Comparison of Control Architectures for Autonomous Mobile Robots

P.B. Verhoeckx
(0663727)

DC 2016.006

Master's thesis

Supervisor:	Prof. dr. ir. H. Nijmeijer
Coach:	dr. ir. D. Kostić
Client:	Van den Akker Engineering

Eindhoven University of Technology
Department of Mechanical Engineering
Dynamics and Control

Eindhoven, February, 2016

Abstract

The level of autonomy of robotic systems is increasing due to intelligent algorithms, improving hardware capabilities and new scientific insights. This can yield increasing robustness against hard- and software failures and may improve robot safety. However, when robots have to cooperate with each other, the level of autonomy of the individual robots (agents) may affect the efficiency of the complete robot system. The desired level of autonomy for cooperative mobile robots, operating in automated warehouses, is investigated in this thesis.

Thereto, the coordination control of individual robots is studied first. This includes generation of the reference trajectories, collision avoidance and motion control. A novel Free Range Routing algorithm with collision avoidance is developed and investigated. Together with the trajectory tracking controller this composes the low-level control of the investigated system.

To examine the effect of different levels of autonomy, the preferred communication topology is investigated and implemented. This enables robots to act based on information on the other robots. It is chosen to use a modular software framework with an accompanying inter-robot communication network.

Various high-level aided collision avoidance methods are introduced, examined and compared. Furthermore, this information exchange enables robot consensus, facilitating formation control and other high-level features. Due to different allocations of robot tasks throughout the system, different levels of dependency on inter-robot communication are obtained.

The performance of the algorithms with several hierarchical structures are validated and evaluated, in terms of various performance criteria, in simulations and experiments. Thereto, a simulation and experimental environment is created and tests are done in operational environments that are representative for Van den Akker Engineering. The different control structures are compared with respect to performance and requirements.

It is concluded that the use of inter-robot communication and high-level control, is beneficial for the overall performance of a transportation system in environments of concern for Van den Akker Engineering. However, since the transportation system must remain operational even when the inter-agent communication is lost, this communication is used only to increase the efficiency of transportation, not for keeping the individual agents operational and safe from collisions.

Acknowledgments

This thesis covers the results of my graduation project which concludes the master program Mechanical Engineering at the Eindhoven University of Technology. This project is the outcome of close collaboration between Van den Akker Engineering and the Dynamics and Control group of the aforementioned university.

On hindsight, when I reflect on this project, I see a period with intellectual challenges, creative freedom, endless interesting and educative discussions complimented with a healthy dose of stress. I could not have obtained these results without the contribution of numerous people which I would like to thank. First of all, I would like to extent my sincerest gratitude to Professor Henk Nijmeijer for the opportunity to graduate under his supervision. I think I speak for all his students when I say that the level of personal commitment to the students their curriculum is greatly appreciated. Special thanks go to Dragan Kostić for the weekly meetings where he greatly helped me to pinpoint the root of my problems, resulting in many productive discussions. I have never worked with a mentor this knowledgeable and considerate. It was highly instructive to realize this research within the industry. Therefore, I would like to thank Arno van den Akker for welcoming me into the company. He has provided me with all necessities for my research and above all showed me the market side of product development. I am also grateful to Michiel Legius, whose work at Van den Akker Engineering is the foundation of the results presented in this thesis. The time we worked together was genuinely helpful to get my project going.

Daarnaast zijn er nog mensen die ik persoonlijk wil bedanken. Natuurlijk mijn ouders, van wie ik onvoorwaardelijke steun heb gekregen. Bedankt voor de vrijheid die ik heb gekregen om mezelf te ontwikkelen maar vooral voor het opvangen als er iets niet uitwerkte zoals ik verwachtte. En natuurlijk mijn vrienden, welke hebben gezorgd voor de nodige afleiding en een fantastische studententijd. Van deze vrienden wil ik Paul Miggiels nog specifiek bedanken. Naast de nodige uren die we samen hebben gestudeerd (soms op merkwaardige tijdstippen) was hij ook kenmerkend voor mijn sociale ontwikkeling (vaak op deze zelfde tijdstippen).

Paul Verhoeckx, 23 november 2015

Contents

List of symbols

a	Robot acceleration
c	Tracking controller gain
d	Distance to goal position
d_{blind}	Safety region 1
d_l	Safety region 2
d_f	Safety region 3
d_b	Safety region 4
d_{evade}	The minimum distance to the obstacle in the direction of α_{evade}
$\overline{d_{evade}}$	Average distance covered to evade obstacles
$d_{i,j}$	Coupling gain
d_{obs}	The minimum distance to the obstacle in the direction of α_{min}
e	Reference tracking error
$\overline{e_{evade}}$	Average tracking error induced by the evasion of obstacles
f	Information on a possible formation
g	Robot goal position
G	Set of goal positions
k_θ	Saturation value
k_χ	Saturation value
N_{obs}	Number of observed obstacles
p	Robot position
$\overline{P}$	Average theoretically required mechanical power
r	Robot reference position
R	Outer radius of mobile robot
r^{coll}	Collision free reference path
r_l	Robot reference position w.r.t the virtual center of the formation
s	Velocity scaling factor
t	Time since the generation of the reference trajectory
t_0	Time at the generation of the reference trajectory
t_b	Blend time of reference trajectory
$\overline{t_{evade}}$	Average time spend evading obstacles
t_f	Finish time of reference trajectory
t_{int}	Expected time required to reach the intersection point of reference trajectories
$\overline{T}$	Average required computation time
u	Robot input velocities
v	Robot forward velocity
v_r	Reference forward velocity

w_α	Weight factor regarding angle to the obstacle
w_ω	Weight factor regarding the local angular velocity of an obstacle
w_d	Weight factor regarding distance to the obstacle
α_{evade}	The absolute smallest angle to the end of the observed obstacle w.r.t the robot heading angle
α_{int}	Angle to the intersection point of reference trajectories w.r.t the robot heading angle
α_L	The most extreme left angle to the obstacle w.r.t the robot heading angle
α_{mean}	The average angle to the obstacle w.r.t the robot heading angle
α_{min}	The absolute minimum angle to the obstacle w.r.t the robot heading angle
α_R	The most extreme right angle to the obstacle w.r.t the robot heading angle
Δt	Sampling time
δx	Influence of scaling on reference position in x-direction
δv_x	Influence of scaling on reference velocity in x-direction
ϵ	Formation error
θ	Robot heading angle
θ_r	Robot reference heading angle
σ	Smooth monotonically increasing weighting function
τ	Artificial force as a result from collision avoidance
ϕ	Saturation function
ω_{obs}	Approximation of local angular velocity of an obstacle
ω_r	Reference steering velocity

1 Introduction

In this chapter the scope of this project is described, together with the accompanying research objectives. Furthermore, the outline of this thesis is presented.

1.1 Motivation of the research

This report is the outcome of a project commissioned by Van den Akker Engineering (VDA) in collaboration with the Eindhoven University of Technology. VDA is an internationally active company specialized in industrial automation and is mainly, but not exclusively, operating in the food industry. A large segment of the tasks consists of unique product manipulations, for which no ready-made solution exists. This fits the innovative nature of VDA which is applied to solutions concerning internal product handling, software (PLC) engineering, electrical engineering, robot manipulation and quality control. A set of mobile robots could provide solutions for internal transport of products, materials and/or tools. The corresponding developments within the industry are closely monitored by VDA. This, among others, resulted in a demand for more knowledge on cooperative mobile robots.

1.2 Autonomous cooperative systems

Ever since the late 1980s, the field of cooperative robotics has grown rapidly. The continuous development of these robotic systems can be explained by the growing opportunities for their applications. One subset of these systems are the Automated Guided Vehicles (AGV), which are taking an increasing role as transportation method in warehouses. Other applications of autonomous mobile robots are (among others) localization, mapping, navigation and exploration. These functions can be used for expeditions to unknown environments, search and rescue missions, military purposes and transportation. The use of AGV's is often motivated by cost reductions, the removal of people from hazardous situations or by efficiency reasons.

For VDA a set of cooperative mobile robots can be used to compete with traditional transportation methods, such as conveyor systems, human operated vehicles or manual labor. It is generally established that for numerous tasks such multi-agent systems are favorable in terms of robustness, flexibility, efficiency and costs. Such systems allow us to easily scale the possible throughput and can result in a more fluent flow of products. The latter is especially useful in the food industry, since limited periods of time between the production and consumption are allowed in this sector.

1.3 Research objectives

The main objective of this research can be formulated as follows:

> *Investigate the ideal level of robot autonomy in a hierarchical control architecture for mobile robots performing transportation tasks.*

This objective can be divided into the following sub-targets:

1. *Effectively and above all safely control motions of a group of autonomous mobile robots.*

2. *Determine a suitable topology of communication among the mobile robots.*

3. *Validate, analyze and compare different control architectures, that are characterized by different levels of robot autonomy, in simulations and experiments based on several performance criteria.*

To enable the study on different control hierarchies for a group of autonomous mobile robots, it is necessary to develop a software infrastructure for motion and coordination control of these robots. Hence, the first task is to investigate the problem of motion control of the individual robots. This requires research on the reference trajectory generation, collision avoidance methods and motion control algorithms. For evaluation of merits and limitations of different control hierarchies, relevant performance criteria have to be determined such as robustness against perturbations, ease of implementation, flexibility, scalability and time- and energy-efficiency. Furthermore, it is necessary to study possible communication topologies that are evaluated in terms of scalability and required bandwidth. The purpose of the communication is to let the robots make decisions by taking information on the other robots into account. What information is communicated for what purposes and where these decisions are made, make up the hierarchy of the system. The layout of this hierarchy influences the performance of a multi-robot system, however this can be at the cost of robustness against the failure of robots or communication channels. The objective is to find the optimum in the allocation of the robot tasks throughout the systems and the corresponding dependency on information of the other robots. To draw conclusions about performance of different hierarchical structures, these structures have to be evaluated in terms of various performance criteria in simulations and experiments.

1.4 Thesis outline

In Chapter 2 an overview of the literature of relevance for this project is given and some conclusions on the to be tried setpoint generation, collision avoidance and hierarchical structures are drawn. These conclusions motivate the particular way for creation of the reference trajectories of the AGVs, which is explained in Chapter3. The trajectory tracking algorithm for the AGVs is presented in the same chapter. In Chapter 4 a local collision avoidance mechanism is introduced. This algorithm is extended by inter-agent communication depended methods in Chapter 5. Furthermore, in the same chapter some other features are presented which become available when the robots communicate among each other. These features make it possible to achieve different configurations of the AGV motion control architecture and hierarchy. The experimental setup described in Chapter 6 is used to verify the performance of the presented motion control methods. The simulation models are used to evaluate the performance of the various control schemes. The performance analysis is carried out in Chapter 7. Finally, conclusions of this project are drawn in Chapter 8 together with several recommendations for future research.

2

Literature study on route planning and control architectures

Numerous researches have been done on the motion control of mobile robots. In this report, the unicycle-type mobile robots are especially examined due to their high degree of mobility. However, this type of mobile robots are of nonholonomic nature which constraints admissible reference trajectories and challenges motion control design. This constraint implies that an unicycle robot is not capable to move sideways without slippage and is further explained in Chapter 3. Besides motion control of the unicycle mobile robots, in this project great attention is paid to trajectory planning, collision avoidance and hierarchical distribution of tasks within a group of mobile robots. In this chapter, an overview is given of different results that can be found in the literature and are relevant for the research topic of this project. Their advantages, restrictions and requirements are classified and discussed.

2.1 Path generation

2.1.1 Fixed-path planning

By subdividing the environment into fixed (virtual) paths, grids or zones, a central trajectory planner tries to find the shortest, collision-free trajectory for each agent. This is often based on the algorithm derived in [1]. This approach is widely used in industry, because of its intuitive nature. Since all information on the agents their future tasks is centrally determined, such a planning can be made very time-efficient.

However, there are some drawbacks to this method. First of all, it is necessary to know the static environment in advance in order to lay out a path, grid or zone network. Also, this approach is limited in its flexibility, since the agents their movement is constraint to the predefined routes. Flexibility could be improved by increasing the number of potential paths or by using finer grids or zones, however that would require more computational

power to calculate the optimal set of trajectories. When the number of agents or the size of the environment is increased, the necessary computational power or time can become an issue. Methods from this approach have been the subject of extensive research, for example see [2–4]. Roughly these methods can be divided into two categories, static and dynamic route planning.

Static route planning

A static planner creates a trajectory for every agent, preventing deadlocks and collisions a priori. This is done by segmenting the routes and linking time periods to these segments. Based on the expected occupation of the segments by the agents, suitable trajectories are created in [5]. Since these calculations are done in advance, the agents do not need extensive on-board computational power for the sake of trajectory generation. In [6] various algorithms are covered which search for the optimal division of the segments. The downside of this method is that the system is sensitive to robot or network malfunctions, or large tracking errors induced in any other way. Once these issues occur, the result of the static planning may become obsolete. To resolve this issue the agents need to stop their motions before a new set of trajectories is created by re-running the algorithm for the static planning.

Dynamic route planning

In the case of dynamic route planning, the reference trajectories of the mobile agents can be altered when necessary. Therefore, the effectiveness of the dynamic route planning is less sensitive to malfunctions or perturbations on the agents. Most often, these planners use an off-line calculated set of reference trajectories that are online updated when necessary [3, 7]. The on-line trajectory generators enable of the individual agents calculate new routes by taking into account the already known trajectories of the other agents. This gives more robustness regarding malfunctions and perturbations. However, since the trajectories are now derived consecutive, these trajectories may be less time-efficient than when all trajectories are derived simultaneous.

2.1.2 Free Range Routing

A more flexible way to determine the reference trajectory for a group of agents is to use a so called Dynamic Free Range Routing, see [8–10]. This set of methods utilize the entire available motion space and accordingly do not need deadlock prevention algorithms. Since the working environment is unknown simplistic future reference trajectories are created, requiring little computational costs. This improves the ability to scale up the system and facilitates the implementation of (new) agents. However, as the trajectories are unknown, the followed routes can be less optimal in the sense of the spend time and distance covered, in comparison to the predefined routes. For example, an agent does not know if it will encounter an obstacle in the future and can therefore not correct for it yet. This is a consequence of the fact that the trajectory planner uses very little information about the surroundings and available routes. Examples of Dynamic Free Range Routing

are the Artificial Potential Field (APF) algorithms described in [11–14], the cost function method [8, 9], a Resource Allocation System (RAS) [15], a Deformable Virtual Zone (DVZ) approach described in [16] and using a Fuzzy Inference System as done in [10, 17]. The major downside of systems using APF or a cost function is the possibility of ending op in a local minimum, which differs from the target. Multiple methods have been suggested to overcome some parts of these problems, for example by switching to a wall-following algorithm or by enhancing the APF [11, 18]. Also an APF method is suggested in [19], which requires information about the environment and computes a trajectory which always ends at the target. However, this method requires knowledge of the environment and demands more computational power, especially in large operating areas. In [20–24] steering potential function is suggested. This collision avoidance method is based on human behavior. Similar as with APF systems an artificial force is created based on the known obstacles. However in this algorithm only the steering velocity is altered, resulting in a path which is deflected from the obstacles.

2.2 Hierarchical structures

Most multi-agent systems consist of a set of robots and one supervisor. The allocation of tasks and responsibilities between the robots and the supervisor determines the level of the robot autonomy. The two ends of this spectrum are the completely centralized and completely decentralized control architectures. These architectures are discussed below.

Centralized control

In the fully central case, for example described in [25], all decisions and calculations are performed at the level of the supervisor. This discards the need of inter agent communication and omits the need of powerful on-board computers. Therefore for sets of miniature robots this hierarchy can be beneficial, since the limited on-board computational power restraints the possible configurations. Although it might be beneficial to have a constant and well defined communication topology, this hierarchy also contains negative aspects. First of all, performance of the agents depends on active communication with the supervisor; the agents are not autonomous. In practice, communication channels can be of limited bandwidth and are subjected to delays. This can have a negative impact on the agents, compromise safety, and even cause them to stop working. Furthermore, the required communication bandwidth grows with the number of agents, limiting the maximum number of agents. Especially, if sensor data that are obtained at the level of the individual agents have to be communicated to the supervisor, the bandwidth limitations become critical.

Decentralized control

A completely decentralized control hierarchy has a hard separation of the information on the individual agents. Often a supervisor is solely used as an user interface for a human operator. Cooperation of multiple robots, which fully depended on local robot

observations, is investigated in [26]. Here an agent will wait to perform a two-agent task, until a second agent is detected. This leads to a very robust but inefficient collaboration. Also for single-agent tasks time-efficiency is expected to be inferior compared to the central case. This is due to path planning which neglects presence of multiple agents and lack robot cooperation during collision avoidance.

2.3 Conclusion

Based on this literature study a set of requirements on a group of collaborating mobile robots is defined. It can be concluded that an ideal multi-agent system should:

a. Perform motions along time- and energy-efficient trajectories;
b. Avoid deadlocks;
c. Avoid collisions;
d. Be flexible w.r.t. the possible routes and destinations;
e. Be easy to be implemented in a new environment;
f. Be able to scale in number of agents;
g. Behave predictable;
h. Be robust against perturbations;
i. Always reach assigned destinations;
j. Have a modular control scheme.

To accommodate the requirements a Free Range Routing algorithm with collision avoidance is investigated in this project. To investigate the effect of the possible hierarchical structures, modular function blocks of the robot tasks are desired which can be allocated to the various entities of the system. This modular architecture also facilitates an easy scaling of the number of robots. The inter-agent communication is required for the performance of multi-agent systems, see [27] and [28]. However, as discussed in [29], the information exchange should be made minimal and algorithms should be robust against the absence of information on the other agents.

3

Setpoint generation and tracking

In this chapter, the algorithm used to create the setpoint motion profile of a mobile robot is explained. The desired profile should allow time- and energy-efficient robot motion towards a goal location while satisfying the robot's nonholonomic constraint, velocity limits and continuity in the initial robot position and velocity. To optimally use space available for robot motion, a free range routing method is proposed, eliminating the need of fixed routes or zones. Thereto, the principle behind a newly developed reference generation method is explained in Section 3.1. This method allows for on the fly alterations of the planned reference trajectory in order to satisfy the velocity limits as explained in Section 3.1.1 and can create formations of robots as is described in Section 3.1.2. Furthermore a tracking controller is introduced in Section 3.2 in order to pursue the robot along the setpoint motion profile by commanding adequate velocity inputs to the robot.

3.1 Setpoint generation

The reference motion trajectory is based on the second order polynomial function, consisting of linear segments with parabolic blends (LSPB). This enables an as large as possible period with a constant velocity, which is beneficial energy-efficiency. An example of the pursued reference velocity profile is shown in Figure 3.1. By inspection of this figure, one can notice that the reference velocity has a linear increase at the beginning (constant acceleration), then remains at the desired constant value, and finally linearly decreases (constant deceleration).
A downside to the currently existing LSPB trajectories, such as described in [30], is that they require an initial and goal velocity of zero. This would inhibit the possibility of on the fly updating the trajectory, while ensuring continuity with respect to the initial position and velocity together with the nonholonomic constraint. This is necessary for compatibility with the collision avoidance algorithm which will is described in Section 4.1. Also the implementation of way points, which can be considered as goals with nonzero velocities, would be impossible. To accommodate these demands a novel LSPB setpoint

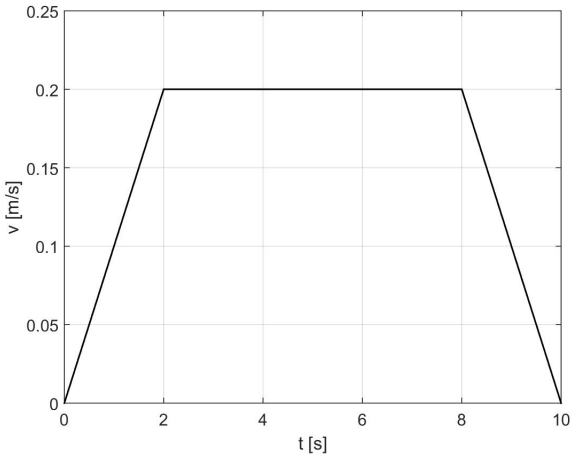


Figure 3.1: Reference velocity profile using the LSPB method.

generator is created. The trajectory is not generated on the direct path s between the initial and goal robot positions p_0 and g respectively, as depicted in Figure 3.2. This is done to avoid difficulties in creating a continuous reference angle θ_r of the robot when its initial heading angle θ_0 differs from the angle towards the goal θ_g. As shown, the created reference path r is not necessarily the shortest path to the goal, but does incorporate the nonholonomic constraints of the robot which is mathematically represented in (3.1). To create this path, two separate trajectories are generated and combined, one trajectory in the global x- and another in the global y- direction. Since both trajectories can have nonzero initial velocities, these velocities can directly be taken into account when creating a continuous reference angle towards the goal.

$$-\dot{x}\left(t\right) sin\left(\theta\left(t\right)\right) + \dot{y}\left(t\right) cos\left(\theta\left(t\right)\right) = 0 \qquad \forall t \tag{3.1}$$

In order to incorporate the maximum velocity of the robot into the reference trajectories, this general maximum reference velocity $v_{r,max}$ has to be subdivided into maximum velocities in the two directions ($v_{xr,max}$ and $v_{yr,max}$). This is done by scaling $v_{r,max}$ based on the initial distance between the robot and the goal in x- and y- directions, as follows:

$$\begin{aligned} v_{xr,max} &= \frac{d_x}{\sqrt{d_x^2 + d_y^2}} v_{r,max}, \\ v_{yr,max} &= \frac{d_y}{\sqrt{d_x^2 + d_y^2}} v_{r,max}. \end{aligned} \tag{3.2}$$

Here d_x and d_y are the absolute initial distances to the goal, in the x and y directions respectively. These maximum reference velocities are the desired velocities during the constant velocity phase of the trajectory. Since the maximum acceleration a_{max} of the robots is relatively high, there is no need to subdivide it between the directions; instead, the same acceleration limit can be assumed for both directions. This limit is used in the first and the last segments of the trajectory.

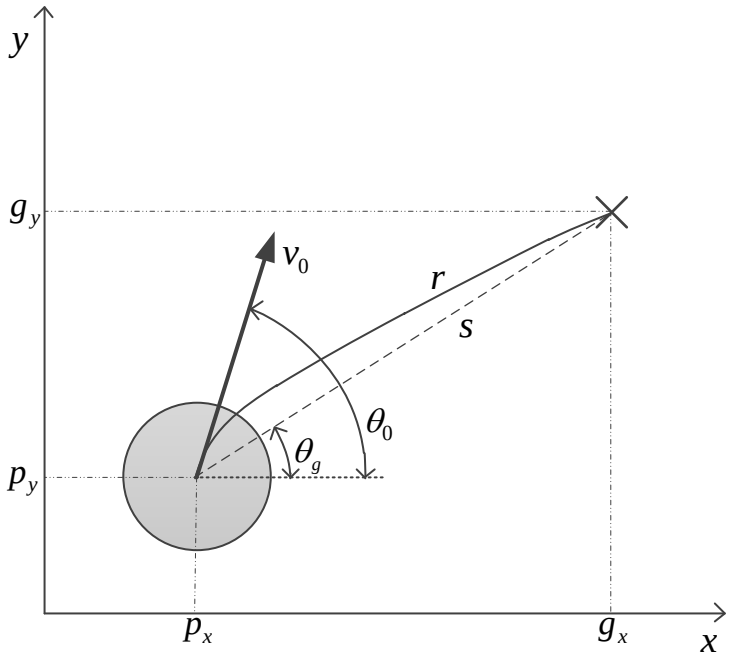


Figure 3.2: Example of a generated reference path.

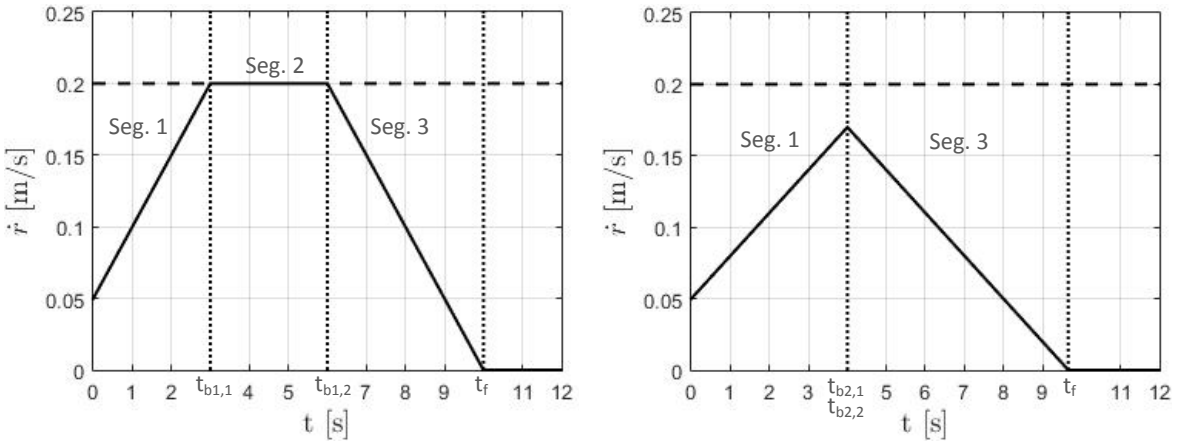
In the next part, the time intervals of the three segments are determined; for convenience, the explanation is given for one direction only.

First of all it is checked if the desired velocity and accelerations are positive or negative. For the velocity this is done by checking the current position relative to the goal, so if $g - p_0$ is positive the desired velocity of the reference in segment 2 ($\dot{r}_{seg2}$) is also positive (for explanation on the segments, see Figure 3.3). Here g and p_0 are scalars indicating the 1D position of the goal and the robot at the moment the reference trajectory is created, respectively.

For the desired acceleration in the first segment ($\ddot{r}_{seg1}$), the absolute current velocity is compared to the (positive) maximum reference velocity $v_{r,max}$, as done in (3.3) resulting in an acceleration towards the desired maximum velocity of the reference $\dot{r}_{seg2}$.

$$\ddot{r}_{seg1} = \begin{cases} -sign\left(\dot{p}_0\right) \cdot a_{max} & \text{for} \quad |\dot{p}_0| - v_{r,max} > 0, \\ sign(\dot{r}_{seg2}) \cdot a_{max} & \text{for} \quad |\dot{p}_0| - v_{r,max} \leq 0. \end{cases} \tag{3.3}$$

The duration of the first segment (t_{b1}) depends on which of the two situations shown in the Figure 3.3 occurs. In the sub-figures, the desired maximum velocity is indicated by the horizontal dashed line, the blending times which make up the segments are indicated by the vertical dash-dot lines. In situation a, depicted in Figure 3.3, $\dot{r}_{seg2}$ is reached at $t = t_{b1}$, while in situation b, depicted in the same figure, it is necessary to start decelerating at $t = t_{b1}$ to prevent an overshoot in the position. At this moment the maximum desired velocity is not yet reached and segment 2 is omitted. Which of these two situations



(a) Situation a. (b) Situation b.

Figure 3.3: Possibilities of reference velocity profiles.

occurs depends on the initial distance towards the goal, the initial and goal velocities and the maximum allowable acceleration and velocity.
For situation a, t_{b1} can be calculated as follows:

$$t_{b1,a} = \frac{\dot{r}_{seg2} - \dot{p}_0}{\ddot{r}_{seg1}}. \tag{3.4}$$

For situation b the blending time $t_{b1,b}$ is calculated by evaluating at what time instant the distance to the goal equals the distance required to decelerate up to the goal velocity $\dot{g}$. The distance to the goal equals

$$\begin{aligned} d(t) =& g - p(t) \\ =& g - p_0 - \dot{p}_0 t - 0.5\ddot{r}_{seg1} t^2. \end{aligned} \tag{3.5}$$

Here $p(t)$ is the current robot position and t is the time from the moment the reference is created. The minimum distance δ_{seg3} needed to slow down to the goal velocity $\dot{g}$ can be calculated by (3.6). Note that for $\ddot{r}_{seg3}$ it is assumed that $sign(\ddot{r}_{seg3}) = -sign(\dot{p}(t_{b1,b}) - \dot{g})$, this is checked afterwards.

$$\begin{aligned} \delta_{seg3}(t) =& \overbrace{\left|\frac{\dot{p}(t) - \dot{g}}{\ddot{r}_{seg3}}\right|}^{\text{time needed}} \cdot \overbrace{\frac{\dot{p}(t) + \dot{g}}{2}}^{\text{average velocity}} \\ =& -\frac{\dot{p}(t) - \dot{g}}{\ddot{r}_{seg3}} \cdot \frac{\dot{p}(t) + \dot{g}}{2} \\ =& -\frac{\dot{p}_0 + \ddot{r}_{seg1} t - \dot{g}}{\ddot{r}_{seg3}} \cdot \frac{\dot{p}_0 + \ddot{r}_{seg1} t + \dot{g}}{2} \\ =& -\frac{\dot{p}_0^2 + 2\dot{p}_0 \ddot{r}_{seg1} t + \ddot{r}_{seg1}^2 t^2 - \dot{g}^2}{2\ddot{r}_{seg3}} \end{aligned} \tag{3.6}$$

To determine $t_{b1,b}$, we use the condition $d(t) = \delta_{seg3}(t)$, which based on (3.5) and (3.6) leads to the following polynomial equation:

$$(\ddot{r}_{seg1}\ddot{r}_{seg3} - \ddot{r}_{seg1}^2)t_{b1,b}^2 + 2\dot{p}_0(\ddot{r}_{seg3} - \ddot{r}_{seg1})t_{b1,b} + 2\ddot{r}_{seg3}(p_0 - g) + \dot{g}^2 - \dot{p}_0^2 = 0. \tag{3.7}$$

Which of the two situations sketched in Figure 3.3 is obtained, depends on whether the reference velocity first reaches $\dot{r}_{seg2}$ (situation a) or the point where it has to start decelerating (situation b). Therefore the actual first blending time of the reference t_{b1} equals the minimum of the smallest real solution of (3.7) and the solution of (3.4), and can be noted as

$$t_{b1} = min(t_{b1,a}, t_{b1,b}). \tag{3.8}$$

If $|\dot{p}_0| - v_{r,max} > 0$ it can occur that $\ddot{r}_{seg1} = \ddot{r}_{seg3}$ for which the polynomial equation (3.7) does not hold. This situation can occur in practice, since $v_{r,max}$ is calculated on the fly using (3.2). In this case $t_{b1} = t_{b1,a}$, assuming $|\dot{g}| \leq |\dot{r}_{seg2}|$.

In situation a, a second blending time (t_{b2}) has to be calculated, which is done in a similar manner. The distance to be traveled in segment 2 equals the initial distance to the goal minus the distance traveled in the segments 1 and 3 or δ_{seg1} and δ_{seg3}, respectively. The distance to be traveled in segment 2 is used to calculate t_{b2} as given by (3.9). Here, δ_{seg3} is determined in a similar manner as in (3.6) with $\dot{p}(t) = \dot{r}_{seg2}$.

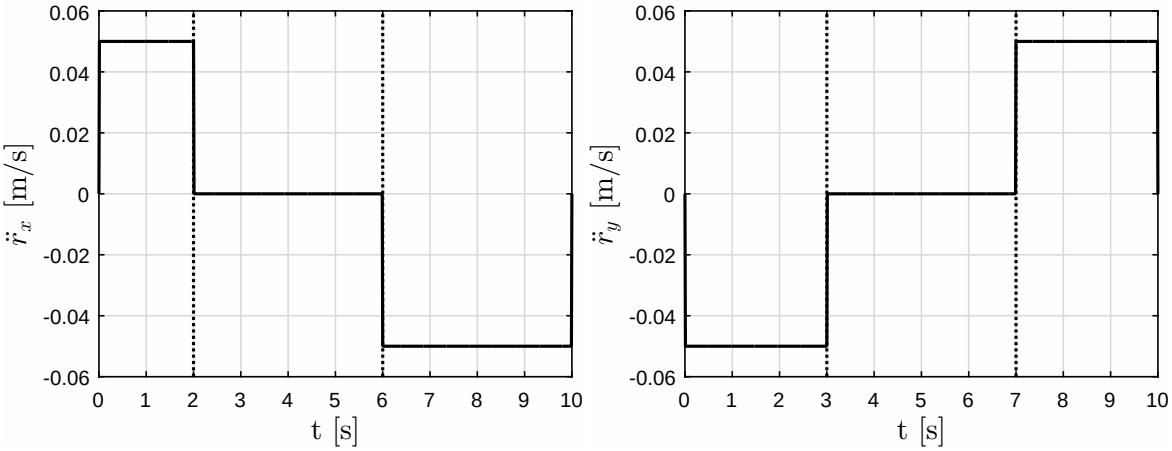
$$\begin{aligned} t_{b2} =& \frac{d_0 - \delta_{seg1} - \delta_{seg3}}{\dot{r}_{seg2}} + t_{b1} \\ =& \frac{g - p_0 - \dot{p}_0 t_{b1} - 0.5\ddot{r}_{seg1}t_{b1}^2 + \frac{\dot{r}_{seg2} - \dot{g}}{\ddot{r}_{seg3}} \frac{\dot{r}_{seg2} + \dot{g}}{2}}{\dot{r}_{seg2}} + t_{b1}. \end{aligned} \tag{3.9}$$

For the second situation depicted in Figure 3.3 it holds that $t_{b2} = t_{b1}$. Furthermore there is a possibility that the time required in segment 2 is negative. This happens when a relatively high initial velocity and low allowable deceleration cause an overshoot of the position. In this case t_{b2} is set to equal t_{b1}. Due to the inevitable position overshoot a new trajectory is required after the ending of the current one.

Finally, the time of arrival t_f at the goal position is calculated. This is done by adding the required stopping time to t_{b2}, resulting in

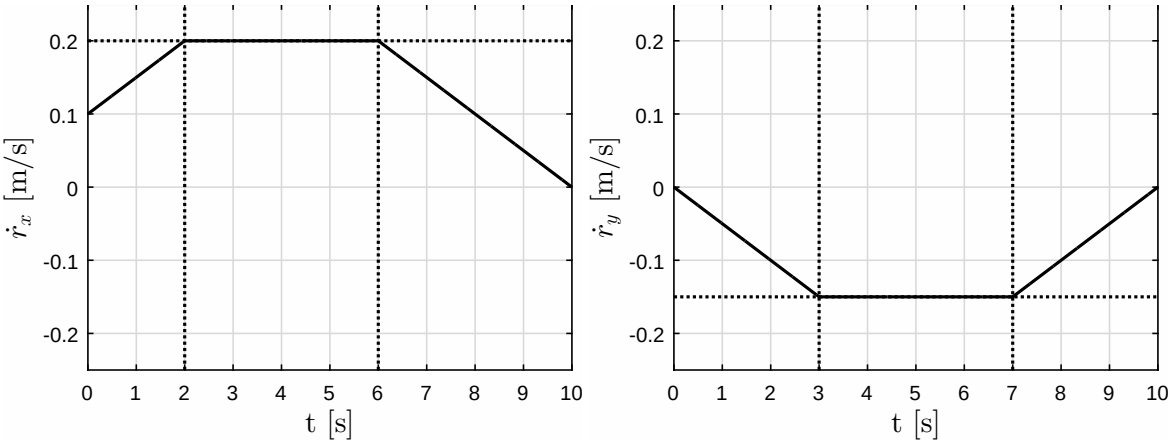
$$t_f = t_{b2} + \left| \frac{\dot{r}_{seg2} - \dot{g}}{\ddot{r}_{seg3}} \right|. \tag{3.10}$$

In the Figures 3.4 to 3.6 an example acceleration, velocity and position are shown against time, for both the x- and the y-directions. Here, the vertical lines again represent the blending times. In Figure 3.7 the resulting 2D velocity and position are depicted. The starting position is indicated by the circle and the final position by a cross.



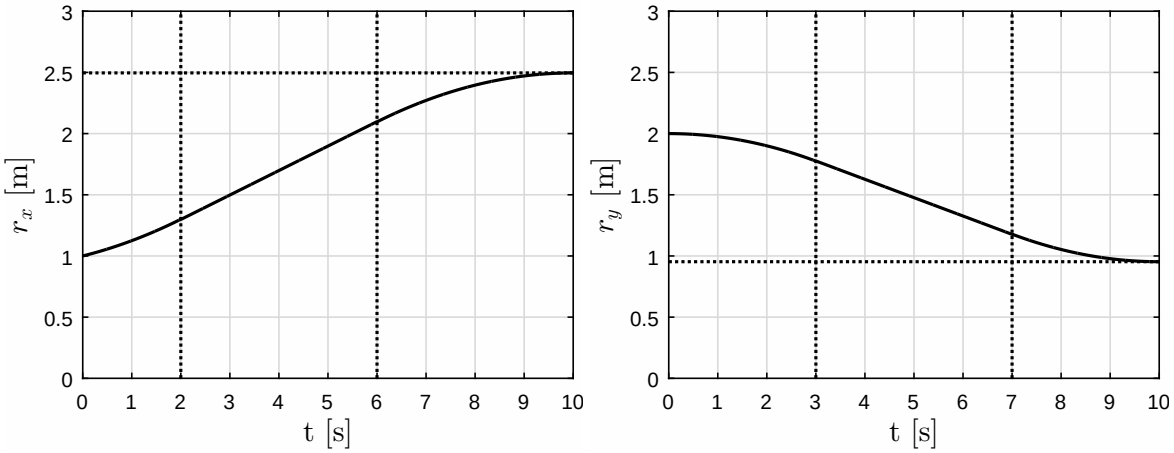
(a) Acceleration in x-direction. (b) Acceleration in y-direction.

Figure 3.4: Reference accelerations against time.



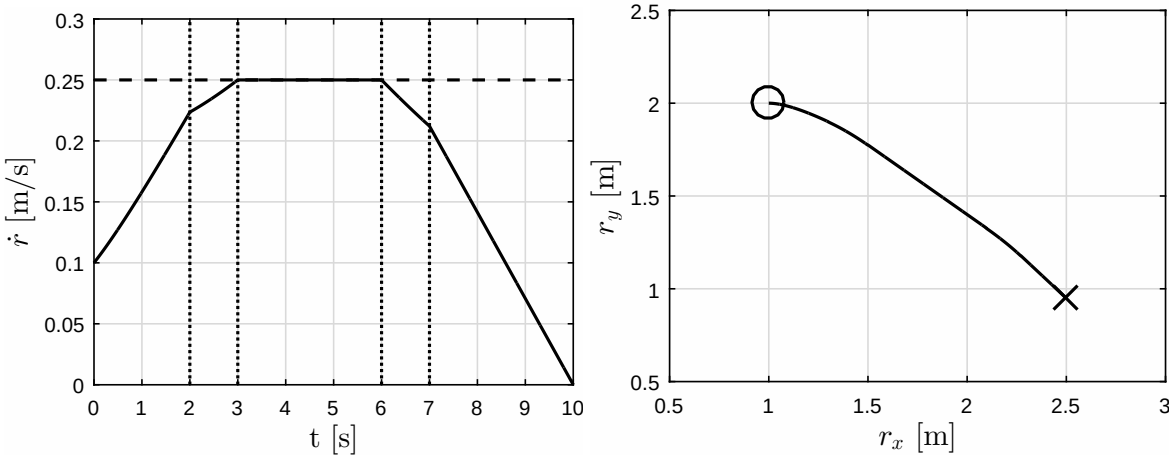
(a) Reference velocity in x-direction. (b) Velocity in y-direction.

Figure 3.5: Velocity against time.



(a) Position in x-direction. (b) Position in y-direction.

Figure 3.6: Reference positions against time.



(a) Overall velocity in 2D. (b) Overall position in 2D.

Figure 3.7: Combined reference velocity (left) and reference position (right).

3.1.1 Velocity limits

The reference motion profiles of the robots have to accommodate the limits on the robot's forward and steering velocities. As for the forward velocity, this is done by (3.2). The steering reference velocity $\omega_r(t)$ is calculated as follows:

$$\omega_r(t) = \frac{\ddot{r}_y(t)\dot{r}_x(t) - \ddot{r}_x(t)\dot{r}_y(t)}{\dot{r}_x^2(t) + \dot{r}_y^2(t)}. \tag{3.11}$$

To ensure that $\omega_r(t)$ remains within the limit on the robot's steering velocity, the robot's reference accelerations in x- and y-directions have to be scaled on the fly. The reference translational velocity is not altered, since a continuous translational velocity profile is required for the tracking controller as explained in Section 3.2.

In order to still reach the goal position and velocity at the end of the reference trajectory, the blending times have to be altered when the accelerations are reduced. Each time step when the reference accelerations are scaled down to meet the limit on the steering velocity, it is on the fly calculated what differences in velocity and traveled distance at $t = t_{b1}$ are induced by this scaling. First, the scaling factor $s(t)$ is calculated, using

$$s(t) = max\left(\frac{|\omega_r(t)|}{\omega_{r,max}}, 1\right) \tag{3.12}$$

where $\omega_{r,max}$ is a positive constant and $\omega_r(t)$ is determined by (3.11).
For convenience, the following calculations are given for the x-direction only, since the same ones apply to y. This factor $s(t)$ is used to obtain the scaled reference acceleration:

$$\ddot{r}_{x,s}(t) = \frac{\ddot{r}_x(t)}{s(t)}, \tag{3.13}$$

To determine the effects of the scaling on the velocity and traveled distance at $t = t_{b1}$, this velocity and distance are calculated with and without scaling. When no scaling is necessary the traveled distance and obtained velocity are given by

$$\begin{aligned} r_x(t_{b1}) &= r_x(t_0) + \dot{r}_x(t_0)t_{b1} + \frac{1}{2}\ddot{r}_{x,seg1}t_{b1}^2, \\ \dot{r}_x(t_{b1}) &= \dot{r}_x(t_0) + \ddot{r}_{x,seg1}t_{b1}. \end{aligned} \tag{3.14}$$

Here t_0 is the initial time when the reference is created.
When the reference accelerations are scaled for one sampling time Δt during segment 1,

the distance and velocity at $t = t_{b1}$ are given by,

$$
\begin{aligned}
r_{x,s}(t_{b1}) &= \overbrace{\underbrace{r_x(t_0) + \dot{r}_x(t_0)t_s + \frac{1}{2}\ddot{r}_{x,seg1}t_s^2}_{r_{x,s^-}} + \underbrace{(\dot{r}_x(t_0) + \ddot{r}_{x,seg1}t_s)}_{\dot{r}_{x,s^-}}\Delta t + \frac{1}{2}\frac{\dddot{r}_{x,seg1}}{s(t_s)}\Delta t^2}^{r_{x,s^+}} \\
&\quad + \overbrace{\left(\dot{r}_x(t_0) + \ddot{r}_{x,seg1}t_s + \frac{\dddot{r}_{x,seg1}}{s(t_s)}\Delta t\right)}^{\dot{r}_{x,s^+}}(t_{b1} - t_s - \Delta t) + \frac{1}{2}\ddot{r}_{x,seg1}\left(t_{b1} - t_s - \Delta t\right)^2 \\
&= r_x(t_0) + \dot{r}_x(t_0)t_{b1} + \frac{1}{2}\ddot{r}_{x,seg1}t_{b1}^2 + \left(\left(1 - \frac{1}{s}\right)\ddot{r}_{x,seg1}\Delta t\right)\left(\frac{1}{2}\Delta t + t_s - t_{b1}\right), \\
\dot{r}_{x,s}(t_{b1}) &= \dot{r}_x(t_0) + \ddot{r}_{x,seg1}t_s + \frac{\dddot{r}_{x,seg1}}{s(t_s)}\Delta t + \ddot{r}_{x,seg1}(t_{b1} - t_s - \Delta t) \\
&= \dot{r}_x(t_0) + \ddot{r}_{x,seg1}t_{b1} - \left(1 - \frac{1}{s}\right)\ddot{r}_{x,seg1}\Delta t. \qquad (3.15)
\end{aligned}
$$

Here, t_s is a time period from the starting time of the reference to the moment when the scaling is applied. Furthermore, r_{x,s^-} and r_{x,s^+} are the x positions at the time instances just before and after the scaling is applied. The influences of the scaling on the velocity and distance are determined by taking the difference between the equations in (3.14) and (3.15).

$$
\begin{aligned}
\delta x(t_s) &:= \delta x(t_s) + \left(1 - \frac{1}{s(t_s)}\right)\ddot{r}_{x,seg1}\Delta t\left(t_{b1} - t_s - \frac{1}{2}\Delta t\right), \\
\delta v_x(t_s) &:= \delta v_x(t_s) + \left(1 - \frac{1}{s(t_s)}\right)\ddot{r}_{x,seg1}\Delta t. \qquad (3.16)
\end{aligned}
$$

In Figure 3.8 a graphical representation of the influence of scaling during segment 1 is given. Once the effects of all applied scalings are known, it can be checked whether an elongation of a certain segment is required. If the scaling is applied in segment 1 of the reference, t_{b1} is adjusted such that the desired maximum velocity again is reached at the end of this segment. The required time for which the blending times t_{b1}, t_{b2} and t_f have to be deferred (δt_v), can be calculated using:

$$
\delta t_{v_x} = \frac{\delta v_x(t)}{\ddot{r}_{x,seg1}}. \qquad (3.17)
$$

Furthermore, the length of segment 2 is adjusted to counteract the effect of the scaling and the alteration of the duration of segment 1 on the traveled distance. The effect of altering the blending times as done in (3.17) on the traveled distance equals the traveled distance between $t = t_{b1}$ and $t = t_{b1} + \delta t_{v_x}$. Now δx can be updated as follows:

$$
\delta x^+(t) = \delta x(t) - \left(\dot{r}_{x,seg2}\delta t_{v_x} - \frac{1}{2}\ddot{r}_{x,seg1}\delta t_{v_x}^2\right). \qquad (3.18)
$$

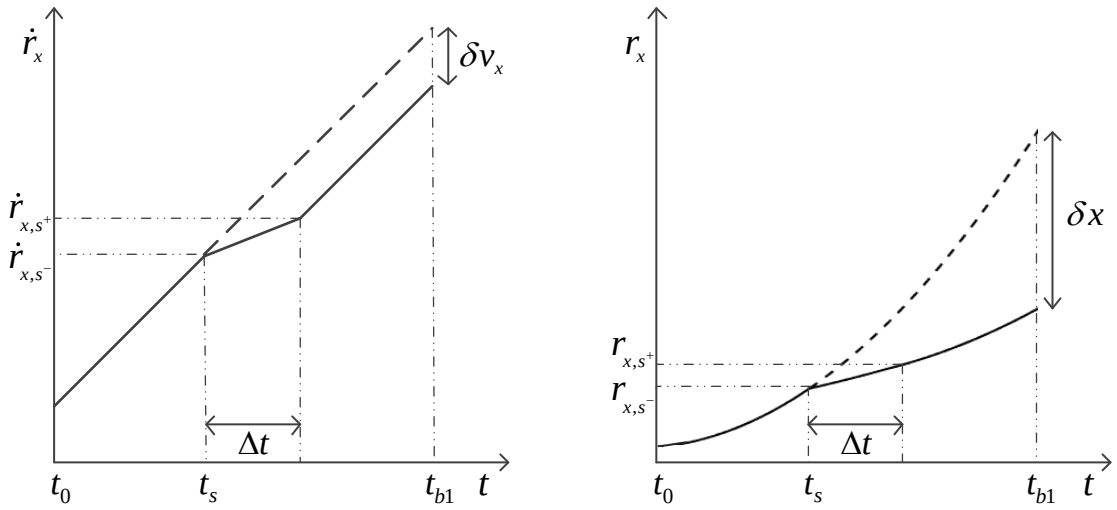


Figure 3.8: Example of the influence of scaling on the reference velocity (left) and position (right).

Here $\delta x^+(t)$ represents the current expected difference in x-position at $t = t_{b1} + \delta t_{v_x}$. The required change in the blending times t_{b2} and t_f to counteract this, is calculated by:

$$\delta t_x = \frac{\delta x^+(t)}{\dot{r}_{x,seg2}}. \tag{3.19}$$

This compensation for velocity and traveled distance works only when the scaling is applied during the first segment of the reference profile. When the scaling is applied during the third segment there is no possibility for compensation and the reference position and velocity will exceed the goal position and velocity, respectively. A new reference might be required to reach the goal. Examples of the reference velocity profiles with and without saturation are given in Figure 3.9. These profiles are shown for both x- and y-directions. When no saturation is applied, large steering velocities arise when

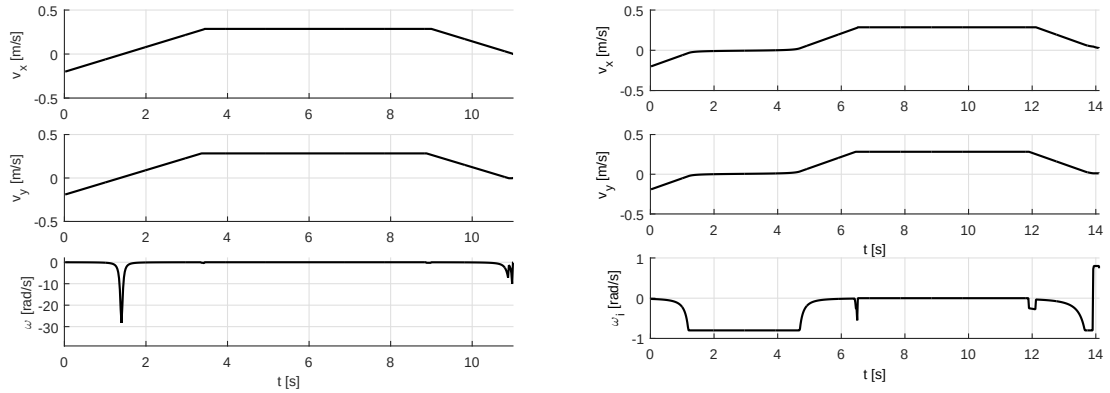


Figure 3.9: Reference velocities without (left-hand side) and with (right-hand side) saturation.

the translational velocity is low and the accelerations are nonzero. This is to be expected based on (3.11). The saturated reference profiles require a longer time to reach the goal. As shown in Figure 3.10, the motion profiles when the saturation is applied have the same final position as the non-saturated ones.

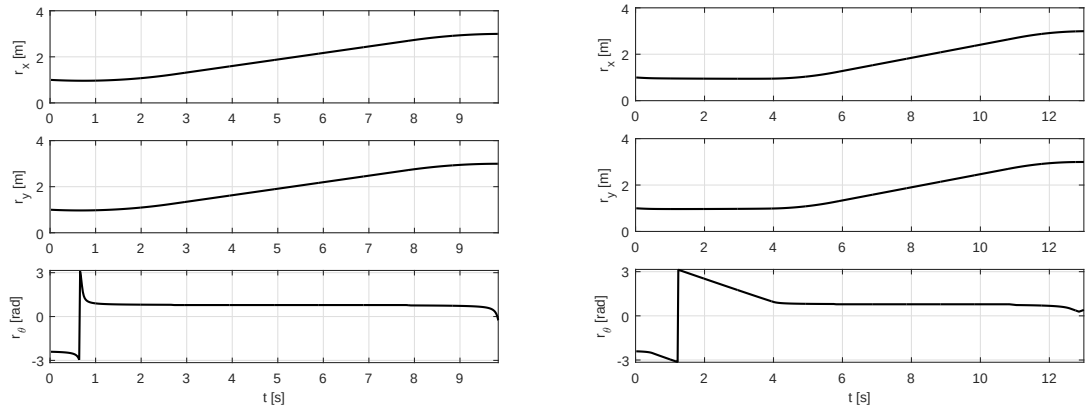


Figure 3.10: Reference positions without (left-hand side) and with (right-hand side) saturation.

3.1.2 Formation control

To investigate the possible advantage of multiple agents maneuvering in a geographical pattern, formation control is considered. This formation is constructed as done by [31] where the reference position of an agent is defined relative to the virtual center of the formation. For convenience but without loss of generality, the leader-follower method is chosen where the virtual center coincides with the leader. The reference trajectory of this leader is determined as described above. When an agent is assigned to follow the leader, its reference position is set at a certain position relative to the leader. This results in:

$$\mathbf{r_i}(t) = \mathbf{r_l}(t) + \mathbf{R}(\theta_l)\mathbf{l_i}(t). \tag{3.20}$$

Here $\mathbf{r_i} \in \mathbb{R}^2$ denotes the reference position of the i^{th} agent ($i \in \{1, 2, \ldots, n\}$), such that $\mathbf{r_i} = [r_{x,i},\ r_{y,i}]^T$, r_l represents the reference position of the leader, $\mathbf{R}$ is the rotation matrix defined by (3.21), and $\mathbf{l_i} \in \mathbb{R}^2$ is a vector denoting the desired position of agent i with respect to the virtual center.

$$\mathbf{R}(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}. \tag{3.21}$$

The allocation of leaders and followers is done by a supervisor and will be discussed in Chapter 5. When $\mathbf{l_i}$ is chosen to be time-invariant, the reference velocity of the follower can be written as:

$$\dot{\mathbf{r}}_\mathbf{f}(t) = \dot{\mathbf{r}}_\mathbf{l}(t) + \omega_{r,l}\mathbf{S}\mathbf{R}(\theta_l)\mathbf{l_f}. \tag{3.22}$$

Here, the property $\dot{\mathbf{R}}(\theta) = \omega_r \mathbf{SR}(\theta)$ is used, with

$$\mathbf{S} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}. \tag{3.23}$$

The reference of the follower is only determined for the current time step, no future reference is computed.

3.2 Tracking controller

In order to pursue the desired position and velocity derived in the previous section, a tracking controller is implemented. The controller calculates the steering and forward velocities, based on the reference velocities and servo errors. These servo errors consist of the position and formation errors that are defined as follows:

$$\begin{aligned} \mathbf{e_i}(t) &= \mathbf{r_i}(t) - \mathbf{p_i}(t), \\ e_{\theta,i}(t) &= \theta_{r,i}(t) - \theta_i(t). \end{aligned} \tag{3.24}$$

Here $\mathbf{e_i} = [e_{x,i},\, e_{y,i}]^T$ and $\mathbf{p_i} = [p_{x,i},\, p_{y,i}]^T$. The formation errors are defined by

$$\epsilon_{\mathbf{i,j}}(t) = \mathbf{e_i}(t) - \mathbf{e_j}(t). \tag{3.25}$$

The particular tracking controller is given by (3.26), and it is derived by Kostić et al. in [32]. This controller has the advantage that it generates forward and steering velocities that meet the robot's velocity limits.

$$\begin{aligned} v_i(t) &= v_{r,i}(t) cos(e_{\theta,i}(t)) + \phi_{k_{\chi,i}(t)}(c_x \chi_i(t)), \\ \omega_i(t) &= \omega_{r,i}(t) + v_{r,i}(t) \frac{sin(e_{\theta,i}(t))}{e_{\theta,i}(t)} \xi_i(t) + \phi_{k_{\theta,i}(t)}(c_\theta\, e_{\theta,i}(t)). \end{aligned} \tag{3.26}$$

Here c_x, c_θ, $k_\chi(t)$ and $k_\theta(t)$ are strictly positive gains, χ and ξ are functions as given in (3.27) and $\phi_{k_\chi(t)}$ and $\phi_{k_\theta(t)}$ are saturation functions.

$$\begin{bmatrix} \chi(t) \\ \xi(t) \end{bmatrix} = \mathbf{R}(\theta_i) \left[\frac{c_i\, \mathbf{e_i}}{\sqrt{1 + c_i\, \mathbf{e_i}^T \mathbf{e_i}}} + \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{d_{i,j}\, \epsilon_{\mathbf{i,j}}}{\sqrt{1 + d_{i,j}\, \epsilon_{\mathbf{i,j}}{}^T \epsilon_{\mathbf{i,j}}}} \right]. \tag{3.27}$$

Here $c \in \mathbb{R}^n$ denote a feedback gain which consists of strictly positive scalars and $d \in \mathbb{R}^{n \times n}$ denotes the coupling gain that can be either positive or zero. These gains allow one to emphasize effects of trajectory tracking v.s. formation control. When no formation is pursued, $c_{i,j} = 0,\ \forall j \in \{1, 2, \ldots, n\}$. A saturation function $\phi_r(kz)$ has the following properties:

$$-r \leq \phi_r(kz) \leq r,\ \forall z \in \mathbb{R}; \quad z\phi_r(kz) > 0,\ \forall z \neq 0; \quad \phi_r(0) \equiv 0, \tag{3.28}$$

and in this case it is chosen to use $\phi_r(kz) = r \cdot tanh(kz)$. The bounds $k_{\chi,i}(t)$ and $k_{\theta,i}(t)$ are chosen to be time-varying, which increases the robustness against disturbances at lower velocities. These are defined as:

$$\begin{aligned} k_{\chi,i}(t) &= v_{max} - |v_{r,i}(t)cos(e_{\theta_i}(t))|\,, \\ k_{\theta,i}(t) &= \omega_{max} - |\omega_{r,i}(t)| - |v_{r,i}(t)| \left(\sqrt{c_i} + \sum_{\substack{j=1 \\ j\neq i}}^{n} \sqrt{c_{i,j}}\right). \end{aligned} \tag{3.29}$$

Here v_{max} and ω_{max} are the maximum allowable robot forward and angular velocities, respectively. It should be noted that these not necessarily equal the maximum allowable reference velocities $v_{r,max}$ and $\omega_{r,max}$. The saturation bounds $k_\chi(t)$ and $k_{theta}(t)$ should remain positive to guarantee trajectory tracking (see [32]), especially the latter one imposes limits to the gains c_i and $c_{i,j}$. In order to achieve good motion performance during pure trajectory tacking or formation control, the gains are tuned for both situations independently and switch when the formation is formed or broken.

3.3 Summary

A method to plan future reference trajectories is explained in this chapter. By merging two separate reference velocity profiles, in the directions spanning the motion space, a reference trajectory is obtained which satisfies the nonholonomic constraint of the mobile robots and can cope with non-zero initial velocities. On the fly alterations of the reference accelerations ensure that the steering velocity limit is met while the final reference position remains the same. Furthermore, a method to create the references of robots pursuing a formation is proposed. Finally, the used tracking controller is introduced which ensures tracking of the reference positions and guarantees to satisfy the robot velocity limits.

4

Low-level control strategies

Many different control architectures and accompanying communication schemes can be used to control a group of mobile robots. The main difference is the level of autonomy given to the agents. In this chapter a completely decentralized control hierarchy is proposed, where the individual agents are completely autonomous in making decisions regarding setpoint generation, obstacle avoidance and motion control. This requires all vital calculations to be executed by the local controller on the on-board computer without the need of a communication network.

In this case a supervisor is used just to increase the ease of use. Once the goal positions are known to the agent, all connections can be terminated and the agent will still find their way. In Figure 4.1 the distribution of responsibilities is shown accompanied with the corresponding communication network. Here G is a set of goal positions, r a reference path and velocity, r^{coll} a collision free reference path, u the robot input velocities and p the robot positions. The Reference generator and Tracking controller are already discussed in the previous chapter. The Reference modifier is discussed in this chapter. Thereto a novel technique to interpret detected obstacles is introduced in Section 4.1.1 and an accompanying collision avoidance method which is inspired on the work in [22, 23] is presented in Section 4.1.2. This latter method uses the current positions of obstacles relative to the agent to regulate the evasive action. In Section 4.1.3 a collision avoidance method is discussed which uses the observed velocities and corresponding directions of obstacles to determine the required avoidance.

4.1 Local collision avoidance

In Chapter 3 the desired trajectory is created under the assumption that no obstacles or other agents will be encountered. In order for the individual agents to be autonomous, a local collision avoidance algorithm is added. It is aimed to create a temporary change to the reference trajectory which is free of collisions, until the thread of collision is over or at least not observed anymore. Hereafter, a new reference trajectory is created on the

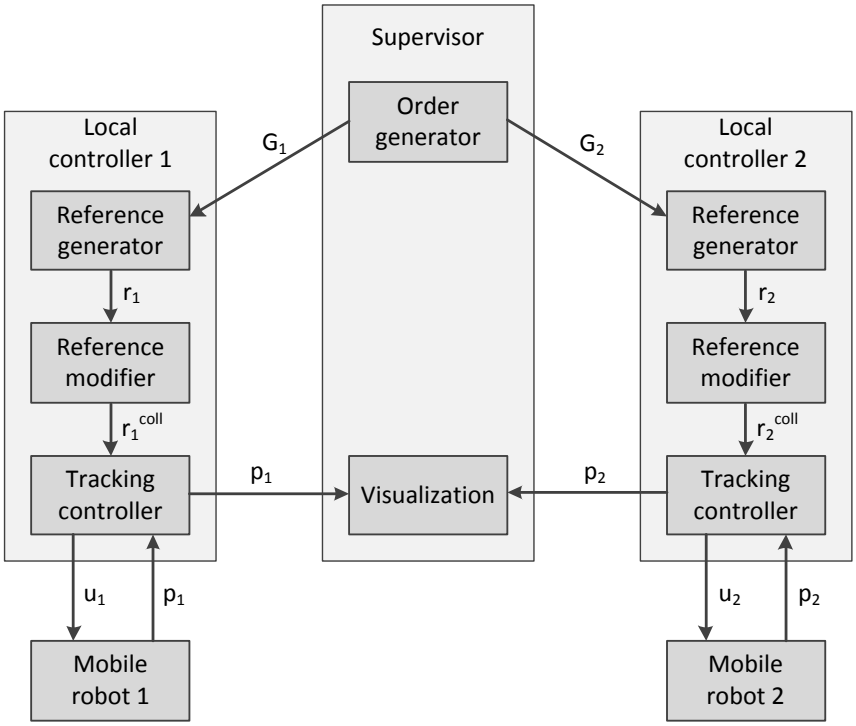


Figure 4.1: Low-level control architecture for two agents.

fly, based on the same assumption. This differs from a conventional collision avoidance algorithm which not alters the reference but acts on the level of the tracking controller. In this section, working mechanisms of two low-level collision avoidance algorithms are explained and the advantages and disadvantages are discussed.

4.1.1 Obstacle perception

In this project three kind of obstacles are considered: static obstacles, dynamic obstacles (e.g. humans) and associate agents. To evade such obstacles, the agents first have to interpret the incoming sensor data. In order to use information on the size of objects, an algorithm allocates the detected points to sets that represent an object. This is favorable since when all sensed points are considered as single objects a wall would induce a much higher evading action than a small object, which would be neglected.

Sets are composed of sensed points which are too close to each other to safely pass through them. Furthermore it is checked if the sensed points are significantly further away from the agent than the goal. If not, the sensed point is neglected. This allows goals to be in the vicinity of the objects. Multiple points in the same set are assumed to be connected by the object. This results in a series of objects with known (angular) size and position. To explain how the collision avoidance method handles these sets, various variables are introduced. The allocation of these variables varies for the situations where the object is located in either one or two quadrants of the vision field, as is represented

in Figure 4.2. Note that the entire field of view consists of the two frontal quadrants. The required variables are explained below and depicted in Figure 4.2, all angles are with respect to the heading angle of the robot.

α_L:	The most extreme left angle to the obstacle.
α_R:	The most extreme right angle to the obstacle.
α_{mean}:	The average angle to the obstacle which equals $(\alpha_L + \alpha_R)/2$.
α_{min}:	The absolute minimum angle to the obstacle.
α_{evade}:	The least extreme angle in order to evade the obstacle (α_L or α_R).
d_{obs}:	The minimum distance to the obstacle in the direction of α_{min}.
d_{evade}:	The minimum distance to the obstacle in the direction of α_{evade}.

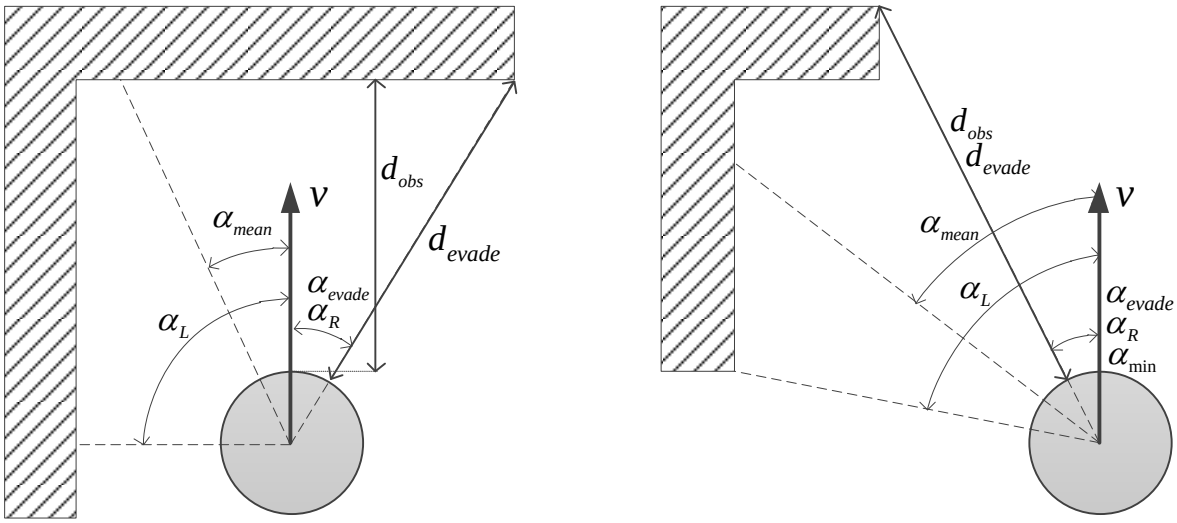


Figure 4.2: Graphical representation of the interpretation of an obstacle which is positioned in one (right) or both (left) quadrants of the vision field.

4.1.2 Position based collision avoidance

The collision avoidance algorithm used in this project is inspired on the work of [22–24]. The reference robot motion is on the fly altered by an artificial force. This force directs the robot forward velocity perpendicular to the heading angle of the robot, inducing an angular velocity. The magnitude of this force is determined using various weight factors, which are based on the distance to the obstacle, its angular width, and the angle between the middle of the obstacle and the heading direction of the robot. All weighting factors contain a monotonically increasing smooth function, as given in (4.1) and depicted in Figure 4.3.

$$\sigma_{a,b}(\zeta) = \begin{cases} 0 & \text{for } \zeta \leq a, \\ -\frac{1}{2}cos\left(\frac{\zeta - a}{b-a}\pi\right) + \frac{1}{2} & \text{for } a \leq \zeta \leq b, \\ 1 & \text{for } b \leq \zeta. \end{cases} \tag{4.1}$$

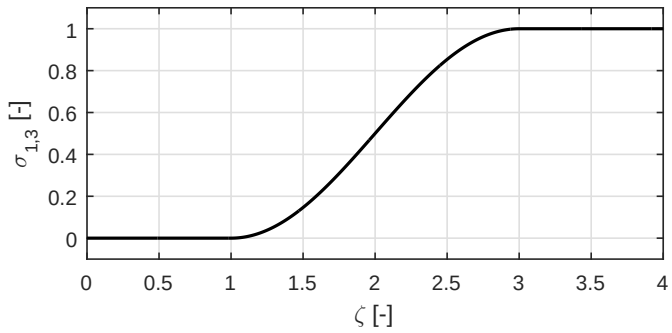


Figure 4.3: Continuous monotonically increasing weighting function $\sigma_{1,3}$.

The weight factor which depends on the distance between an obstacle and the robot has the following form (4.2)

$$w_d = \left(1 - \sigma_{d_f,d_b}(d_{obs})\right)\left(1 + \frac{d_l}{d_{obs} - d_{blind}}\right), \tag{4.2}$$

where d_{obs} is the distance to the obstacle, d_l, d_f and d_b are safety distances of increasing magnitudes and d_{blind} describes the blind spot of the agents in the proximity of the sensors. For reasons explained later, the value of d_b can vary for different obstacles. An example of the used safety areas around an agent is depicted in Figure 4.4. As depicted in Figure 4.5 this weighting function is zero for distances larger than d_b, slowly increases between d_b and d_f and goes to infinity as the distance goes to d_{blind}.

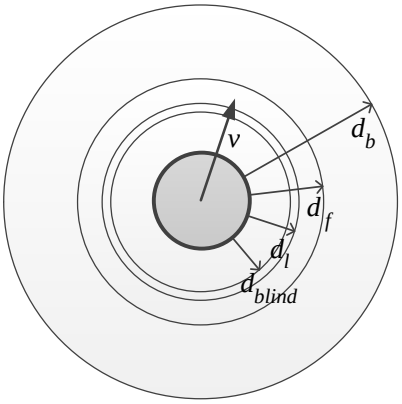


Figure 4.4: Graphical representation of the safety areas.

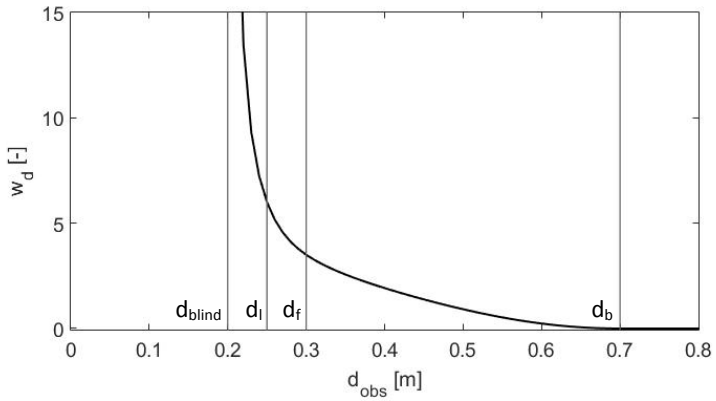


Figure 4.5: Distance depending weighting function.

The angular width of an object and the angle between the object and the heading direction are incorporated in another weighting function. This makes the collision avoidance not only more sensitive to larger objects but also to objects which are directly in front of the robot. The resulting weighting factor is given below.

$$w_\alpha = \left(1 - \sigma_{c_1+\gamma,c_2}\left(|\alpha_{mean}|\right)\right). \tag{4.3}$$

Here c_1 and c_2 are constants and

$$\gamma = atan2(R, d_{evade}) + |\alpha_{mean} - \alpha_{evade}|. \tag{4.4}$$

where R is the radius of the robot. As it can be seen in Figure 4.6, γ increases for wider obstacles resulting in a larger evading action.

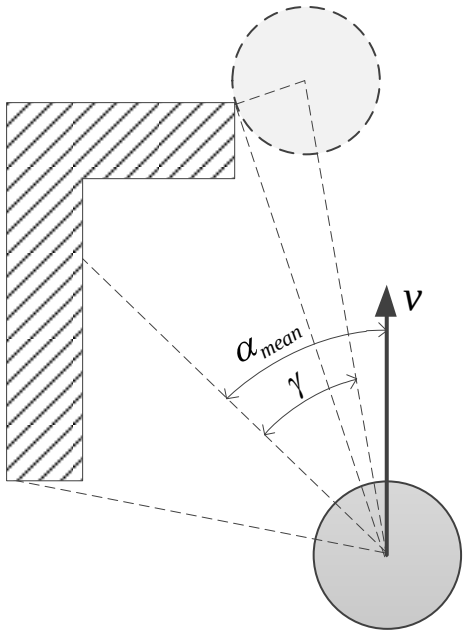


Figure 4.6: Graphical representation of the required angles for the weighting factor regarding the angles.

To determine the magnitude of the artificial force (τ) which takes care of the collision avoidance, the two weight factors are multiplied for all obstacles and then summed over the number of observed obstacles (N_{obs}). Finally, it is multiplied with an overall constant gain c_3, which allows one to alter the influence of the collision avoidance.

$$\tau_i = c_3 \sum_{k=1}^{N_{obs,i}} \left(dir(i,k) \cdot w_d(i,k) \cdot w_\alpha(i,k) \right). \tag{4.5}$$

Here $dir(i,k)$ is the direction of evasion of agent i with respect to object k, and equals ± 1 as calculated by:

$$dir(i,k) = sign\left(\alpha_{mean}(i,k) \right). \tag{4.6}$$

The direction of this force is chosen to point perpendicular to the right with respect to the heading direction. The magnitude can however be positive or negative as shown in (4.5). This on the fly calculated force is separated into global x- and a y-components that are added to the in advance calculated reference accelerations $\ddot{r}_x(t)$ and $\ddot{r}_y(t)$, respectively.

Finally, at each time step the new reference velocity and position are determined as follows

$$\begin{aligned}\ddot{r}_\tau(t) =& \ddot{r}_{pred}(t) + \tau(t),\\ \dot{r}_\tau(t) =& \dot{r}_\tau(t-\Delta t) + \ddot{r}_\tau(t-\Delta t)\Delta t,\\ r_\tau(t) =& r_\tau(t-\Delta t) + \dot{r}_\tau(t-\Delta t)\Delta t.\end{aligned} \tag{4.7}$$

Here, $\ddot{r}_{pred}$ is the (scaled) reference acceleration which is calculated in advance as explained in Section 3.1. Again this is done for both motion directions. When no obstacle is observed for several time instances, a new reference is created respecting the current position and velocities.

As discussed in Section 4.1 this collision avoidance algorithm alters the reference trajectory on the fly. This reference trajectory is used by the tracking controller given in (3.26), which guarantees global asymptotic stability of the tracking error dynamics (see[33]). However, in this report no mathematical prove is given that no collisions will occur when using this algorithm. It is recommended that this is done for further research. Nevertheless, numerous simulations and experiments have been performed from which can be concluded that the method seems to work well.

A drawback of this position based collision avoidance is the possibility that multiple agents evade each other in the same direction. This problem can be solved using velocity based collision avoidance, as is explained in the next section.

4.1.3 Velocity based collision avoidance

This method tries to predict the future course of an expected collision regarding the same type of obstacles as introduced in Section 4.1.1. To do so, this method calculates the intersection crossing order, by evaluating the local angular velocity of other agents and obstacles, as done in [21]. It uses the fact that when two approaching objects observe each other under a constant angle with respect to their heading angle, they will collide. Also, if the absolute angle to the obstacle is decreasing, the obstacle will precede the agent and vice versa. This allows one to locally determine the crossing order of the intersection points without needing to know the position and velocity of other agents. Due to these properties, the observed angular velocity of obstacles can be used to determine the direction and magnitude of the evading action.

As with the position based collision avoidance method, the detected obstacles are divided into sets, each representing a different object. Now the average angle to each obstacle (α_{mean}) is numerically differentiated over two consecutive time instances and it is compensated for the rotation of the observing agent i, as shown by (4.8).

$$\omega_{obs_k,i} = \frac{\alpha_{mean,k}(t) - \alpha_{mean,k}(t-\Delta t)}{\Delta t} + \omega_i(t-\Delta t) \tag{4.8}$$

This approximately results in the local angular velocities of all k obstacles. However, in order to compare the angle to an obstacle at different time instances, one has to

know which two sets represent the same obstacle at the two time instances. Since sets can merge, separate, disappear or be added between two instances, a transformation is created between the current sets and the previous ones. Only if a set is distinctively recognized in the previous time instance, without being split or merged, it can be used for this method. Other obstacles, that cannot be connected to the previously found obstacles are neglected for one time instance or can be subject to the position based collision avoidance algorithm.

Once the local angular velocity of the obstacle is known, it can be used in yet another weighting function w_ω, which is depicted in Figure 4.7. This function induces a more powerful evasive action for obstacles with low local angular velocities. To prevent switching behavior in the evading direction, the function is slightly asymmetric around $\omega_{obs} = 0$. For example, a simply rotating agent observing a stationary obstacle should find a local angular velocity of the obstacle equal to 0. However, due to the limited number of discrete sensors angles, the observed local angular velocity can be subjected to discretization errors. Since these errors can alternate between being positive and negative, this would create a rapidly vibrating evading action. Typical values for moving obstacles are well outside the asymmetric domain of the function. As can be seen, w_ω can be positive and

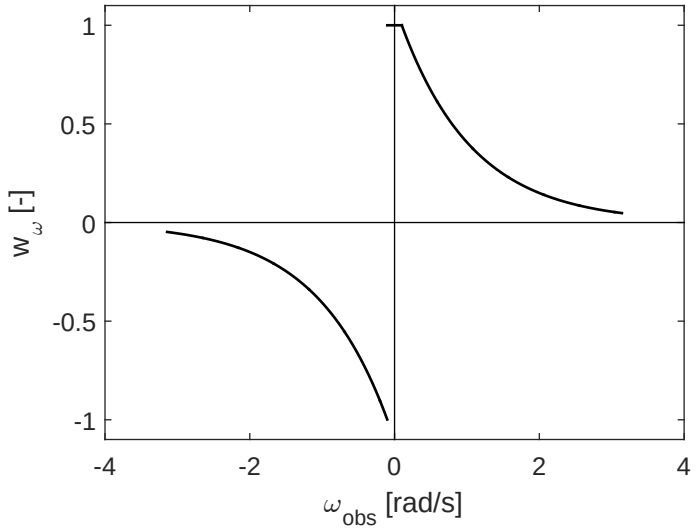


Figure 4.7: Weighting function depending on local angular velocity of obstacles.

negative and is added to the existing w_d and w_α derived in Section 4.1.2, resulting in the artificial force:

$$\tau = c_4 \sum_{k=1}^{N_{obs}} \left(w_\omega(k) w_d(k) w_\alpha(k) \right). \tag{4.9}$$

Here c_4 is a constant gain to set the overall influence of the collision avoidance algorithm. After implementation in the simulation, various problem arose for this method. First of all, there is a problem due to the finite measurement range of the sensors. When an obstacle is located at the edge of this range, the obstacle can be observed with more or

less points than in the previous time instance. This changes α_{mean}, resulting in a false local angular velocity. This can largely be solved by not evaluating the change of α_{mean} but by combining the changes in α_L and α_R; mathematically speaking, we can replace (4.8) with:

$$\omega_{obs_k,i} = \begin{cases} sign\left(\dot{\alpha}_{L,k}\right) \cdot max\left(\left|\dot{\alpha}_{L,k}\right|, \left|\dot{\alpha}_{R,k}\right|\right) + \omega_i(t - \Delta t) & \text{for} \quad sign(\dot{\alpha}_{L,k}) = sign(\dot{\alpha}_{R,k}), \\ 0 & \text{for} \quad sign(\dot{\alpha}_{L,k}) \neq sign(\dot{\alpha}_{R,k}). \end{cases} \tag{4.10}$$

Here $\dot{\alpha}_{L,k}$ and $\dot{\alpha}_{R,k}$ are the first order backwards derivatives of respectively α_L and α_R regarding obstacle k. The angular velocity is only registered if the outmost points on the left- and right-hand sides of the observed obstacle move in the same direction. All other changes are assigned to the time-variating number of observed points of the obstacle.

Another problem is that the accuracy of the estimated local angular velocity depends on the density of the sensing points. If the sensing resolution is limited this discretization error can be significant. The expected average of this discretization error is inversely proportional to the angular sensing resolution. The test setup used in the scope of this project consists of 12 sensing angels and also experiences this problem. More details on this setup can be found in Chapter 6. By means of simulations and experiments it is shown that this method is not applicable to the existing experimental setup, due to this large angular increment of the sensors.

Simulations are also performed where a device with higher angular resolution is considered, e.g. a laser range scanner. In Figure 4.8 the locally observed angular velocity of a static obstacle is shown. Hereto, a robot which purely rotates in front of a static obstacle is simulated. As it can be seen, in the simulation using 1000 sensors the error of the detected local angular velocity (e_ω) is significantly smaller than when 12 sensors are used.

In Figure 4.9 the robot paths around the obstacle are depicted that correspond to different sensor resolutions, are depicted. When 12 sensors are used, the agent collides with the obstacle. Note here that the lines shown in Figure 4.9 depict paths of the midpoint of the robot, which has a radius of 175 mm. The possible advantage of velocity based collision avoidance method becomes clear when one compares Figure 4.10 with Figure 4.11. Here the simulated robot paths are depicted that are achieved with the position based- and velocity based collision avoidance methods, both using 1000 sensors. The position based algorithm steers movements of both agents in the same direction, causing them to drive parallel to each other, while the velocity based method provides a more intuitive evasive action. The average required time to reach the goal positions is 55 seconds for the position based collision avoidance algorithm versus 37 seconds for the velocity based one. This simulation shows potential of the velocity based methods which recommends it for further research. Especially, since it uses information on the other agents without direct information exchange. However, as demonstrated this method is not effective for the test setup used in this project and therefore not further investigated in this report.

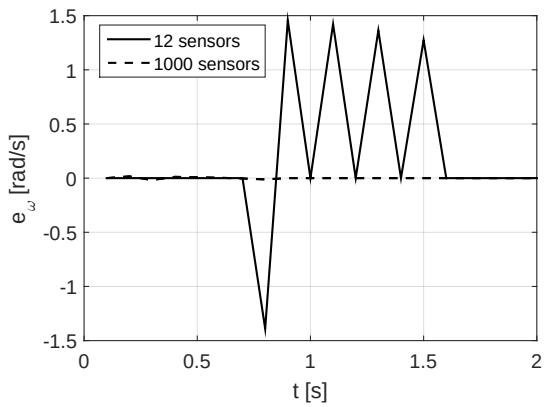


Figure 4.8: Observed local angular velocity of a static obstacle.

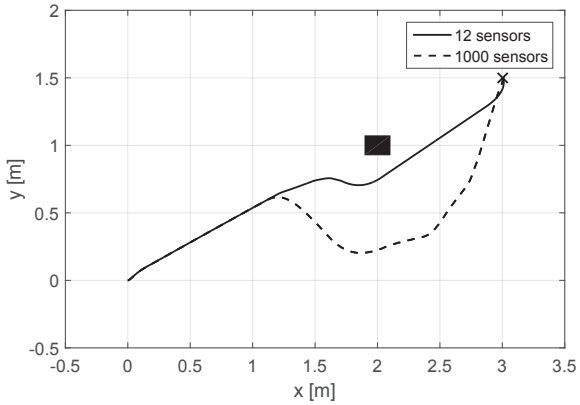


Figure 4.9: Resulting paths around a static obstacle.

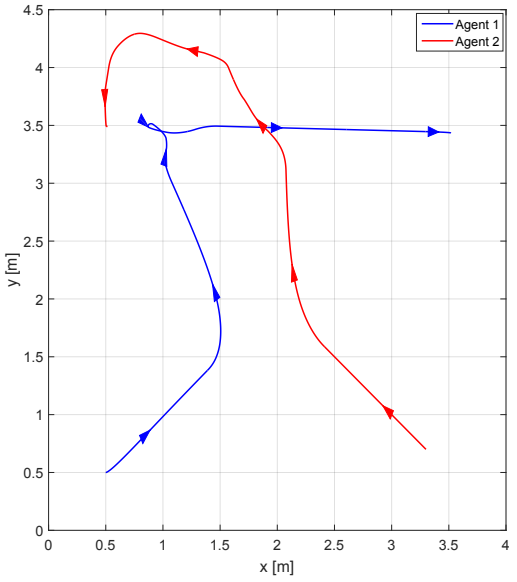


Figure 4.10: Resulting paths using position based collision avoidance.

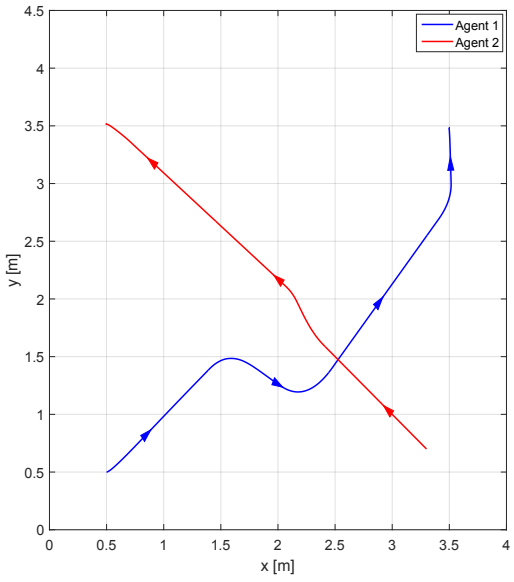


Figure 4.11: Resulting paths using velocity based collision avoidance.

4.2 Summary

In this chapter a technique to interpret detected obstacles and two different low-level collision avoidance methods are presented. This manner of obstacle perception creates sets of points which represent an object and derives the corresponding distance to the agent, angle of the object relative to the agent its heading angle and angular width of the object. These parameters are used as input for the collision avoidance methods, which use smooth weighting functions to calculate the desired magnitude of the evasion. The in Section 4.1.2 presented position based collision avoidance algorithm uses the current observed position of obstacles to determine an evading direction. This however, can result in inefficient evading maneuvers for dynamic obstacles. The velocity based collision avoidance method described in Section 4.1.3 is constructed in order to prevent this problem. This method utilizes the change in the detected angle to the obstacle with respect to the heading angle of the agent to compute an evading direction. This latter method however requires significant sensing capabilities and is therefore not applicable to the hardware configuration used in this project.

5

High-level control strategies

Various features that possibly could improve the efficiency of the transportation system require information about current and future states of the agents. That information can be supplied by an advanced supervisor which actively steers the system instead of only facilitating it. These high-level control methods are introduced in this chapter.

In Figure 5.1 the separation of tasks is given for the investigated high-level control strategies. Here G is a set of goal positions, f contains information on a possible formation, r is a reference path and velocity, r^{coll} a collision free reference path, u the robot input velocities and p the robot positions. The dashed lines are only required if the agents should be able to drive in formation. As can be seen, more information is being shared with other agents and the supervisor compared to the architecture depicted in Figure 4.1. The total number of communication channels can be limited by only exchanging data between agents when they are in each others vicinity. Since the inter-agent communication is solely used for collision avoidance and formation control, no information is required from agents positioned far away. In Section 5.1 collision avoidance methods are proposed that use inter-agent communication. In the Section 5.2 some additional features are discussed that could improve the time-efficiency of the system.

5.1 Communication aided collision avoidance

Here, two collision avoidance methods are presented that can work as a high-level layer on top of the low-level position-based collision avoidance method discussed in Section 4.1.2. For the sake of safety and robustness, these top layers are designed such that their functions are allowed to fail, for example when the communication network malfunctions. In this case the low-level controller will remain active and all agents will pursue their goals as described in Section 4.1. This becomes clear when one investigates Figure 5.1. When all inter-agent and supervisory communication would fail, the possibility of driving in a formation would be lost as well as the possibility to visualize the data in real-time. Besides that only the *Reference modifier* uses data about the other agents. It is key to not make this

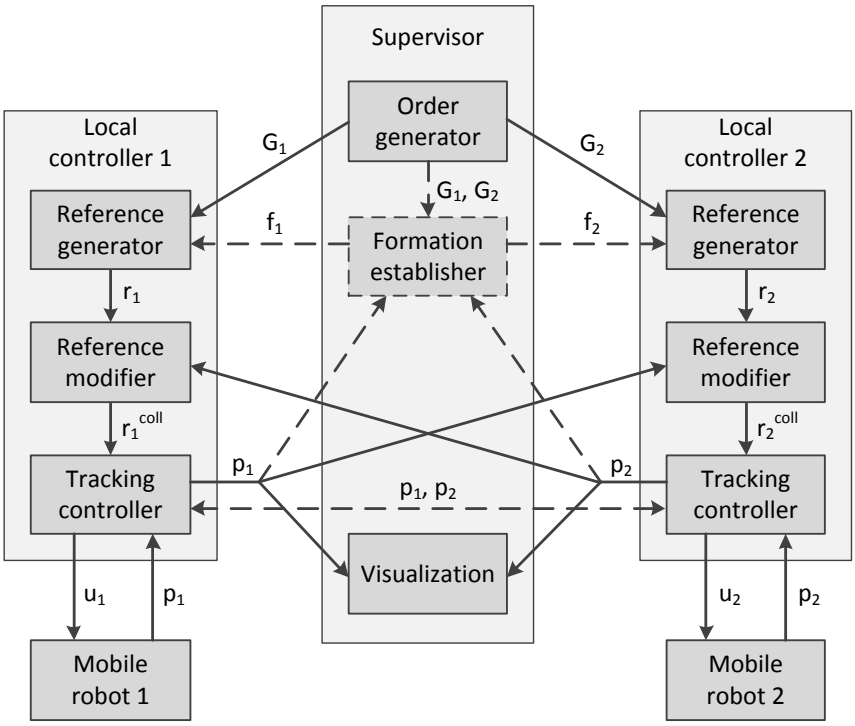


Figure 5.1: High-level control architecture for two agents.

task fully dependent on this information, by ensuring the agents can safely fulfill the given tasks without information on the other agents.

5.1.1 Waiting based collision avoidance

The manner by which multiple agents evade each other described in Section 4.1.2 is not efficient. It might be more efficient for agents to wait until the threat of collision is gone. This can reduce the traveled distance of the agents, decreasing the chance of detecting more obstacles to avoid. To do so, the agent needs to be able to distinguish other agents from static or dynamic objects. Therefore, when an object is detected, its position is communicated to the supervisor which checks whether this object correspond to an agent or obstacle. If the object is recognized as another agent by the supervisor, this is communicated back to the first agent together with the planned future reference trajectory of the detected agent. The first agent can now calculate if an (almost) collision is expected to occur with the detected agent. This is done by investigating whether the agents come within a certain distance from each other within a defined time period. If this is the case, the agent which is expected to reach the point of collision later than the other one is required to wait when it comes within a set distance from this point. The period of time it has to wait equals the expected time the other agent needs in order to reach this intersection point. This waiting method is an addition to the position based collision avoidance method described in 4.1.2 which is meant to provide back-up safety. If an obstacle is not (recognized as) an agent, this method still relies on the

low-level collision avoidance algorithm to evade it. When an agent is detected but no collision is to be expected, the range at which the low-level collision avoidance algorithm comes into play (d_b) is reduced for this agent and both agents continue their motions. This requires different settings regarding the low-level collision avoidance for different detected objects. If all agents are requested to wait a deadlock can occur. In such an event, the agent for which is waited for the most is mobilized again. In Figure 5.2 a simplified flow diagram for this collision avoidance method is depicted. If no communication between an agent and the supervisor is established, then the agent does not recognizes the detected objects as the agents. As can be seen in the first step of this diagram, this causes the low-level collision avoidance to be active.

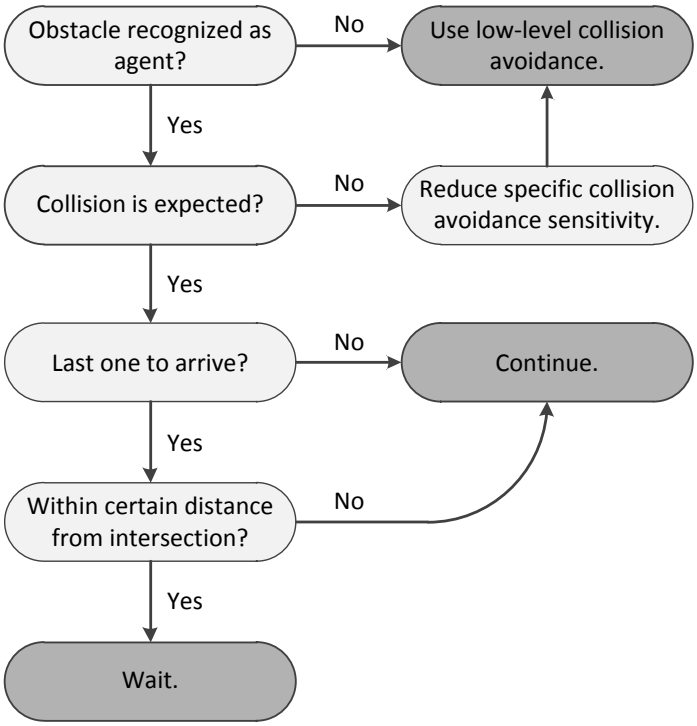


Figure 5.2: Simplified flow diagram for the waiting based collision avoidance.

5.1.2 Evading based collision avoidance

The collision avoidance method described in Section 4.1.3 aims to evade an obstacle in the most efficient direction. As discussed in the same section, issues concerning the determination of the intersection crossing order may arise when this is tried to be determined locally by the sole agent. These issues can be overcome by means of inter-agent communication. Similarly as in the previous subsection, when an agent is recognized, it's planned future reference is communicated to other agents and in the case of an anticipated collision the expected times of arrival at this intersection point are

calculated. In mathematical terms (4.5) can be replaced by (5.1).

$$\tau_i = c_3 \sum_{k=1}^{N_{obs,i}} \left(dir(i,k) \cdot w_d(i,k) \cdot w_\alpha(i,k) \right). \tag{5.1}$$

Here dir represents the evading direction and equals ± 1, w_d and w_α are the weighting functions as given in 4.2 and 4.3 respectively. If agent i recognizes the k^{th} observed object as an agent (j) in the thread of their collision, the corresponding evading direction is calculated as follows:

$$dir(i,k) = sign(\alpha_{int,i} - \alpha_{i,j}) \cdot sign(t_{int,i} - t_{int,j}). \tag{5.2}$$

Here t_{int} is the arrival time at the intersection point, while the angles are defined in Figure 5.3. If this obstacle is not identified as an associate agent, the direction is calculated as in the decentralized case, see (4.6). In the example sketched in this figure both agents

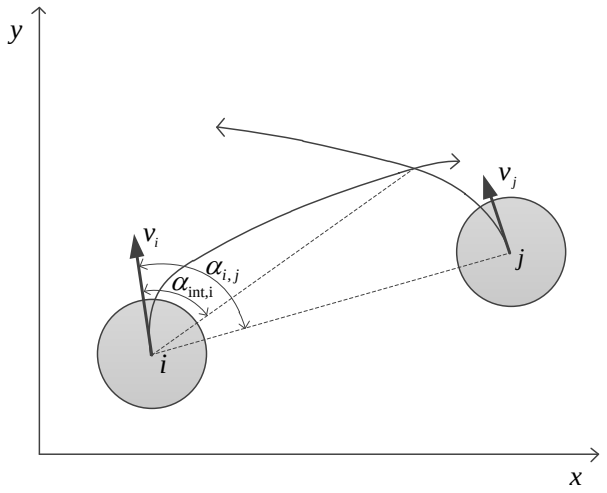


Figure 5.3: Explanation on evading direction of agent i.

evade each other at their right-hand side, while in the decentralized case described in Section 4.1.2 agent i evades to the opposite side. In order to prevent intensive switching of the evading direction, this direction is fixed until either of the future references is changed. To enable use of the fixed directions, the evading direction has to be stored for every obstacle which is recognized as an agent. A simplified flow diagram of this evading method is shown in Figure 5.4. Also in this diagram can be seen that when the communication between the agent and the supervisor fails, the low-level collision avoidance is used.

In rapid changing environments the evading algorithm might perform less effective. Since the forward velocity of an evading agent is not scaled down while the angular velocity is saturated, there exists a minimum evading radius. If two agents are close to each other this might lead to an unfeasible evading action, resulting in a collision. A possible remedy is to scale down the forward velocity when evading, however this greatly

complicates the determination of the expected future reference on which this algorithm is based. Due to time restrictions, this remedy is not explored in the scope of this project, but it might be a worthwhile option for future research.

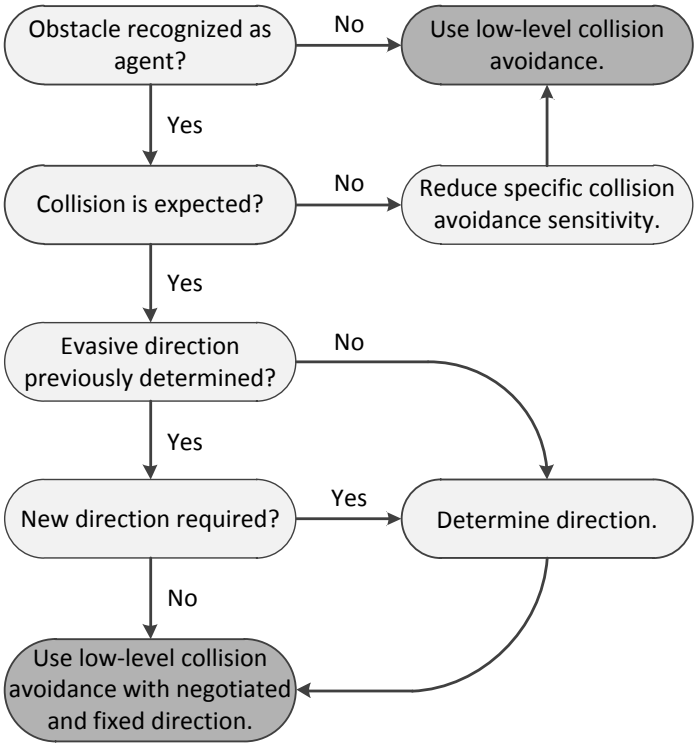


Figure 5.4: Simplified flow diagram for the evading based collision avoidance.

5.2 High-level features

This section describes some functions that rely on multi-agent communication which could possibly improve the performance of the transportation system.

5.2.1 Formation control

When multiple agents travel to (approximately) the same target position, it might be beneficial to let them drive in formation. In this case, as discussed in Section 3.1.2, agents can be assigned to be leaders or followers. The allocation of formations and the corresponding individual roles can be done on the fly, when several criteria are met. Criteria a, b, and c given below have to be met at the time the formation is established, criteria d and e have to hold all the time the formation is in place. If not, the formation is aborted and the agents continue as individuals. The threshold given below are the ones used in this project and are empirically determined.

a. The agents are not yet part of a formation.
b. The distance between the agents is at most half of the distance to their goals.
c. The distance between the agents their goal positions is at most half of the distance from the agents to their respective goals.
d. The difference of the global angle from the agents to their goal is smaller than $\pi/4\,[rad]$.
e. The smallest distance from an agent to its goal is larger than $0.5\,[m]$.

When an agent is assigned to join the formation as follower, it still needs to evade obstacles. In the thread of a collision, the follower neglects its leader if their relative distance is not smaller than their designated formation distance. The detection range for collision avoidance with the leader ($d_{b,l}$) is reduced to a value smaller than $(||\mathbf{l_f}|| - R)$, where R is the radius of the robot and l_f the vector as used in Section 3.1.2. With this setting the follower normally evades obstacles and other agents evades its leader only when necessary. The formation control law is discussed in Section 3.2 and is considered to investigate the effect of allocating priorities to groups of agents instead of to the individually robots.

In the case of the waiting algorithm described in Section 5.1.1, only the leaders evaluate the possible future collisions with other agents. If multiple leaders are expected to collide, the group which has to wait is chosen such as to achieve the minimum total waiting time. We determine that time by multiplying the number of agents with the time they have to wait. For the evading algorithm the leaders locally decide their evading direction, while the followers pursue their reference and use the low-level collision avoidance to prevent collisions.

5.2.2 Consensus on target positions

Since the current goals of all agents are known by the supervisor, the supervisor can be used to predict possible sub-optimal situations. For instance, if multiple agents have the same target position the supervisor can calculate who is expected to arrive first and the goal position is reserved for this agent. If another agent is planning to arrive at this goal later, a temporarily goal is created at a fixed distance from the original goal which acts as a buffer. If due to any circumstance the agent which claimed the goal is delayed, other agents can occupy the goal. This prevents multiple agents to perform idle movements around the same goal.

5.2.3 Communication on unexpected obstacles

If an unexpected obstacle, such as a broken agent, is detected this can be communicated with the other agents in the neighborhood. These agents individually check if their predicted future reference trajectories come close to this obstacle, and if necessary create a strategically placed waypoint. This results in a detour which leads the agent around the obstacle. In Figure 5.5 the effect of this algorithm is depicted. On the left-hand side, paths of the agents are shown that are achieved using the low-level collision avoidance

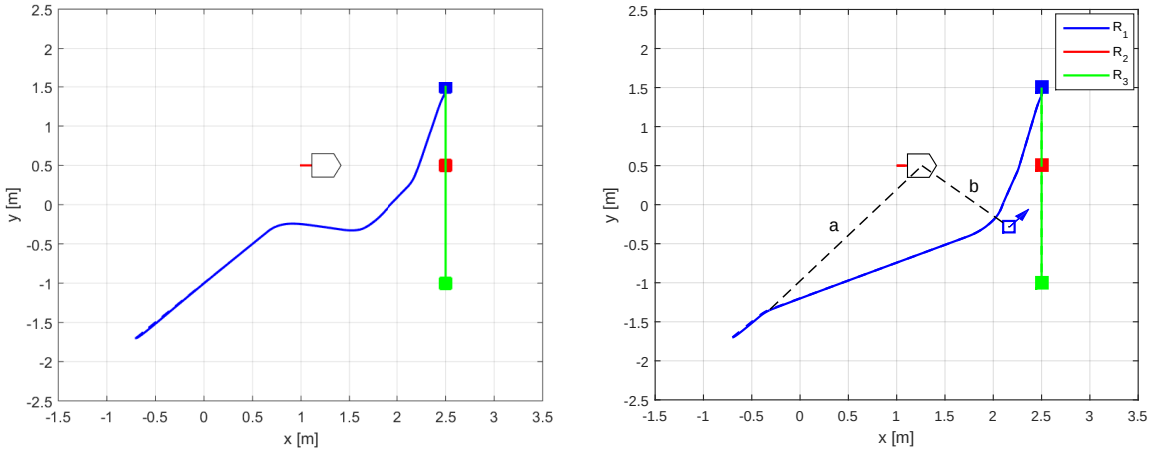


Figure 5.5: Broken agent with (right-hand side) and without (left-hand side) waypoint induced detour.

algorithm. On the right-hand side , the paths are shown when the position of the broken agent is communicated. Here, agent 2 (in red) breaks down which is eventually observed by agent 3 (in green). This is communicated to agent 1 (in blue), which is then redirected. The reference path of agent 1 is altered by inserting the waypoint represented by the blue open square. This waypoint is located at a set distance on line b, which is perpendicular to the line between the broken and the redirected agent (line a). This leaves two possible positions for the waypoint; here, the one closest to the final goal (filled blue square) is chosen. The set goal velocity at this waypoint, represented by the blue arrow, equals the maximum allowable reference velocity in the same direction as line a. In simulations, this algorithm gives smoother robot paths than when the algorithm is omitted. However, no significant benefit regarding time of motion are found. Surely the algorithm could be optimized, resulting in a shorter path and thus increasing the time-efficiency. Due to time restrictions, this is not done in the scope of this project.

5.3 Summary

In this chapter two collision avoidance algorithms are introduced that consider the current and expected future robot positions of the other agents. The accompanying hierarchical structure is designed such that it is robust against failure of agents and communication channels. In the absence of information on the other agents the low-level control will remain active. Therefore, failures of agents or communication channels would influence the efficiency but not the safety of the system. Furthermore, other high-level features such as formation control, consensus on target positions and communication on unexpected obstacles are presented that can increase the performance of the overall transportation system. These high-level collision avoidance methods and features demand higher computational power, which is going to be taken into account when evaluating the control methods in Chapter 7.

6

Experimental setup

To enable experimental evaluations, a test setup is developed. The hardware of this framework is based on the work of Michel Legius presented in [34]. In this project, three mobile robots are used which are described in Section 6.1. The inter-agent communication network is described in Section 6.2. The test environment used in this project is presented in Section 6.3.

6.1 Hardware components

In this project, Turtlebot 2 mobile robots are used which is an adapted version of the Kobuki platform. This circular robot with a diameter of 354 mm weighs 6.3 kg and has a maximal forward velocity of 0.65 m/s, see [35] for more information. This low cost robot allows integration of additive sensors and is capable to carry an external computer. This setup is equipped with 12 analog infrared proximity sensors (Sharp GP2Y0A02YK0F), which properties are listed in [36]. These sensors are integrated on the frontal edge of the robot, creating a discretized field of view of 180°. The analog output signals of these sensors are guided to an Arduino MEGA 2560 microcontroller [37], where these signals are converted into digital signals. This microprocessor is powered by the Kobuki's internal battery. Since there is some interference between the input voltages, the range of reliable sensor measurement is significantly less than advertised. For distances between 20 and 70 cm their measurements are considered as reliable, outside this range the data is neglected. The positioning of the agents is based on the local odometry of the Kobuki robots.

Furthermore, each agent contains an on-board computer to run the low-level control software, interpret the sensor data and communicate with other entities in the system. An Intel NUC D54250WYK computer is used for these tasks. This compact computer contains an Intel Core i5 processor. It is connected to the microcontroller and the mobile base via USB cables, and includes a WiFi adapter for further communication. This on-board computer is powered by an external powerbank [38]. The complete robot is depicted

in Figure 6.1 and a more detailed picture of the sensor assembly is given in Figure 6.2. To establish a network of agents and a supervisor, an Asus rt-ac66u wireless network router is used. This router is solely used by the setup to prevent data overflow, which could result in time delays or the loss of data.

For the high-level control performed by the supervisor, an external laptop is used. For this purpose, a HP Compaq 8510w is chosen.

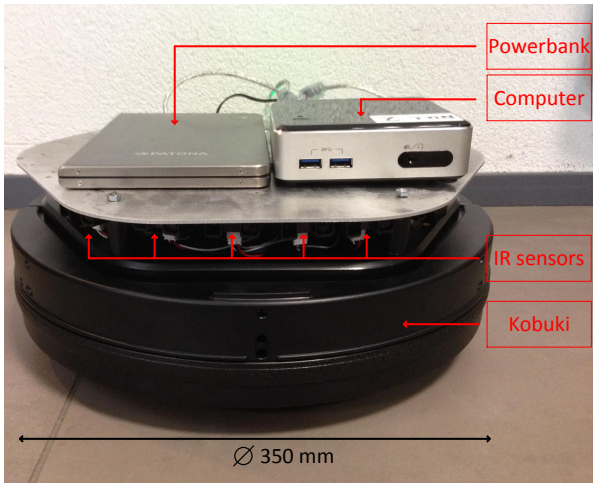


Figure 6.1: Experimental setup of one agent.

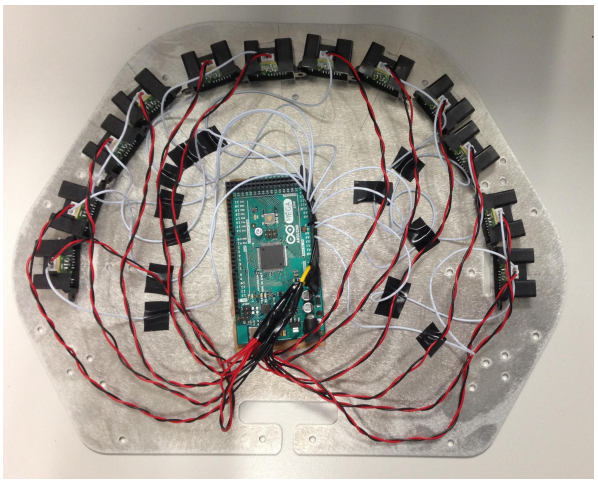


Figure 6.2: Infrared sensors and Arduino board of an agent.

6.2 Network configuration

A modular network structure is developed to accommodate varying number of agents that participate. That facilitates the addition or removal of agents such as to accommodate on the fly generated requests on throughput of the system.

The local controllers on the individual agents run in MATLAB 2013b (64x) [39] and by using a ROS I/O package [40] communicates to a local ROS Hydro network [41]. Here, the ROS master is chosen to be located on the on-board computers. This ensures that each agent contains an identical software module. Using an open/source environment, the Turtlebot is included in the ROS network and the message passing is done through ROS. The supervisory controller is equipped with MATLAB 2015a which by the use of the Robotics Systems Toolbox [42] is able to connect to multiple ROS masters. A schematic representation of the obtained network for two agents is depicted in Figure 6.3. To circumvent the need for an internet connection, the supervisory computer is used as a local server. A shared folder is created which is accessible for all agents in the local network. This ensures that all agents use the same software and simplifies software adjustments on the local computers. Since these on-board computers have no display, a

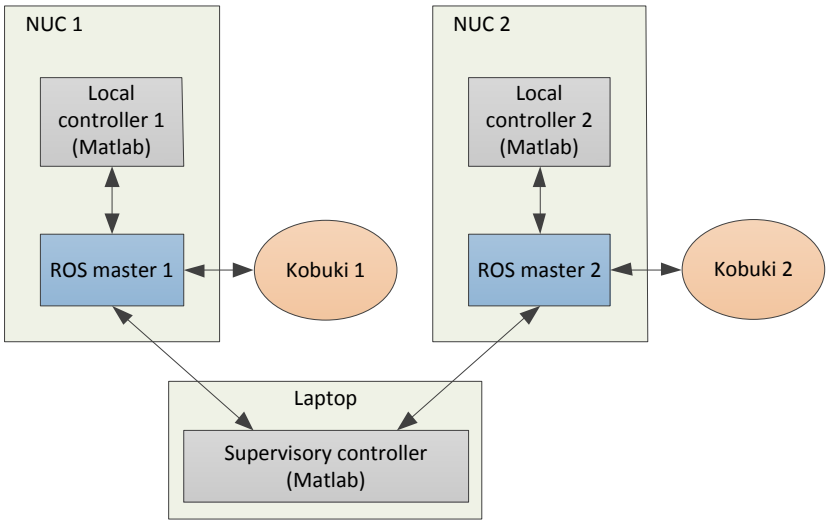


Figure 6.3: Network setup for two agents and a supervisor.

remote desktop application is used to operate these computers without connecting them to an external screen.

6.3 Test environment

In consultation with Van den Akker Engineering a representative test environment is created to perform simulations and experiments. In Figure 6.4, we depict the resulting arena where several starting points (S), workstations (WS) and objects are placed. Sets of consecutive way points can be assigned to each agent to specify their individual tasks.

6.4 Summary

The robot hardware described in this chapter is chosen because of its versatility, low cost and adequate size to carry a computer and the required sensors. The sensors are chosen for their ease of implementation, wide collective field of view and again low cost. A small-sized but powerful on-board computer is chosen, such as to allow execution of computationally intensive navigation and control algorithms; this in turn makes it possible to investigate different sensory configurations and algorithms in the future without concern about available computational power. The inter-agent communication is achieved by a modular network structure, which facilitates scaling of the number of robots. The performance of these robots is tested in an area which is a representative of the operation environment as intended by Van den Akker Engineering.

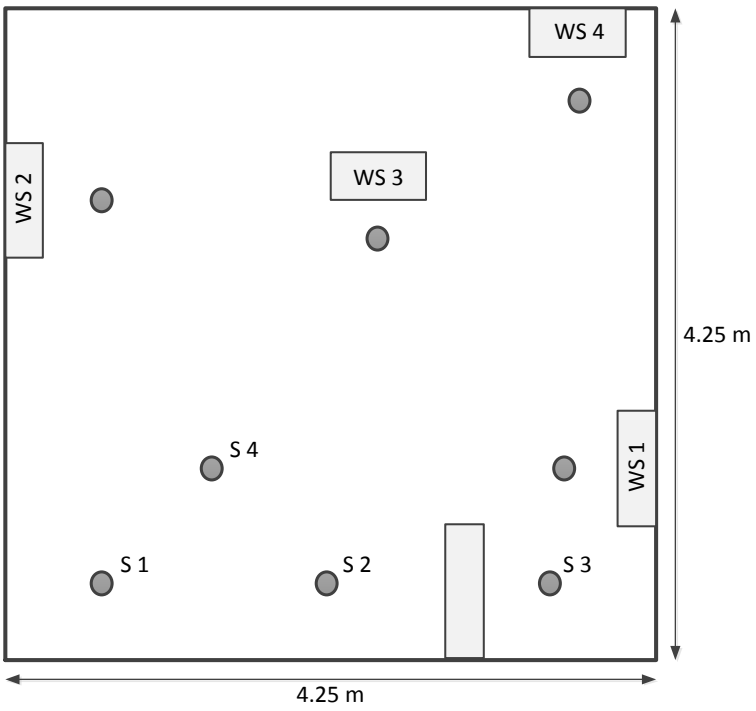


Figure 6.4: Top view of the test arena.

7 Results and discussion

In order to test and compare the various control hierarchies described in the Chapters 4 and 5, several experimental and simulation case-studies are performed. In these simulations, the test environment described in Section 6.3 is mimicked. This chapter contains the validation, evaluation and comparison of the control algorithms.

In the next section the simulation environment is discussed and validated. In Section 7.2 it is explained how the control architectures are tested. The performance indices for evaluation of the control algorithms are explained in Section 7.3 and in Section 7.4 the achieved simulation results are compared and discussed.

7.1 Validation

The simulation study helps the development of control algorithms and is a convenient instrument to examine the effect of software changes. However, there are several differences between the simulation and experimental environments. First of all, in simulations the robots are described by the kinematics model only, where the effects of mass, inertia, friction and slippage are neglected. Also, experiments are affected by errors made in the initial placement of the robots by the human operator, since the agents calculate their position relative to this initial position by relying on servo feedback information. Together with possible slippage this placement error causes localization errors which do not exist in the simulation environment. The error caused by the finite resolution of the wheel encoders, which provide the servo feedback, are assumed to be neglectable. Furthermore, the detection of obstacles and other agents with the use of sensors is mimicked in the simulations. Due to computational costs, these virtual sensors have a lower resolution than the real ones but are not subjected to noise as is the case with the real sensors. Finally, in the simulations a fixed sampling time is used, which is also pursuit but not always achieved during the experiments.

In order to verify whether the simulations can be used as an instrument for performance analysis, the simulation environment is validated by comparing the outcome of tests

executed in both simulations and experiments. First, a simple straight line movement over $2\,m$ is made to show the basic differences between the two test approaches. In Figure 7.1 it can be seen that the position errors in the experiments as described in (3.24) increase during the acceleration phase, while the input forward velocity exceeds the reference velocity, as shown in Figure 7.2. The errors in the experiments can be attributed to the effect of the robot inertia, which is not incorporated in the simulations. Also during the constant forward velocity there is a difference between the simulation and experimental results. According to the kinematic model, the change of the translational errors (e_x and e_y) during the constant velocity phase should be equal to the surface area between the input velocity and reference velocity in the respective directions over this period of time. For the x-direction this yields:

$$\Delta e_x = e_x(t_{b2,x}) - e_x(t_{b1,x}) = \int_{t_{b1,x}}^{t_{b2,x}} \Big(v_r(t)cos(\theta_r(t)) - v(t)cos(\theta(t)) \Big) \Delta t. \tag{7.1}$$

In this case the error should decrease with $0.18\,m$ while only a decrease of $0.03\,m$ is obtained in the experiment. This difference can be explained by the friction which exists in the experiments and not in the simulation, resulting in a difference between the input velocity and achieved velocity of the robot. Finally, a rotational error is also present which is likely caused by the slippage, neglected inertia and again friction. Although the aforementioned differences between the simulation and experimental cases are present, these differences are small in comparison with the distances traveled by the robot. In Figure 7.3, the Cartesian and angular robot motions are plotted against time, and are roughly similar both in the simulation and experiment.

Also the detection of, and anticipation on, obstacles are validated. Experiments where a static obstacle is evaded are executed multiple times and compared to the simulation. The repetition of the experiments is done to average out differences in initial position, slippage and sensor input. In Figure 7.4 the resulting paths are shown, together with the position of a square object which is evaded. As it can be seen, the path obtained in the simulation is similarly shaped as the experimental one. Some apparent differences can again be attributed to the effects of friction, neglected inertia and slippage.

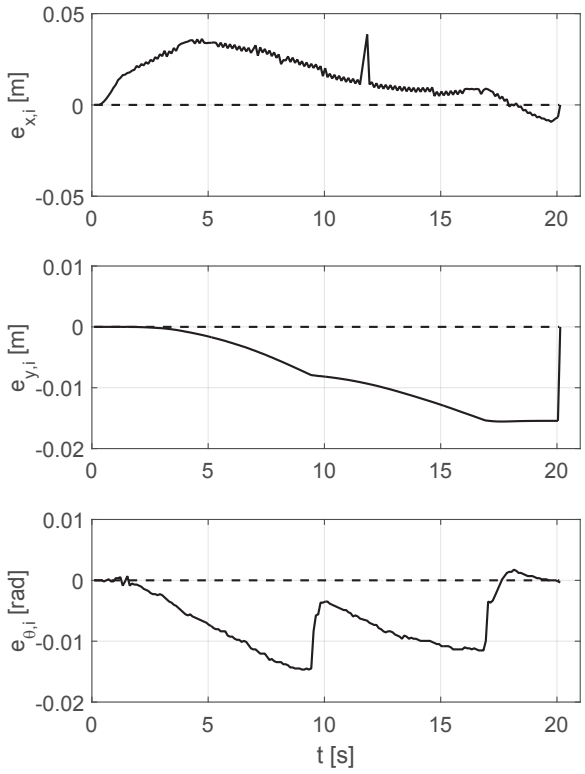


Figure 7.1: Position and orientation errors in simulation (dashed) and experiment (solid).

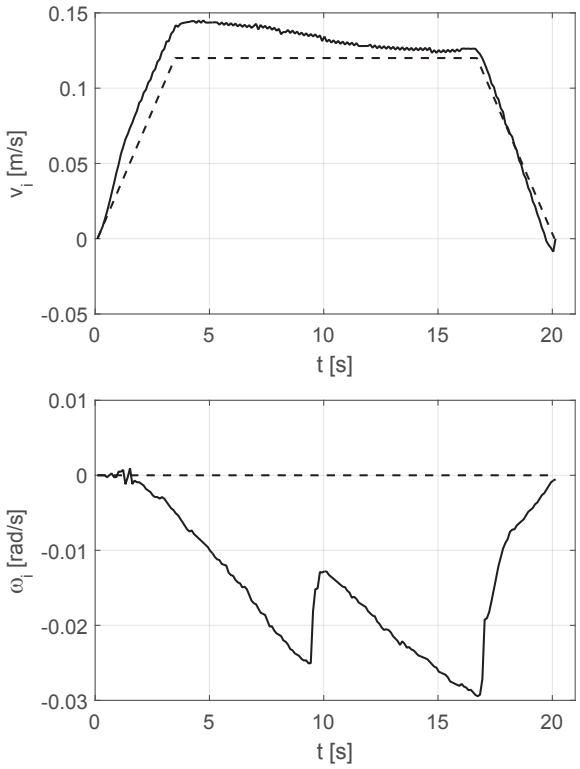


Figure 7.2: Input robot velocities in simulation (dashed) and experiment (solid).

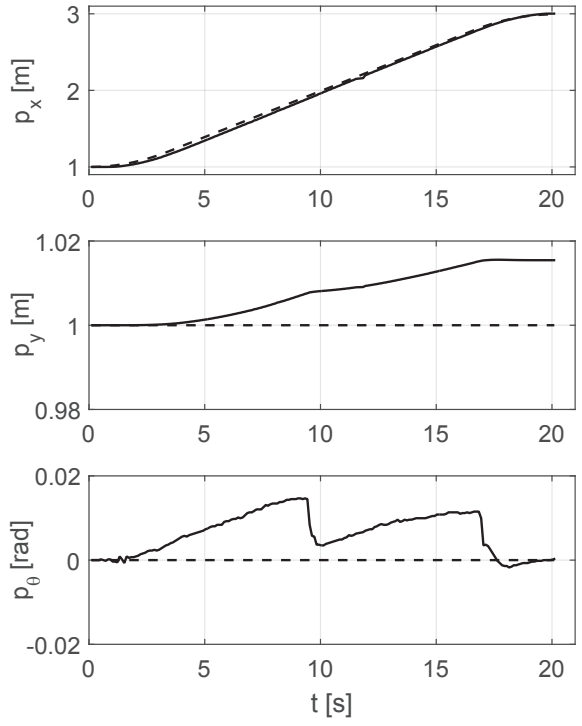


Figure 7.3: Cartesian and angular robot motions in a simulation (dashed) and an experiment (solid).

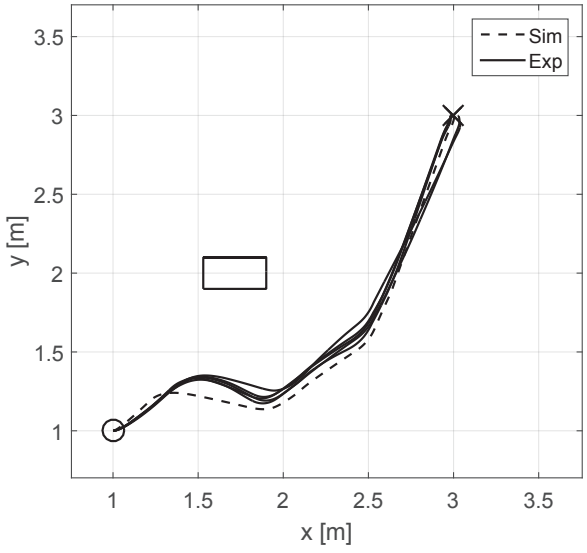


Figure 7.4: Paths of an agent evading a rectangle obstacle in simulation (dashed) and experiments (solid).

7.2 Monte Carlo analysis

Various experiments are performed where the agents are routed from the starting position to the workstations and back to their initial positions. This is done using the different control methods. Critical inspection of the obtained results reveals that the experimental results very much depend on the initial conditions. Also the precision of the manual positioning on the initial locations can greatly affect the outcome of the experiment. This sensitivity is shown in a simple experiment where an obstacle is placed such that it is on the edge of being evaded at the left- and right-hand side. This experiment is repeated 5 times, where it is tried to position the robot identically. The achieved robot paths are shown in Figure 7.5, where the agent starts at the circle and drives to the goal indicated by the cross. As it can be seen, the small differences in the initial positioning together with the friction, slippage, sensor noise input and/or sample frequency variations can result in different outcomes. Since executing enough experiments in order to average out these effects is not practical, simulations are partly used for evaluation of the different control algorithms.

Thereto, a Monte Carlo simulation analysis is used where an initial position and a set of goals is quasi-randomly chosen for each robot. This quasi-random way means that the chosen positions have to adhere to a couple of rules to prevent deadlocks and initial collisions:

1. Each robot has a unique initial position;
2. The final goal position of each robot is not part of the sets of goal positions of all other robots;
3. No identical subsequent goal positions of the individual robots are allowed;
4. Each simulation scenario appears only ones.

For each initial position and the set of goals, simulations are performed using the various control methods. This process is repeated until numerous simulations are done, in this case 137 unique simulations are performed. All starting positions and workstations depicted in Figure 6.4 can be used as the initial and/or goal position.

7.3 Quantitative performance evaluation

Several criteria are used to assess the performance of a control algorithm. These indices are listed and explained in this section.

Time required

First of all, the time required to execute the given robot task ($\overline{t_{evade}}$) is considered. The main difference between the algorithms is the method used to avoid collisions. To emphasize these differences, the execution time achieved with each algorithm is reduced with the time achieved in a zero measurement. This zero measurement represents the minimum time required to complete the task when no obstacles have to be evaded. This

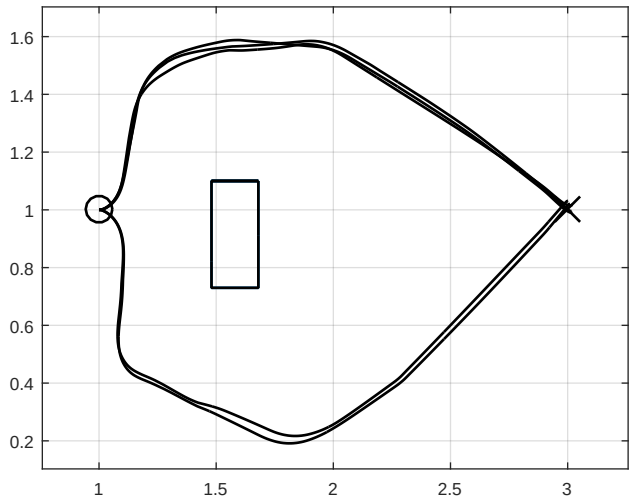


Figure 7.5: Paths of repeated experiments.

is calculated by simulating the task where the collision avoidance algorithm is disabled and agents pass through objects and other agents. This yields the shortest and fastest possible trajectories which are not necessarily safe but do obey the nonholonomic constraints and input velocity limits. This minimum time is subtracted from the times achieved by the individual agents which than is averaged over the number of data sets:

$$\overline{t_{evade}} = \frac{1}{nN} \sum_{j=1}^{N} \sum_{i=1}^{n} \left(t_{end,i,j} - t_{min,i,j} \right). \tag{7.2}$$

Here $t_{end,i,j}$ is the time agent i spends to perform experiment j, $t_{min,i,j}$ is the time obtained in the zero measurement for agent i and task j, n is the number of agents and N the number of simulations.

Distance covered

Secondly, the distance which agents travel to evade obstacles ($\overline{d_{evade}}$) is used to quantify the different control schemes. Thereto, the total distance covered is adjusted with the distance covered in the zero measurement, where nothing is evaded. This normalized averaged distance is calculated as follows:

$$\overline{d_{evade}} = \frac{1}{nN} \sum_{j=1}^{N} \sum_{i=1}^{n} \left(d_{i,j} - d_{min,i,j} \right). \tag{7.3}$$

Here, $d_{i,j}$ is the distance covered by agent i in simulation j and $d_{min,i,j}$ is the minimum distance required for agent i to perform this task in the baseline simulation.

Obtained tracking error

Furthermore, the tracking performance of the reference robot motion ($\overline{e_{evade}}$) is evaluated for all control methods. We consider the average position error which is compensated for the mean position error made in the zero measurement and subsequently averaged over all agents and simulations, as done below:

$$\overline{e_{evade}} = \frac{1}{nN} \sum_{j=1}^{N} \sum_{i=1}^{n} \left(\frac{1}{t_{end,i,j}} \int_{t=0}^{t_{end,i,j}} |e_{i,j}(t)| dt - \frac{1}{t_{min,i,j}} \int_{t=0}^{t_{min,i,j}} |e_{min,i,j}(t)| dt \right). \quad (7.4)$$

Here, $e_{i,j}$ and e_{min} are the time dependent Cartesian differences between the actual and reference positions of the agents in simulation j and in the corresponding zero measurement respectively.

Computational effort

The next performance criterion is the computational cost of the algorithms, which is given by an average calculation time of control signals ($\overline{T}$) in experiments. This is done based on experiments rather than in simulations, to achieve realistic data. Since the calculations are done both by the local controller as by the supervisory controller, the total computational time is averaged over the number of agents. This is useful since it tells which algorithm is the most convenient to scale up or demands the least powerful computer. This minimum required calculation time is expressed as follows:

$$\overline{T} = \frac{1}{nN} \sum_{j=1}^{N} \left(\overline{T_{sup,j}} + \sum_{i=1}^{n} \overline{T_{local,i,j}} \right). \quad (7.5)$$

Here, $\overline{T_{sup,j}}$ is the average time required for the supervisor to fulfill its tasks during experiment j and $\overline{T_{local,i,j}}$ is the average execution time of the local controller of agent i during experiment j.

Energy efficiency

For the final performance criterion an average theoretical power consumption ($\overline{P}$) is calculated in order to evaluate the energy efficiency of the algorithms. This is done by examining the theoretical mechanical power which is needed to generate the velocity profiles that are obtained in experiments:

$$\overline{P} = \frac{1}{nN} \sum_{j=1}^{N} \sum_{i=1}^{n} \left(\frac{1}{t_{end}} \int_{t=0}^{t_{end,i,j}} \left(m_i \frac{dv_{i,j}(t)}{dt} v_{i,j}(t) + I_i \frac{d\omega_{i,j}(t)}{dt} \omega_{i,j}(t) \right) dt \right). \quad (7.6)$$

Here m_i is the mass of the robot and I_i is the mass moment of inertia of the robot which is assumed to be a cylinder with a uniformly distributed mass. Furthermore, $\frac{dv_{i,j}(t)}{dt}$ and $\frac{d\omega_{i,j}(t)}{dt}$ are the first order derivatives of respectively the forward and angular robot velocities. These time derivatives are determined by numerical differentiation, which causes

high frequent components in the resulting accelerations that are not present on the real robots. To reduce the presence of these components, non-causal low-pass filtering is applied. This filtering technique prevents phase distortions of the filtered data and is possible since the filtering is performed on the recorded data instead of real-time data. The experimental data is used to incorporate the effect of the robot inertia and better represent the actual dynamics of the system. In Figure 7.6 the acceleration signals determined by numerical differentiation of the experimentally measured robot forward velocity is shown before and after low-pass filtering together with the reference acceleration profile. As it can be seen, the acceleration signal before the filtering contains some high frequent components. When the frequency spectrum of this signal is compared with the spectrum of the reference acceleration, see Figure 7.7, one can observe differences between these spectra at high frequency that gives us a hint about a cutoff frequency of the low-pass filter which is needed to suppress the high frequent components. Based on the signal spectra we design a fourth order Butterworth filter with a cutoff frequency of $3\,Hz$, which is then used filter out the acceleration signal determined by numerical differentiation. The filtered signal is also presented in Figure 7.6. Here, we should make a remark that not all high frequent components are necessary caused by numerical differentiation. Some might be present in the actual robot dynamics. Consequently, the filtered signal might be a conservative estimation of the real robot acceleration.

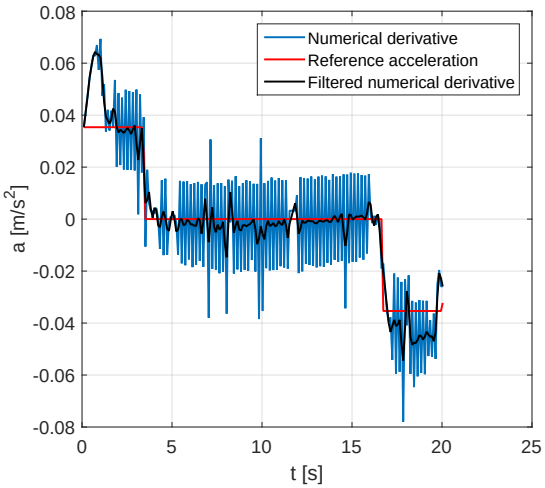


Figure 7.6: Acceleration signals

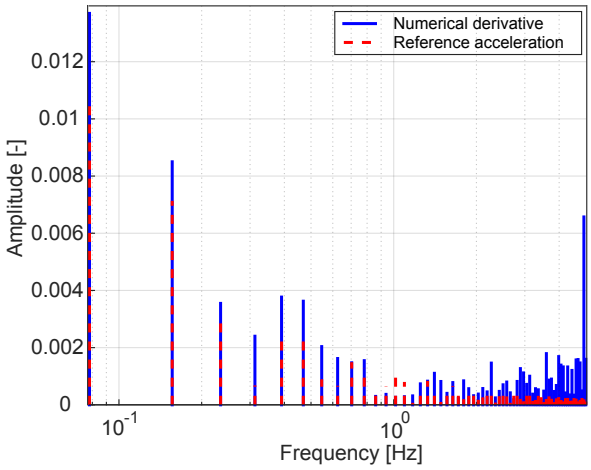


Figure 7.7: Frequency spectra of acceleration signals

7.4 Comparison of control structures

The performance indices $\overline{t_{evade}}$, $\overline{d_{evade}}$ and $\overline{e_{evade}}$ that are defined in Section 7.3 are calculated using simulations. In each simulation 3 agents are considered. The values for $\overline{T}$ and $\overline{P}$ are determined by averaging over 5 experiments which also accommodate 3

agents. In this section the numerical vales of all performance indices are evaluated and discussed for the three control architectures.

Time required

In Figure 7.8 the times spend evading obstacles are given. It can be seen that the high-level control algorithms are beneficial for the time required to fulfill the tasks. These differences can be allocated to the different manners of inter-agent evasion, which are more efficient for these high-level control methods. The evading based supervisor results in the most time-efficient trajectories, decreasing the average time required to prevent collisions with more than 25% compared to the decentralized case.

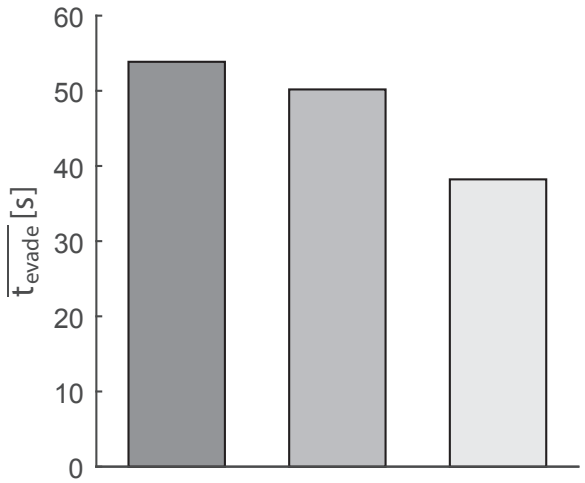


Figure 7.8: Average times spend evading obstacles.

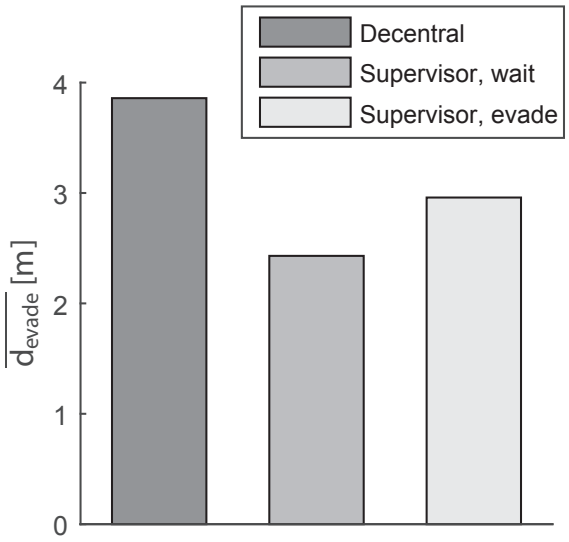


Figure 7.9: Average distance covered while evading obstacles.

Distance covered

Also both controllers with the high-level control layer perform better on the average covered distance than the decentralized one, as shown in Figure 7.9. The waiting algorithm performs the best on this aspect, which is to be expected since it halts in order to prevent collisions instead of creating a detour.

Obtained tracking error

This last argument also explains the differences found in the achieved tracking error, presented in Figure 7.10. After detection of an obstacle the waiting based algorithm halts while the others alter their reference which induces a tracking error. The decentralized method and the evading based supervisory algorithm experience a similar average tracking error, as it is anticipated since they evade using the same principle.

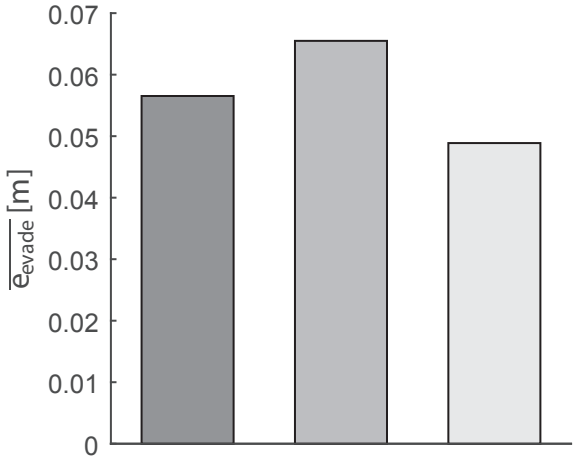


Figure 7.10: Average obtained tracking error due to the evasion of obstacles.

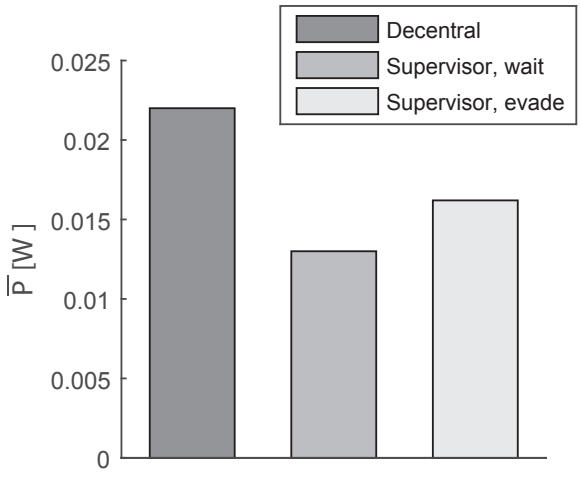


Figure 7.11: Average theoretically required power.

Energy efficiency

The energy-efficiency is also improved by means of the high-level control, as shown in Figure 7.11. For the waiting algorithm this is caused by covering less distance than the others and less angular accelerations are induced by the evasion of the other agents. This is however counteracted by the required translational decelerations and accelerations in order to stop for the other agents. The evading based high-level control strategy outperforms the decentralized case on the energy consumption due to less covered distance and a shorter average operation time. The average used electrical power presented in Figure 7.11 is very low. This is partly due to the energy-efficient reference velocity profiles. Furthermore, the experiments are done with lightweight robots at low velocities and with low accelerations. However, the theoretically used energy calculated by (7.6) does not incorporate any friction, which for the considered experimental setup makes likely a significant part of the power consumption, on one hand, but is hard to be quantified, on another. Therefore, this power consumption is given as an indication of the relative differences between the different control methods.

Computational effort

These gains in performances come at a cost, namely the required computational power. This influences the minimal achievable sampling time for a given setup. As indicated in Figure 7.12, $\overline{T}$ is approximately 5 times larger for the high-level control strategies. That demands computers of higher computational power. However, it should be noted that the execution times ($\overline{T}$) might be decreased by programming the algorithm in C++ or by compiling the Matlab script to an executable file.

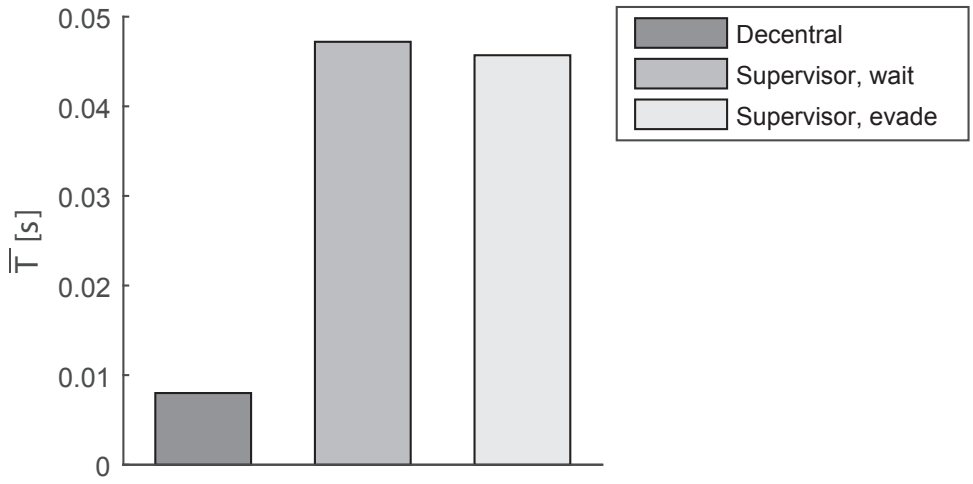


Figure 7.12: Average minimal achievable sampling time.

Overall performance evaluation

By inspection of the results shown in Figures 7.8 to 7.12, it becomes clear that the addition of a high-level control layer is beneficial for the system performance. This addition does not compromise the low-level safety in case of a broken agent or a defect communication network, but does gain efficiency. Therefore, it is recommended to use such a complementary layer if the available computational power allows for it. Which of the two high-level controllers can be considered as the best one depends on the most relevant performance properties for that specific use. This is caused by the Pareto optimality found in these results, which states that no control method performs best on all criteria. On average the evading algorithm faster accomplishes the goals than the waiting method. However, by inspection of the simulation results where the waiting algorithm outperforms the evading algorithm, it seems that the evading algorithm performs less time-efficient in crowded working areas. This can be expected since the evading direction is based on the original calculated future reference. When evading obstacles, the real future reference does not equal the predicted future reference. If this continues for a longer period of time without an update of the reference, an agent might start performing evading maneuvers in the non-optimal directions, which decreases time- and energy efficiencies. To verify this hypothesis, a crowded situation is simulated. Here, 5 agents are simulated in a confined space, first with the evading and than with the waiting supervisory control method. The resulting average spend time and covered distance are presented in Figure 7.13. These simulations support the hypothesis that the evading algorithm may perform worse in congested situations, however no hard claim can be made.

Further research is necessary to investigate which control strategy perform best in specific situations. For different tasks and environmental conditions a different control strategy could be beneficial. Even a combination of the two could be an option.

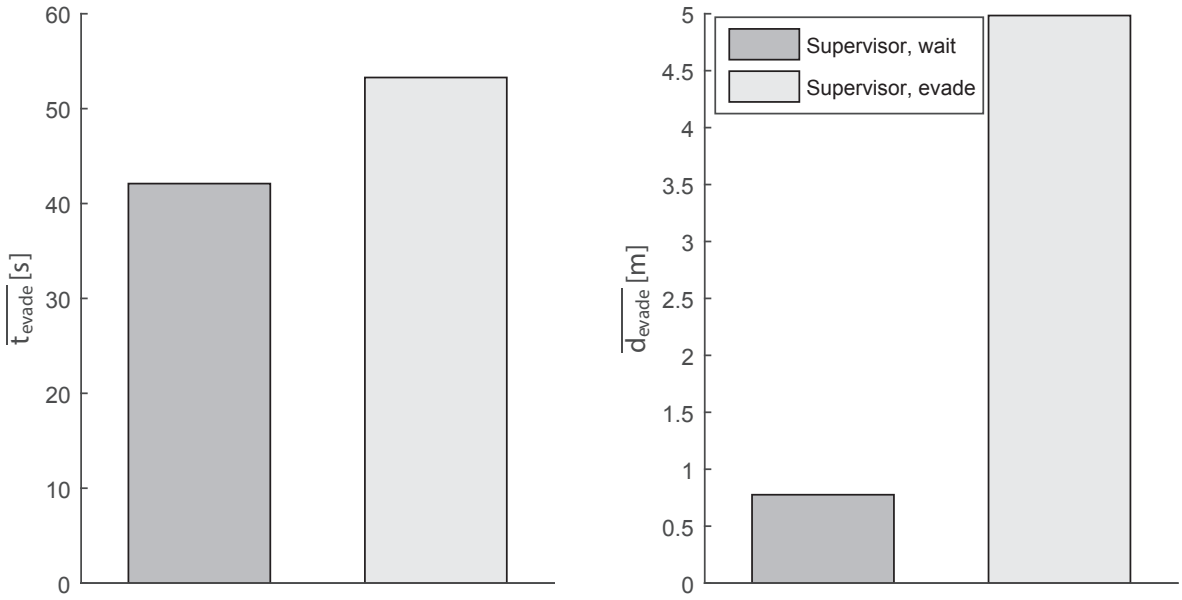


Figure 7.13: Quantitative values of the performance measures achieved in simulations of a crowded area.

7.4.1 Effect of formation control

To investigate the effect of allocating priorities or evasive actions to groups of robots instead of individual agents, it is tried to form formations by using the method described in Section 3.1.2. On hindsight it is found that the manner by which the geometrical pattern of the formation is constructed invokes problems, especially due to the fact that the virtual center of the formation coincides with the leading robot. Therefore, when the leading robot rotates, for example in order to avoid an obstacle, the robot reference position of the following agent rapidly changes. This induces large tracking errors, which requires significant effort from the following agents to resolve. These rapid changes in the reference position of the following agents, in combination with a finite sampling time, also result in non-smooth input velocities. Due to these problems, no useful results on the performance of such transportation system using formation control on groups of agents are obtained in this project.

However, the motivation behind the implementation of formation control, does not require a strict geometrical pattern of agents. It might be adequate for agents to be sufficiently close to each other and moving in roughly the same direction. In that case, the method of allocating groups of agents can be done in a similar manner as described in Section 5.2.1 which seems to work well. Further research into such methods of allocating priorities could be of interest.

7.5 Summary

In this chapter simulation environment is validated using experimental data, although differences between the simulation and experimental environment are observed and discussed. Also it is argued why, if possible, it is beneficial to examine the different control architectures using simulations rather than experiments. The thereto used performance criteria are given and explained. Finally the performance of the control architectures regarding these indices are evaluated, compared and discussed. Based on this comparison it is concluded that the use of high-level control algorithms are beneficial for the sake of efficiency, without undermining the safety of the system.

8

Conclusions and recommendations

8.1 Conclusions

In this project the ideal level of autonomy, for robots performing transportation tasks, is investigated. The thereto developed software infrastructure includes setpoint generation, collision avoidance and motion control algorithms. In combination with the established communication topology this allows for a wide variety of possible robot task allocations throughout the different control levels of the system. This also implicates different levels of dependency on communication. Three defined control architectures, with different levels of autonomy, are considered in this report. Using an experimental setup and a developed simulation environment these control methods are validated, analyzed and compared regarding several performance criteria.

For the setpoint generation, Dynamic Free Range Routing is used as is discussed in Chapter 2. This method omits the commitment to predefined routes, segments or zones. This eases the implementation in new operating environments and optimally uses the available motion space. Furthermore, it minimizes the appearance of live- and deadlocks in the motion planning due to practically infinite number of possible motion profiles.

The reference 2D motion profiles are constructed by a novel reference trajectory generation method, which is described in Chapter 3. This method composes two 1D motion profiles, both based on adapted Linear Segment with Parabolic Blend (LSPB) trajectories, to span the complete motion space. This adaption enables non-zero initial- and/or goal reference velocities, which facilitates on the fly reference updates and the implementation of waypoints. LSPB trajectories are advantageous regarding the energy efficiency because of the constant velocity phase whose duration can directly be specified. The robot motion control uses the kinematics of an unicycle mobile robot.

A collision avoidance algorithm is incorporated to prevent collisions between the robots and obstacles. This algorithm uses smooth weighting functions to calculate an evading route for each individual robot in real time such as to avoid collisions. In particular, the heading angle of the robot is altered, redirecting the reference robot path around the obstacle. This results in generic and intuitive evading maneuvers, that are explained in Chapter 4.

To improve the efficiency of the transportation system, an inter-agent communication network is established. Thereto, a modular network structure is created which simplifies the scaling of the number of agents. This communication facilitates the use of control methods which requires information on the (future) states of neighboring agents. Two variations of high-level control strategies are developed in Chapter 5. Using the experimental setup described in Chapter 6, in combination with the developed simulation environment, these control algorithms are evaluated. In Chapter 7 these results are compared for various system performance indices. It is concluded that the use of inter-agent communication and high-level control, is beneficial for the overall performance of a transportation system in environments of concern for Van den Akker Engineering. However, since the transportation system must remain operational even when the inter-agent communication is lost, this communication is used only to increase the efficiency of transportation, not for keeping the individual agents operational and safe from collisions.

8.2 Recommendations

Based on the findings presented in this thesis, various subjects can be recommended for future research. First of all, the low-level collision avoidance method described in Section 4.1.3 can be further investigated. This method shows promising results in simulations but is not yet tested on an experimental setup. This would require a higher sensing resolution than the one used in this project. When the method becomes practically feasible, it is able to use information on neighboring agents without inter-agent data communication. This could further increase the robustness of the transportation system against communication failures or can lower the required communication bandwidth.

Furthermore, the current global positioning of agents is based on the odometry of the robots and their given initial position and orientation. This is sensitive to slippage, the encoder resolution and errors in the initial placement of the robots. In [34] the addition of a depth sensing camera is investigated, where the global localization is derived from the position relative to known beacons and is compared to the assumed position. This method can be further investigated for example by using simultaneous localization and mapping (SLAM), where the global positioning is based on a constructed map of the operating environment. Also the implementation of radio-frequency identification (RFID) for the provision of global localization of agents might be of interest, for example as done in [43]. When individual global positions are known, the information on the environment obtained by other agents and/or data obtained in the past can be used

concurrently. This information on the surroundings and the positioning relative to it, also enables the implementation of global planning. This could yield reference trajectories that take the known obstacles into account, which can increase the efficiency of the system.

As discussed, the evading based algorithm presented in Section 5.1.2 is not intrinsically safe. Besides the finite sampling time this is caused by the obtained minimum evading radius of a robot. This radius depends on the forward robot velocity when the angular velocity is saturated during an evasive action. Therefore, it is recommended to investigate the effects of real-time scaling of the forward velocity, where the adaption of the expected future reference position is incorporated. This latter is of importance since the high-level controllers require the future reference positions of the agents. Also a mathematical description of the conditions under which the system is collision free is of interest.

In Chapter 7 the performance of three control algorithms is investigated in numerous simulations. It is observed that performance of each algorithm may very much depend on the actual circumstance, e.g. initial- and goal positions and the placement of obstacles. It is recommended to work out a method which in real-time switches between the algorithms presented in the Sections 5.1.1 and 5.1.2, such as to achieve the optimal performance based on the actual robot situation.

No useful results on the performance of agents driving in formation are obtained in this report. As discussed in Section 7.4.1, this is caused by incompatibility between the used collision avoidance method and the definition of the robot formation given in Section 3.1.2. It can therefore be interesting to investigate the allocation of priorities or evading directions not for a strict formation but for a sufficiently compact group of robots with arbitrary relative positions.

Finally, as indicated in Section 7.4, the execution time of the control algorithms can be decreased by their implementation in C/C++ or by compilation into executables. This would also omit the need for using Matlab to run these algorithms.

Bibliography

[1] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[2] Mohamed El. Path Planning in a Dynamic Environment. *International Journal of Advanced Computer Science and Applications*, 5(8):86–92, 2014.

[3] Maria Pia Fanti. Event-based controller to avoid deadlock and collisions in zone-control AGVS. *International Journal of Production Research*, 40(6):1453–1478, jan 2002.

[4] Iris F.a. Vis. Survey of research in the design and control of automated guided vehicle systems. *European Journal of Operational Research*, 170(3):677–709, may 2004.

[5] M.P. Fanti, B. Maione, S. Mascolo, and A. Turchiano. Event-based feedback control for deadlock avoidance in flexible production systems. *IEEE Transactions on Robotics and Automation*, 13(3):347–363, jun 1997.

[6] Ling Qiu, Wen-Jing Hsu, Shell-Ying Huang, and Han Wang. Scheduling and routing algorithms for AGVs: A survey. *International Journal of Production Research*, 40(3): 745–760, jan 2002.

[7] Christopher Oboth, Rajan Batta, and Mark Karwan. Dynamic conflict-free routing of automated guided vehicles. *International Journal of Production Research*, 37(9): 2003–2030, nov 1999.

[8] Mark B Duinkerken, Tiemen ter Hoeven, and Gabriel Lodewijks. Simulating the Operational Control of Free Ranging AGVs. In *Winter Simulation Conference*, pages 1515–1522, 2006. ISBN 1424405017.

[9] M.B. Duinkerken, M. van der Zee, and G. Lodewijks. Dynamic Free Range Routing for Automated Guided Vehicles. *2006 IEEE International Conference on Networking, Sensing and Control*, pages 312–317, 2006.

[10] M Majdi, Hassan Shadkam Anvar, R Barzamini, and S Soleimanpour. Multi AGV Path Planning in Unknown Environment Using Fuzzy Inference Systems. In *International Symposium on Communications, Control and Signal Processing*, pages 172–177, 2008. ISBN 9781424416882.

[11] Mohamed Jasim Mohamed. Obstacles Avoidance for Mobile Robot Using Enhanced Artificial Potential Field. 9(1):71–82, 2013.

[12] Johann Borenstein and Yorem Koren. Real-Time Obstacle Avoidance for Fast. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1179–1187, 1989.

[13] S S Ge and Y J Cui. Dynamic Motion Planning for Mobile Robots Using Potential Field Method. *Autonomous Robots*, 13:207–222, 2002.

[14] Farbod Fahimi, C. Nataraj, and Hashem Ashrafiuon. Real-time obstacle avoidance for multiple mobile robots. *Robotica*, 27(02):189, apr 2008.

[15] Spyros A Reveliotis and Elzbieta Roszkowska. Conflict Resolution in Free-Ranging Multivehicle Systems : A Resource Allocation Paradigm. *IEEE Transactions on Robotics*, 27(2):283–296, 2011.

[16] L. Lapierre, R. Zapata, and P. Lepinay. Combined Path-following and Obstacle Avoidance Control of a Wheeled Robot. *The International Journal of Robotics Research*, 26(4):361–375, apr 2007.

[17] Lazhar Khriji, Farid Touati, Kamel Benhmed, and Amur Al-yahmedi. Q-Learning Based Mobile robot behaviors Coordination. In *International Renewable Energy Congress*, pages 293–299, 2010.

[18] R. Volpe and P. Khosla. Manipulator control with superquadric artificial potential functions: theory and experiments. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(6):1423–1436, 1990.

[19] Hossein Adeli, M H N Tabrizi, Alborz Mazloomian, Ehsan Hajipour, Mehran Jahed, and Collision Free Path. Path Planning for Mobile Robots using Iterative Artificial Potential Field Method. *International Journal of Computer Science Issues*, 8(4):28–32, 2011.

[20] Wesley H. Huang, Brett R. Fajen, Jonathan R. Fink, and William H. Warren. Visual navigation and obstacle avoidance using a steering potential function. *Robotics and Autonomous Systems*, 54(4):288–299, apr 2006.

[21] WH Warren and BR Fajen. Behavioral Dynamics of Visually-Guided Locomotion. *Coordination: Neural, Behavioral and Social Dynamics*, 2008.

[22] Brett R. Fajen and William H. Warren. Behavioral dynamics of steering, obstable avoidance, and route selection. *Journal of Experimental Psychology: Human Perception and Performance*, 29(2):343–362, 2003.

[23] Hendrik Reimann, Ioannis Iossifidis, and Gregor Sch. End-effector obstacle avoidance using multiple dynamic variables. In *41st International Symposium on Robotics*, number Isr, 2010.

[24] F.H.A Monsieurs. Collision-free cooperative control of unicycle mobile robots with saturated input velocites. Technical report, 2014.

[25] Paul Rybski, Sascha Stoeter, Maria Gini, Dean Hougen, and Nikolaos Papanikolopoulos. Performance of a Distributed Robotic System Using Shared Communication Channels. *Control*, 18(5):211–225, 2002.

[26] Christopher Amato, Gd Konidaris, and Gabriel Cruz. Planning for Decentralized Control of Multiple Robots Under Uncertainty. Technical report, 2014.

[27] T. Arai, E. Pagello, and L.E. Parker. Guest editorial advances in multirobot systems. *IEEE Transactions on Robotics and Automation*, 18(5):655–661, 2002.

[28] Sisdarmanto Adinandra. Hierarchical Coordination Control of Mobile Robots. Technical report, 2012.

[29] R. Vidal, S. Rashid, C. Sharp, O. Shakernia, J. Kim, and S. Sastry. Pursuit-evasion games with unmanned ground and aerial vehicles. *Proceedings - IEEE International Conference on Robotics and Automation*, 3:2948–2955, 2001.

[30] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*, volume 141. 2006. ISBN 0471649902.

[31] A Sadowska, D Kostic, N Van De Wouw, H Huijberts, and H Nijmeijer. Distributed Formation Control and Trajectory Tracking of Multiple Unicycle Mobile Agents. Technical report, 2011.

[32] D. Kostic, S. Adinandra, J. Caarls, N. van de Wouw, and H. Nijmeijer. Saturated control of time-varying formations and trajectory tracking for unicycle multi-agent systems, dec 2010.

[33] D. Kostić, S. Adinandra, J. Caarls, N. Van De Wouw, and H. Nijmeijer. Saturated control of time-varying formations and trajectory tracking for unicycle multi-agent systems. *Proceedings of the IEEE Conference on Decision and Control*, pages 4054–4059, 2010.

[34] Michel Legius. Control and Integration of a Flexible Transportation System Based on Autonomous Mobile Robots. Technical Report April, Van den Akker Engineering; University of Technology Eindhoven, Eindhoven, 2015.

[35] Yujin. Kobuki homepage, 2015. URL `http://kobuki.yujinrobot.com/home-en`.

[36] Sharp Gp2Y0a02Yk0F datasheet. 2006. URL `https://www.sparkfun.com/datasheets/Sensors/Infrared/gp2y0a02yke.pdf`.

[37] Arduino - ArduinoBoardMega2560, 2014. URL `https://www.arduino.cc/en/Main/ArduinoBoardMega2560`.

[38] Patona Powerbank, 2015. URL http://www.pts-trading.de/universal-powerbank-for-laptop-mobile-phone-pda-dvd-mp3mp4-ipad-ipod-iphone-ultra-thin-aluminum-jacker-16000mah-from-patona-p-3641.html.

[39] MATLAB The Language of Technical Computing, 2015. URL http://nl.mathworks.com/products/matlab/.

[40] Matworks. ROS I / O Package Getting Started Guide. Technical report, 2014.

[41] Powering the world's robots. URL www.ros.org.

[42] Robotics System Toolbox - Simulink and MATLAB - MathWorks Benelux. URL http://nl.mathworks.com/products/robotics/.

[43] S E M Janssen. Control of mobile robots based on RFID localization. Technical report, University of Technology Eindhoven, 2015.