Linköping University | Department of Electrical Engineering
Master's thesis, 30 ECTS | Electrical Engineering
2021 | LiTH-ISY-EX–21/5389--SE

Trajectory Tracking for Automated Guided Vehicle

Trajektoriaföljning för en autonom truck

Johan Gustafsson
Anton Holgersson

Supervisor: Daniel Arnström, ISY, Linköping University
Examiner: Martin Enqvist, ISY, Linköping University

External supervisors: Håkan Therén, Toyota Material Handling
Andreas Westerlund, Toyota Material Handling

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Abstract

The purpose of this thesis is to investigate different control strategies on a differential drive vehicle. The vehicle should be able to drive in turns at high speed and slowly when it should park next to a charger. In both these cases, good precision in both orientation and distance to the path is important. A PID and an LQ controller have been implemented for this purpose.

The two controllers were first implemented in a simulation environment. After implementing the controllers on the system itself, tests to evaluate the controllers were made to imitate real-life situations. This includes tests regarding driving with different speeds in different turns, tests with load distributions, and tests with stopping accuracy. The existing controller on the system was also tested and compared to the new controllers.

After evaluating the controllers, it was stated that the existing controller was the most robust. It was not affected much by the load distribution compared to the new controllers. However, the LQ controller was slightly better in most cases, even though it was highly affected by the load distribution. The PID controller performed best regarding stopping accuracy but was the least robust controller by the three. Since the existing controller has a similar performance as the LQ controller but is more robust, the existing controller was chosen as the best one.

Acknowledgments

First of all, we would like to thank Toyota Material Handling for the opportunity of this exciting and meaningful thesis. We are grateful for the possibility, despite the COVID-19 pandemic, that we were allowed to evaluate our findings on a real system with practical elements. Thanks to the team on Toyota, in which we have become involved, and special thanks to our supervisors Håkan Therén and Andreas Westerlund for guidance and assistance throughout this work.

Finally, thanks to our supervisor Daniel Arnström and examiner Martin Enqvist at Linköping University for quick responses, feedback, and ideas to keep the work going.

Linköping, May 2021
Johan Gustafsson and Anton Holgersson

Contents

List of Figures vii

List of Tables ix

Nomenclature x

1 Introduction 1
- 1.1 Background . . . 2
- 1.2 Objective . . . 2
- 1.3 Research Questions . . . 2
- 1.4 Limitations . . . 3
- 1.5 Related Work . . . 3
- 1.6 Thesis Outline . . . 3

2 System Overview 4
- 2.1 System . . . 4
- 2.2 Kinematic Model . . . 6

3 Trajectory Tracking 8
- 3.1 Trajectory . . . 8
  - 3.1.1 Orientation . . . 8
  - 3.1.2 Tangential Velocity . . . 9
  - 3.1.3 Angular Velocity . . . 10
  - 3.1.4 Interpolation . . . 10
  - 3.1.5 Reference Selection . . . 11
- 3.2 Error Model . . . 11
- 3.3 Positioning . . . 13
- 3.4 PID Controller . . . 13
- 3.5 LQ Controller . . . 17
  - 3.5.1 Tracking and Gain Selection . . . 17
  - 3.5.2 Discrete Algebraic Riccati Equation . . . 18
  - 3.5.3 Time Delays . . . 20

4 Method 21
- 4.1 Simulation . . . 21
- 4.2 Implementation and Testing of Controllers . . . 23
- 4.3 Comparisons Between Controllers . . . 24
  - 4.3.1 Trajectory Tracking . . . 24
  - 4.3.2 Load Distribution . . . 24
  - 4.3.3 Stopping Accuracy . . . 26

5 Evaluation 27
- 5.1 Simulation . . . 27

5.2 PID Controller 30
5.3 LQ Controller 33
5.4 Comparisons 35
5.4.1 General Controller Comparison 35
5.4.2 Load Distribution 35
5.4.3 Stopping Accuracy 41
5.5 Method Evaluation 46

6 Conclusion **47**
6.1 Conclusion 47
6.2 Future Work 48
6.3 Sustainability 48

Bibliography **49**

List of Figures

1.1 A differential drive AGV for baggage logistics. 1

2.1 The construction of the AGV. 4
2.2 A block diagram of the main nodes in the system. 5
2.3 Global and local coordinate system. 6
2.4 The velocity at the contact point between a wheel and the ground. 7

3.1 Calculation of reference orientation. 9
3.2 Tracking errors in relation to current and reference pose. 11
3.3 Feedback and feedforward control. 14
3.4 Block diagram of the system. 15
3.5 The system controlled by two PID controllers. 16
3.6 PID controllers with feedforward. 17
3.7 LQ controller with feedforward. 18

4.1 Simulink scheme used for simulations. 22
4.2 Relationship between classes in C++. 23
4.3 Weight placement for load distribution tests. 25
4.4 Driving schedule for load distribution comparison. 25
4.5 Stopping points used in the comparison of stopping accuracy. 26

5.1 The trajectory used in simulations. 28
5.2 PID control signals in simulation without delay. 28
5.3 Simulation results for the PID controller used without delay. 29
5.4 Simulation results for the LQ controller used without delay. 29
5.5 Simulation results for the LQ controller used with delay. 30
5.6 Trajectory tracking results in the xy-plane for the PID controller. 30
5.7 The orientation data before and after lowpass filtering in the PID controller. 31
5.8 Reference and desired angular velocities when using the PID controller. 31
5.9 The reference velocity and the desired velocities for the left and right wheel calculated by the PID controller 31
5.10 Predicted future position. 32
5.11 The data used in the PID controller. 32
5.12 Calculation time for the PID controller. 33
5.13 Trajectory tracking results in the xy-plane for the LQ controller. 33
5.14 Reference and desired angular velocity using the LQ controller. 34
5.15 The reference velocity and the desired velocities for the left and right wheel calculated by the LQ controller 34
5.16 The time to calculate control signals for one iteration using the LQ controller. . . . 34
5.17 The lateral distance between the AGV and the trajectory for different load distributions. 36
5.18 The lateral distance between the AGV and the trajectory for weight placed in the back-left corner. 37

5.19 Lateral distance probability for different weight placements. 39
5.20 The orientation error for different load distributions. 40
5.21 The orientation error when weight is placed in the back-left corner. 41
5.22 The stopping accuracy results with its total Euclidean distance. 42
5.23 The orientation error for stopping accuracy tests. 43

List of Tables

4.1 A description of the stop scenarios used in stopping accuracy tests. 26

5.1 Parameter values for the PID and the LQ controller used in simulations. 27
5.2 Total Euclidean distance during driving for different weight placements. 38
5.3 Total absolute orientation error during driving for different weight placements. . . 38
5.4 Average Euclidean distance for each stopping point. 41
5.5 Average orientation error for each stopping point. 43

Nomenclature

Acronyms

Acronym	Meaning
AGV	Automated Guided Vehicle
DARE	Discrete Algebraic Riccati Equation
FMS	Fleet Manager System
IIR	Infinite Impulse Response
LIDAR	Light Detection And Ranging
LQ	Linear Quadratic
MIMO	Multiple Input Multiple Output
PID	Proportional Integral Derivative
SDA	Structure-preserving Doubling Algorithm
SIMO	Single Input Multiple Output
SISO	Single Input Single Output
TMH	Toyota Material Handling
TS	Toyota Smartness
Wheel track	Distance between two wheels on the same axis

Quantities

Symbol	Description
x	X position
y	Y position
θ	Orientation
v	Tangential velocity
ω	Angular velocity
$[x, y, \theta]$	Pose
$[x, y, v]$	Part of a trajectory received by the controller
$[x, y, \theta, v, \omega]$	The complete trajectory
acc_{max}	Maximum acceleration
dec_{max}	Maximum deceleration
$\dot{\phi}$	Angular velocity on the wheel

1 Introduction

The core of this thesis is to evaluate control strategies regarding trajectory tracking for an Automated Guided Vehicle (AGV). Today, there is an *existing* controller that is running on the AGV. New controllers will be implemented, and the performance of these controllers will be evaluated in different driving scenarios. Furthermore, the new controllers will be compared with the existing controller. A differential drive AGV with its baggage logistics top module is illustrated in Figure 1.1. This chapter will give a background and introduction to the work.

Figure 1.1: A differential drive AGV supplemented with a conveyor belt used for baggage logistics.

1.1 Background

The automation of industries and warehouses has never been as high as it is today. Engineers work hard to create customized solutions suitable for different kinds of applications. The reason for this high demand is its many benefits. Automating a warehouse can increase the output since a robot has the capacity to work night and day, without a break, every day of the week. This transition will also reduce labor costs and enable workers to focus on more important tasks while the automated robot can handle repetitive work. Furthermore, the safety of a workplace can increase when human errors are eliminated. The mistakes made by humans due to, for example, emotional stress or hunger can be devastating, and operating a heavy forklift in such conditions can, in the worst case, lead to a tragedy [1]. Additionally, the efficiency and even the accuracy can increase, and space in facilities can be saved.

Toyota Material Handling (TMH) has been working with driverless products for many years. In their line-up, there are automated warehouse trucks, tow trains, and shuttle solutions. A recent development is a differential drive vehicle suited for various tasks, including baggage logistics solutions at airports and relocation of heavy pallets. The goal for each baggage truck is to handle individual bags seamlessly. Many trucks form a fleet and work together to handle all the baggage operations at an airport. To ensure efficiency and safety, it is important that the AGV can follow a trajectory with precision.

To work in different environments with different tasks, the truck must be adaptable concerning the physical shape and extensions of the AGV and the software running on it. Hence, the controller needs to be versatile.

Today, a controller for this AGV is purchased from an external company, and its structure is unknown. An in-house solution of the controller might benefit the adaptability of a variety of applications. It might also give TMH the freedom to monitor more parts of the software while debugging and have better control over the system.

1.2 Objective

The thesis aims to investigate how new controllers can be implemented and communicate with the current system on an AGV. This includes an investigation of the performance of various control strategies for differential drive vehicles.

The high-level goal of the thesis is to improve the trajectory following for the AGV compared to the existing controller. A sub-goal is to create an in-house controller that has the same performance as the existing one.

1.3 Research Questions

From the objectives in Section 1.2, three problem statements are formulated:

1. Which control strategies can be used in a trajectory following differential drive AGV?
2. Among the possible control strategies, which is the best one regarding tracking error, robustness, and complexity?
3. What are the benefits and drawbacks of developing an in-house controller for an AGV?

1.4 Limitations

There are limitations, delimitations, and assumptions to the thesis. The delimitations are that only differential drive vehicles are treated, error handling such as bulldozing does not need to be concerned, and there is no wheel slip. A limitation is that the position and orientation of the AGV given by the positioning system will not be 100% accurate. Assumptions are that the AGV will drive on a flat floor, its weight is fixed, and the trajectory, global position, and orientation are known.

1.5 Related Work

Differential drive vehicles can be modeled with forward as well as inverse kinematics. A four-wheeled robot, much like the AGV with two drive wheels and two caster wheels can be modeled with two different coordinate systems. The derived model and a reference trajectory can construct an error model for the controller to use [2]. Further, [3, 4] describe how to restructure the error model and linearize it around an operation point.

For accurate trajectory tracking, the trajectory must be well constructed. To increase the number of reference points, either linear interpolation [5] or cubic spline interpolation [6] can be used.

Fundamental control theory is used to motivate the implementation of controllers on the system. This includes information about Proportional–Integral–Derivative (PID) and Linear Quadratic (LQ) controllers and how these can be implemented as well as multivariable systems, discrete-time theory, and non-linear systems [7, 8, 9]. This information will be used in combination with the literature containing information about various control strategies.

LQ controllers are more advanced than PID controllers, partly because they are model based. [10] gives more information and a deeper understanding than the previously mentioned literature about LQ controllers.

1.6 Thesis Outline

In Chapter 2, a brief introduction to the system is given. It covers the hardware as well as the communication between the nodes in the system. Further, a kinematic model of the AGV is presented. Information and construction of the trajectory for the AGV to track are presented in Chapter 3. Based on related work, an error model and two control strategies are proposed. The PID controller represents a simple approach, while the LQ controller is more advanced. It is also described how these controllers can be implemented on the AGV. Chapter 4 treats the workflow during the thesis. It is described how simulations, implementation and testing, and comparisons were carried out.

The results for the controllers are presented in Chapter 5. This chapter includes both our solutions and the existing solution, comparisons between them, and discussions. The chapter also contains a discussion regarding the method used in the thesis. Finally, the result and discussion from the previous chapter are concluded in Chapter 6. The initial research questions are answered, and proposals for future work are also given.

2 System Overview

2.1 System

The AGV used in the thesis is a differential drive robot with two separately driven wheels located at each side of the robot's body, shown in Figure 2.1. They are placed on an axis straight through the robot's center point. Further, there are two undriven caster wheels attached in the front and back of the robot, which enhance the balance and enable easy turning. The LED lights in each corner of the AGV indicate if the robot is turning and if errors have occurred.

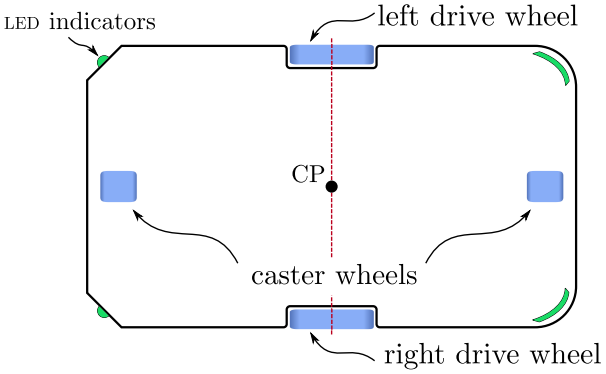


Figure 2.1: Bird's eye view sketch of the AGV, where its center point is denoted *CP*.

Several nodes are running on the industrial-grade computer mounted on the AGV. Toyota Smartness (TS) is the main node in the system. It receives and sends information to other nodes. Hence, TS enables communication between them. The cycle time of the system is 20 milliseconds. The major nodes and the relation between them are shown in Figure 2.2.

A Fleet Manager System (FMS) is running on an external computer. It can communicate with several AGVs, and one of its tasks is to send trajectories to the AGVs. It can also receive log data from the nodes. The transfer of data is performed via TS.

The driven wheels are each powered by a permanent magnet AC motor. Two motor controllers, one for each wheel, of type PID are used to control the wheels' speeds. The controllers take target velocity for the given wheel as the reference and calculate a control signal to the motor to obtain the reference velocity on the wheels. The rise time for this system is dependent on the weight of the AGV. The cycle times in the communication between the nodes sum up to a delay in the system. The size of this delay must be known by the new controllers so that a prediction of the future state of the AGV can be made, and thereby appropriate control signals can be calculated.

The existing control system is of unknown type. It takes a trajectory and current pose (x- and y-coordinate, and orientation θ) as input sent from TS. Using this, target velocities for the left and right wheel are calculated, sent to TS. These target velocities are then sent to the speed controllers. There are requirements for a controller used on the AGV. Its CP is not allowed to deviate more than 0.02 m from the trajectory and differ more than 4° in its orientation compared to the trajectory.

There is also a positioning system that runs on the AGV. Using a Light Detection And Ranging (LIDAR), it calculates the current position and orientation of the AGV. This information is sent to TS.

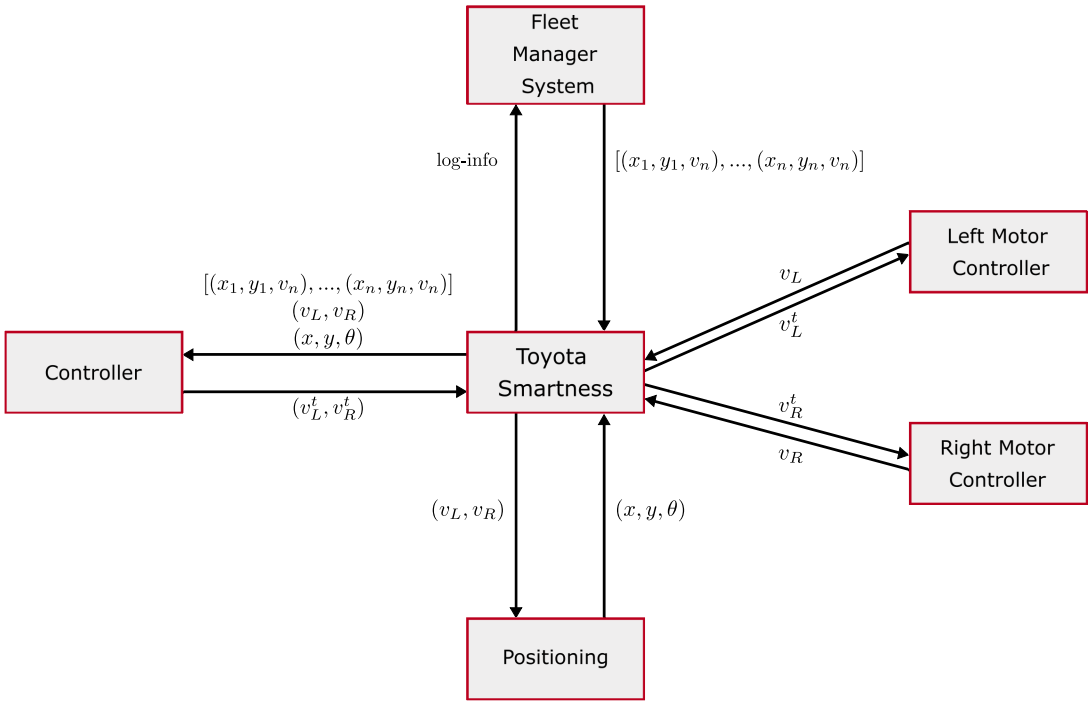


Figure 2.2: Velocity of the left and the right wheels are denoted v_L and v_R, their target velocities are denoted v_L^t and v_R^t, the current pose is described by (x, y, θ), and the vector $[(x_1, y_1, v_1), ..., (x_n, y_n, v_n)]$ represents the parts of the trajectory that includes reference position and velocity.

2.2 Kinematic Model

To control the AGV, a model of it needs to be derived. First, two different coordinate systems are defined to describe the AGV's position in its operating environment, as shown in Figure 2.3. The global coordinate system g with axis $[x, y]$, is fixed to the environment. The reference coordinates of the AGV are expressed in this system. The second coordinate system is a local one, denoted l with axis $[x', y']$. The local system is fixed to the AGV, where the x'-axis is pointing in the same direction as the robot's orientation, and its origin is located between the wheels on the wheel axis and is called the center point CP.

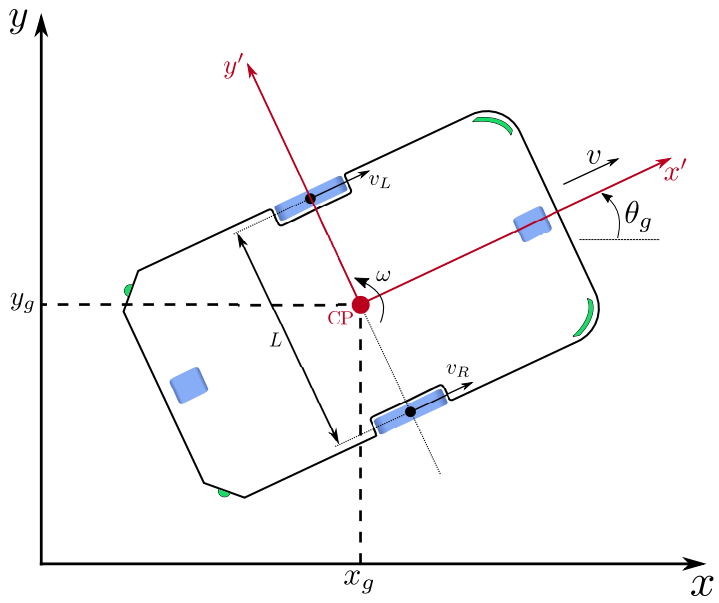


Figure 2.3: The AGV with its fixed coordinate system placed in the global frame. The AGV's center point is placed in $[x_g, y_g]$, its orientation is described by the angle θ_g, and it has a tangential velocity v. The wheel track is denoted L, and the angular velocity is denoted ω.

The pose of the AGV in the global coordinate system at a certain time t can be described as $p_g(t) = [x_g(t), y_g(t), \theta_g(t)]$. The relation between a point $p_g(t)$ in the global system and $p_l(t)$ in the local system of the robot can be expressed as $p_g(t) = R(\theta_g(t))p_l(t)$, where $R(\theta_g(t))$ is the rotation matrix

$$R(\theta_g(t)) = \begin{bmatrix} \cos\theta_g(t) & -\sin\theta_g(t) & 0 \\ \sin\theta_g(t) & \cos\theta_g(t) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.1}$$

For convenience, the time-dependent variables will no longer be denoted with (t), and reference to a pose $[x_g, y_g, \theta_g]$ in the global coordinate system will not be indexed with g. Further, two assumptions regarding a differential drive robot that yields non-holonomic constraints that restrict its motion [11]. The first one has to do with lateral slip. The AGV can only move forward and backward in curved or straight motions without any sideward movement. Thus, the center point CP of the robot has a zero velocity at its lateral axis. Hence,

$$\dot{y}_l^{CP} = 0 \tag{2.2}$$

and this velocity can, by using the rotation matrix, be expressed in the global coordinate system as

$$-\dot{x}^{CP}\sin\theta + \dot{y}^{CP}\cos\theta = 0. \tag{2.3}$$

The second assumption is pure rolling which means that the wheels do not slip against the floor; they only roll. A model of one wheel is shown in Figure 2.4. The velocity for each wheel, the right R, and the left L, at the contact point in the coordinate system of the AGV is expressed as

$$v_R = r\dot{\varphi}_R, \tag{2.4}$$
$$v_R = r\dot{\varphi}_L. \tag{2.5}$$

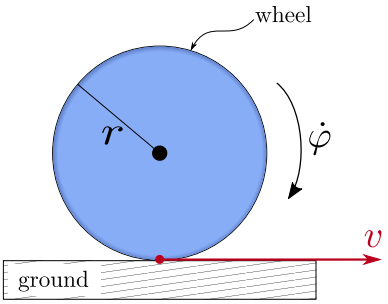


Figure 2.4: A driven wheel with angular velocity $\dot{\varphi}$ and radius r in contact with the ground. The velocity at the contact point is denoted v.

The kinematic model does not consider forces affecting the motion of the AGV. Using a kinematic model, the linear and the angular velocity of the AGV can be described with equations. The linear velocity v is the average of both the right and left wheel velocities

$$v = \frac{v_R + v_L}{2}, \tag{2.6}$$

and the angular velocity ω is the difference between them divided with the wheel track L [12]

$$\omega = \frac{v_R - v_L}{L}. \tag{2.7}$$

The kinematic model and motion equations can be expressed as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}, \tag{2.8}$$

where v and ω is the controller output [13]. Devising a control strategy that generates v and ω such that (x, y, θ) follows the desired trajectory is the main problem considered in this thesis.

3 Trajectory Tracking

3.1 Trajectory

Parts of the trajectory will be sent to the controller in small segments. The first segment is sent when the AGV starts, and then segments are sent continuously while the AGV is moving. New segments will be added to the old, which forms a long trajectory called a *mission*. Trajectory points that are behind the AGV are removed to save calculation time when the trajectory is treated, explained in this section. Each segment contains multiple points where each of them includes x- and y-position and a reference velocity. The number of points for a segment depends on the velocity and can differ between 9 and 18, with a higher number for a higher velocity and a distance between each point of 10 cm.

The current pose, reference pose, reference velocity, and angular velocity must be known to calculate the control signals. The received trajectory lacks orientation θ and angular velocity ω, and hence, they need to be estimated to form a complete trajectory.

3.1.1 Orientation

The orientation at reference point i can be calculated with the known position in point $i-1$ and $i+1$ as

$$\theta_i = \arctan(\frac{\Delta y}{\Delta x}) = \arctan(\frac{y_{i+1} - y_{i-1}}{x_{i+1} - x_{i-1}}), \tag{3.1}$$

where the notations are shown in Figure 3.1. The orientation interval is $[-\pi, \pi]$, and hence, the four-quadrant inverse tangent can be used.

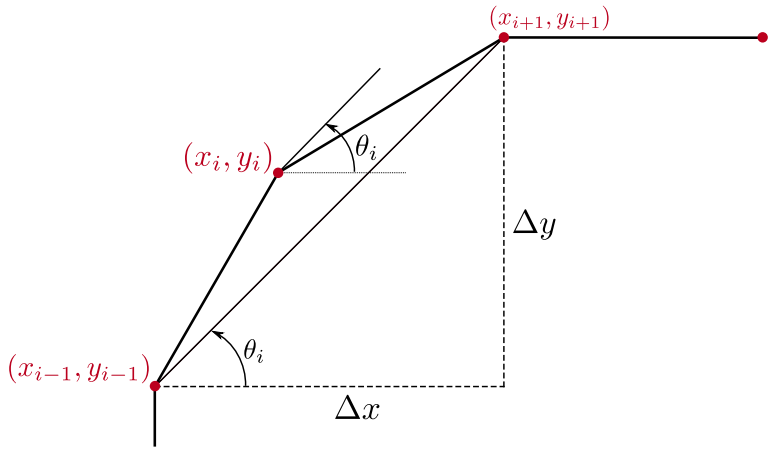


Figure 3.1: Part of a trajectory with four points marked in red. The reference orientation θ_i is calculated using the previous and the next points position.

3.1.2 Tangential Velocity

There is no indication when the AGV shall stop. Hence, it is always important to assume that the current trajectory segment is the last one. The last velocity point in each trajectory segment is therefore put to zero. This will not affect the AGV, except when it is on the last segment since each segment's number of trajectory points is large. Furthermore, the criteria

$$v_i = \sqrt{2 \cdot acc_{max} \cdot \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} + v_{i-1}}, \tag{3.2}$$

is used for maximum acceleration, acc_{max} of the AGV. It is used to re-calculate the trajectory velocities if there is a sudden increase between two points, resulting in a smoother velocity curve during driving. When verifying the acceleration, two points at a time are treated. The verification is performed from the beginning to the end of the trajectory, and the velocity is limited if necessary. Similarly, the criteria

$$v_{i-1} = \sqrt{2 \cdot dec_{max} \cdot \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} + v_i}, \tag{3.3}$$

is used to verify that the deceleration dec_{max}, is not too high. Additionally, the verification is done from the end to the beginning.

At the beginning of a mission, checks for maximum acceleration are not performed. The first trajectory point will be 10 cm in front of the AGV's current position. For the AGV to not exceed the maximum acceleration and not move with the velocity associated with that point right away, a check is critical. When a control signal, described in Section 3.4 and 3.5, is calculated,

$$v_{new} = v_{prev} + acc_{max} \cdot T_s \tag{3.4}$$

is used. If the new velocity v_{new} forces the AGV to exceed the maximum acceleration, it will be recalculated using the previously calculated velocity v_{prev} and sample time T_s.

3.1.3 Angular Velocity

The angular velocity is defined as

$$\omega = \frac{d\theta}{dt}, \tag{3.5}$$

where θ is the orientation of the robot. Moreover

$$\bar{\omega}_i = \frac{\theta_{i+1} - \theta_{i-1}}{t_{i+1} - t_{i-1}} \tag{3.6}$$

is the average angular velocity in the time interval $[t_{i+1}, t_{i-1}]$ [14]. The difference $\Delta t = t_{i+1} - t_{i-1}$ can be calculated if the distance between the points Δs, and the average velocity $\bar{v}$, is known by

$$\Delta t = \frac{\Delta s}{\bar{v}}. \tag{3.7}$$

This implies that the average angular velocity, and hence the reference angular velocity, between two points, is approximated by

$$\bar{\omega}_i = \Delta\theta \frac{\bar{v}}{\Delta s}. \tag{3.8}$$

3.1.4 Interpolation

After completing the steps described above, the trajectory points will now consist of $(x, y, \theta, v, \omega)$. The criteria of the trajectory for the controller are therefore fulfilled. However, to obtain good trajectory tracking, it might be relevant to interpolate the trajectory points by, for example, making a linear interpolation [5]. This is necessary for all components of a point. To interpolate between two points, the formula

$$f(n_{1,i}) = f(n_{1,0}) + \frac{n_{1,i} - n_{1,0}}{n_{2,0} - n_{1,0}}(f(n_{2,0}) - f(n_{1,0})), \tag{3.9}$$

can be used, where $f(n_{1,i})$ is the value at the interpolated trajectory point $n_{1,i}$ with desired index i. In this thesis, ten interpolation points for each original point proved to be appropriate, hence $i = \{0, 1, ..., 9\}$. The new point at index i in between the first two original points in the trajectory can be expressed as $n_{1,i}$. The interpolation of a y-coordinate between the origin points with index three and four can be calculated by

$$y(n_{3,1}) = y(n_{3,0}) + \frac{n_{3,1} - n_{3,0}}{n_{4,0} - n_{3,0}}(y(n_{4,0}) - y(n_{3,0})). \tag{3.10}$$

To interpolate the orientation, it needs to be continuous, hence not limited to the interval $[-\pi, \pi]$. After the interpolation, a wrap angle function can be used to ensure that the value of the orientation is restricted to $[-\pi, \pi]$ once again.

An alternative way is using a cubic spline function. The benefit of a spline function is to, unlike linear interpolation, create a smooth curvature through several data points. The approach is further described in [6]. However, this approach is only used in simulations in this thesis, not in the real system.

3.1.5 Reference Selection

Every time the controller should calculate a new control signal, it must know where the closest point on the path is. This is determined by looking at the Euclidean distance to all points on the trajectory. The point to which the Euclidean distance is the smallest will be the reference point. The equation for finding the Euclidean distance d is

$$d = \sqrt{(x_i - x)^2 + (y_i - y)^2}, \tag{3.11}$$

where x_i and y_i denote the coordinates of point i on the trajectory while x and y denote the coordinates of the AGV. Note that with this way of choosing a reference point, the point does not have to be in front of the AGV; it can also be behind it. This is the approach taken in this thesis. The method can be extended with the condition that the chosen point needs to be in front of the AGV.

Another solution is not to limit the search for a reference point to the trajectory's discrete points. Instead, a point can be chosen that is located somewhere between the closest discrete point and the second closest discrete point. It can be illustrated as a continuous straight line drawn from the closest discrete point to the second closest discrete point. Among all continuous values on this line, the point that has the smallest Euclidean distance to the AGV will be chosen.

3.2 Error Model

While tracking the reference trajectory, described in Section 3.1, the kinematic model

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} \cos\theta_r & 0 \\ \sin\theta_r & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_r \\ \omega_r \end{bmatrix}, \tag{3.12}$$

for a reference point is defined. Further, a difference between the robot's pose and the reference pose is expressed as a tracking error **e**, shown in Figure 3.2.

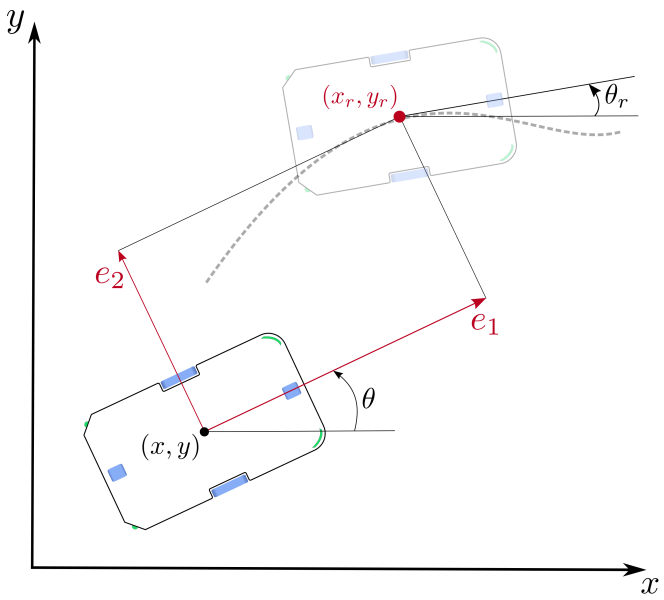


Figure 3.2: The AGV's current pose (x, y, θ) in the global coordinate frame relative to the reference pose (x_r, y_r, θ_r) on the trajectory. The tracking errors e_1 and e_2 are visualized.

Transforming the tracking error to the robot coordinate system by the rotation matrix (2.1), it can be described by

$$\mathbf{e} := \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix}. \tag{3.13}$$

Taking the derivative of the tracking error (3.13) and combining the AGV's kinematics (2.8) and the kinematics of the reference point (3.12) yields [15]:

$$\begin{aligned}
\dot{e}_1 &= -\omega \sin\theta(x_r - x) + \cos\theta(\dot{x}_r - \dot{x}) + \omega\cos\theta(y_r - y) + \sin\theta(\dot{y}_r - \dot{y}) = \\
&= \sin\theta\Big((\dot{y}_r - \dot{y}) - \omega(x_r - x)\Big) + \cos\theta\Big((\dot{x}_r - \dot{x}) + \omega(y_r - y)\Big) = \\
&= \sin\theta\Big(\sin\theta_r v_r - \sin\theta v - \omega(x_r - x)\Big) + \cos\theta\Big(\cos\theta_r v_r - \cos\theta v + \omega(y_r - y)\Big) = \\
&= /(2.8), (3.12)/ = \\
&= -v\big(\sin^2\theta + \cos^2\theta\big) + v_r\big(\sin\theta\sin\theta_r + \cos\theta\cos\theta_r\big) + \\
&\quad + \omega\big(\cos\theta(y_r - y) - \sin\theta(x_r - x)\big) = \\
&= -v + v_r\cos(e_3) + \omega e_2
\end{aligned} \tag{3.14}$$

Using similar calculations, it can be shown that

$$\begin{aligned}
\dot{e}_2 &= -\omega\Big(\cos\theta(x_r - x) + \sin\theta(y_r - y)\Big) - \sin\theta(\dot{x}_r - \dot{x}) + \cos\theta(\dot{y}_r - \dot{y}) = \\
&= -\omega e_1 + v_r\sin(e_3) && (3.15) \\
\dot{e}_3 &= \omega_r - \omega && (3.16)
\end{aligned}$$

Rewriting the derivatives (3.14, (3.15), (3.16)) in matrix form gives

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \begin{bmatrix} \cos e_3 & 0 \\ \sin e_3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_r \\ \omega_r \end{bmatrix} + \begin{bmatrix} -1 & e_2 \\ 0 & -e_1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}, \tag{3.17}$$

where the feedforward part of the control signals, v_r and ω_r, are reference velocity and reference angular velocity. Moreover, v and ω are the linear and angular velocities that are the control signals to the system [2]. If the control signals are expressed as

$$\begin{aligned}
v &= v_r\cos e_3 - v_{fb}, \\
\omega &= \omega_r - \omega_{fb},
\end{aligned} \tag{3.18}$$

where v_{fb} and ω_{fb} are the feedback part of the control signal, (3.17) can be rewritten as [3]:

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \begin{bmatrix} 0 & \omega & 0 \\ -\omega & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} + \begin{bmatrix} 0 \\ \sin e_3 \\ 0 \end{bmatrix} v_r + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_{fb} \\ \omega_{fb} \end{bmatrix}. \tag{3.19}$$

The model of the AGV can be described in state-space form by

$$\dot{x} = Ax + Bu \tag{3.20}$$

$$y = Cx \tag{3.21}$$

where x is the state of the system, u is the input signal to the system, and y is the output from the system. A, B and C are matrices with dimensions $n \times n$, $n \times m$, and $p \times n$, where n is the number of states, m is the number of control signals, and p is the number of system outputs

[7]. Linearizing (3.19) around the operation point $e_1 = e_2 = e_3 = 0$ and $v_{fb} = \omega_{fb} = 0$ gives [4]:

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \begin{bmatrix} 0 & \omega_r & 0 \\ -\omega_r & 0 & v_r \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}. \tag{3.22}$$

Controllability is an important property when studying the system. By using the matrices B, AB, ..., $A^{n-1}B$, the controllable states can be derived [7]. If a state x can be taken from origin to x^* by a control signal, the state vector is said to be controllable. Furthermore, if all state vectors are controllable, the complete system is controllable. The columns in the matrix

$$\mathcal{S} = (B \quad AB \quad ... \quad A^{n-1}B) \tag{3.23}$$

span a linear space that is equal to the set of controllable state-space vectors. Using Model (3.22) and (3.23) gives

$$\mathcal{S} = \begin{bmatrix} 1 & 0 & 0 & 0 & -\omega_r^2 & v_r\omega_r \\ 0 & 0 & -\omega_r & v_r & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.24}$$

If $\text{rank}(\mathcal{S}) = n$, the system is controllable [16]. When $v \neq 0$ and $\omega \neq 0$, $\text{rank}(\mathcal{S}) = 3$ and hence, the system is controllable. However, when $v = 0$ and $\omega = 0$, $\text{rank}(\mathcal{S}) = 2$ and the system is not controllable.

3.3 Positioning

There is noise in the position and orientation given by the positioning system. The noise is significant at some locations in the lab. Therefore, lowpass filtering of the position and orientation of the AGV can be suitable to suppress the noise. A filter that considers the current and all previous input values and the previous output value when determining a new output value is called Infinite Impulse Response (IIR) filter. The equation

$$y(n) = \sum_{i=0}^{\infty} h(i)x(n-i) \tag{3.25}$$

describes the IIR filter, which is both memory and computationally efficient [17].

The position and orientation given by the sensors are filtered by using the equation

$$y(i) = y(i-1) + \frac{x - y(i-1)}{\alpha} \tag{3.26}$$

where $y(i)$ is the filtered output, $y(i-1)$ is the previous filtered output, x is the new raw data, and $\alpha \geqslant 1$ is a lowpass filter parameter. Increasing α will reduce the bandwidth of the filter and also results in increased phase shift. Setting $\alpha = 1$ results in no filtering of the data [18].

3.4 PID Controller

A simple approach for trajectory tracking is to use a (PID) controller. PID controllers are mostly used for Single Input Single Output (SISO) systems. However, System (3.22) is a Multiple Input Multiple Output (MIMO) system since it uses two control signals, v and ω, and has three outputs, e_1, e_2, and e_3. An example of how a PID controller can control a non-SISO system is described in [19]. The problem in the article is to stabilize an inverted pendulum.

The system has two outputs and one input. The outputs are the position of the cart and the angle of the pendulum, while the input is a signal that adjusts the cart's speed. This makes this system a Single Input Multiple Output (SIMO) system. An appropriate control signal can be calculated by adding the outputs of two separate PID controllers. One of the controllers takes reference position and current position as input, and the other controller takes reference orientation and current orientation as input.
Furthermore, a method to improve reference tracking is to use feedforward control. This will add an extra degree of freedom to the controller that can increase robustness. The control signal to the system will then be

$$u = u_{fb} + u_{ff}, \tag{3.27}$$

where u_{fb} denotes the feedback part of the control signal, and u_{ff} denotes the control signal's feedforward part. These terms can be calculated by

$$u_{fb} = F(y_r - y), \tag{3.28}$$
$$y_r = G_m r, \tag{3.29}$$
$$u_{ff} = F_f r, \tag{3.30}$$

where F is the controller's feedback part, G_m is a reference model, r is the reference signal, and F_f is the feedforward part of the controller. Choosing $G_m = 1$ can, in some cases, be preferable. Choosing $F_f = \frac{G_m}{G}$ results in ideal feedforward while choosing $F_f = \frac{1}{G(0)}$ results in neutral feedforward [8]. A block diagram of the combined feedback and feedforward controller for a SISO system is shown in Figure 3.3.

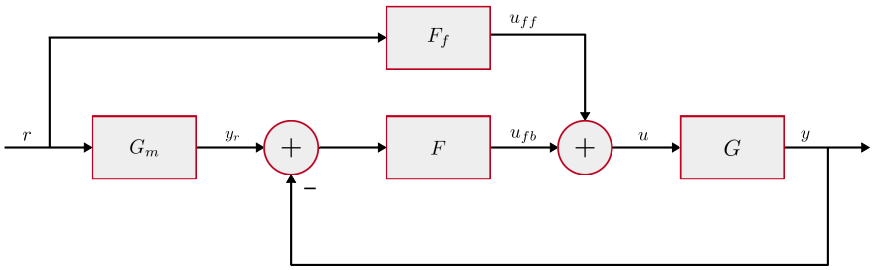


Figure 3.3: Feedback and feedforward control. G_m is a reference model, F_f is the feedforward, block F is the feedback controller and G is the system.

A transfer function matrix can be used to describe a MIMO system. The system can hence be expressed as

$$Y(s) = \begin{bmatrix} g_{11}(s) & \dots & g_{1m}(s) \\ \vdots & \ddots & \vdots \\ g_{l1}(s) & \dots & g_{lm}(s) \end{bmatrix} U(s) \tag{3.31}$$

where $Y(s)$ is the output, $U(s)$ the input, and $g_{ij}(s)$ is a transfer function from input j to output i [20]. Transfer functions can express a system given on state-space form. The transfer function is calculated by

$$G(s) = C(sI - A)^{-1}B + D. \tag{3.32}$$

Using (3.22) and (3.32) results in

$$G(s) = \begin{bmatrix} \frac{s}{s^2+\omega_r^2} & \frac{v_r\omega_r}{s^3+s\omega_r^2} \\ \frac{-\omega_r}{s^2+\omega_r^2} & \frac{v_r}{s^2+\omega_r^2} \\ 0 & \frac{1}{s} \end{bmatrix} \tag{3.33}$$

which is the model of the differential drive vehicle in terms of a transfer function matrix [7]. Combining (3.31) and (3.33), it is clear that the first control signal, v, will affect e_1 and e_2, while the second control signal, ω, will affect all states. When following a straight line ($\omega_r = 0$), v will only affect e_1 while ω will affect e_2 and e_3. A block diagram describing the system is shown in Figure 3.4.

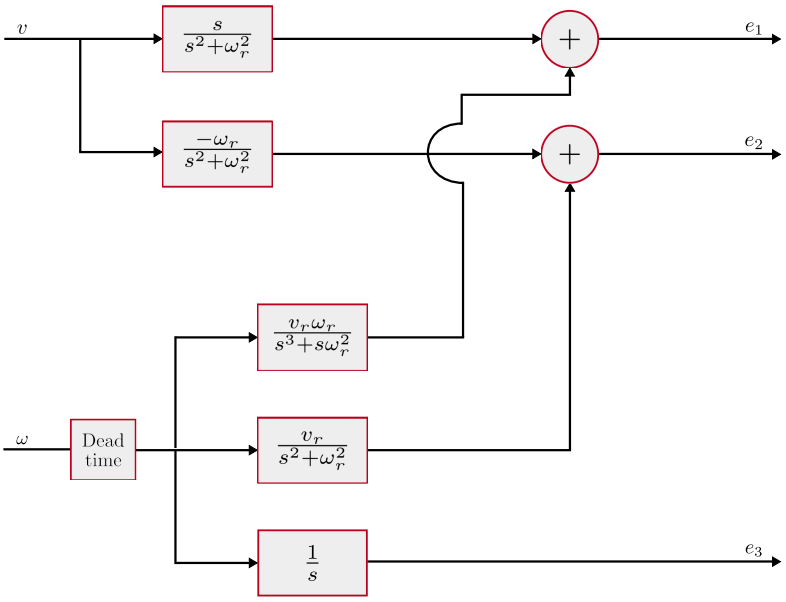


Figure 3.4: A block diagram that describes how the control signals, v and ω, affect the states, e_1, e_2, and e_3.

When implementing a trajectory tracking controller, it must be determined which outputs are important to control. A proposed method for controlling a vehicle that uses steering angle and speed as inputs is described in [21], which considers the orientation error and distance to the path as inputs to a feedback controller and reference steering angle as input to a feed-forward controller. The outputs from the two controllers are added and form the desired steering angle, one of the system's control signals. The other control signal, the velocity of the vehicle, is calculated using a reference velocity and a method for obstacle avoidance, which limits the velocity. The longitudinal error is not relevant to use in any of the controllers. This method can be applied to this thesis, where e_2 and e_3 should be used to calculate a suitable steering angle, which in this case will be an angular velocity ω of the AGV since it is a differential drive vehicle and does not have traditional steering. Furthermore, the second control signal, v, should be calculated using the reference velocity and, if possible, signals that indicate that the velocity must be decreased. Hence, it is unnecessary to use e1 in the controller when applying the method in this thesis.

A suitable implementation of PID controllers is shown in Figure 3.5. It is inspired by the control strategy used in the SIMO system described above. Additionally, it is based on that ω can affect all states, and the measured signals e_2 and e_3 are the most important to use in a trajectory tracking controller. One of the PID controllers will use the difference between reference orthogonal error and current orthogonal error as input. In contrast, the other PID controller uses the difference between reference orientation error and current orientation error as input. Reference orthogonal error and orientation will, of course, be equal to zero. Both controllers will, from their inputs, calculate suitable control signals that will be added to form the desired angular velocity, ω_{fb}, to the system. This can be described by

$$\omega_{fb,2} = k_{P,2} \cdot e_2 + k_{I,2} \int_0^t e_2(t)dt + k_{D,2} \frac{de_2(t)}{dt}, \tag{3.34}$$

$$\omega_{fb,3} = k_{P,3} \cdot e_3 + k_{I,3} \int_0^t e_3(t)dt + k_{D,3} \frac{de_3(t)}{dt}, \tag{3.35}$$

$$\omega_{fb} = \omega_{fb,2} + \omega_{fb,3}, \tag{3.36}$$

where $k_{P,i}$, $k_{I,i}$, and $k_{D,i}$ is the controller parameters and $\omega_{fb,i}$ is the outputs from the controllers, for $i = \{2, 3\}$.

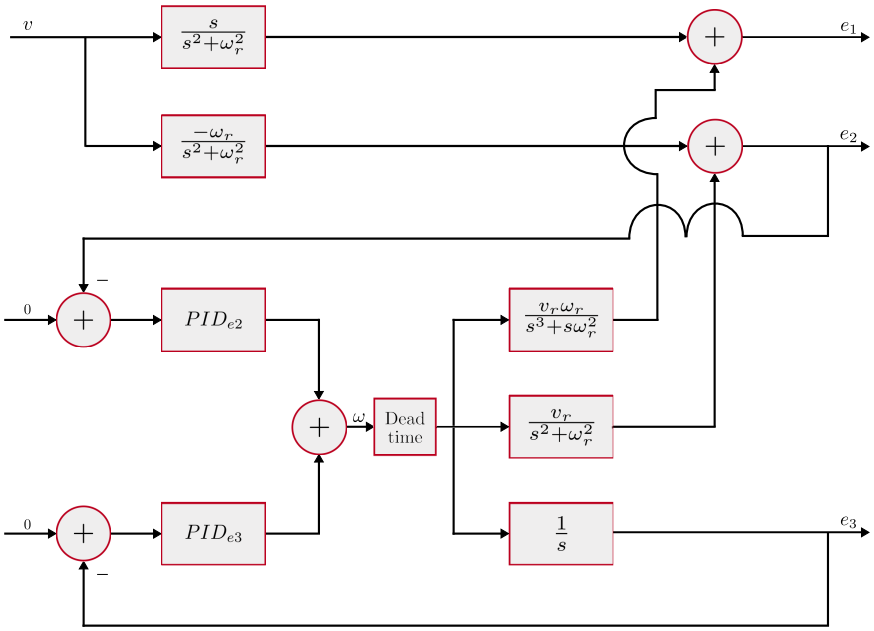


Figure 3.5: The system controlled by two PID controllers.

The feedback controller that consists of the two PID controllers can be extended with a feedforward controller. The feedforward controller will use reference velocity, v_r, and reference angular velocity, ω_r, given by the trajectory. Since no feedback loop exists to compute the desired velocity, v_{fb}, of the AGV, the final controller will output the control signals

$$v = v_r, \tag{3.37}$$

$$\omega = \omega_r + \omega_{fb}. \tag{3.38}$$

The complete controller is shown in Figure 3.6.

Since the system has large dead time, it is necessary to consider this when creating the controller. Therefore, the future state of the AGV will be predicted, which is described in Section 3.5.3. From the future state, the future predicted errors $e_{1,future}$, $e_{2,future}$, and $e_{3,future}$ can be derived and used in the two PID controllers.

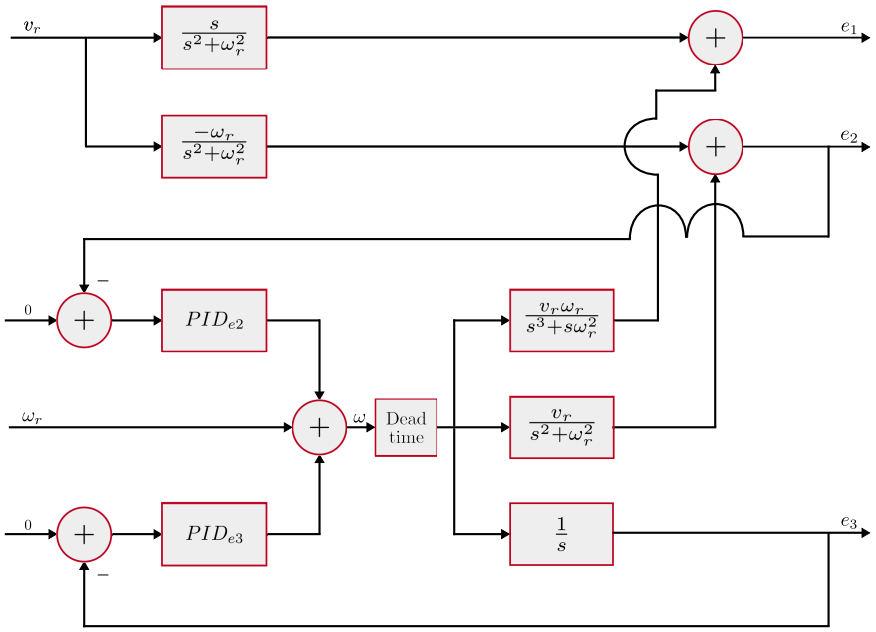


Figure 3.6: PID controllers with feedforward.

3.5 LQ Controller

A control strategy that can be used is LQ control. If it is assumed that A and B are constant and the system is controllable, a controller that takes the system from a non-zero initial state to zero can be constructed. The controller can keep both the state error of the system and the control signal itself small by minimizing the quadratic performance index

$$J = \int_{t_0}^{T} (x^T Q x + u^T R u) dt \tag{3.39}$$

from time t_0 to T. In the integral (3.39) R and Q are positive definite matrices [10]. Based on the performance index (3.39) when $T \to \infty$, the optimal control signal is calculated by

$$u = -Kx = -R^{-1} B^T P x, \tag{3.40}$$

where P is a positive semidefinite matrix that is calculated by solving the Algebraic Riccati Equation [9]

$$A^T P + PA + Q - PBR^{-1}B^T P = 0. \tag{3.41}$$

3.5.1 Tracking and Gain Selection

The control law in (3.40) is suitable when all controlled outputs should be steered to zero. For this purpose, an error model was created and linearized, as described in Section 3.2. The states, hence the errors, will be forced to zero. The control signals can be expressed as

$$v = v_r \cos(e_3) + v_{fb}, \tag{3.42}$$

$$\omega = \omega_r + \omega_{fb}, \tag{3.43}$$

where v_r and ω_r is the reference velocity and angular velocity given by the trajectory while v_{fb} and ω_{fb} are the feedback control signals. The feedback control signals are calculated

based on (3.40), where the errors are the state. Hence, the feedback control signals become

$$\begin{bmatrix} v_{fb} \\ \omega_{fb} \end{bmatrix} = - \begin{bmatrix} k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,1} & k_{2,2} & k_{2,3} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix}. \tag{3.44}$$

From (3.44) it is obvious that if the robot is traveling on the trajectory with the correct orientation ($e_1 = e_2 = e_3 = 0$), the feedback control signals will be zero [3]. A block diagram illustrating the controller given by (3.42) and (3.43) are shown in Figure 3.7.

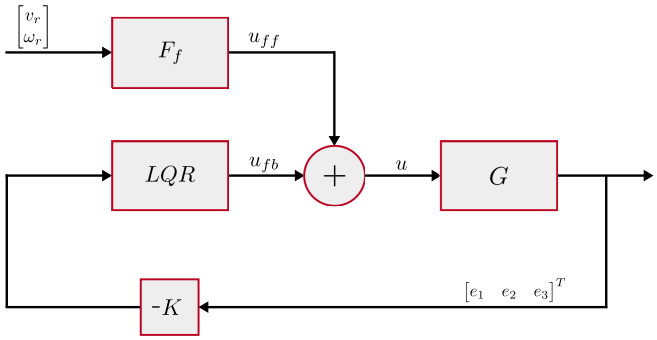


Figure 3.7: An LQ controller that uses feedforward to control the system. u_{fb} is the feedback contribution based on the errors e_1, e_2, and e_3 and u_{ff} is the feedforward contribution based on reference velocity and angular velocity. Adding these two contributions form the control signal u, where u is a vector containing control signals v and ω to the system.

The velocity and angular velocity of the robot will vary along the trajectory, and therefore there are different operating points, which might require different controllers. Gain scheduling can be used for this purpose to create controllers for several operating points [9]. Hence, a set of predefined linearized error models, defined in (3.22), can be created. The A and B matrices of these models, along with the chosen Q and R matrices, will be used to determine suitable controller gain matrices, K, for all operating points offline. These gains are placed in a look-up table so that they can be used online by the controller. Since the controller parameters will change over time, the controller will be nonlinear [8].

Another alternative to save memory is by calculating a gain matrix, K online every time the controller determines a new control signal. This requires that the Algebraic Riccati Equation in (3.41) is solved online. This is the approach taken in this thesis.

3.5.2 Discrete Algebraic Riccati Equation

The discrete representations of the A and B matrices in the state-space description are denoted F and G. These matrices can be calculated by

$$F = e^{AT_s}, \tag{3.45}$$

$$G = \int_0^{T_s} e^{At} B dt, \tag{3.46}$$

where T_s is the sampling time and the factor e^{At} can be calculated by

$$e^{At} = \mathcal{L}^{-1}(sI - A)^{-1}, \tag{3.47}$$

where $\mathcal{L}$ is the Laplace transform. The relationship between the input signal, the states, and the output at the sampling moments can be described by

$$x(kT_s + T_s) = Fx(kT_s) + Gu(kT_s), \tag{3.48}$$

$$y(kT_s) = Cx(kT_s), \tag{3.49}$$

if the input signal is piecewise constant in the interval $[kT_s, kT_s + T_s]$ [8]. Using Equation (3.22), (3.45), (3.46), and (3.47) results in

$$F = \begin{bmatrix} \cos(T_s\omega_r) & \sin(T_s\omega_r) & \frac{v_r}{\omega_r}(1 - \cos(T_s\omega_r)) \\ -\sin(T_s\omega_r) & \cos(T_s\omega_r) & \frac{v_r}{\omega_r}\sin(T_s\omega_r) \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.50}$$

$$G = \begin{bmatrix} \frac{\sin(T_s\omega_r)}{\omega_r} & \frac{v_r}{\omega_r}(T_s - \frac{\sin(T_s\omega_r)}{\omega_r}) \\ \frac{1}{\omega_r}(\cos(T_s\omega_r) - \frac{1}{\omega_r}) & -\frac{v_r}{\omega_r^2}(\cos(T_s\omega_r) - 1) \\ 0 & T_s \end{bmatrix}. \tag{3.51}$$

Similar to (3.41), the Discrete Algebraic Riccati Equation (DARE) can be expressed as

$$P - F^T PF + F^T PG(R + G^T PG)^{-1}B^T PF - Q = 0. \tag{3.52}$$

Instead of minimizing (3.39), it is

$$J_d = \sum_k (x_k^T Q x_k + u_k^T R u_k) \tag{3.53}$$

that will be minimized. Solving for P in (3.52) gives the optimal controller gain

$$K_d = (R + G^T PG)^{-1} G^T PF, \tag{3.54}$$

which can be used in a state feedback controller [22].

An algorithm to solve the DARE is called the structure-preserving doubling algorithm (SDA). The algorithm is described below. In the algorithm, I is the unit matrix. The algorithm will converge quadratically to the solution P [23].

Algorithm 1: Structure-preserving doubling algorithm

Result: P
Input:
$j \leftarrow 0$;
$K_0 \leftarrow F$;
$L_0 \leftarrow GR^{-1}G^T$;
$P_0 \leftarrow Q$;
while $\|P_{j+1} - P_j\| > \epsilon\|P_{j+1}\|$ **do**
  $K_{j+1} \leftarrow K_j(I + L_jP_j)^{-1}K_j$;
  $L_{j+1} \leftarrow L_j + K_j(I + L_jP_j)^{-1}L_jK_j^T$;
  $P_{j+1} = P_j + K_j^T P_j(I + L_jP_j)^{-1}K_j$
end
return P_{j+1}

To calculate gain matrices online, the reference velocity and angular velocity, v_r and ω_r, are used. With these variables, the F and H matrices can be calculated. Then (3.52) is solved using SDA, which returns the matrix P. This matrix is used in (3.54), which gives the gain matrix K_d.

3.5.3 Time Delays

The system to be controlled contains time delays. However, using a predictor approach allows a gain matrix, K, to be used even though it has been calculated with no caution to time delays. This strategy uses the Smith predictor. The feedback part of the control signal in (3.42) and (3.43) will change from the form in (3.40) to

$$u_k = -K \cdot \left(A^d x_k + \sum_{i=1}^{d} A^{i-1} B u_{k-i} \right), \tag{3.55}$$

where k is the current time index, and d is the delay in the system. The gain matrix, K is multiplied with the future predicted state $\hat{x}_{k+d}$ instead of the current state, which is the case without time delays [24].

Handling delays in this way is suitable to use in this thesis. However, rather than using the future predicted state in (3.55), it is the future predicted error that is to be multiplied with K. This also means that when choosing a point on the trajectory as a reference, it is the point closest to the predicted position that must be chosen rather than the closest point on the trajectory at the sampling moment.

4 Method

The workflow in this thesis has been divided into several parts, starting with a literature study and time to familiarize ourselves with the system. Thereafter, simulation, implementation, and testing of controllers and comparisons were performed, described in detail below.

4.1 Simulation

Both the PID and the LQ controller has been tested in a simulation environment before they were implemented on the truck. The simulation was done using Simulink and MATLAB, where the Simulink model is shown in Figure 4.1. The block *Trajectory correction* modifies the given trajectory, according to Section 3.1. For example, the velocity in the trajectory is modified so that the speed, in the beginning, will be ramped up and the speed, in the end, will be ramped down. This is to allow smooth acceleration and deceleration. Furthermore, it calculates angular velocities in every point and interpolates the trajectory. However, the trajectory in the simulation is very long. In reality, only short sections of the trajectory are available when finding the closest reference points. The modifications of the trajectory that must be made will therefore differ in reality and simulations.

The block *Reference finder* finds the closest reference point (x_r, y_r, θ_r, v_r, ω_r) based on the current position and orientation and the trajectory. The block *Transformation to Robot coordinates* transforms the global errors to inertial errors by using the rotation matrix. The blocks *LQR* and *PID* contains an LQ and a PID controller, respectively. The *LQR* block uses the function *lqr* in MATLAB to calculate a suitable gain matrix K, which is used to determine control signals. The gain matrix cannot be determined this easily on the real system since the function *lqr* only can be used in MATLAB. Therefore, it is instead calculated by solving the Riccati equation, as described in Section 3.5. In the simulation, the controllers were implemented in MATLAB code and tested by running simulations where the AGV should follow the given trajectory.

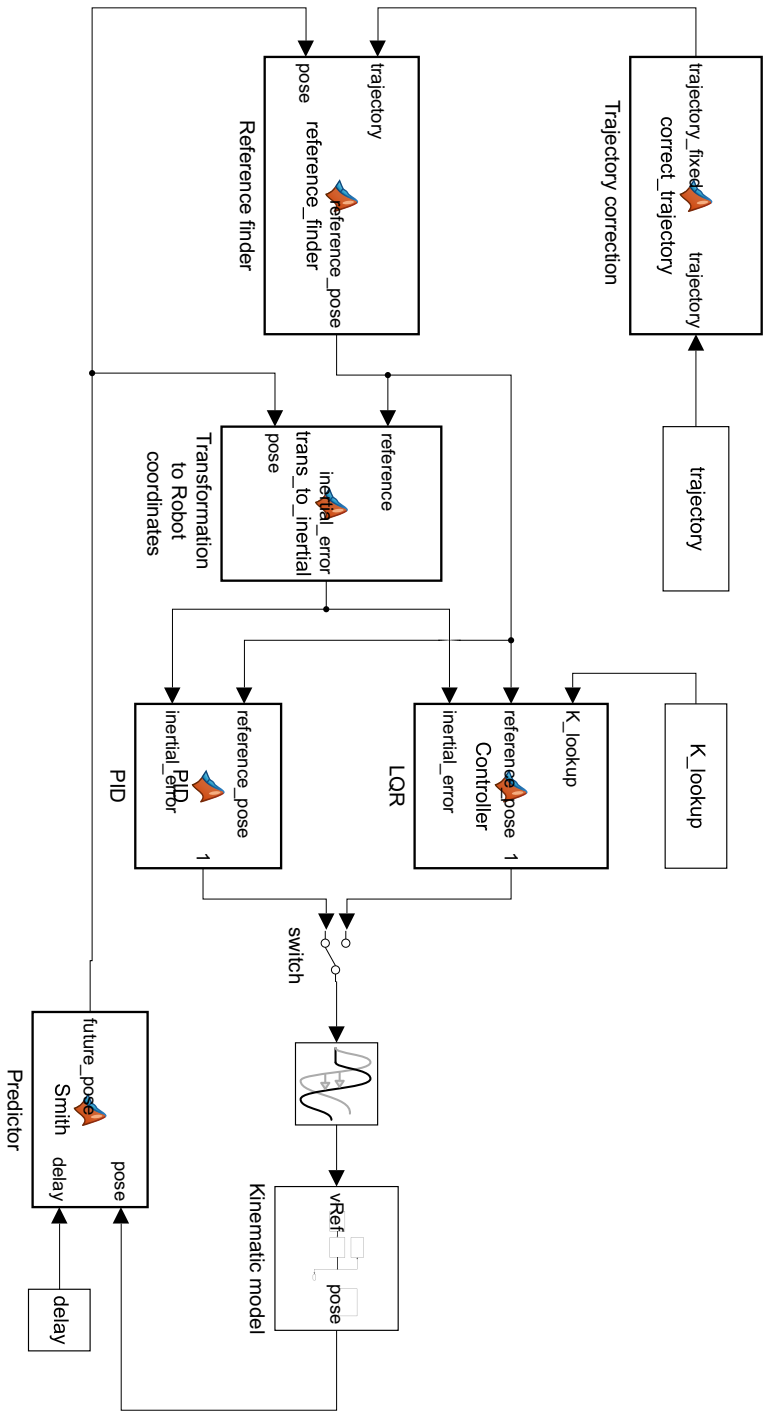


Figure 4.1: Simulink scheme used for simulations.

There is also a switch that determines which controller that affects the system. Hence, the user can choose which control signals that will be passed on to the *Kinematic Model*. The block *Kinematic Model* represents the AGV. It takes tangential and angular velocity as input, and the output is a position and orientation. The sample time of this block is 0.02 s since this is the cycle time in the real system. The block *Predictor* estimates the future state of the AGV based on previous control signals and current position and orientation.

A block called *Transport Delay* is used to simulate a delay in the system. This block is placed before the *Kinematic Model*. The delay occurs in the real system when a control signal has been sent from the controller to the motor controller. The size of the delay can be chosen in MATLAB. For the real system, it is approximately 150 milliseconds.

4.2 Implementation and Testing of Controllers

On the real system, the PID and the LQ controllers were implemented in C++ code. This programming language was chosen since most of the other nodes in the system were written in it.

The implemented program consists of five classes: *Smartness*, *TrajectoryHandler*, *Controller*, *LQ*, and *PID*. The communication between them is shown in Figure 4.2.

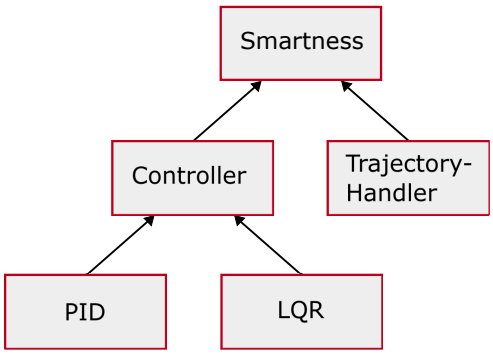


Figure 4.2: Relationship between classes in C++.

The class *Smartness* is responsible for communication with the other nodes in the system and controls when the other classes in the program should perform tasks. This class will receive all necessary information from the node Toyota Smartness, such as current position and trajectory. It will send control signals back to Toyota Smartness.

The class *TrajectoryHandler* modifies the trajectory described in Section 3.1. It scales the values so that the dimension of the data is correct, interpolates the trajectory points, calculates the reference angles and the angular velocities in every point and modifies the velocity at the beginning and end of the trajectory.

The class *Controller* works as a superclass to the classes *LQ* and *PID*. Only one of these sub-classes is performing tasks during a mission. These classes are responsible for finding the closest point on the trajectory and, from this, calculate control signals.

4.3 Comparisons Between Controllers

To evaluate the performance of the controllers, several tests have been carried out. Each test has been repeated to ensure credibility and consistency. Not only have the PID and the LQ controllers been tested, but the results have also been compared with the existing controller used on the AGV today.

4.3.1 Trajectory Tracking

To evaluate and compare the trajectory tracking abilities of each controller, a mission was made that includes straight lines with high and low speeds and curves with one and two meter radius. Combinations of these sections also occur. It can, for example, be a left-hand curve with a one-meter radius followed by a right-hand curve with a two-meter radius. These types of sections have been chosen since TMH frequently uses them.

When comparing the PID and LQ controller with the existing controller, the trajectory must be constructed differently. This is because our controllers interpret the reference velocity in the trajectory differently compared to the existing controller. Our controllers aim to maintain the reference velocity, and hence the outer wheel in a curve will move at a higher velocity than the reference velocity. The existing controller interprets the reference velocity as a maximum velocity that no part of the AGV is allowed to travel faster than. This results in the outer wheel in a curve having the same velocity as the reference velocity. Hence, the total velocity of the AGV will be smaller than the reference velocity. Therefore, a trajectory constructed for our controllers must be modified if the existing controller controls the AGV, so that the reference velocity in the curves becomes higher. Using (2.6), (2.7), and $v = Rw$, where R is the radius of the curve, results in

$$v_{outer} = \frac{v(L + 2R)}{2R}, \tag{4.1}$$

where v_{outer} is the speed the outer wheel must have to make the AGV travel with velocity v in a curve. For example, if the new controllers uses v in (4.1) as reference velocity, the existing controller must use v_{outer} instead.

4.3.2 Load Distribution

The controllers have been implemented and evaluated on different AGVs with the same framework and dimensions. The center of gravity and weight affect the performance. To evaluate how robust the controllers are, a 55 kg weight has been placed at different locations on the AGV while performing a predefined driving schedule. This is an important test since the AGV must have good tracking abilities both unloaded and loaded, and there is no guarantee that the load will be placed in the center of gravity. A suitcase at an airport can be unevenly packed and even be misplaced on the AGV's conveyor belt.

Tests will first be conducted when the AGV is driving without load. Thereafter, the 55 kg weight is first placed in the middle of the AGV, then at the front-right corner, and finally at the back-left corner. The placement and driving schedule are illustrated in Figure 4.3 and 4.4, respectively. To perform the tests, the AGV is instructed to drive to predefined points. These points can be of type *Load*, *Unload*, and *Stop*. At the *Load* and *Unload* points, the AGV will stop and shortly after drive again. At the *Stop* point, it will stop and stay there. In this driving schedule, the AGV starts by facing west while standing at the point marked *Stop* and is asked to drive to *Load*, then to *Unload*, and finish at *Stop*. Hence, it will first drive on the outer part of the area, pass the point *Unload* without stopping, and finally reach *Load*. Then, it drives on the trajectory in the middle until it reaches *Unload*. Finally, it will drive on the outer part of the area, pass the point *Load* without stopping, and finally reach *Stop*.

Figure 4.3: An illustration of the 55 kg weight placement for different tests. The blue box is placed at the front-right corner of the AGV, the red in the middle, and the green at the back-left corner.

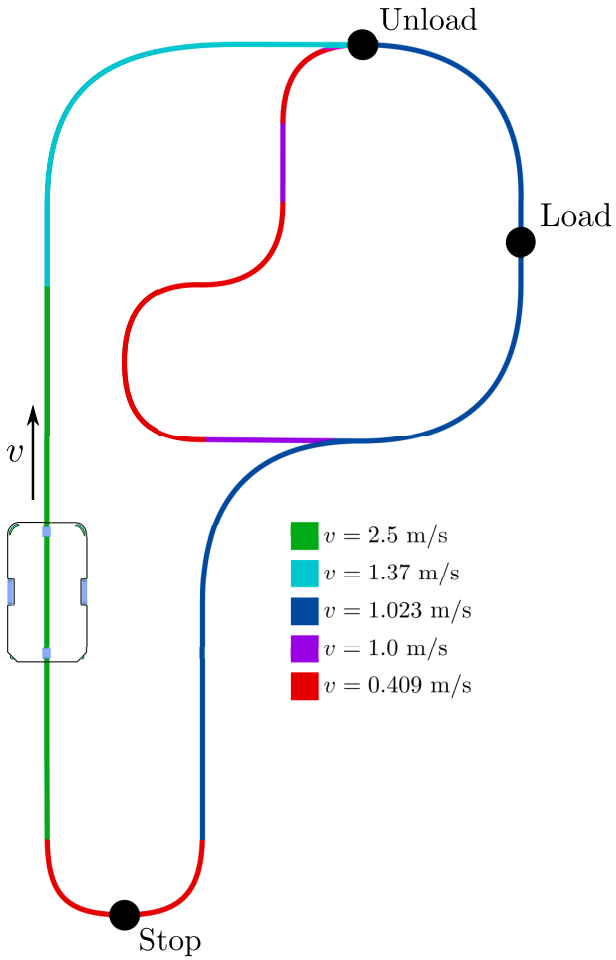


Figure 4.4: The driving schedule with different velocity profiles illustrated. The AGV will perform different actions corresponding to the points marked in black.

4.3.3 Stopping Accuracy

An important aspect of trajectory tracking, not least for this application, is how accurate the AGV can stop at the final position of a trajectory. Furthermore, the orientation accuracy at these points has also been put to the test. To determine the accuracy, all three controllers have been tested in six different stop scenarios. All scenarios have been tested eight times for each controller, starting from their previous stopping point at a standstill, further described in Table 4.1. Figure 4.5 illustrates the trajectories leading up to the stopping points.

Table 4.1: A description of the stop scenarios and their properties in the form of curve radius and vehicle speed. The points are referred to as stopping point numbers.

Stopping point	Description	Curve radius	Vehicle speed
1	From standstill to full acceleration with a sudden stop at a straight path	-	2.5 m/s
2	A right 90° curve followed by a stop	2 m	1.37 m/s
3	An s-shaped curve from right to left.	2 m	1.02 m/s
4	An s-shaped curve from left to right.	1 m	0.49 m/s
5	A slow crawling pace in a straight line	-	0.1 m/s
6	A 180° right curve	1 m	0.37 m/s

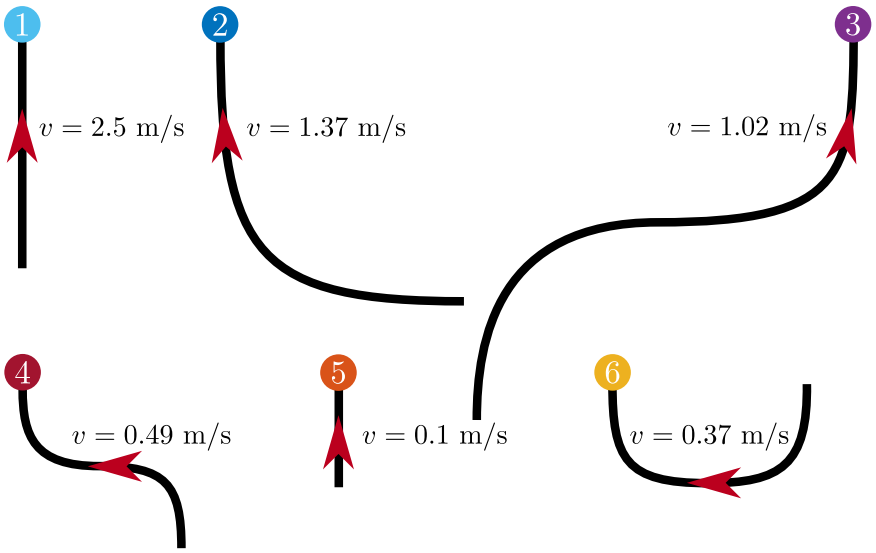


Figure 4.5: An illustration of the trajectories with stopping points.

5 Evaluation

In this chapter, the controller results from simulations and when using the AGV are presented, compared, and discussed. The method used in the thesis, introduced in Chapter 4, is also discussed.

5.1 Simulation

As mentioned in Chapter 4, simulations of the PID and LQ controllers has been made in Simulink. Noise has not been added to the simulation. Results from this, with and without delay, are presented in this section. The AGV has followed the outer part of the trajectory described in Figure 4.4. Hence, the trajectory followed in simulation is the one shown in Figure 5.1. The parameters used for the controllers that were proposed in Chapter 3 are presented in Table 5.1.

Table 5.1: Parameter values for the PID and the LQ controller. The parameters indexed with $i = 1, 2, 3$ affects the states, e_i. The parameters R_v and R_ω affects the control signals v and ω, respectively.

PID		**LQ**	
Parameter	Value	Parameter	Value
$k_{P,2}$	6	Q_1	1
$k_{I,2}$	0.01	Q_2	300
$k_{D,2}$	0.5	Q_3	1
$k_{P,3}$	4	R_v	10
		R_ω	10

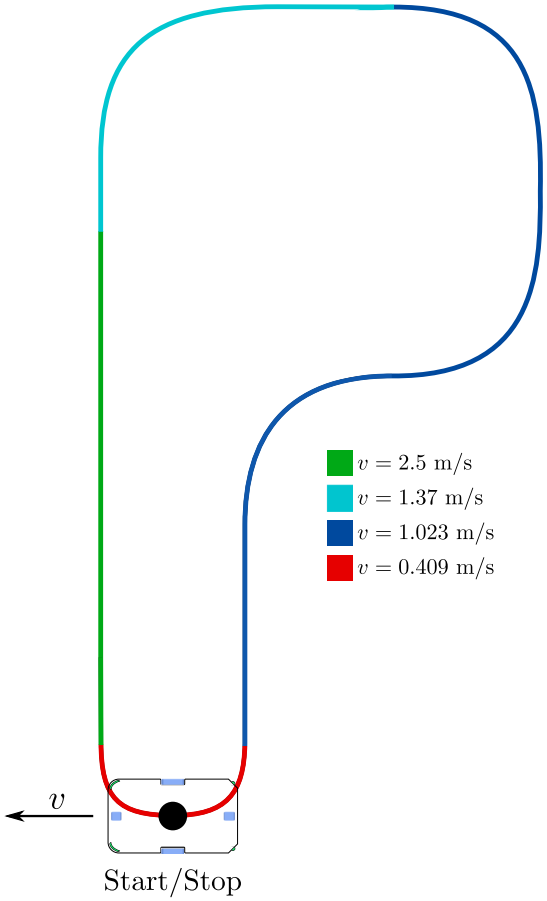


Figure 5.1: The trajectory used in simulations. The starting point is furthest down, and the AGV is first oriented west.

The first test was conducted when the PID controller was running without delay in the simulation. The desired velocities on the left and right wheel, v_L and v_R from this test, are shown in Figure 5.2.

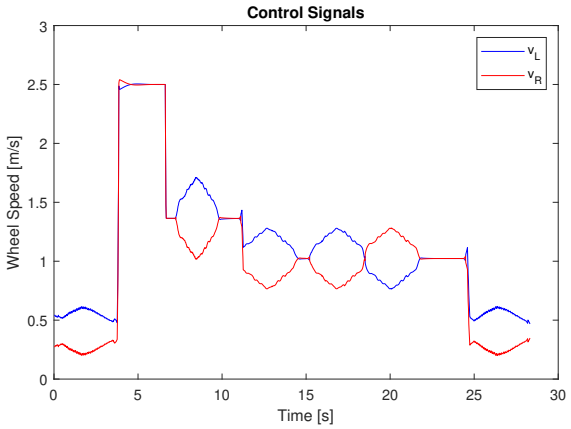
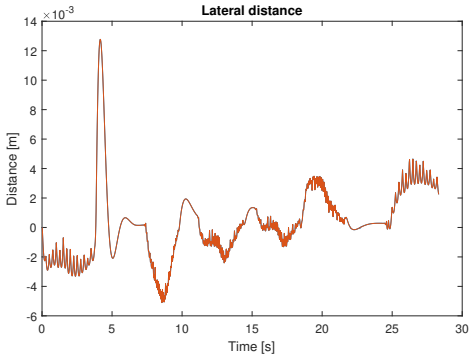


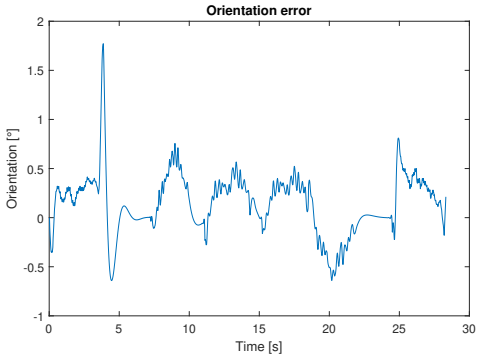
Figure 5.2: Calculated control signals when the PID controller is running in simulation without delay.

The distance between the trajectory and the AGV, the lateral distance, and the orientation error are illustrated in Figure 5.3. The maximum deviation from the path is 13 mm, and the maximum orientation error is 1.75°. Results from the same tests but with the LQ controller are shown in Figure 5.4. In this case, the maximum deviation from the path is 11 mm, and the maximum orientation error is 1.75°. The figures show that the highest peak, which occurs at around 4 seconds, is smaller when the LQ controller is used. Overall, the performance of the controllers is similar.

The LQ controller with delay compensation has also been tested in simulations with a simulated delay of 150 ms. Figure 5.5 shows the lateral distance from the AGV to the trajectory and the orientation error. The maximum deviation from the path is 14 mm, and the maximum orientation error is 2.1°. Both the lateral distance and orientation error are larger than when no delay was present, but the increase is small. Since the delay is 150 ms and the cycle time is 20 ms, the prediction of a future pose will be $150/20 = 7.5$ steps ahead, which is not an integer. Therefore, a perfect prediction cannot be obtained, which explains the increasing errors. Looking at Figures 5.3a, 5.4a, and 5.5a, the signals *look* noisy, especially around 3 and 27 s. This occurs since the model has a sample time of 0.02 s and that the trajectory consists of discrete reference points. Hence, the distance from the vehicle to the closest point at one sample time can differ slightly compared to the distance at the next sample time.

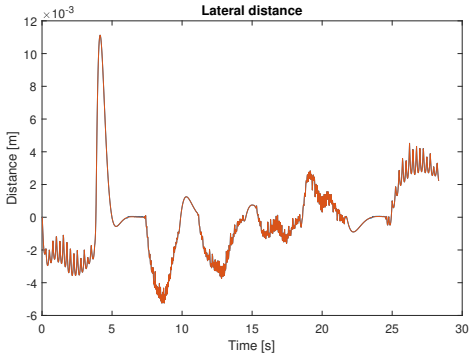


(a) Lateral distance from the AGV to the trajectory.

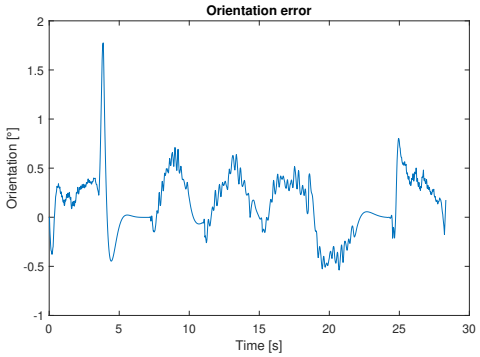


(b) Orientation error compared to the reference in the trajectory.

Figure 5.3: Results from simulation when the PID controller is used without delay.

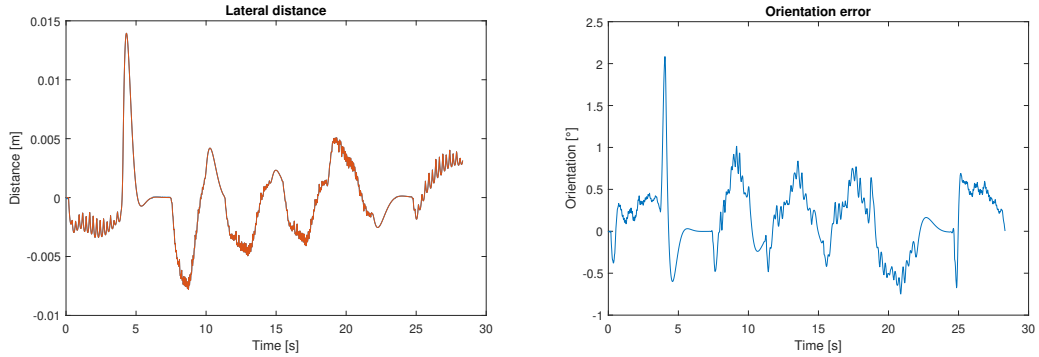


(a) Lateral distance from the AGV to the trajectory.



(b) Orientation error compared to the reference in the trajectory.

Figure 5.4: Results from simulation when the LQ controller is used without delay.



(a) Lateral distance from the AGV to the trajectory.

(b) Orientation error compared to the reference in the trajectory.

Figure 5.5: Results from simulation when the LQ controller is used with a delay of 150 ms.

5.2 PID Controller

In this section, results from driving when the AGV is unloaded and controlled by the PID controller are presented. Figure 5.6 illustrates the trajectory, the AGV's unfiltered and filtered route, and the boundaries that the AGV must be inside, according to TMH's requirements. The AGV exceeds the boundaries at the zoomed-in part of the trajectory, and the filtered path is smooth compared to the unfiltered one.

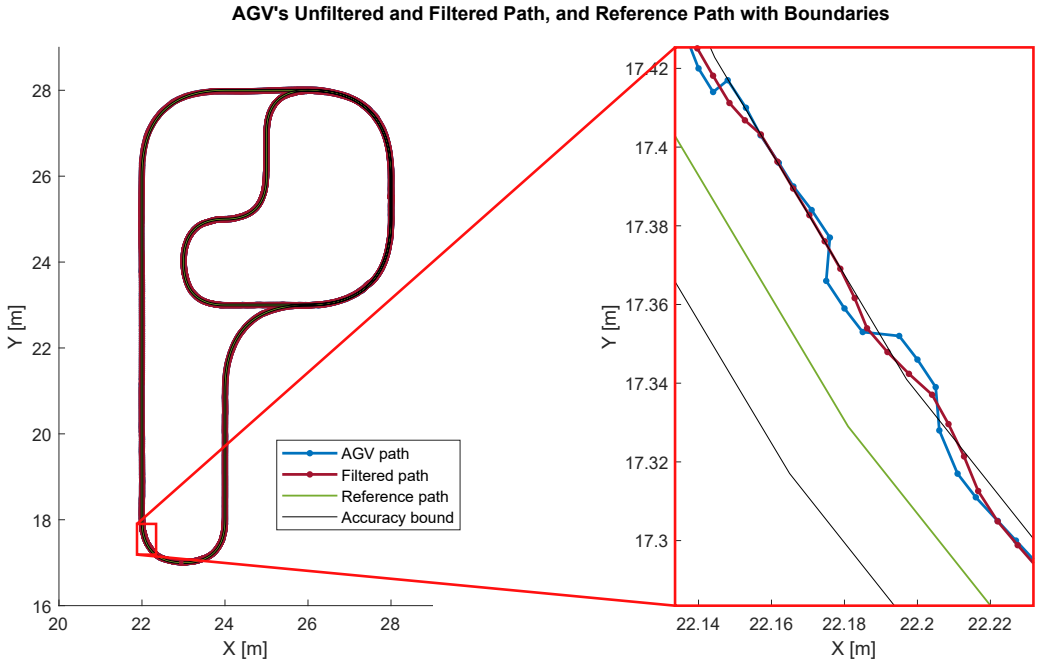


Figure 5.6: Trajectory tracking results in the xy-plane for the PID controller. The right figure is a zoomed-in part of the left figure's bottom-left corner.

The lowpass filtering of the AGV's orientation is shown in Figure 5.7. The filter smooths the high-frequency noise for the orientation in the same way it did for the position. Comparing the raw data of the orientation and the filtered orientation shows that the filtering does not add much phase shift in the system. The increase in delay is around 40 ms for the types of trajectories considered here, which is added to the existing delay of 150 ms.

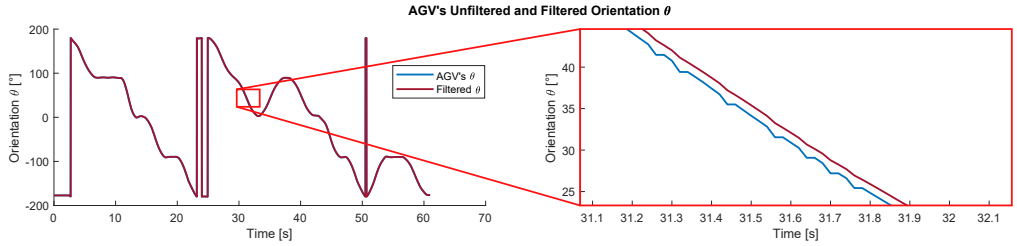


Figure 5.7: The orientation data before and after lowpass filtering in the PID controller. The right figure is a zoomed-in part of the left figure.

The reference angular velocity, calculated using the information in the trajectory, and the desired angular velocity (control signal), calculated using e_2 and e_3, are shown in Figure 5.8. The desired angular velocity is converted to velocities on the left and right wheel of the AGV, and these velocities are shown in Figure 5.9.

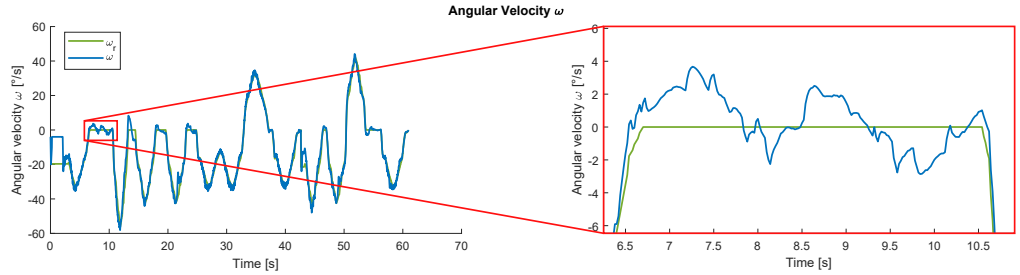


Figure 5.8: Reference and desired angular velocities, ω_r and ω when using the PID controller. The right figure is a zoomed-in part of the left figure.

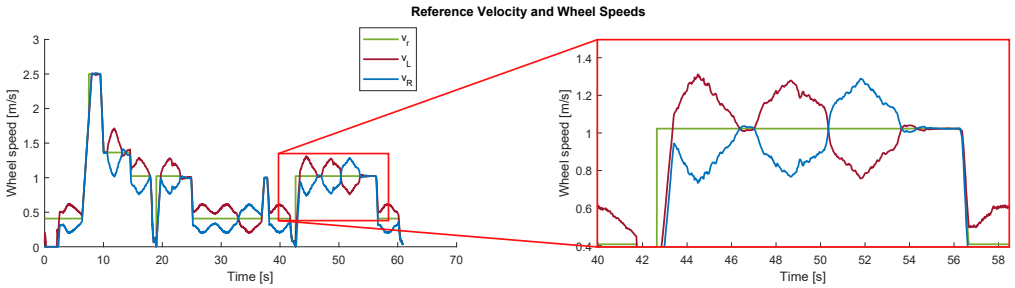


Figure 5.9: The desired velocities on the left and right wheel, v_L and v_R calculated by the PID controller and the reference velocity, v_r given by the trajectory. The right figure is a zoomed-in part of the left figure.

As described in Section 3.5.3, the future position and orientation of the AGV are predicted using the kinematic model of the AGV. In Figure 5.10, at the last point on the path where the AGV has driven, a position is predicted in front of it. It can also be seen that the AGV has been driving where the system in previous iterations has predicted a future position. This indicates that the prediction works as intended. The peak in the figure occurs due to poor accuracy in the positioning.

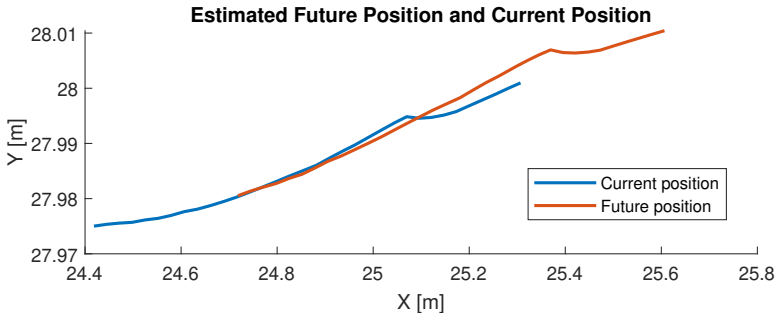


Figure 5.10: Result of the predicted future position. The AGV has been driving from left to right using the PID controller and has estimated future positions.

Figure 5.11 illustrates how much all parts in the two PID controllers contribute to the control signal ω_{fb}. It is also shown how the lateral error changes and the sum of it. The difference in the lateral error capture measurement noise, despite lowpass filtering the signals. Hence, the D-part in the controller did not improve the performance and was turned off.

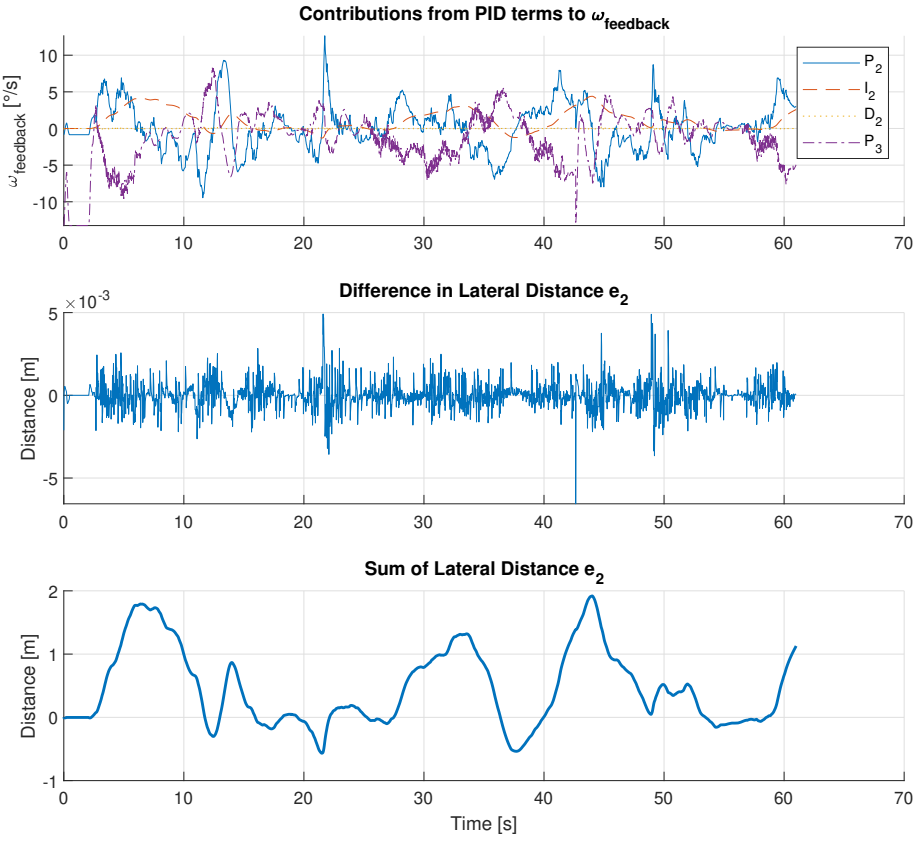


Figure 5.11: The data used in the PID controller. The top graph includes all parts used in the PID controller, the middle graph displays change in e_2 used in the D-part, and the bottom part displays the sum of e_2 used in the I-part.

The time to calculate control signals in every iteration, the cycle time, is illustrated in Figure 5.12. The time to modify the trajectory and calculate control signals are included in the total time shown. However, the treatment of the trajectory only occurs when new trajectory segments are received, and it is these calculations that take most of the time, which gives the high peaks in the graph. Hence, the calculation time for the PID controller is the bold bottom line and is around 250 μs.

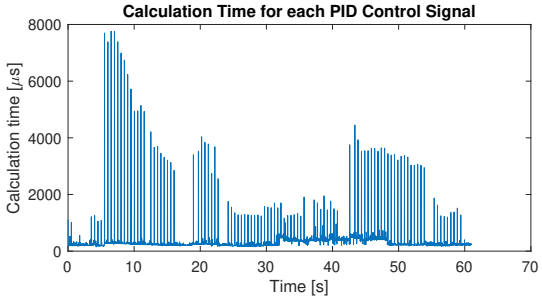


Figure 5.12: Calculation time for the PID controller. The calculation time consists of the time to calculate control signals and to modify the trajectory.

5.3 LQ Controller

In this section, results from driving when the AGV is unloaded and controlled by the LQ controller are presented. Figure 5.13 illustrates the trajectory, the AGV's route, and the boundaries that the AGV must be inside.

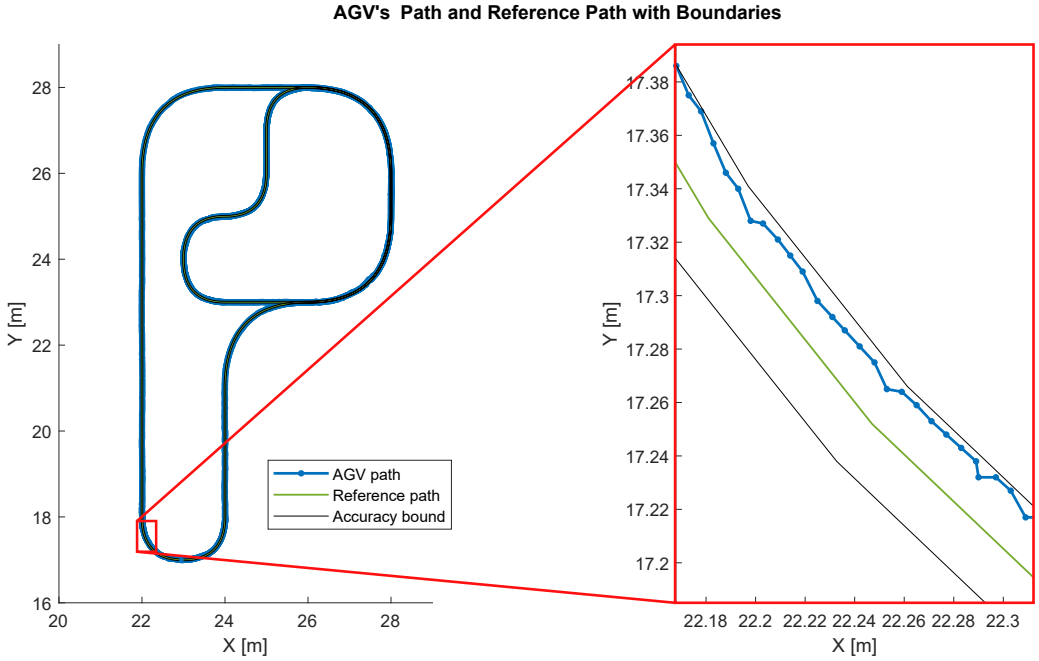


Figure 5.13: Trajectory tracking results in the xy-plane for the LQ controller. The right figure is a zoomed-in part of the left figure's bottom-left corner.

The desired angular velocity and the velocities on the left and right wheel of the AGV are shown in Figure 5.14 and 5.15. As shown, the control signals are relatively noisy.

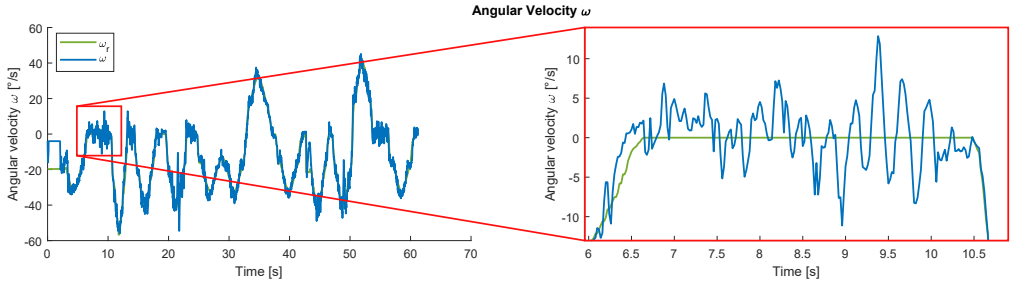


Figure 5.14: Reference and desired angular velocity, ω_r and ω when using the LQ controller. The right figure is a zoomed-in part of the left figure.

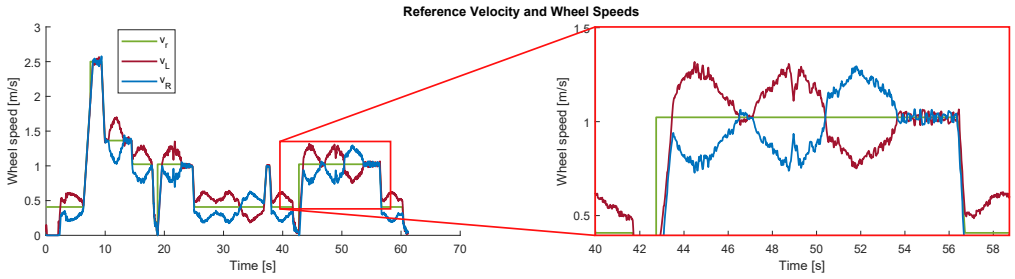


Figure 5.15: The desired velocities on the left and right wheel, v_L and v_R, calculated by the LQ controller and the reference velocity, v_r, given by the trajectory. The right figure is a zoomed-in part of the left figure.

The time to calculate control signals in every iteration is illustrated in Figure 5.12. The calculation time for determining control signals is around 2100 μs.

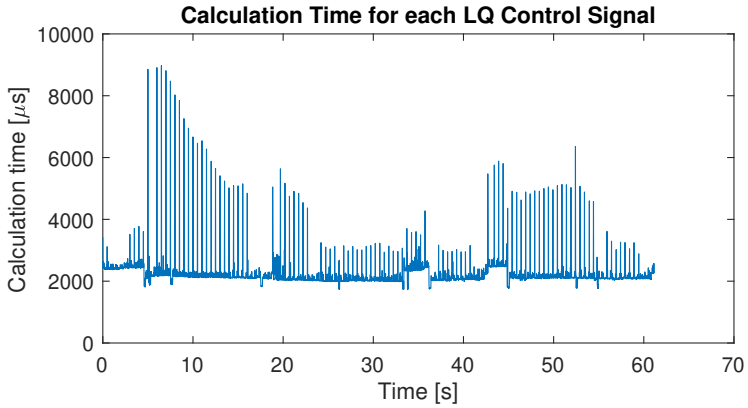


Figure 5.16: The time to calculate control signals for one iteration using the LQ controller.

5.4 Comparisons

Comparisons have been made between all controllers. First, the new controllers are compared by looking at general controller properties. Then, all three controllers are compared by testing them with different load distributions on the AGV and finally by their stopping accuracy.

5.4.1 General Controller Comparison

In this section, the PID and LQ controllers are compared by looking at their characteristics. By starting with the signals sent to the motor controllers, v_L and v_R, that are illustrated in Figure 5.9 for the PID controller and in Figure 5.15 for the LQ controller, the signals calculated by the LQ controller are noisier than the control signals calculated by the PID controller. This is partly because the position and orientation, used as inputs, are lowpass filtered in the PID controller, and hence most of the noise in the measurements is filtered out. The performance in the LQ controller was decreased when lowpass filtering the measurement signals. Both the lateral error and the orientation error became larger, which is why no filtering occurs in this controller. Studying the control signal ω in Figure 5.8 and 5.14, calculated by the two controllers, the noise is even more distinct in the LQ controller. The feedback part of the control signal ω calculated in the LQ controller switches sign very often. A combination of aggressive tuning and noise can be a contributing factor to this. For example, assume that the AGV is 1 cm to the right from the path with the correct orientation. If the controller is too hard tuned, the feedback part of ω will be very high ($13°/s$ as in Figure 5.14 at 9.4 s), causing the vehicle to rotate heavily to the left. This leads to a large orientation error after a few iterations, and the feedback part of ω will eventually become negative to compensate for this error. The control signals calculated by the PID controller is much smoother, which can lead to smoother trajectory tracking. However, it must be noted that a jerky behavior cannot be seen on the AGV when the LQ controller is running, although the control signals look noisy. The lowpass filtering in the motor controller might be an explanation for this.

The calculation times, illustrated in Figures 5.12 and 5.16, differ for the controllers. The calculation time is approximately 8 times higher for the LQ controller since it solves a Riccati equation iteratively in every iteration. The benefit of this is that many controller gains are used that are applicable for a large variety of operating points. Since the determination of control signals is small compared to the trajectory modification and the calculation time is shorter than the cycle time of 20 ms, the increased calculation time will not affect this application.

5.4.2 Load Distribution

Results and a discussion about tests with different load distributions are presented in this section. Four types of scenarios have been tested: Without weight, the weight placed in the center, the weight placed in the front-right corner, and the weight placed in the back-left corner of the AGV, illustrated in Figure 4.3. The lateral distances from the AGV to the trajectory, for all tests, are shown in Figure 5.17 and 5.18.

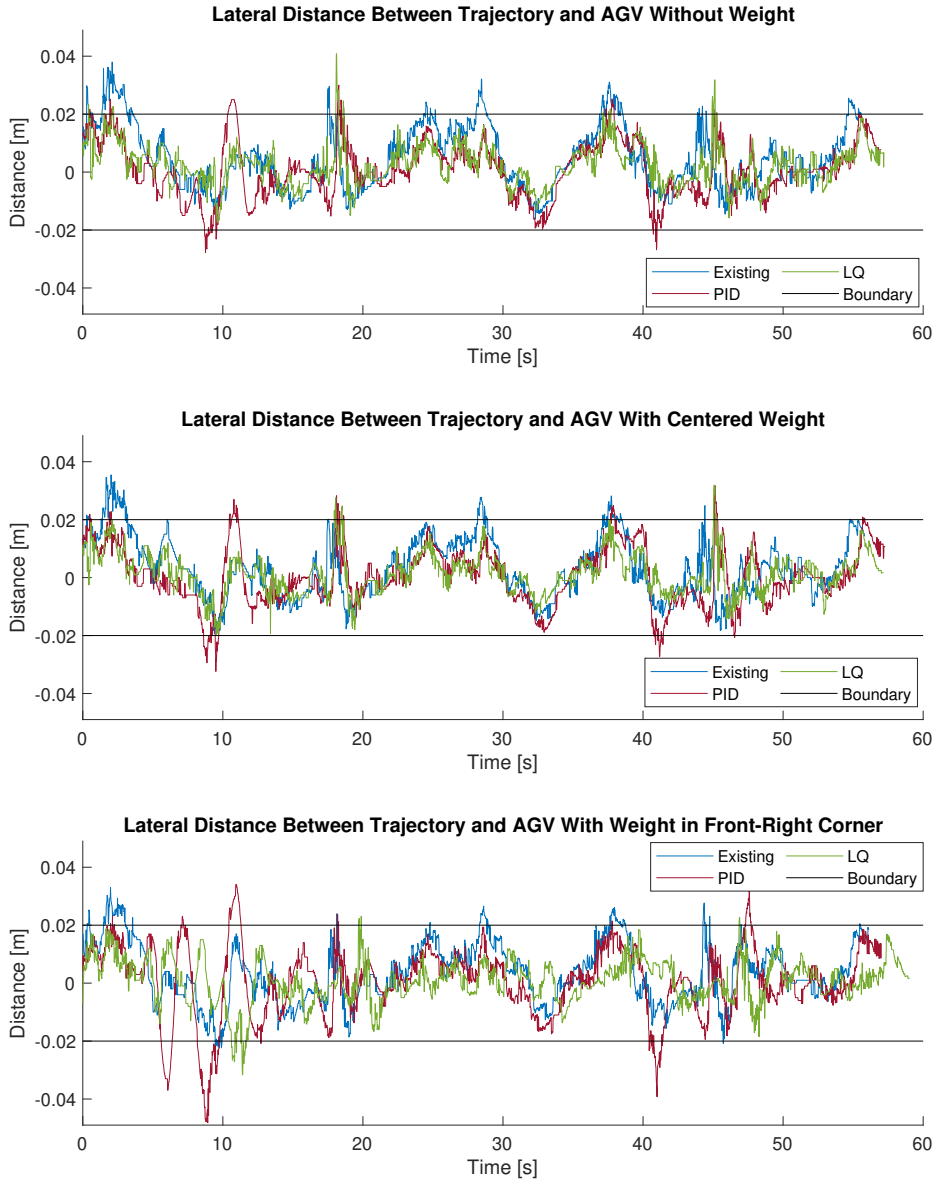


Figure 5.17: The lateral distance between the AGV and the trajectory. Results for all three controllers are displayed for different weight placements. First without any weight, then with weight in the center, and finally in the front-right corner of the AGV.

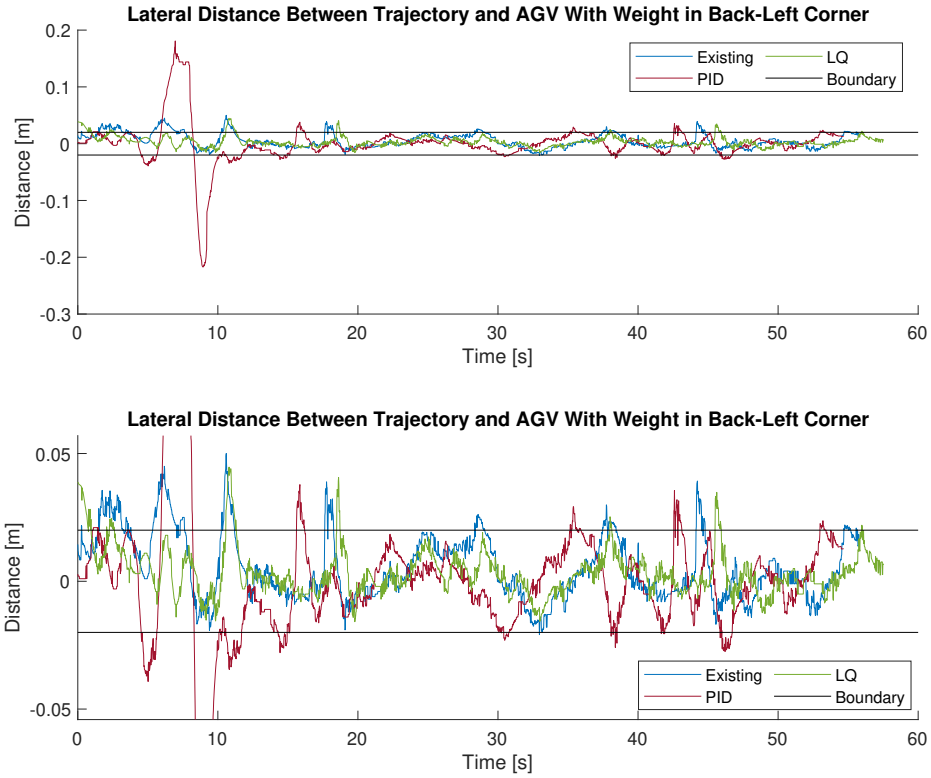


Figure 5.18: The lateral distance between the AGV and the trajectory. Results for all three controllers are displayed with the weight placed in the back-left corner of the AGV. The bottom figure is a zoomed-in version of the top figure.

It can be seen that the LQ controller makes the AGV drive inside the boundaries except for a very short period of times when the AGV is unloaded and when the weight is placed in the middle and the front-right corner. This is seen as peaks in the graph. Neglecting these peaks, the LQ controller will keep the AGV inside the boundaries. The PID controller does not keep the AGV inside the boundaries as good as the LQ controller. It can also be seen that the AGV will oscillate in the lateral distance when using the PID and LQ controller when the weight is placed in the front-right and back-left corner. These oscillations do not occur when the AGV uses the existing controller. It should also be noted that the AGV barely was able to follow the trajectory with the PID controller when the weight was placed in the back-left corner.

If the absolute value of all the lateral distances to the trajectory in every sampling point is summed, the total Euclidean distance can be obtained. A value of this for all controllers and all types of missions is presented in Table 5.4. None of the controllers is greatly affected by a load in the middle of the AGV compared to when it is unloaded. The LQ and the existing controller perform slightly better with a load mounted, while the PID controller performs slightly worse. If the weight is placed in the front-right corner, there is a noticeable decrease in performance for the PID controller. In contrast, the LQ and the existing controller are hardly affected by the change in load distribution regarding lateral distance to the trajectory. Placing the weight at the back-left corner has a noticeable impact on all controllers. The LQ controller performs best, the existing controller performs second best, and the PID controller performs worst. However, the percentage increase differs between the controllers. The LQ controller has an increase in total Euclidean distance of 19%, while the existing controller only increases 12%. The PID controller has an increase of 137%. Hence, the existing controller is more robust than the new controllers, but the overall performance of the LQ controller, in

these test cases, is better than the existing one. If the AGV is loaded with a heavier weight, it is possible that the existing controller will perform better in some cases due to its robustness.

Table 5.2: Total Euclidean distance during driving for different weight placements.

Controller	Without weight [m]	Weight in center [m]	Weight in front-right [m]	Weight in back-left [m]
Existing	25.7	24.2	24.4	28.9
PID	21.4	22.5	26.4	50.9
LQ	16.2	15.7	16.6	19.3

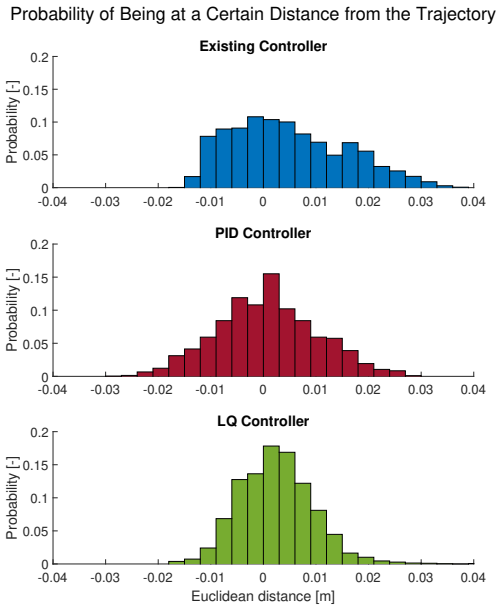
The probability of being at a certain distance from the trajectory is shown in Figure 5.19. Comparing Figure 5.19a, b, c, and d, shows that the AGV will have the highest probability to be close to the path when the LQ controller controls it. In all cases, the probability graphs for the LQ controller are concentrated around zero. It should also be noted that the bars in the PID case continue all the way to -0.2 and 0.2 m when the weight is placed in the back-left corner, which is not seen in the figure. Furthermore, none of the controllers meets the requirement that the Euclidean distance from the path must be smaller than 2 cm. However, the requirement that the maximum orientation error is 4° is met in all cases for the existing controller and in all cases except the case with the weight in the back-left for the LQ and PID controller.

The orientation errors for all tests are shown in Figure 5.20. It can be seen that when the AGV is unloaded, the PID and the LQ controllers barely make the AGV be inside the orientation boundaries at $-4°$and 4°. They are close to the boundaries at two points, and generally, the LQ controller oscillates more than the others. The existing controller is not even close to the orientation boundaries when the AGV is unloaded. There is no big difference between the results when the AGV is unloaded or when the weight is placed in the middle. However, when the weight is placed in the front right corner, the orientation error oscillates more, and the peaks are higher for the PID and LQ controllers. The existing controller is not greatly affected. It should be noted that all controllers can still keep the orientation error inside the boundaries. When the weight is placed in the back-left, none of the controllers achieves the requirements, and the PID controller performs poorly.

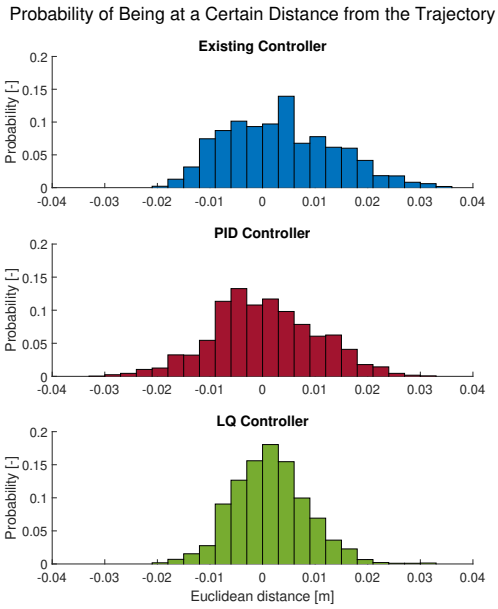
The absolute orientation errors in every sampling point are summed, which gives the total absolute orientation error for all controllers. A value of this for all controllers and all types of missions is presented in Table 5.3. The existing controller performs much better than the new controllers in all cases. The difference is greatest when the weight is placed at the back-left on the AGV. The existing controller is hardly affected by the load distribution since the increase in orientation error from when the AGV is unloaded compared to when it has the weight placed in the back-left is only 7%. The increase for the PID and LQ controller is 66% and 23%, respectively. This also confirms the robustness of the existing controller, as stated above.

Table 5.3: Total absolute orientation error during driving for different weight placements.

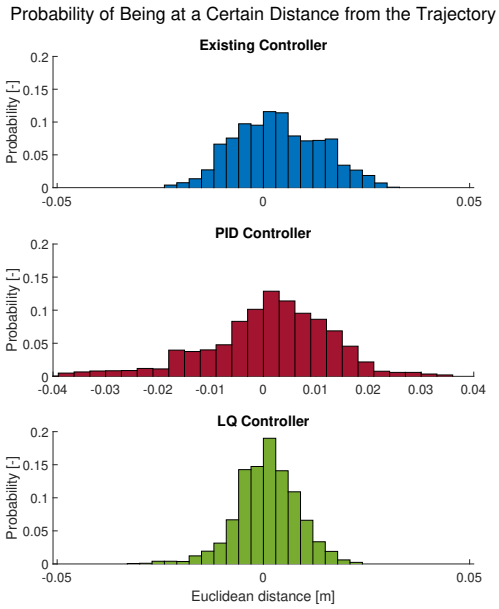
Controller	Without weight [°]	Weight in center [°]	Weight in front-right [°]	Weight in back-left [°]
Existing	1966	1915	2072	2108
PID	2640	2665	3146	4386
LQ	2543	2504	2942	3127



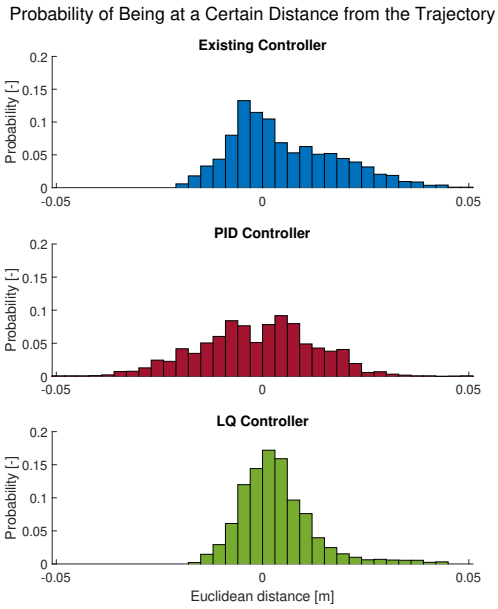
(a) Mission without any weight.



(b) Mission with weight in the center.



(c) Mission with weight in the front-right corner.



(d) Mission with weight in the back-left corner.

Figure 5.19: The probability of being at a certain lateral distance from the trajectory during the drive for all controllers with different weight placements.

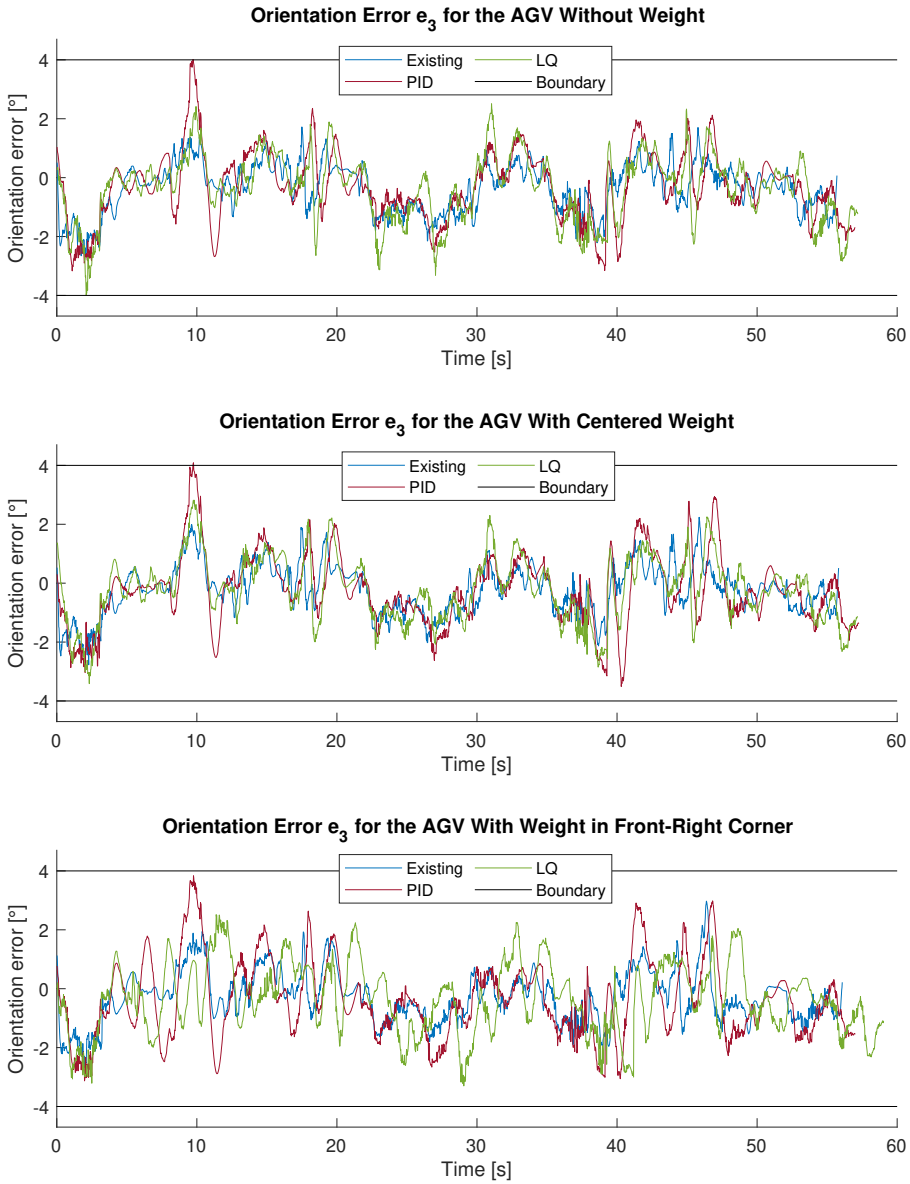


Figure 5.20: The orientation error for different weight placements. Results for all three controllers are displayed, first without any weight, then with weight in the center, and finally in the front-right corner of the AGV

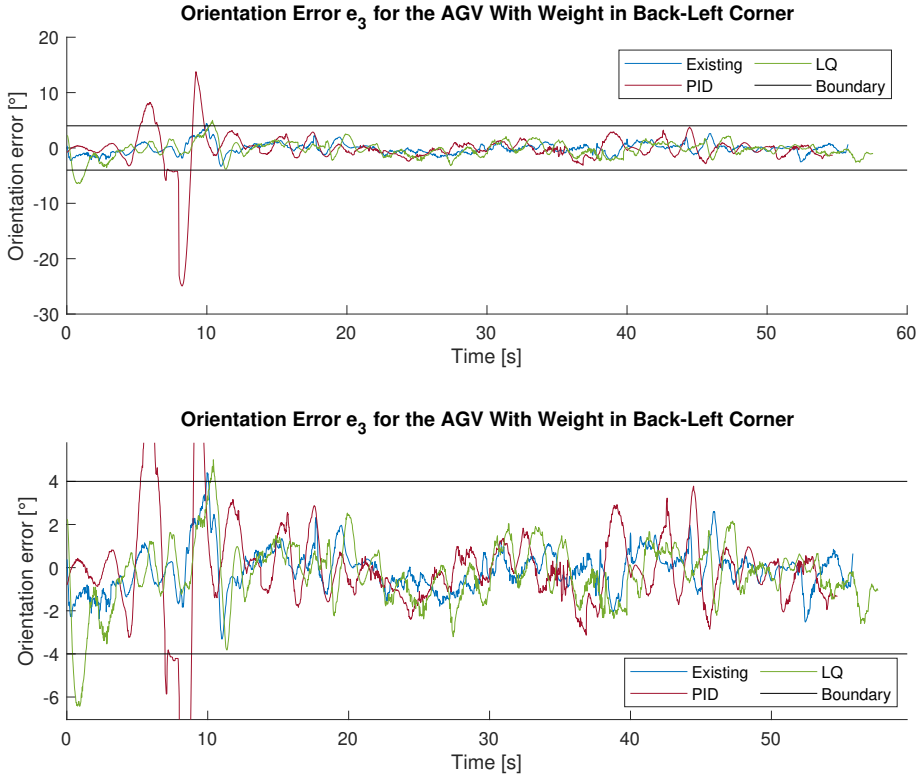


Figure 5.21: The orientation error with the weight placed in the back-left corner of the AGV. Results for all three controllers are displayed. The bottom figure is a zoomed-in version of the top figure.

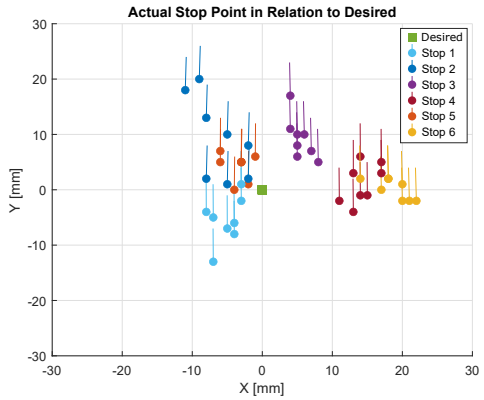
5.4.3 Stopping Accuracy

Results from the eight stopping accuracy tests for each stopping point, described in Table 4.1, are presented in Figure 5.22. Illustrations of the actual stopping points of the AGV relative to the desired stop are shown in Figure 5.22a, c, and e for the existing, PID, and LQ controller, respectively. Note that there is an inaccuracy in both the x- and y-axis of 5 mm. Further, the corresponding total Euclidean distances are shown in Figure 5.22b, d, and f. A summary in the form of the average distance between the actual stop and the desired stop for each point and the total average distance is presented in Table 5.4.

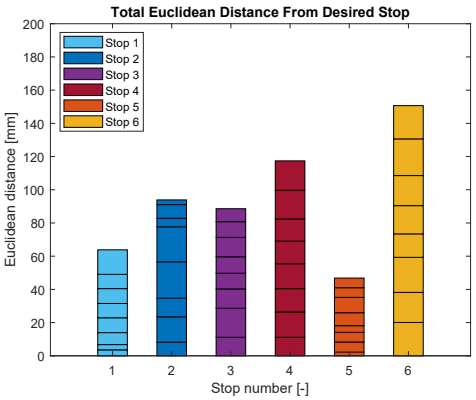
Table 5.4: Average Euclidean distance for each stopping point separately, as well as the total average for each controller. The units are millimeters.

Controller	Stop 1	Stop 2	Stop 3	Stop 4	Stop 5	Stop 6	Total
Existing	7.9796	11.7364	11.0736	14.6758	5.855	18.8342	11.692
PID	8.5117	24.7703	5.8322	10.3428	8.6392	11.1714	11.545
LQ	15.7641	18.2240	12.9569	7.7247	7.8036	11.1398	12.269

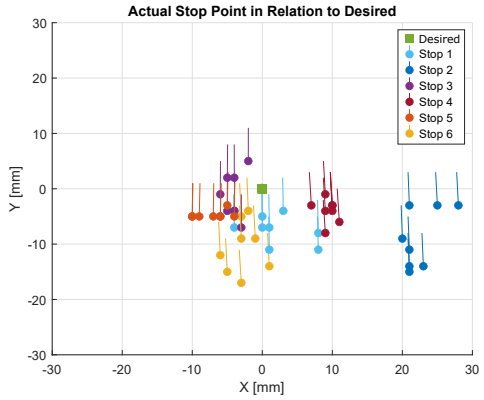
In Figure 5.23, the sum of all orientation errors at each point for the AGV is illustrated. The absolute average error for each stop and the total error for each controller are presented in Table 5.5.



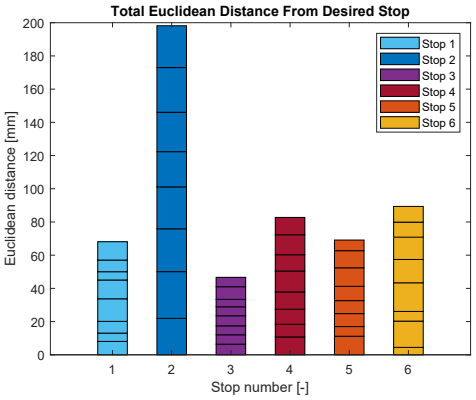
(a) Stopping result for the existing controller.



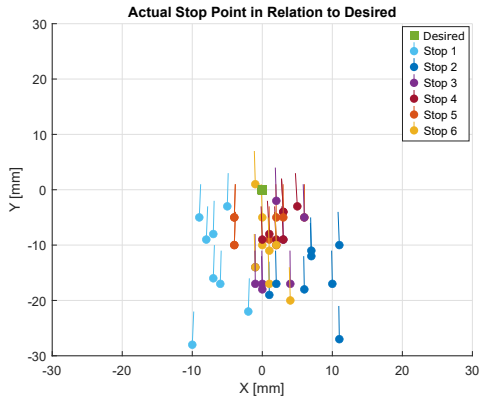
(b) Total Euclidean distance with the existing controller.



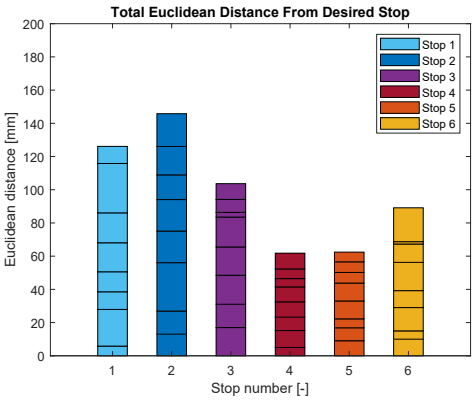
(c) Stopping result for the PID controller.



(d) Total Euclidean distance with the PID controller.



(e) Stopping result for the LQ controller.



(f) Total Euclidean distance with the LQ controller.

Figure 5.22: The stopping results along with the total Euclidean distance for all three controllers. A stopping point is illustrated in one color and appears eight times, once for each test. Consequently, every bar in the figures to the right is divided into eight sections. The straight lines in the left figures symbolize the orientation of the AGV at the stop.

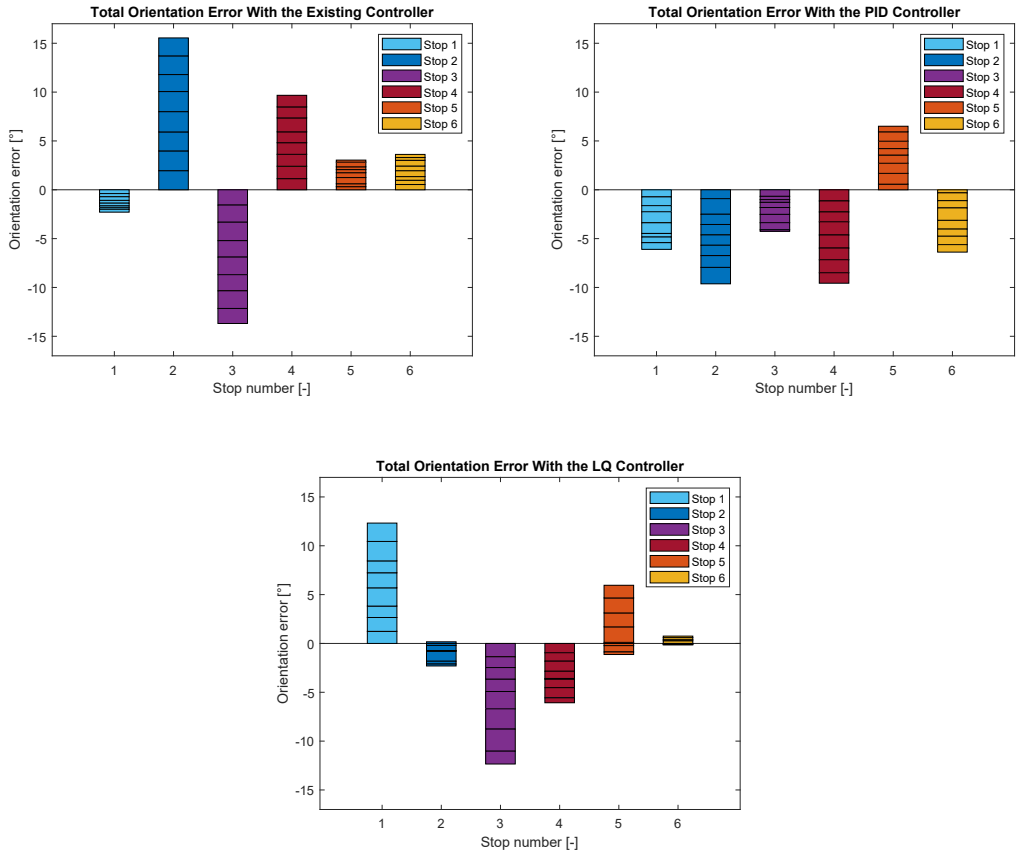


Figure 5.23: The sum of the orientation error for each stop and controller. The AGV is oriented to the right if the orientation error is positive and vice versa.

Table 5.5: Average absolute orientation error for each stopping point separately and the total average for each controller. The units are degrees.

Controller	Stop 1	Stop 2	Stop 3	Stop 4	Stop 5	Stop 6	Total
Existing	0.2863	1.943	1.711	1.209	0.3799	0.4525	0.997
PID	0.7624	1.2036	0.5318	1.1959	0.8132	0.7977	0.884
LQ	1.5404	0.3094	1.5430	0.7592	0.8863	0.1126	0.858

Below, the results are discussed for each stopping point. All three controllers will be evaluated and compared. The PID , the LQ , and the existing controllers are denoted F_{PID}, F_{LQ}, and F_E, respectively.

Stop 1

Comparing the three controllers' performance at stopping point 1 while the AGV moves at a high speed of $v = 2.5$ m/s on a straight path, it is clear that all controllers will make it stop too early. F_E tends to stop slightly to the left and F_{LQ} even more so, while F_{PID} stays closer to the path. However, the average Euclidean distance is smallest for F_E tightly followed by F_{PID}. The worst is F_{LQ} that stands out as it stops earlier than the other controllers. Furthermore, the stopping points for each test for ‹are not as close to each other compared to the other controllers.

The same ranking order is reflected when comparing the average orientation error. F_E performs best, followed by F_{PID}, and last F_{LQ}. In addition, the two first mentioned will make the AGV face to its left, which makes sense for F_{PID} as the AGV should move to the left to get closer to the stopping point. However, for F_E, the AGV should instead face right to come closer to the stop. F_{LQ} has the correct orientation to move closer to the desired stop.

Stop 2

At stopping point 2, the fast right curve with a 2 m radius, F_E makes the AGV stop too late, while the new controllers make it stop too early. Further, F_E will stop on the left side of the desired stopping point, and the new controllers will make it stop to the right. Unlike the last stop, all controllers will have somewhat more scattered stops. The average Euclidean distance will be far best for F_E, followed by F_{LQ} and lastly F_{PID}.

While comparing the orientation error, it is clear that F_{LQ} performs much better than the other two. This implies that it has prioritized the orientation more than the distance to the path compared to the other controllers. The second best is F_{PID} followed by F_E in the last place. Moreover, they are all facing in the direction to get closer to the stop in the end.

Stop 3

While moving at a speed of $v \approx 1$ m/s and performing an s-shaped curve, F_E stops a bit too late to the right, F_{PID} stops fairly close in the y-direction and a tiny bit to the right, while F_{LQ} stops early but roughly on the trajectory. The average Euclidean distance is similar for F_E and F_{LQ}, but F_{PID} outperforms the other two.

Evaluating the average orientation error gives the same result. F_{PID} controller is much better than F_E and F_{LQ}, which is fairly similar. Further, it is worth noticing that F_{LQ} is facing left even though it is on track.

Stop 4

When making the opposite s-shaped curve, compared to the previous point, with a smaller radius and lower velocity, all the controllers will stop to the right of the desired point. F_E is furthest right but stops at approximately the correct longitudinal position, and F_{PID} does not stop as far right, but a bit too early. F_{LQ} controller stops even better regarding the left-right plane but stops the earliest of them all. This concludes in an average Euclidean distance rating, where F_{LQ} performs best, then F_{PID}, and lastly F_E.

Regarding the average orientation error, F_{LQ} controller performs the best. F_{PID} and F_E are approximately the same in the absolute orientation error presented in the table. However, F_E is facing right simultaneously as it stands on the right side of the stopping point. This is not the case for F_{PID}.

Stop 5

Stopping point 5 consists of a straight path with the slowest velocity of all tests at $v = 0.1$ m/s. This seems to yield the best overall results for the three controllers. The same pattern as seen before is repeated here. F_E stops a bit too late, F_{PID} and F_{LQ} a bit early. Moreover, the first two stops on the left side while the last one stops closer to the trajectory. The average Euclidean distance is best for F_E, followed by F_{LQ} and lastly F_{PID}.

Looking at the orientation error, F_E keeps the top position for this stop, followed by F_{PID}, which is marginally better than F_{LQ}. All three controllers face to the left, which makes sense regarding where they have stopped.

Stop 6

At the 180° right curve with velocity $v = 0.37$ m/s, stop 6, F_E stops to the right, F_{PID} stops a bit to the left and F_{LQ} stops on the trajectory. Seen longitudinally, F_E stops well while the other two stops too early but approximately the same. The average Euclidean distance shows that the last two have the shortest distance, despite stopping too early, while the existing controller stops further away.

F_{LQ} has the best average orientation error, followed by F_E and lastly F_{PID}.

Discussion

The results from the stopping accuracy tests vary depending on which type of stop is considered. One controller is not superior since they perform better or worse in different scenarios. The total average Euclidean distance for all stops together is quite similar. The same conclusion can be made regarding the total average absolute orientation error.

F_E has the second-best overall Euclidean distance. It often over-shoots the stopping point and has more spread in the left-right plane for some stops. It seems that F_E performs worse in curves with a small radius. Moreover, it performs worst when evaluating the orientation error. It is even wrongly oriented in two stopping points. However, it is the best controller regarding straight paths.

F_{PID} is marginally the best when looking at Euclidean distance. It rarely over-shoots the stopping point, stops quite close longitudinally, but differs a bit in the left-right plane. The controller stops very consistently, except for stopping point 2, where it has the worst performance of all stops for all controllers. If tests from stopping point 2 are neglected, F_{PID} is undoubtedly the best regarding Euclidean distance. Further, it is the second-best regarding the orientation error. The performance can be improved by increasing the P-part in the controller with e_3 as input, alternatively using the D-part in the PID controller with e_2 as input. However, as stated in Section 5.2, the use of this will add noise in the controller.

F_{LQ} controller has the highest total Euclidean distance since it often stops too early. Additionally, the stopping point for the same stop type is more scattered than for the other two controllers. On the other hand, it stops on the trajectory and not to the right or the left as often. F_{LQ} is the best regarding the orientation error. It performs well except when driving straight followed by a stop, compared to the other controllers.

Comparing the longitudinal distance to the stopping point, both F_{PID} and F_{LQ} tend to stop too early, where F_{LQ} is the worst. This can be an indication that the size of the delay used in the controllers is too large. For instance, if the controller predicts that the next reference point is the last one, the desired velocity will be zero. But in reality, the AGV will not reach that point since the predicted state was too far ahead and therefore stops too early. Hence, it is possible that the value of the delay used in the controllers must be decreased. The performance can also be improved by using e_1 for the controller to be more aware of how close the AGV is to the final reference point.

5.5 Method Evaluation

The first work after the literature study consisted of building a model and implementing controllers in Simulink. This allowed us to gain knowledge about what characterizes a differential drive vehicle. For example, a PID controller was not as easy to implement as first thought, due to that the AGV is a MIMO-system. The simulation environment allowed us to experiment and learn how to implement this type of controller. It was also a huge advantage to implement the system that would predict a future state in the simulation since it was easy to evaluate and validate the result. Furthermore, the simulation environment allowed us to understand how the modification of the trajectory could be done. For example, it could be seen if interpolation of the trajectory points was sufficient or if a spline function based on all points would be necessary for the performance. The calculation of orientation and angular velocity in all points could also be evaluated easily. Hence, the implementation in Simulink made us well prepared for the real implementation on the AGV. The work with the simulation took a large amount of time, resulting in that the real implementation on the system was not started as early as it could have been. However, we were not dependent on being in the lab at this point, which was beneficial since it had high demand from other workers.

The implementation on the real system differed in many ways compared to simulations. In the simulation, the model consisted of blocks with MATLAB code, while the real implementation required different classes in C++ to communicate with each other and the existing systems on the AGV. However, because we had gained knowledge about the systems that would form our controllers, the implementation progressed well. Hence, the time spent on simulation together with implementation on the real system might have been shorter than if we would have started implementing on the real system immediately.

The tests to evaluate the controllers and compare them to each other could be divided into three parts: Trajectory tracking, load distribution, and stopping accuracy. These tests allowed us to compare calculation times and differences in the calculated control signals. The robustness of the controllers could be compared by varying the load distribution between missions. The stopping accuracy tests indicated how well the vehicle could stop, which is crucial when the AGV should perform loading, unloading, or charging. From these tests, it became clear in which situations the controllers had a bad and good performance and their overall robustness.

Furthermore, in the tests regarding stopping accuracy, six types of stopping points were investigated with different velocities. If the tests had been performed again, it might have been more suitable to use the same stopping points, but where the velocity of the trajectory leading to the points would have been the same for all points. By neglecting one variable, it would have been clearer how the shape of the trajectory leading to the stopping points affects the controllers.

6 Conclusion

In this thesis, a PID and an LQ controller have been implemented for trajectory following on a differential drive vehicle, the AGV. A vehicle model was implemented in a simulation environment together with the controllers before they were implemented on the real system. The controllers have been compared to an existing controller on the AGV regarding performance and robustness, and conclusions about this are presented in this chapter.

6.1 Conclusion

The purpose of the thesis was to investigate the performance of our implemented controllers and reach a conclusion on whether an in-house control solution is suitable for TMH or not.

Both a PID and an LQ controller can be used to control a differential drive vehicle. The LQ controller performs better than the PID controller in almost all situations and is also more robust. The only situation where the PID controller performs better is at the stops, where the longitudinal error to the stopping point is lower. The PID controller does, however, require less computational time to calculate control signals. If this is of great importance, the PID controller might be a better choice.

The LQ controller is more complex than an ordinary PID controller. But since the AGV is a MIMO-system, implementing a PID controller is not straightforward, nor is the tuning. Therefore, the general implementation is most likely easier with a state-space controller, such as LQ, rather than a PID controller that is more suitable for SISO-systems. To conclude, the LQ controller is a more suitable control strategy than the PID due to the performance, robustness, and ease of tuning.

The performance of the LQ controller is better than the existing controller regarding distance to the trajectory. However, the existing controller keeps the orientation error smaller than the LQ controller can and is also more robust. Regarding stopping, the PID controller performs best when looking at Euclidean distance and orientation error combined. The PID controller is, however, not as robust as the other two controllers. In the controllers' current states, the conclusion is that the existing controller is the best to use since its robustness, combined with its overall performance, stands out compared to the other controllers.

There are both benefits and drawbacks to developing and using an in-house control system. One of the benefits is that TMH has knowledge of the system and has great competence in how the control system can be modified if the system is changed. For example, if the AGV is fitted with a hefty device, a new tuning of the controller might be needed. If knowledge about the controller exists, the tuning will likely be easier and better. Another benefit is that if a system on the AGV is changed, that requires changes in other systems, TMH does not have to involve another company to make changes in the systems. Instead, they have the competence inside the company about how the control system works and how the other systems work that it needs to communicate with.

A drawback of creating an in-house application is that it requires development costs. Another drawback is that there must always exist employees at TMH that know the control system if something must be changed. This can be both costly and a risk. Further, the existing controller is developed by a company that is also responsible for the positioning system. The existing controller and the positioning system on the AGV work as one system, and the position of the AGV is available to the controller as soon as it has been determined. This means that developing an in-house control system that is not a part of the system that determines the position results in an extra delay of 40 ms. Based on the benefits and drawbacks, a recommendation to TMH is to start working on a control system since it is important to have as much knowledge as possible about their own systems.

6.2 Future Work

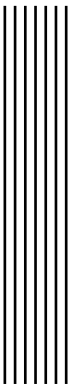
Future work can be done on the implemented controllers. Starting with the PID controller, more gains for different operating regions can be added. This can make the controller perform better in certain areas. It can also be relevant to investigate if the D-part in the controller can be used. This requires that most of the noise in the measurements is removed. The stopping accuracy for both controllers can be improved. This can, for example, be done by using the longitudinal error, e_1, to check the distance to the stopping point. It is also possible to change the value of the delay in the controllers to achieve better stopping accuracy. This can also lead to better overall performance during missions. Furthermore, the controllers can be integrated better with the other systems by fault handling.

If there were more time, it could be suitable to perform tests that covered more scenarios. For example, when performing load distribution tests, the weight itself could differ in mass and could also have been placed in the back-right corner and the front-left corner. Furthermore, tests using opposite s-curves compared to stop 3 and 4 but with the same radius would be interesting to study. However, many scenarios have been covered, and the collected data is sufficient to make conclusions about the controllers' performance.

It is also possible to investigate if something can be improved with the trajectory tracking in the simulation since the signals look noisy. It might be possible that a spline of the x- and y-coordinates can fix this problem.

6.3 Sustainability

The results from the thesis can be valuable for TMH if they wish to improve their AGV. The use of AGVs in baggage logistics areas at airports can lead to a reduced need for traditional logistic solutions. This can, for example, be massive systems of conveyor belts that require large facilities consuming unnecessary energy. Hence, replacing these systems with AGVs can lead to more space and energy-efficient baggage handling solutions. AGVs can also be used in other applications as well and thereby replace traditional and inefficient solutions.



Bibliography

[1] K. S. Park and K. T. Jung. "Considering performance shaping factors in situation-specific human error probabilities". In: *International Journal of Industrial Ergonomics* 18.4 (1996), pp. 325–331.

[2] S. Nurmaini, K. Dewi, and B. Tutuko. "Differential-Drive Mobile Robot Control Design based-on Linear Feedback Control Law". In: *IOP Conference Series: Materials Science and Engineering*. Vol. 190. 012001. IOP Publishing. 2017.

[3] G. Klancar, D. Matko, and S. Blazic. "Mobile robot control on a reference path". In: *Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation Intelligent Control, 2005.* IEEE. 2005, pp. 1343–1348.

[4] R. L. S. Sousa, M. D. do Nascimento, F. G. Nogueira, and B. C. Torrico. "Trajectory tracking control of a nonholonomic mobile robot with differential drive". In: *2016 IEEE Biennial Congress of Argentina (ARGENCON)*. IEEE. 2016, pp. 1–6.

[5] L. Eldén and L. Wittmeyer-Koch. "Numeriska beräkningar - analys och illustrationer med MATLAB". In: Studentlitteratur AB, 2001, p. 107. ISBN: 9789144020075.

[6] S. McKinley and M. Levine. "Cubic spline interpolation". In: *College of the Redwoods* 45.1 (1998), pp. 1049–1060.

[7] T. Glad and L. Ljung. *Reglerteknik: Grundläggande teori*. Studentlitteratur AB, Lund, 2016.

[8] M. Enqvist, T. Glad, S. Gunnarsson, P. Lindskog, L. Ljung, J. Löfberg, T. McKelvey, A. Stenman, and J.-E. Strömberg. *Industriell reglerteknik Kurskompendium*. Reglerteknik, Institutionen för systemteknik, Linköping University, 2014.

[9] T. Glad and L. Ljung. *Control Theory: Multivariable and Nonlinear Methods*. Taylor Francis Ltd, 2000.

[10] B. Anderson and J. B. Moore. *Optimal Control: Linear Quadratic Methods*. Dover Publications, 2007.

[11] M. Kalyoncu and F. Demirbaş. "Differential Drive Mobile Robot Trajectory Tracking With Using PID and Kinematic Based Backstepping Controller". In: *Selcuk University Journal of Engineering, Science and Technology* 5 (2017), pp. 1–15.

[12] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, 2010.

[13] I. Maurović, M. Baotić, and I. Petrović. "Explicit Model Predictive Control for trajectory tracking with mobile robots". In: *2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE. 2011, pp. 712–717.

[14] NASA. *Angular Displacement, Velocity, Acceleration*. Available at https://www.grc.nasa.gov/www/k-12/airplane/angdva.html (2021/02/21).

[15] F. Xie and R. Fierro. "First-state contractive model predictive control of nonholonomic mobile robots". In: *2008 American Control Conference*. IEEE. 2008, pp. 3494–3499.

[16] MIT. *Controllability*. Available at https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-30-feedback-control-systems-fall-2010/lecture-notes/. Oct. 2010.

[17] Circuit Globe. *Difference Between FIR Filter and IIR Filter*. URL: https://circuitglobe.com/difference-between-fir-filter-and-iir-filter.html (visited on 05/02/2021).

[18] G. Kistner. *Frame Rate-Independent Low-Pass Filter*. URL: http://phrogz.net/js/framerate-independent-low-pass-filter.html (visited on 05/03/2021).

[19] S. Omatu, T. Fujinaka, Y. Kishida, and M. Yoshioka. "Self-tuning neuro-PID for SIMO systems". In: *1999 European Control Conference (ECC)*. 1999, pp. 4324–4330.

[20] R. Vilanova and A. Visioli. *PID Control in the Third Millennium*. Springer, 2012.

[21] T. Hellstrom and O. Ringdahl. "Follow the Past: a path-tracking algorithm for autonomous vehicles". In: *International journal of vehicle autonomous systems* 4.2-4 (2006), pp. 216–224.

[22] A. R. Norman. "On the Discrete-Time Algebraic Riccati Equation and Its Solution in Closed-Form". In: *IFAC Proceedings Volumes* 44.1 (2011), pp. 162–167. URL: https://www.sciencedirect.com/science/article/pii/S1474667016436049.

[23] E. K.-W. Chu, H.-Y. Fan, W.-W. Lin, and C.-S. Wang. "Structure-Preserving Algorithms for Periodic Discrete-Time Algebraic Riccati Equations". In: *International Journal of Control* 77.8 (2004), pp. 767–788. URL: https://doi.org/10.1080/00207170410001714988.

[24] H. Zhang, L. Li, J. Xu, and M. Fu. "Linear Quadratic Regulation and Stabilization of Discrete-Time Systems With Delay and Multiplicative Noise". In: *IEEE Transactions on Automatic Control* 60.10 (2015), pp. 2599–2613.