Performance metrics of model training over 25 training epochs.

To assess the model's learning and adaption to the new dataset some performance matrices have been extracted and displayed in Figure 7.1. By analyzing the *box_loss* and *cls_loss*, high accuracy can be concluded as the model improves over the training epochs, both in locating and classifying objects correctly. The third column in Figure 7.1 covers the distribution focal loss denoted *dfl_loss*, indicating an increased correlation between the estimation of the bounding box coordinates and the ground truth specified in the dataset.

The four plots on the right in Figure 7.1 highlight the final model's performance. These results display high values in both *Precision* and *Recall*, indicating that the model can distinguish well between desired objects and non-desired objects. Higher values for these metrics imply fewer false- positives and negatives. Additionally, based on the metrics *mAP50* and *mAP50-95*, the results suggest a rapid improvement in the model's ability to predict the bounding boxes for objects with at least 50 % overlap, as well as IOU thresholds ranging from 50 to 95 %. These results indicate that the model has achieved a high level of accuracy in consistent object detection and tracking. However, some fluctuations in the precision metric suggest that the model was more inconsistent in detecting true positives in the early stages of training.

7.1.2 State Estimation Accuracy

Estimating the mobile unit's state is the key component of the perception system and should be done with high accuracy to yield a stable and responsive system. Within this project, the main states estimated with the perception system were the Cartesian x and y coordinates of the mobile unit and the linear velocity. The remaining variables in the state vector, $\mathbf{X}$, were deemed more sufficient to estimate with the internal motion model. By giving the system a set of initial states, the states in the next iteration can be estimated by the current actions set by the motion controller, thus mitigating the need to estimate the pose with the perception system.

As the estimated position in the 2D space is critical to ensure that the mobile unit follows the desired trajectory as intended, the accuracy of the estimated x and y coordinates are evaluated in a small test. The test involved comparing the measured true position to the estimated one given by the perception system. Table 7.1 shows the average deviation between the true and estimated position in the 2D space.

Table 7.1: Standard deviation of position measurements

Position measurement	Deviation from true position
x	0.0366 m
y	0.0133 m
$\|x, y\|$	0.0389 m

The results presented in Table 7.1 show that the estimated position can deviate approximately 4 cm from the true position. The position can deviate in any direction. The deviation test was performed with a camera height of 2.28 m above the ground, facing directly downwards.

7.1.3 Process Acceleration

To evaluate the performance of each acceleration method presented in Chapter 4.4.4, a sample video similar to the application is used. The sample video was a unique set of frames, different from the frames in the dataset used for training the model. With this, the latency (including pre-processing, inference, and post-processing) of the system is measured. Thereafter, each framework is compared to find which acceleration method yields the lowest computational time without compromising accuracy to a large extent. The results from the performed tests are presented in Figure 7.2 and Table 7.2.

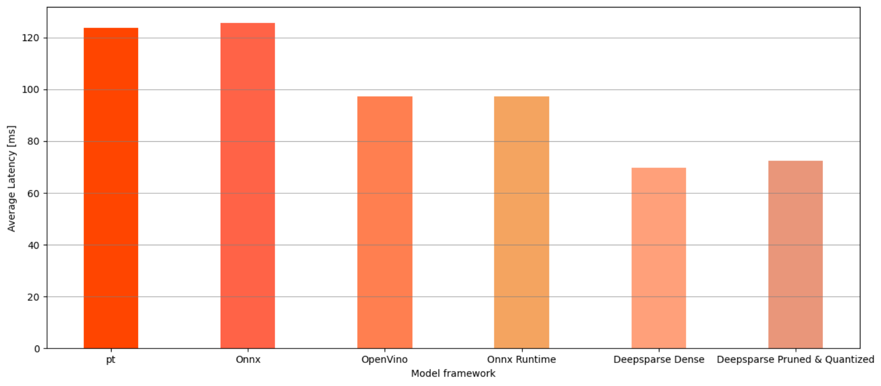


Figure 7.2: Average latency for different object detection frameworks during runtime.

Table 7.2: Average accuracy measurements of different acceleration methods.

Method	Model Accuracy
.pt	74%
ONNX	79%
OpenVino	81%
ONNX Runtime	69%
DeepSparse Dense	91%
DeepSparse Quantizied	54%

Table 7.2 presents different acceleration methods and their accuracy rating. The accuracy rating is measured based on the average confidence score of the object detection during the test, where the period and frames were the same for all methods. As shown in Figure 7.2 the DeepSparse framework provides the lowest latency on average, particularly with a dense model configuration. When analyzing the results in Table 7.2, the comparison between dense and quantized DeepSparse models reveals a significant insight into the trade-offs between speed and accuracy. Both DeepSparse models indicate a reliable level of performance across different configurations. However, the dense model stands out for its balance of speed and precision, as the loss of accuracy for the quantized model was significantly higher. Thereby, the dense model with the Deepsparse engine was used for the remaining tests.

7.2 Motion Control

In the motion control system, two controllers were implemented: a PID controller and an MPC controller, each with significantly different levels of complexity. With this evaluation, the main goal was to establish the complexity level required to achieve the desired results navigating the tight demonstration area with the given mobile units. To evaluate the controllers, the average deviation between the reference and the measured position of the mobile unit is computed at each point, as shown by (7.1) over the simulation time, T_{tot}:

$$e_{avg} = \frac{1}{T_{tot}} \sum_{k=0}^{T_{tot}} \sqrt{(x_{r,k} - \tilde{x}_k)^2 + (y_{r,k} - \tilde{y}_k)^2}. \tag{7.1}$$

Additionally, the maximum path deviation e_{max} and the standard deviation σ are computed. This can be analyzed by (7.2), which is used to evaluate the reliability and efficiency of the mobile unit navigation.

$$\sigma = \sqrt{\frac{\sum_{k=0}^{T_{tot}} (e_k - e_{avg})^2}{T_{tot}}} \tag{7.2}$$

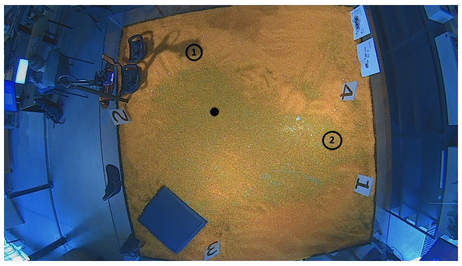
Other than evaluating the two motion controllers' ability to follow the desired reference states, the solution time of each controller, its ability to solve more complex problems, and reach all desired stations within a given tolerance is investigated.

During the performed tests both in simulation and on the hardware, the PID controller followed a set reference velocity while the setup for the MPC controller allowed for adaptive velocity planning. This gave the MPC controller the ability to regulate its speed while performing different manoeuvres, incorporating it into the optimization problem.

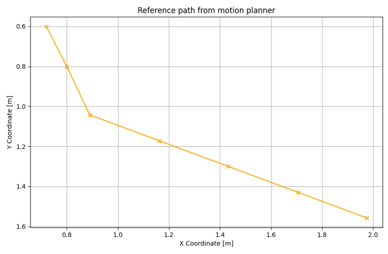
7.2.1 Test Scenarios

To evaluate both motion control strategies used within the project, a set of test scenarios where defined. Each controller is evaluated in all scenarios, both in simulation and on the hardware to verify performance and discrepancies between the simulation results and the hardware performance. Below are some of the tests described in more detail.

Figure 7.3 shows a simple test similar to a step response. However, due to the turning capabilities of the mobile units used within this project, the reference change is not a 90-degree turn. The test was done to evaluate the controllers settling time for a reasonable reference change.



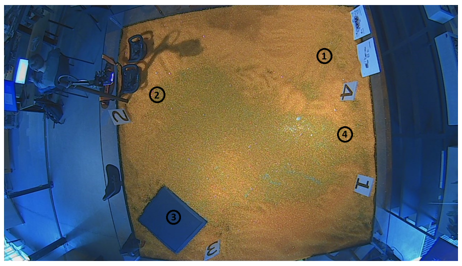
(a) Demonstration area with stations and a helping node (●).



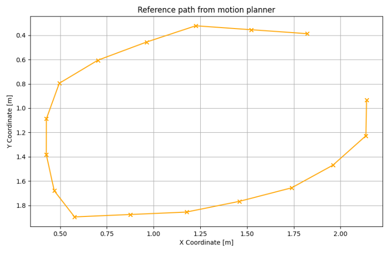
(b) Generated reference path from the motion planner.

Figure 7.3: Simple step response test.

The second test, presented in Figure 7.4, contains several stations, made to resemble a realistic run, constructed in a way similar to what the company wants to use for demonstration purposes in the future. This entails going to several stations shown in Figure 7.4a and completing different tasks. Figure 7.4b shows the generated reference states given by the motion planner during the run.



(a) Demonstration area with several stations.



(b) Generated reference path from the motion planner.

Figure 7.4: Full cycle test between stations.

With the given stations in each test, Figure 7.3b and 7.4b show the reference trajectory generated by the motion planner. With this approach, a simple and feasible path can be generated with more direct control over the exact path, giving the demonstration area more flexibility to get the system to behave as intended. One example of this is displayed in Figure 7.3 where a helping node is added between the two stations, indicating where the path should start its sharp turn.

7.2.2 Control Tuning

The tuning of the controllers was performed manually. The parameters for each controller were kept the same for both the PID and MPC during all hardware tests and simulations. This intent was to highlight the deviation between the simulator and hardware performance and also see how a statically tuned system would affect the results for different reference changes.

The PID's proportional gain (P) was set to an aggressive value to ensure quick correction to enable sharp turns. The integral action (I) included an anti-windup mechanism to reduce potential instability issues. The derivative component (D) was set to a low value to minimize oscillations and avoid excessive system changes. The parameters used for the PID controller during all physical tests and simulations are presented in Table 7.3.

Table 7.3: Tuning parameters for PID-controller.

K_p	1000
K_i	0.5
K_d	5

For the MPC controller, a prediction horizon of $N = 5$ was chosen to balance the travel between acceptable planning and avoiding shortcuts that could lead to missing stations. The constraints and cost functions were implemented as described in Chapter 6.2.2.

The tuning of the MPC controller aimed at penalizing the position in x and y the most to minimize the deviation between the true position and the reference position. The weights for the velocity v and θ were configured to allow the mobile unit to slow down and reverse if necessary, but without permitting backward travel along the remaining path. The input and state weights used for the MPC controller are presented in (7.3) and (7.4).

$$R = R_d = \begin{pmatrix} 0.01 & 0 \\ 0 & 0.01 \end{pmatrix} \tag{7.3}$$

$$Q = Q_t = \begin{pmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0.1 \end{pmatrix} \tag{7.4}$$

7.2.3 Simulation

Each controller was initially tested and tuned in simulation, both to minimize troubleshooting and to validate the motion controllers separately from the perception system. Within this project, the simulator used in [42] has been further developed, shown in Figure 7.5. However, due to time constraints, the simulator can only run the rigid motion model from Chapter 6.1 and an articulated version is yet to be implemented.

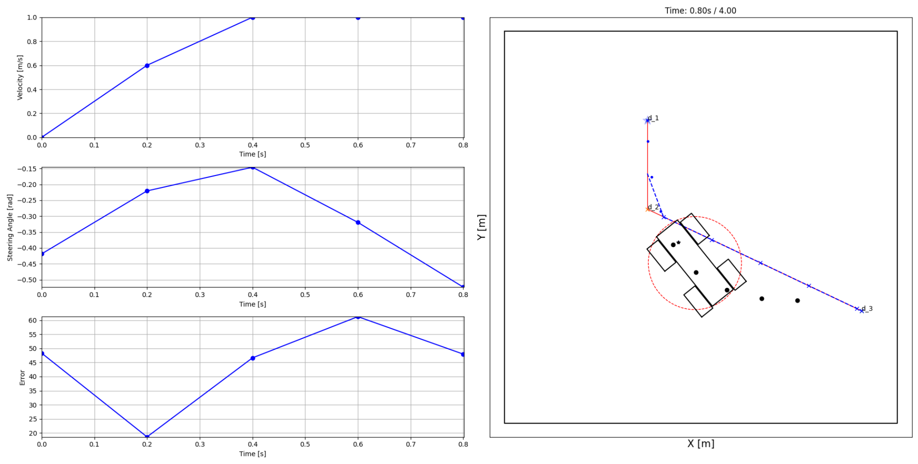
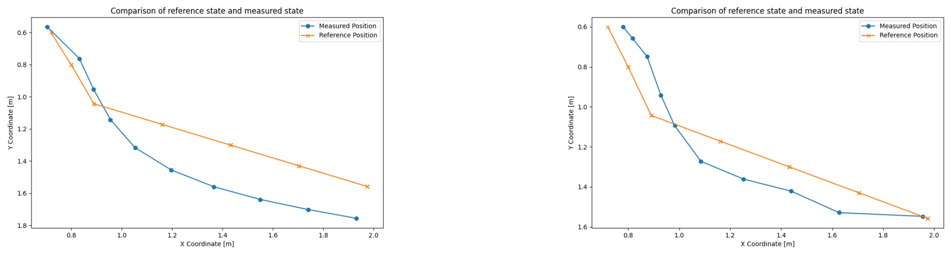


Figure 7.5: Simulator interface.

The simulation results for the two test scenarios presented in Chapter 7.2.1 are shown and evaluated below. Figure 7.6 and Table 7.4 relate to the step test, and Figure 7.7 and Table 7.5 show the results from the test involving multiple stations.



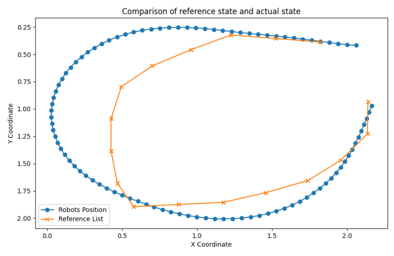
(a) Reference path deviation for PID controller.

(b) Reference path deviation for MPC controller.

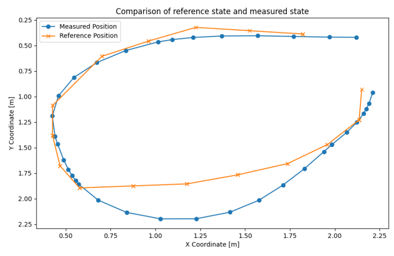
Figure 7.6: Controller comparison in simulation in a step test.

Table 7.4: Plot of reference path deviation error in simulation.

	PID	MPC
e_{avg}	0.5953	0.4547
e_{max}	1.3023	1.1027
σ	0.4427	0.3591



(a) Reference path deviation for PID controller.



(b) Reference path deviation for MPC controller.

Figure 7.7: Controller comparison in simulation in a full cycle test.

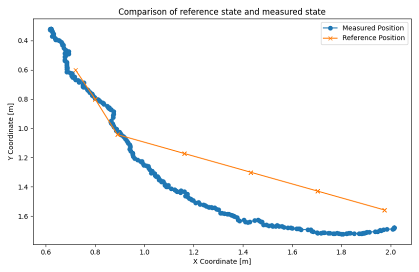
Table 7.5: Reference path deviation error in simulation.

	PID	MPC
e_{avg}	1.1719	0.1386
e_{max}	1.6682	0.3664
σ	0.4732	0.1094

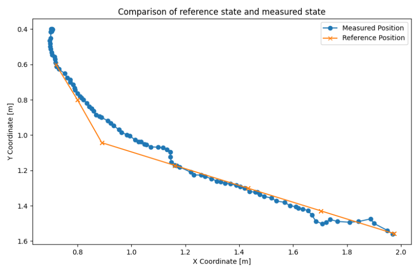
Based on the results presented in tables 7.4 and 7.5 the PID controller has a larger maximum and average deviation from the desired path compared to the MPC controller. The same holds for the standard deviation indicating a general trend of the MPC more consistently adhering to the desired reference path. Additionally, figures 7.6 and 7.7 show significant overshoots from the reference trajectories, especially at sharper reference changes. This is true for both controllers but to a larger extent for the PID controller.

7.2.4 Hardware

To validate the performance of the controllers on the target hardware, tests were conducted using the articulated truck. The plots and the reference path deviation from the step test are presented in Figure 7.8 and Table 7.6.



(a) Reference path deviation for PID controller during step test.



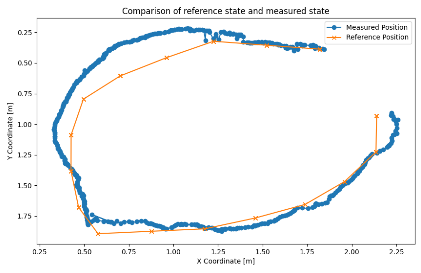
(b) Reference path deviation for MPC controller during step test.

Figure 7.8: Controller comparison on the target hardware for a step test.

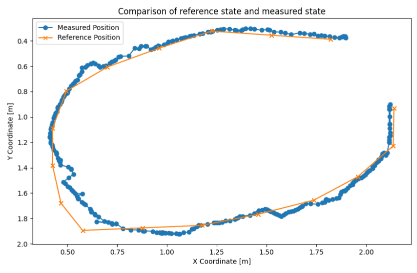
Table 7.6: Reference path deviation error on the target hardware for a step test.

	PID	MPC
e_{avg}	0.6500	0.1378
e_{max}	1.2358	0.4130
σ	0.2839	0.1181

The reference path deviation results for a full cycle test are presented in Figure 7.9 and Table 7.7.



(a) Reference path deviation for PID controller in a full cycle test.



(b) Reference path deviation for MPC controller in a full cycle test.

Figure 7.9: Controller comparison on the target hardware for a full cycle test.

Table 7.7: Reference path deviation error on the target hardware for a full cycle run.

	PID	MPC
e_{avg}	0.9974	0.1269
e_{max}	1.7785	0.5695
σ	0.5407	0.1183

For the step test, Table 7.6, demonstrates that the MPC-controller has a lower average error, e_{avg}, and a lower maximum error, e_{max}, compared the PID-controller. The standard deviation σ is larger for the PID controller, indicating worse consistency in its performance, compared to the MPC. In the full cycle test, with the data presented in Table 7.7 the MPC-controller again demonstrates a lower average error e_{avg} and a lower max error e_{max} compared the PID-controller. The standard deviation σ of the MPC-controller's error is 0.1183, which is lower than the PID controller's σ of 0.5407, implying better reliability and precision of the MPC controller.

One reason for implementing a controller with an increased complexity was also to enable the system to handle more constrained situations. This could entail more complex manoeuvres to sufficiently adhere to the given references. One scenario where this could be beneficial is when the mobile unit should go between a start and a goal state and when reaching the goal state having a heading direction towards the start state. An example of how the MPC controller solves this problem is shown in the video at the following link: ***Reversing MPC demo***.

7.3 Full System Evaluation

With each subsystem validated and tested, a holistic system assessment can be performed. All subsystems are coupled, meaning that their individual performance heavily depends on the other parts of the system. This chapter focuses on evaluating the performance and accuracy of the full system running on the target application. To give an overview of how the system works within the demonstration area, a video of the running full system in the sandbox at OrangePoint can be found at the following link: ***Full system demo***.

7.3.1 Hardware Assessment

A goal of the project was to demonstrate how advanced autonomous systems can be run on B&R's hardware units for sales purposes. At OrangePoint there is currently an APC 3100, specified in Chapter 3.2, which was initially used as the target hardware unit for the application. Therefore, tests were done on the hardware unit to determine if it was sufficient for the application or had to be upgraded with a more powerful CPU.

Based on the performance tests, the current APC was not sufficient as it contained an older CPU which was missing some critical features for the application. One of these was the *Advanced Vector Extensions 2* (AVX2), which was a necessity to run the Deepsparse pipeline [45] which provided the best results for accelerating CPU inference in this project. When running the perception system on the target hardware without hardware acceleration the resulting average inference time was 0.6 seconds with maximum peaks at approximately 2.1 seconds. An average inference time approximately 8 times slower than with hardware acceleration.

7.3.2 Solution Time

Additional measurements were captured during the test scenario with multiple stations on the physical hardware presented in Chapter 7.2.4. The system latency was measured and is shown in Table 7.8. The latency indicates the possible sampling time, Δt, for each controller and what update frequency is feasible at runtime for the entire system.

Table 7.8: Average update time, max update time and solver time in seconds for full cycle test with different controllers.

	PID	MPC
Δt	0.0923	0.2030
Δt_{max}	0.1858	1.1539
T_{solver}	-	0.0010

Table 7.8 displays the captured Δt for the full system with two different controllers. The solution time for the PID controller is considered negligible while the MPC's solver time is displayed as T_{solver}. All measurements are average values from multiple tests of the full-system runs. The presented Δt is the average time required for one iteration of the full system. Δt_{max} is the maximum measured time required for one iteration on the full system. The solution time for the MPC controller, T_{solver}, is included in the Δt and Δt_{max}.

7.3.3 Reliability and Accuracy Assessment

Any latency in the system can directly impact its performance, resulting in undesired behaviours and potential collisions. Such latencies can also disrupt the vehicle's current position, causing it to deviate from the desired path and miss desired stations. Observed potential sources of latency include a built-up buffer in the camera, high computational time, or communication latency. These latencies can negatively affect the overall system reliability and accuracy by introducing an offset to the true path. Additionally, unforeseen lighting conditions or unexpected objects in the demonstration area can create difficulties for the perception system in identifying the truck. This may result in tracking errors or false positives, as illustrated in Figure 4.5, despite the model being trained. Misidentified objects can lead to defective steering commands or even a terminated run due to safety features, thereby compromising system reliability and accuracy.

The initial state of the mobile unit is also an important aspect of the system's accuracy. The heading position of the mobile unit is estimated using a motion model and if the initial state is faultily inputted, it could result in a consistent heading offset. In terms of robustness, the MPC controller also displays a sensitivity to different initial conditions leading to different solutions to the given problem in similar runs. Additionally, the MPC controller sometimes fails to find a feasible solution, leading to a terminated run. This issue could arise from to strict boundary constraints applied, limiting the feasible region too much.

8
Discussion

The aim of the thesis work was to develop an autonomous system that could navigate and control a mobile unit within a demonstration area, running on a hardware unit from B&R. The following chapter discusses the project's outcomes and results related to the project's aim and requirements.

8.1 Perception Evaluation

Based on the test results, the perception system has sufficient accuracy within the demonstration area. With good lighting conditions and not many other similar objects within the sandbox the mobile units can be tracked with a high confidence score within the whole area. By training most layers in the model with a niched dataset containing only data from the demonstration area, the model is very good at identifying the mobile unit. However, in a different environment or if too much changes in the sandbox, the system could behave very differently. This is not verified but should be noted as a new dataset could be desirable if future changes yield significant altercations to the current area.

Hardware acceleration techniques, especially the Deepsparse engine showed good results, allowing the full system to be run in real-time, an aspect critical for the project. Despite this, inference is still the main bottleneck for the system. With only CPU, and no GPU available, pruning and quantization should, in theory, yield better results than showed by this project in terms of accuracy. The undesired results presented within this project could be down to settings for quantization and pruning the model or during the transfer learning and should be investigated further.

8.2 Motion Model Evaluation

Within this project, a mobile unit with a rigid body was utilized to validate the system's performance during development. However, the final product was requested for another mobile unit with an articulated body. The non-articulated motion model was developed because the articulated mobile unit was unavailable during development due to its use in company demonstrations. With the uncertainty of testing time with the intended mobile unit, the system was primarily modelled with a rigid body both in the simulator and for the MPC controller. However, efforts were made to also implement an augmented motion model. Unfortunately, the articulated motion model was never evaluated in the tests presented for this project, mainly due to time limitations.

Differences between the rigid- and articulated motion models have previously been discussed, where the latter was presented to yield an improved turning capability but could lead to increased complexity in more “tricky” situations, such as reversing. This claim was also verified when comparing the mobile units which both had similar turning capabilities in terms of steering angle but resulted in different turning radii. The resulting tests presented in Chapter 7.2 showed a hardware performance that generally provided better results than in simulation. One reason could be that a rigid motion model was used in the simulation while an articulated mobile unit was used for the hardware tests. This opens up the possibility that implementing an articulated motion model both in simulation and for the MPC could yield improved performance. However, this is still to be evaluated.

8.3 Motion Control Evaluation

By analyzing the test results for each control strategy, the pros and cons of each approach can be determined. The PID and MPC results are discussed below to determine which controller is the most suitable for the application in this project.

The PID is a lateral controller, meaning that it only aims to minimize the deviation from the desired path by correcting the mobile unit’s steering angle. This approach proved good results for simpler paths but was very tuning-dependent at different types of reference changes. Not much time was spent improving this approach, as it was mainly intended as a simpler benchmark for other controllers. However, gain-scheduling control could be implemented to make the PID approach better suited for a varying track. Additionally, a longitudinal velocity controller could be used to incorporate an improved velocity planner that could reduce the overshoots. But with multiple controllers, co-dependencies could arise, where the controllers could affect each other negatively, destabilizing or limiting each other.

To avoid co-dependencies and achieve full state feedback control, an MPC controller was evaluated and compared to the previous approach. Based on the results, the MPC controller showed a higher accuracy and more desired actions compared to the PID approach. By involving more states in a single optimization problem the MPC was able to solve more complex problems and reach all stations more efficiently, effectively taking its own decisions on how to achieve the desired results. However, with this implementation, the system became more unpredictable and sensitive to external conditions. For example, by altering the initial conditions slightly the controller could take different approaches to solve the same problem, which can be undesired for demonstration purposes. The MPC also requires more tuning time than the PID controller, containing several parameters and weights that in different combinations affect the system’s behaviour drastically. However, the resulting outcome is shown to be more reliable and efficient than the PID approach.

8.4 Latency and Hardware Performance

The current APC at OrangePoint struggles to run the full system. Without support for AVX2, the system's average inference time is measured to approximately 0.6 seconds; an inference time approximately three times slower than what is possible for the full system featuring the MPC controller on hardware similar to the upgraded version of the APC.

To maintain a smooth run of the mobile unit at the demonstration area, a system frequency of 3-4 Hz is recommended. This frequency ensures that the system can complete all operations within the necessary time frame, with time available should some operations demand additional processing time. This buffer allows for the system to "catch up" if latency occurs, without causing collision or deviating too much from the desired trajectory.

Operating the full system with a sampling time of 0.6 seconds is not suitable, as the manoeuvering of the mobile unit would be affected by such latencies. To ensure a smooth full system run, an upgrade to the more powerful APC is recommended. Since the MPC's solver time is minimal, its impact on the full system is negligible when considering the hardware capabilities.

9 Conclusions

In this conclusion the research questions defined in Chapter 1.4 will be revised, presenting a summary of the insights and results gained during the project. Additionally, potential areas of future improvements and research are outlined.

1. To what extent can camera-based state estimation of a mobile unit be used to determine different kinematic properties, including position, speed, and relative angle of joints?

The primary focus of state estimation for the mobile unit was its position, linked to the reliability of the trained perception model. The properties, including the velocity and heading angle of the mobile unit, were computed to match the reliability of the position estimation. This was calculated using the motion models presented in Chapter 6.1. With the coordinates x and y updated at each Δt, all kinematic states were reliably determined based on the mobile unit's centre point, regardless of its orientation or location within the demonstration area. The implementation of the relative angle of joints has not yet been deployed, as the project's development focused on a rigid motion model.

2. What type of control strategy will ensure sufficient motion planning and control of the autonomous mobile unit within the predefined area to ensure safe and efficient navigation without operator input?

Based on the controllers compared within this project, the MPC approach was deemed most suitable for the application. This was based on its accuracy combined with its ability to handle different situations. Despite requiring more computational resources than the PID controller, integrating multiple conditions and constraints into a single problem formulation allows for a more comprehensive solution that considers all relevant factors. Although the controller is not perfect, the results show good performance with room for future improvements.

3. What are the key factors affecting the reliability and accuracy of the autonomous system controlling the mobile unit, and how can these factors be managed?

From the system testing, several factors were identified and established as critical components for the reliability and accuracy of the system. The key feature affecting system reliability was the latency of the system, which was heavily dependent on the hardware components. This could be solved in multiple ways, by upgrading the hardware, or by employing hardware acceleration tools, which in this project, proved a significant decrease in latency.

The system's accuracy was primarily influenced by two factors. The achievable turning radii of the mobile units, and the tuning of the controllers. To mitigate the problem of a limited turning radius within the small demo area, a controller capable of both forward and reverse motion was developed.

The project resulted in a complete autonomous system that enabled a single mobile unit to move between stations within the demonstration area. However, there are several things that should be further developed in the future to achieve improved performance and increase reliability within all subsystems.

9.1 Future Improvements and Development

Considering the perception system being computationally demanding, more efforts should be directed towards hardware acceleration. Further investigation of pruning and quantization could yield a lighter model that can reduce the computational complexity. Hopefully, this would also reduce the latency of the system without compromising the accuracy. Additionally, the perception system could be augmented with more features such as static and dynamic obstacle detection. This could allow the system to interpret the environment more effectively and detect undesired parts of the demonstration area that would lead to collisions. This feature could be added by simply augmenting the dataset to the current model.

Another continuation of the project could be improving the MPC controller. By augmenting the motion model and implementing a relative joint angle, as defined in Chapter 6.1.2, the control of the vehicle could be improved further. This could improve performance while executing reversing manoeuvres and gain more precision. To further improve the planned trajectory, the objective function could also be extended, penalizing horizons outside the boundary of the area or within obstacles. This approach could be similar to what is presented in [42]. This could reduce the number of hard constraints, thus potentially reducing the solution time at each iteration while still achieving a collision-free trajectory.

Finally, to make the system more suited for the demonstration area, the user console could be updated. A user-friendly HMI interface could be implemented to define desired stations, constraints, and different driving modes for the mobile unit.

Bibliography

[1] Vijay Kumar and Sarah Tang. Autonomous flight. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:29-52, 2018.

[2] B&R Automation. OrangePoint B&R Innovative Machine Building Lab. `https://www.br-automation.com/sv/academy/orangepoint-br-innovative-machine-building-lab/`. [Accessed: 20-03-2024].

[3] National pilot committee for digital ethics. Ethical issues regarding "autonomous vehicles", 2021. Accessed: [01-06-2024].

[4] Yali Amit, Pedro Felzenszwalb, and Ross Girshick. *Object Detection*, pages 875–883. Springer International Publishing, Cham, 2021.

[5] Naif Alsharabi. Real-time object detection overview: Advancements, challenges, and applications. *Journal of Amran university*, 3:12, 11 2023.

[6] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[7] Prajkta P. Khaire, Ramesh D. Shelke, Dilendra Hiran, and Mahendra Patil. *Comparative Study of a Computer Vision Technique for Locating Instances of Objects in Images Using YOLO Versions: A Review*, pages 349–359. Springer Nature Singapore, 2023.

[8] T. Diwan, G. Anirudh, and J.V. Tembhurne. Object detection using yolo: challenges, architectural successors, datasets and applications. *Multimedia Tools and Applications*, 82(6):9243–9275, 2023.

[9] Zhen Dong. *Hardware-Aware Efficient Deep Learning*. PhD thesis, EECS Department, University of California, Berkeley, Oct 2022.

[10] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021.

[11] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[12] Christodoulos A. Floudas. *Deterministic Global Optimization*. Springer, 2000.

[13] B&R Industrial Automation. Automation pc 3100 mobile. `https://www.br-automation.com/en/products/industrial-pcs/automation-pc-3100-mobile/`, 2024. [Accessed: 26-04-2024].

[14] Ark Vision Systems. Arkcam basic+ mini. [Accessed: 18-04-2024].

[15] Tianlong Lei, Jixin Wang, and Zongwei Yao. Modelling and stability analysis of articulated vehicles. *Applied Sciences*, 11(8), 2021.

[16] Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, 2019.

[17] Jeong-ah Kim, Ju-Yeong Sung, and Se-ho Park. Comparison of faster-rcnn, yolo, and ssd for real-time vehicle type recognition. In *2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)*, pages 1–4, 2020.

[18] Juan R. Terven and Diana M. Cordova-Esparza. A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. *Machine Learning and Knowledge Extraction*, Feb 2024. arXiv:2304.00501v7 [cs.CV].

[19] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. Yolov9: Learning what you want to learn using programmable gradient information, 2024.

[20] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box, 2022.

[21] Sedat Dogan, Mahir Temiz, and Sıtkı Külür. Real time speed estimation of moving vehicles from side view images from an uncalibrated video camera. *Sensors (Basel, Switzerland)*, 10:4805–24, 05 2010.

[22] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023.

[23] Muhammad Awais, Muzammal Naseer, Salman Khan, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, and Fahad Shahbaz Khan. Foundational models defining a new era in vision: A survey and outlook, 2023.

[24] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. Grounded sam: Assembling open-world models for diverse visual tasks, 2024.

[25] Xin Xiao and Xinlong Feng. Multi-object pedestrian tracking using improved yolov8 and oc-sort. *Sensors*, 23(20), 2023.

[26] Ruicheng Feng, Jinjin Gu, Yu Qiao, and Chao Dong. Suppressing model overfitting for image super-resolution networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[27] ONNX Runtime Team. Onnx runtime architecture. ONNX Runtime Documentation, 2023. [Accessed: 2024-04-24].

[28] Nico Galoppo Mingyu Kim, Vladimir Paramuzov. Techniques for faster ai inference throughput with openvino on intel gpus. *OpenVINO Blog*, April 2023. [Accessed: 2024-04-24].

[29] Neural Magic. Deploy on cpus. https://docs.neuralmagic.com/index/deploy-workflow, 2024. [Accessed: 07-03-2024].

[30] Jonas Berlin, Georg Hess, Anton Karlsson, William Ljungbergh, Ze Zhang, Knut Åkesson, and Per-Lage Götvall. Trajectory generation for mobile robots in a dynamic environment using nonlinear model predictive control. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 942–947, 2021.

[31] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, 2016.

[32] David González, Joshué Pérez, Vicente Milanés, and Fawzi Nashashibi. A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 17(4):1135–1145, 2016.

[33] Xianhua Li, Yuping Gu, Liang Wu, Qing Sun, and Tao Song. Time and energy optimal trajectory planning of wheeled mobile dual-arm robot based on tip-over stability constraint. *Applied Sciences*, 13(6), 2023.

[34] Alexander Dötlinger, Jean-François Stumper, and Ralph Kennel. Receding horizon based trajectory planning and two-degree-of-freedom tracking control for fast sampling constrained systems. In *2013 IEEE International Symposium on Sensorless Control for Electrical Drives and Predictive Control of Electrical Drives and Power Electronics (SLED/PRECEDE)*, pages 1–6, 2013.

[35] University of Illinois at Urbana-Champaign. Planning Algorithms - Node 658. `https://msl.cs.uiuc.edu/planning/node658.html`, 2024. [Accessed: 02-04-2024].

[36] Amro Elhassan. Autonomous driving system for reversing an articulated vehicle. Master's thesis, The Royal Institute of Technology, Stockholm, 2015.

[37] Tiago P. Nascimento, Carlos E. T. Dórea, and Luiz Marcos G. Gonçalves. Nonholonomic mobile robots' trajectory tracking model predictive control: a survey. *Robotica*, 36(5):676–696, 2018.

[38] Alejandro Piñón, Antonio Favela-Contreras, Francisco Beltran-Carbajal, Camilo Lozoya, and Graciano Dieck-Assad. Novel strategy of adaptive predictive control based on a mimo-arx model. *Actuators*, 11(1), 2022.

[39] F. Allgöwer, Rolf Findeisen, and Zoltan Nagy. Nonlinear model predictive control: From theory to application. *J. Chin. Inst. Chem. Engrs*, 35:299–315, 05 2004.

[40] Anders Forsgren, Philip Gill, and Margaret Wright. Interior methods for nonlinear optimization. *Society for Industrial and Applied Mathematics*, 44:525–597, 12 2002.

[41] Atsushi Sakai, Daniel Ingram, Joseph Dinius, Karan Chawla, Antonin Raffin, and Alexis Paques. Pythonrobotics: a python code collection of robotics algorithms. *CoRR*, abs/1808.10703, 2018.

[42] Filip Bertilsson, Martin Gordon, Johan Hansson, Daniel Möller, Daniel Söderberg, Ze Zhang, and Knut Åkesson. Centralized versus distributed nonlinear model predictive control for online robot fleet trajectory planning. In *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, pages 701–706, 2022.

[43] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[44] Florian A. Potra and Stephen J. Wright. Interior-point methods. *Journal of Computational and Applied Mathematics*, 124(1):281–302, 2000. Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations.

[45] Inc. Neural Magic. Installing deepsparse. `https://docs.neuralmagic.com/get-started/install/deepsparse/`, 2024. Accessed: 2024-06-07.