Hierarchical Coordination Control of Mobile Robots

Sisdarmanto Adinandra
This work has been carried out as part of the FALCON project under the responsibility of the Embedded Systems Institute with Vanderlande Industries as the carrying industrial partner. This project is partially supported by the Netherlands Ministry of Economic Affairs under the Embedded Systems Institute (BSIK03021) program.

A catalogue record is available from the Eindhoven University of Technology Library

*Hierarchical Coordination Control of Mobile Robots.*

Copyright ©2012 by Sisdarmanto Adinandra. All rights reserved.

This thesis was prepared using pdfLaTeX documentation system.
Cover Design: Swevel Universal Media, Yogyakarta, Indonesia.
Hierarchical Coordination Control of Mobile Robots

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Eindhoven,
op gezag van de rector magnificus, prof.dr.ir. C.J. van Duijn,
voor een commissie aangewezen door het College voor Promoties
in het openbaar te verdedigen
op donderdag 6 september 2012 om 16.00 uur

door

Sisdarmanto Adinandra

geboren te Yogyakarta, Indonesië
Dit proefschrift is goedgekeurd door de promotor:

prof.dr. H. Nijmeijer
Summary

Hierarchical Coordination Control of Mobile Robots

In the last decade, robotic systems have penetrated human life more than humans can imagine. In particular, the multi-mobile robotic systems have faced a fast growth due to the fact that by deploying a large collection of mobile robots the overall system has a high redundancy and offers the capability of handling more complex tasks. A group of mobile robots increases robustness against failures and provides flexibility to system changes. In order to achieve a fully autonomous operation, the control algorithm to coordinate the robots becomes more important and decisive. The main innovations of this thesis are a general three-layer hierarchical coordination control architecture and performance evaluation of coordination control algorithms for a group of mobile robots.

In the first part of this thesis, a modular framework for simulation and experiments of coordination control of mobile robots is presented. All connections between the modules follow the subscriber/publisher paradigm on the exchanged data, i.e., a component publishes data and other components can subscribe to that data. The data is identified by a named magazine, and actual data packets are called issues of the magazine. By means of simulation and experiments, it is demonstrated how the modularity in the framework allows straightforward modification of system configurations, design parameters, and control algorithms.

In the second part, hierarchical coordination control of mobile robots is presented. The hierarchy consists of three layers: a high-level control for motion planning of the robots, a low-level control for motion execution, and a flexible layer to accommodate shifting of responsibilities. One advantage of using a hierarchical approach is the isolation of control design in each layer. Changes in the control strategy of a layer do not necessarily require adaptation of other layers.

The proposed control algorithms are used to coordinate a group of unicycle mobile robots that realize the transportation system of an automated warehouse. Using the framework, series of simulations with different design parameters are conducted, as well as real-time experiments. Subsequently, performance analysis of the results is carried out. It is shown that the proposed algorithms are flexible to system changes and scalable to variation in the warehouse demands. The algorithms are robust against failures in the transportation systems. It is found that an algorithm, denoted
high-level control, with higher throughputs requires more information sharing between the robots. This algorithm is less robust against failures compared to the algorithm, denoted low-level control, that yields lower throughput. In addition, a cost analysis of the proposed control algorithms is given. It is studied that the cost of realizing the transport using a group of mobile robots, coordinated by the proposed control algorithm, has a similar performance-to-cost ratio compared to conveyor systems. This conclusion is very promising having in mind that the proposed control algorithm is not necessarily the optimal one.

In the third part, the problem of simultaneous tracking of individual references and formation keeping for a group of mobile robots is investigated. The control algorithm is developed using dynamic feedback linearization. The stability proof is analyzed using the theorem on interconnected systems. Using a root-mean-square-like indicator, the trade off between individual tracking and formation keeping, as well as the influences of communication topologies are analyzed. From the analysis of real-time experiment results, it is found that the best formation keeping is obtained when all robots communicate. As a trade off, this requires a larger communication bandwidth and yields large individual tracking errors. Furthermore, the analysis suggests that to achieve optimal individual tracking and formation keeping simultaneously, there is less need to share information between the robots.

In the fourth part, as a complement to the hierarchical control approach, a coordination control algorithm based on Model Predictive Control (MPC) is presented. Focusing on the practical aspects, a sequentially decentralized MPC, i.e. a single MPC computes the control signals of all robots where priority rules are used to determine which robots handled earlier in optimization procedure, is implemented. Using the similar automated warehouse environment, the sequentially decentralized MPC is validated in simulations and real-time experiments. For comparison purposes, a centralized MPC, i.e. a single MPC computes the control signals of all robots simultaneously, is also implemented. Using completion time as indicators, the influences of the MPC parameters are investigated. It is found that the centralized MPC is better than the sequentially decentralized MPC, but with the cost of high computation load and limited number of robots in real-time application. Relevance and performance comparison between the MPC and the hierarchical control approach are presented. It is found that regardless of the control algorithm choices, in terms of completion time, a better performance is obtained when more information is used to compute the control. As a trade off, the system becomes less robust against perturbations and requires larger communication bandwidth.
Contents

Summary v

1 Introduction 1
  1.1 Cooperative robotic systems ........................................ 1
    1.1.1 Basic motion tasks of a single robot .......................... 2
    1.1.2 Coordination control of multi-mobile robotic systems ....... 4
    1.1.3 Performance measures ............................................ 8
  1.2 Flexible Autonomous Logistic CONcept (FALCON) project ......... 9
  1.3 Research objectives of the thesis .................................. 10
  1.4 Contributions .......................................................... 11
  1.5 Thesis outline .......................................................... 12

2 Preliminaries 15
  2.1 Outline ................................................................. 15
  2.2 Mathematical notions ................................................ 15
    2.2.1 Stability of interconnected systems (Khalil, 1996) ............ 16
    2.2.2 Elementary graph theory ......................................... 16
  2.3 Non-holonomic kinematic model of unicycles ....................... 17
  2.4 Control architecture ................................................ 20
  2.5 An automated warehouse .............................................. 22
  2.6 Performance indicators .............................................. 23
    2.6.1 Completion time ($t_{\text{complete}}$) ......................... 23
    2.6.2 Robustness ($P_{\text{robustness}}$) ............................. 24
  2.7 Summary ............................................................... 24

3 Experimental Setup 25
  3.1 Introduction .......................................................... 25
  3.2 Hardware components ............................................... 26
## Contents

### 6 Simultaneous Trajectory Tracking and Formation Keeping for a Group of Unicycles

- 6.1 Introduction ........................................... 77
- 6.2 Dynamic feedback linearization of a unicycle mobile robot ............... 78
- 6.3 Simultaneous tracking and formation keeping ............................. 80
- 6.4 A performance measure for a group of unicycles .......................... 86
- 6.5 Experimental results and performance analysis ............................ 87
  - 6.5.1 Experimental results .................................... 87
  - 6.5.2 Performance analysis .................................... 91
- 6.6 Conclusions ................................................ 96

### 7 Model Predictive Control for a Group of Unicycles: Simulation and Experimental Results

- 7.1 Introduction ............................................. 97
- 7.2 Preliminaries on MPC ..................................... 99
  - 7.2.1 General principle of MPC ................................ 99
  - 7.2.2 MPC for systems with fast dynamics ....................... 100
- 7.3 A sequentially decentralized MPC ................................ 101
- 7.4 Discrete-time model of unicycle mobile robots ........................... 102
  - 7.4.1 A discrete time kinematic model of unicycle mobile robots ....... 102
  - 7.4.2 Obtaining predicted outputs ............................... 103
- 7.5 Cost function .............................................. 104
  - 7.5.1 Trajectory tracking problem ................................ 104
  - 7.5.2 Collision avoidance with other unicycles ....................... 105
  - 7.5.3 The complete cost function ................................ 106
  - 7.5.4 Priority rules ........................................... 107
- 7.6 Optimization algorithm ..................................... 107
  - 7.6.1 Steepest descent optimization method ........................ 108
  - 7.6.2 Line search optimization ................................ 109
  - 7.6.3 Implementation: computation time reduction ...................... 110
- 7.7 Simulation and experimental results ................................... 110
  - 7.7.1 Simulation results ...................................... 111
  - 7.7.2 Experimental results ..................................... 113
- 7.8 Comparison with the hierarchical control approach ....................... 116
  - 7.8.1 Relevance to the hierarchical control approach .................. 116
  - 7.8.2 Performance comparison for the automated warehouse case study . 116
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.9 Conclusions</td>
<td>119</td>
</tr>
<tr>
<td>8 Conclusions and Recommendations</td>
<td>121</td>
</tr>
<tr>
<td>8.1 Conclusions</td>
<td>121</td>
</tr>
<tr>
<td>8.2 Recommendations</td>
<td>124</td>
</tr>
<tr>
<td>Bibliography</td>
<td>126</td>
</tr>
<tr>
<td>Abbreviations</td>
<td>133</td>
</tr>
<tr>
<td>Samenvatting</td>
<td>136</td>
</tr>
<tr>
<td>Acknowledgment</td>
<td>139</td>
</tr>
<tr>
<td>Curriculum Vitae</td>
<td>141</td>
</tr>
</tbody>
</table>
This thesis addresses the problem of coordination of mobile robots, with a specific application in the area of transport in automated warehouses. This chapter provides a general overview of coordination control algorithms. Several applications in which a group of mobile robots is coordinated to achieve a specific task and their respective control approaches are reviewed in detail. Existing control algorithms are analyzed and open problems related to the design of the controllers are summarized. In addition, a brief overview of transport systems in warehouses is given. Relevant issues for controlling transportation are addressed. Next to this overview, the research objectives of this thesis are formulated and its research contributions are presented. Finally, the outline of this thesis is given. This research is conducted within the framework of the Flexible and Autonomous Logistic CONcept (FALCON) project.

1.1 Cooperative robotic systems

In the last decade, the fast technological evolution in several fields and the demand to reduce human labor costs have led to an unpredictably fast development and wide-range adoption of robotic systems. It is foreseen that in the near future robots in personal and professional applications will become part of our daily lives. From providing entertainment for children and test-beds for research, assisting elderly at home, aiding surgery and helping humans during disasters, robotic systems have blended life more than human could have imagine. The class of multi-robot systems is seen as a very promising candidate for the next generation of robotic applications.

A multi-robot system gains its popularity due to the limitations encountered when a single robot is used to perform a certain task. It is widely agreed that the use of multi-robot systems, especially teams or groups of mobile robots which exhibit cooperative behavior, present several advantages over the use of a single robot, see
for example: Arai et al. (2002), Cao et al. (1997), and Siciliano and Khatib (2008). A multi-robot system presents high redundancy since the failure of a single or several robots does not cause complete standstill of the overall system. A multi robot system allows decomposition of a complex task into a set of subtasks that can be handled by cooperation of the robots in the group. Moreover, the group of robots may perform the assigned task in a more reliable, faster, or cheaper way, beyond what is possible with single robots.

The applications of multi-robot systems, specifically in the field of autonomous mobile robots, are widely spread from heterogeneous multi-purpose bio-inspired robots (Bonani et al., 2010), cooperative cleaning (Jager and Nebel, 2002), push-box cooperation (Gerkey and Mataric, 2002), to exploration (Burgard et al., 2005), automated highway systems (Naus et al., 2010; Swaroop and Hendrik, 1996), football playing (Robocup, 2012), and logistics (Giuzzo, 2008). In each application, different approaches have been developed to control the robots such that the collective, and individual (if any), goals can be achieved. Some of the aforementioned applications are illustrated in Figure 1.1. 1

As robotic systems become more complex, the control algorithms that regulate the robots are becoming more decisive and complex to meet different requirements such as robustness, safety, flexibility and scalability. Real-time applications and daily routine require sophisticated control algorithms so that the requirements are met. Different control approaches like behavioral based, e.g. Gerkey and Mataric (2002), master-slave/leader-follower, e.g. Das et al. (2002), or virtual structure, e.g. Lewis and Tan (1997) have been widely used and validated in different areas of applications. Typically, the chosen approach is the one that is most beneficial for a certain application.

In several applications, each mobile robot is modeled using either a kinematic or dynamic model of the unicycle robot. The unicycle has a nonholonomic constraint that prohibits the robot to move sideways. In other applications, each mobile robot is modeled by using the model of an omni-directional wheeled robot. In this case, the model does not contain nonholonomic constraints so that the robot can freely move sideways. Overviews of other applications of multi mobile robot systems and their controllers can be found, for example, in Arai et al. (2002) and Pettersen et al. (2006).

1.1.1 Basic motion tasks of a single robot

Although this thesis focuses on the cooperative control between mobile robots, it is worthwhile to review the basic motions that can be achieved by a single mobile robot. Considering a unicycle mobile robot, in most literature the basic motion tasks of a single robot are divided into three categories: point-to-point, path following, and trajectory tracking. These basic motions are illustrated in Figure 1.2.

Point-to-point motion

This basic motion is also called point stabilization. In this motion, the robot must
1.1 Cooperative robotic systems

Figure 1.1: Examples of cooperative autonomous mobile robot systems.

reach a desired configuration starting from a given initial configuration. Examples of controllers for point stabilization are described in de Wit and Sørdalen (1992), Samson (1995), and Do et al. (2004).

**Path Following**

In path following, a reference point on the robot must follow a geometric path, without an associated timing law given beforehand, in Cartesian space starting from a given initial configuration. The controller has to ensure that the robot moves along the assigned path, regardless of the time taken to reach the path. Examples of path following controllers are found in Aguiar and Hespanha (2007) and Coelho and Nunes (2005).

**Trajectory Tracking**

Different to path following, in trajectory tracking a reference point on the robot must follow a trajectory, i.e. a geometric path with an associated timing law in the Cartesian space starting from a given initial configuration. The tracking controller ensures that the assigned path can be followed asymptotically or exponentially in time. Examples of trajectory controllers are presented by Jiang and Nijmeijer (1997),
The control algorithms mentioned in the references above use a feedback or combination of feedback and feedforward control design. The use of a feedback solution exhibits an intrinsic degree of robustness. However, especially in the case of point stabilization, the design of feedback control laws faces a structural obstruction that may lead to unsatisfactory transient performance.

For the same geometric path, path following control is less aggressive than trajectory tracking control. Since there is a timing along the path, the trajectory tracking will try, as fast as possible, to achieve the desired path. On the other hand, the no-timing property allows path following to take more time in achieving the desired path. This characteristic can be either an advantage or disadvantage depending on the application.

1.1.2 Coordination control of multi-mobile robotic systems

As has been mentioned, a considerable progress in research related to the collective behavior of multiple robots has been accomplished. The study of multi-robot systems naturally extends from research single-robot systems, but is also a discipline in itself: multiple-robot systems can accomplish tasks which are impossible to achieve by using a single robot, since ultimately a single robot, no matter how capable, is spatially limited. Multiple-robot systems are also different from other traditional distributed systems such as computers, databases, or networks because of their implicit real-world environment, see Cao et al. (1997).

In order to fully exploit the potential of multi-mobile robotic systems, one of the theoretical and technological issues which remains open for improvement and exploration is the development of novel coordination control algorithms. These control algorithms must meet different performance criteria that typically depend on the application. Regardless of the application, the most recurrent control approaches in
Multi-mobile robotic systems can be classified in the following groups: the master-slave/leader-follower, the virtual-structure, the behavioral based, and the motion-planning approaches.

**Master-slave approach**

In a master-slave or leader-follower approach, one of the robots in the group acts as the master or leader, while the remaining robots become the slaves or followers. The master’s objective is to complete a certain task which is typically related to the task of the group as a whole. An example is to guide the slaves through a course with obstacles. Based on the information topology as depicted in Figure 1.3, the $n$ slaves generate individual reference trajectories so that the group task can be achieved.

![Figure 1.3: Communication topology in the master-slave motion coordination.](image)

In master-slave coordination, the information flows in one direction from the master to the slaves. The advantage is that even if the slaves have limited sensing and control capabilities, the group still can achieve its collective goal since the master directs the necessary movement to the slaves. However, the unidirectional information flow also brings some disadvantages. Since the motion of the master is independent from the slaves, the master uses no knowledge about the slaves. Thus, if one or more slaves fail in completing their individual tasks, the group task cannot be finished. This implies that master-slave coordination is not robust against perturbations at the slave’s sides. Furthermore, any failure at the master side will cause the failure of the whole group.

Many classical control techniques like feedback linearization (Desai et al., 2001), (Fierro et al., 2002), backstepping (Li et al., 2005), and sliding-mode control (Defoort et al., 2008), have been used to achieve master-slave coordination. In other works by Das et al. (2002) and Fierro et al. (2002), a master-slave coordination is achieved by using the so-called separation-bearing and separation-separation controllers.
Virtual structure approach

In the virtual structure approach, all members of the group generate their reference trajectories based on a common reference known as the virtual center. The virtual center can be a robot of the group. The resulting geometry of the desired motion of the members is called the virtual structure. A typical information flow between a virtual center and \( n \) robots of the group is shown in Figure 1.4.

![Figure 1.4: Communication topology in the virtual structure motion coordination.](image)

As depicted in Figure 1.4, due to the nature of the information flow, all robots communicate with each other and with the virtual center if it is a robot. In a virtual structure, mutual coordination between the robots inherently possesses a certain robustness against perturbations. Since typically all robots are coupled, any perturbation appearing at any robot is propagated to the other members of the group. Thus, the non-perturbed robots can react and compensate for the perturbation so that the virtual structure remains the same.

The first virtual structure concept was introduced in Lewis and Tan (1997). It is assumed that all robots possess global knowledge of the system. The mobile robots are considered as particles that are intended to stay inside the virtual structure and the virtual structure looks to adjust the robot’s positions. Closely related works can be found in Beard et al. (2000) and Young et al. (2001), among others.

Recent development in the virtual structure approach can be found in van den Broek et al. (2009), Kostić et al. (2010a), and Sadowska et al. (2011). In these works, each robot is equipped with a controller that enables simultaneous tracking of an individual reference trajectory and formation keeping with other robots. The assigned trajectories are generated from the virtual center. The formation keeping is introduced by means of coupling between the robots. This introduction increases the group’s ability to withstand perturbations. In Mastellone and Stipanović (2008), van den Broek et al. (2009), and Kostić et al. (2010a), the coupling between the robots requires all robots communicate. This demand is reduced in Sadowska et al. (2011), where a robot is allowed to communicate only with the robots in its neighborhood. The reduced communication load is a very important property since the group should achieve high robustness at minimum communication cost. It is to be
1.1 Cooperative robotic systems

noted that the definition of mutual coordination in the mentioned works is closely related to the one provided by Nijmeijer and Rodriguez-Ángeles (2003) for mutual synchronization of robotic manipulators.

**Behavior-based approach**

The behavior-based approach was first introduced by Brooks (1986). The work presents a set of so-called behaviors or motion primitives that are weighted in order to compute the robots’ ultimate behavior. As an example, the control input of the robot depends on the number of motion primitives such as tracking, localization, collision avoidance, and others. The behavior-based approach for formation control was proposed by Balch and Arkin (1998) for a group of unicycle robots by weighting several independent actions for the robots. An advantage of the behavior-based approach is that it is intuitive and allows decomposition of a complex task into modular subtasks. A modularity is very useful when the number of robots in the group becomes large. The modular actions can be reusable and easy to maintain. However, the resulting group’s dynamics do not lead to straightforward mathematical analysis. As a consequence, the stability proof of the closed-loop system is difficult to analyze and the system cannot be predicted accurately.

**Motion planning approach**

In the three approaches discussed above, a controller is designed based on how the motion of the robots must be executed. The approaches mainly modify one of the three basic motions of a single robot so that collective behavior can be achieved. On the other hand, coordination can be achieved at a higher level where either the final destinations, geometric paths or reference trajectories are generated. In this case, coordination is achieved via motion planning of the robots. For example, Švestka and Overmars (1998) show how the robot coordination can be achieved via coordinated path-planning algorithms. The algorithm ensures, in advance, that the motion of all robots is guaranteed to be successful. This high-level planning approach is widely used in coordinating a group of robots such as Automated Guided Vehicles (AGV) inside manufacturing systems, see for example: Peters et al. (1996), Le-Anh and de Koster (2006), Vis (2006), Weyns et al. (2008), Goel and Gruhn (2008), Lacomme et al. (2010), and Xidias and Azariadis (2011).

A big advantage of using a motion planning approach is the optimality of the solution, either local or global solution. Since motion planning basically predicts the behavior of the system in advance, an optimization technique can be used in order to design the coordination algorithms. However, the resulting algorithms work like a feedforward controller. This means that it is not robust against perturbations in the system. A feedback mechanism may be introduced, but typically increases the computation time for finding the solution.

**Other approaches**

Coordination in multi-mobile robotic systems can also be achieved via simple rules like the use of traffic lights rule, see for examples Asama et al. (1991), Wang (1991), Kato et al. (1992), and Arora et al. (2000), or the artificial-potential function (APF) concept presented by Latombe (1991). These approaches are easy to understand and
implement so that modularity and scalability of the systems can easily be achieved. However, formal mathematical tools to analyze the stability of the systems and to predict their behaviors are missing. Thus, for a large scale and complex system, these approaches pose some potential risks of having unfinished group tasks.

The work in consensus systems also provides an approach for coordinating in multiple-robot systems. The applications range from flocking, rendezvous, to formation control. Typically, an integrator or double integrator is considered as the system's dynamics. By exploiting algebraic graph theory, the works on robot consensus are able to mathematically formulate the relation between the communication topologies and the stability of the system. Details about the fundamentals of consensus algorithms and their analysis can be found in Olfati-Saber et al. (2007) and Ren and Beard (2008). The technique in consensus can be applied to any nonlinear system that is (dynamically) feedback linearizable, including unicycle mobile robots.

1.1.3 Performance measures

The next important property of multi-mobile robotic systems is the notion of performance measures of the systems. Examples of performance measures can be either the time needed by the group to complete a task or how fast the system recovers from perturbations. Since any control algorithm has particular advantages and disadvantages, performance analysis and comparison of different control algorithms are required. This analysis and comparison are useful to combine advantages and eliminate the disadvantages of different control approaches.

To the best of the author's knowledge, there are only few references available on the topic of performance analysis of cooperative multi-mobile robotic systems. For instance, the work in Tanner (2004); Tanner et al. (2002a,b, 2004) presents results on performance of formation control of separation-bearing and separation-separation controllers (master-slave coordination). This work explores the input-to-state stability (ISS) property of the system to construct an indicator that measures the effect of the leader's input to the formation. Combined with the communication topology, the measure gives an analytic performance measure of "how strong" the formation is. Similar works are also found in Ögren and Leonard (2003) and Chen and Serrani (2004).

The above references also show the difficulty in formulating an ISS for nonholonomic system. For a unicycle mobile robot, dynamic feedback linearization can help transforming the system into a system for which the ISS property is easier to formulate. In Tanner (2004), the author presents a result where the ISS property of a unicycle with dynamic extension can be formulated in a particular topology induced by a metric appropriate for the extended systems. A feedback control law renders the closed-loop system locally ISS in the particular topology. Although it has potential applications including robustness analysis of formation control, the proposed metric is not trivial, and therefore generalization is difficult.

In Sadowska et al. (2011), the author proposed a formation geometry performance index that shows the discrepancy between actual formation shape and the desired one. In particular, the performance index measures the difference between actual
and desired distances between all pairs of robots in the formation. Although the index measures the geometry of the formation, it is possible to have a rotation or a reflection (a mirror image) of the formation shape.

If multiple-robot systems are considered as multi-agent systems, the performance of the system is sometimes seen from the perspective of how many messages have to be exchanged between the robots in order to achieve the desired behavior. As an example, Gerkey and Matarić (2002) propose a control architecture that accommodates auctions between the robots and compare the effectiveness of the controller with respect to the number of messages passing in the system to other controllers. In a small-scale multiple robot system, this type of analysis may not be important. However, once the scale becomes larger, the communication bandwidth also becomes limited, and therefore the analysis of numbers of exchanging messages turns out to be crucial.

With regard to consensus theory, Olfati-Saber et al. (2007) present the so-called disagreement vector to measure how fast the consensus can be achieved considering different communication topologies. The measure also checks at which conditions the consensus starts to lose its stability.

1.2 Flexible Autonomous Logistic CONcept (FALCON) project

This thesis is a result of research performed as part of the FALCON project. The project is done by a joint consortium of industrial and academic partners under the responsibility of the Embedded System Institute with Vanderlande Industries as the carrying industrial partner. The project considers the development of a new generation of warehouses with maximum degree of automation. Warehouses are key aspects of modern supply chains and play a vital role in the success, or failure of today's business (Frazelle, 2002). Some major roles of warehouses are (Gu et al., 2007):

1. Buffering the material flow along the supply chain to accommodate variability caused by factors such as product seasonality and/or batching in production and transportation.
2. Consolidation of products from various suppliers to combine delivery to customers.
3. Value-added-processing such as kitting, pricing, labeling.
4. Product customization.

The main goal of the FALCON project is to bridge the gap between component design and system design, thereby filling the lack of integrated warehouse design support (Hamberg and Verriet, 2011). The project covers three major topics:

- The development of tools and methods for the design of warehouses
The development of control methods to make a warehouse system perform as intended

The development of robotic item-picking solutions

As part of large automated systems, transportation in warehouses plays critical roles in delivering high performance. Although the conventional conveyor systems deliver high throughput, they are sensitive to failures. Redundant conveyors require additional extra investment that is most-likely not covered by the limited budget. The fixed capacity of the conveyors is another drawback. Once the warehouse needs to scale-up, additional space and conveyors may be needed, which means the system needs to be overhauled. For the future automated warehouses, a highly robust, flexible, and scalable transportation system must be available.

Design of such a system with its controller typically causes high development cost. Therefore, a simulation and/or experiment tool that allows design, test, and comparison of controllers with quick results will be beneficial. The tool should allow both simple and complex modeling of the systems, as well as different control architectures and algorithms. The tool should allow quick adjustment and analysis while still giving satisfactory accuracy. This is crucial for having insight into the system performance in the early warehouse design phase, as well as comparing its performance to existing transport systems.

1.3 Research objectives of the thesis

The discussion presented in Section 1.1.2 shows that there are different ways to regulate a group of mobile robots. There is a need to developed a novel coordination control algorithm that maximizes the advantages and minimizes the disadvantages of different control approaches. By doing so, it is expected that the control algorithm can simultaneously meet different performance criteria such as robustness against perturbations, flexibility, and scalability. In addition, as suggested by Arai et al. (2002) and Chen and Wang (2005), to fully exploit the benefits of multi-mobile robotic systems there is a need to look for applications that involve a large collection of robots and complex group tasks.

From Section 1.1.3, it is known that information on performance analysis and comparison of different control algorithms is rarely available. The available results are also limited to simple tasks and environments. There is a lack of tools, mathematically or experimentally, that allows performance comparison between different control algorithms in an easy way. This comparison speeds up the process of combining advantages and disadvantages of different control algorithms because for a specific indicator the comparison shows which is the “best” control algorithm.

Section 1.2 mentioned that a future automated warehouse requires a flexible, scalable, and robust transportation system. One way to achieve these properties simultaneously is to replace the conventional conveyor systems with a group of mobile robots. This idea is not new, see for example: Giuzzo (2008) and Ylog Logistics (2011). The idea of using a group of mobile robots for transportation comes from the use of AGVs to transport materials between workstations in manufacturing systems.
1.4 Contributions

In most cases, the group is regulated at the motion planning level of the robots (Vis, 2006; Gu et al., 2010), i.e. a controller decides the paths for the robots so that the complete goals can be achieved while taking into account threat of collisions. This approach assumes that the motion controller executes the planned paths perfectly. This approach yields an optimal transport throughput but is less robust against unwanted situations in the system, e.g. a broken robot blocks the paths of other robots. Furthermore, it is rarely investigated how the performance of the group of robots changes when the robots are fully regulated using a motion controller approach or a combination of motion planning and motion controller.

Based on the summary of open problems in multiple mobile robot systems mentioned in the previous paragraphs, the main objective of this thesis is formulated as follows:

To perform analysis on coordination control algorithms for a group of mobile robots

Furthermore, this main objective is divided into the following sub-objectives:

1. Formulate coordination control algorithms based on a hierarchical control approach by extending the trajectory tracking control concept for a single mobile robot.

2. Develop a framework to facilitate validation, by simulation and experiment, of different coordination control algorithms.

3. Conduct performance analysis and comparison of coordination control algorithms by considering a transportation system of an automated warehouse as a case study.

1.4 Contributions

Regarding the research objectives, the contributions of this thesis are summarized as follows:

- A hierarchical control approach to coordinate a group of mobile robots. The control approach allows separation of control design in each layer as well as shifting of control responsibilities. The hierarchy consists of three main layers, namely high-level motion planner, low-level motion executor, and adjustable layer to accommodate the shifting of responsibilities, [Chapter 2];

- A modular-framework that allows quick switches between simulation and experiments, as well as changing and testing different control algorithms. The framework supports as many robots in simulation and up to 15 robots in the experiments. The modularity provides an easy hardware (computation) implementation, [Chapter 3];

- A low-level motion control approach for a group of mobile robots that meets both requirements on robustness against failures and flexibility to system changes. The controller uses a reactive-to-dynamic control approach that covers
the motion execution of each robot in the group, while at the same time co-
ordinates the movement with other robots to avoid collision and complete the
group tasks. The algorithm is validated in an automated warehouse, [Chapter
4];

• A performance comparison between high- and low-level coordination control
for a group of mobile robots. A high-level controller that regulates a group of
mobile robots is proposed. Similarly, the high-level controller is validated in
an automated warehouse. A detailed performance analysis between the two
algorithms is provided. Furthermore, a cost comparison with a conventional
conveyor system is presented, [Chapter 5];

• A coordination controller that is able to simultaneously track an individual
trajectory and to keep a certain formation with other robots designed using
dynamic feedback linearization. Using the theorem of interconnected systems,
the stability of the controller is analyzed. The influence of communication
topologies is investigated by means of experiments, [Chapter 6];

• A practical Model Predictive Control (MPC) for a group of unicycle mobile
robots. A sequentially decentralized MPC is proposed as a complement to the
hierarchical control approach. The effectiveness of the MPC algorithm is val-
ified in an automated warehouse. Relevance and performance comparison
with the hierarchical control approach are presented, [Chapter 7];

Parts of the contributions have been presented in the following publications: Adinan-
dra et al. (2010) (Chapter 3, 5), Adinandra et al. (2011b) (Chapter 3, 5), Adinandra
et al. (2011a) (Chapter 4), and Adinandra et al. (2012) (Chapter 7).

1.5 Thesis outline

Chapter 2 discusses some preliminaries on necessary mathematical background and
information about an automated warehouse. The hierarchical control approach and
performance indicators for evaluation are introduced in this chapter.

In Chapter 3, description of the experimental setup is given. The chapter starts with
the description of hardware used in experimental setup followed by discussion on
the framework/software modular design. The experimental setup is used to validate
the control algorithms presented in Chapter 4, 5, 6, and 7 respectively.

Chapter 4 discusses the design of a low-level coordination control. The control al-
gorithm is designed by expanding the basic trajectory tracking controller of a single
mobile robot. The algorithm is validated to coordinate a group of mobile robots that
realizes transportation of an automated warehouse. Simulation and experimental
results, as well as a performance analysis are presented.

Chapter 5 describes the high-level coordination control. Using a combination of pre-
diction and negotiation methods, the high-level control works at the level of gener-
ting the trajectories of the robots. Using similar scenarios as in Chapter 4, simulation
and experimental results are presented. Performance analysis and comparison with the low-level coordination control are provided in this chapter.

Chapter 6 discusses the design of a (low-level) coordination control algorithm that is able to simultaneously track individual reference trajectories and maintain a certain desired formation with other robots. The trajectory tracking controller for a single robot designed using dynamic feedback linearization is revisited, followed by the extension to a group of mobile robots. A stability analysis and the influence on how the robots communicate are presented.

Chapter 7 presents a coordination control algorithm based on the MPC approach. The chapter starts with the discussion of MPC for systems with fast dynamics, continued with the proposed sequentially-decentralized MPC. Simulation and experimental results using similar scenarios as in Chapter 4 and 5 are presented. Relevance and performance comparison of the MPC algorithm with the hierarchical control approach are discussed.

Finally, Chapter 8 gives the conclusions of this thesis. The findings from previous chapters are summarized in order to develop a performance comparison and feasibility analysis for using a group of mobile robots to realize transportation in warehouses. Various control parameters that influence the performance and feasibility are discussed. At the end, several issues and challenging research questions for future development are addressed.
This chapter presents necessary mathematical notions used throughout the thesis. In addition, the concept of hierarchical control architecture is explained as well as the benchmark case study. The performance indicators are defined at the end of this chapter.

2.1 Outline

This chapter recalls several concepts and definitions which are useful to design the coordination control algorithms as well as analyzing their stability and performance properties. To start with, Section 2.2 gives a stability theorem on interconnected systems and graph theory.

A description of the nonholonomic kinematic model of unicycles is given in Section 2.3 as well as the trajectory tracking controllers suitable for unicycles. Section 2.4 presents an introduction to hierarchical coordination control and the proposed control architecture used in this thesis. Section 2.5 explains an automated warehouse as the environment to validate the coordination control algorithms. Finally, Section 2.6 provides information on the indicators used to evaluate the performance of the coordination control algorithms.

2.2 Mathematical notions

The notation regarding vector and matrix norms in this thesis is as follows. The vector 1- and 2- norms of vector \( \mathbf{a} \) are denoted \( \| \mathbf{a} \|_1 \) and \( \| \mathbf{a} \|_2 \), respectively. The matrix sum norm, Frobenius norm, and induced matrix 1- and 2- norms of matrix \( \mathbf{A} \) are denoted as \( \| \mathbf{A} \|_{\text{sum}} \), \( \| \mathbf{A} \|_F \), \( \| \mathbf{A} \|_1 \), and \( \| \mathbf{A} \|_2 \), respectively.
2.2.1 Stability of interconnected systems (Khalil, 1996)

Consider the interconnected system

$$\dot{x}_i = f_i(t, x_i) + g_i(t, x), \quad i = 1, 2, \ldots, m,$$

(2.1)

where $x_i \in \mathbb{R}^{n_i}$, $n_1 + n_2 + \ldots + n_m = n$, and $x = \begin{bmatrix} x_1^T & \ldots & x_m^T \end{bmatrix}^T$. The function

$$V(t, x) = \sum_{i=1}^{m} d_i V_i(t, x_i), \quad d_i > 0$$

(2.2)

is a composite Lyapunov function for the collection of the $m$ isolated subsystems for all values of the positive $d_i$. Suppose that for $i = 1, 2, \ldots, m$, $V_i(t, x_i)$ satisfies

$$\frac{\partial V_i}{\partial t} + \frac{\partial V_i}{\partial x_i} f_i(t, x_i) \leq -\alpha_i \varphi_i(x_i),$$

(2.3)

$$\left\| \frac{\partial V_i}{\partial x_i} \right\| \leq \beta_i \varphi_i(x_i),$$

(2.4)

for all $t \geq 0$ and $\|x\| < r$ for some positive constants $\alpha_i$ and $\beta_i$, where $\varphi_i : \mathbb{R}^n \rightarrow \mathbb{R}$ are positive definite and continuous. Furthermore, suppose that the interconnection $g_i(t, x)$ satisfies the bound

$$\left\| g_i(t, x) \right\| \leq \sum_{j=1}^{m} \gamma_{ij} \varphi_j(x_j),$$

(2.5)

for all $t \geq 0$ and $\|x\| < r$ for some nonnegative constants $\gamma_{ij}$. In addition, define an $m \times m$ matrix $S$ whose elements are defined by

$$s_{ij} = \begin{cases} \alpha_i - \beta_i \gamma_{ii}, & i = j, \\ -\beta_i \gamma_{ij}, & i \neq j, \end{cases}$$

(2.6)

and $S$ is required to be an $M$-matrix. To satisfy an $M$-matrix condition it is sufficient to have the leading principal minors of $S$ positive or $\det S > 0$.

**Theorem 2.1.** (Khalil, 1996) Consider system (2.1) and suppose there are positive definite decrescent Lyapunov functions $V_i(t, x_i)$ that satisfy (2.3) and (2.4) and that $g_i(t, x)$ satisfies (2.5) for all $t \geq 0$ and $\|x\| < r$. Suppose that the matrix $S$ defined by (2.6) is an $M$-matrix. Then, the origin is uniformly asymptotically stable. Moreover, if all the assumptions hold globally and $V_i(t, x_i)$ are radially unbounded, the origin is globally uniformly asymptotically stable.

2.2.2 Elementary graph theory

The following elementary graph theory definitions are taken from Godsil and Royle (2001).
Definition 2.2. A graph $\mathcal{G}$ is a triple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ where $\mathcal{V}$ is an index set representing vertices, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes edges such that an ordered pair $(i, j) \in \mathcal{E}$ iff there is an edge between two vertices $i, j \in \mathcal{V}$, and $\mathcal{A}$ is the adjacency matrix which has entries $a_{ij}$ such that:

$$a_{ij} = \begin{cases} w_{ij}, & \text{if} \ (i, j) \in \mathcal{E}, \\ 0, & \text{otherwise}, \end{cases}$$

(2.7)
in which $w_{ij} > 0$ is a weight. Thus, for all $i, j \in \mathcal{V}$ we have $(i, j) \in \mathcal{E}$ iff $a_{ij} \neq 0$.

Definition 2.3. $N_i \subset \mathcal{V}$ is the set of neighbors of vertex $i \in \mathcal{V}$ defined by

$$N_i = \{ j \in \mathcal{V} | j \neq i \text{ and } a_{ij} \neq 0 \}.$$ (2.8)

Definition 2.4. The Laplacian matrix $\mathcal{L}$ associated with the communication graph $\mathcal{G}$ is an $n \times n$ matrix whose elements $l_{ij}$ are defined as follows:

$$l_{ij} = \begin{cases} \sum_{j=1}^{n} a_{ij}, & \text{if } j = i, \\ -a_{ij}, & \text{otherwise}. \end{cases}$$ (2.9)

Definition 2.5. A graph $\mathcal{G}$ is called undirected if $(i, j) \in \mathcal{E}$ whenever $(j, i) \in \mathcal{E}$. It is said to be connected if any two vertices may be connected by a path regardless of the sequence of the vertices involved en route. Otherwise the graph is said to be unconnected.

2.3 Non-holonomic kinematic model of unicycles

Throughout this thesis, a nonholonomic kinematic model of unicycles is considered as the mobile robot model. One of the reason is the fact that robot used for experiment (see Chapter 3) can be easily modeled as a unicycle. Figure 2.1 illustrates the schematic representation of a unicycle mobile robot.
The position at time $t$ of point $Q$, located at the mid-distance from the driving wheels of the robot, with respect to the global coordinate frame $O_x$, is denoted by the coordinates $(x(t), y(t))$, while the orientation, i.e. the angle between the heading direction of the robot and the $O_z$-axis of the global coordinate frame is denoted by $\theta(t)$. The state of the system is denoted by $\mathbf{q} = [x(t) \ y(t) \ \theta(t)]^T$. The posture kinematic model of the unicycle is given as

$$
\begin{bmatrix}
\dot{x}(t) \\
\dot{y}(t) \\
\dot{\theta}(t)
\end{bmatrix} =
\begin{bmatrix}
\cos \theta(t) & 0 \\
\sin \theta(t) & 0 \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
u_1(t) \\
u_2(t)
\end{bmatrix}.
$$

(2.10)

Here, $u_1(t)$ and $u_2(t)$ are the control inputs, with $u_1(t) = v(t)$ the forward/translational velocity and $u_2(t) = \omega(t)$ the rotational/steering velocity respectively.

### Trajectory tracking controller of unicycles

Throughout this thesis, out of the three basic motion tasks for a single mobile robot explained in Section 1.1.1, the trajectory tracking is chosen because of the time property of the desired paths. In this thesis, the coordination control algorithms are designed by extending the trajectory tracking control concept.

The main control objective in the trajectory tracking problem is to accurately track a given reference position and orientation $\mathbf{q}_r = [x_r(t) \ y_r(t) \ \theta_r(t)]^T$. The geometry of the paths is free to choose as long as it satisfies the nonholonomic constraint $-\dot{x}_r(t) \sin \theta_r(t) + \dot{y}_r(t) \cos \theta_r(t) = 0$. Because of this constraint, the reference orientation, forward and steering velocities of the robot are computed using the following formulas:

$$
\begin{align*}
\theta_r(t) &= \arctan\left(\frac{\dot{y}_r(t)}{\dot{x}_r(t)}\right) + k\pi, \quad k = 0, 1, \\
v_r(t) &= \sqrt{\dot{x}_r^2(t) + \dot{y}_r^2(t)}, \\
\omega_r(t) &= \frac{\dot{x}_r(t)\dot{y}_r(t) - \dot{y}_r(t)\dot{x}_r(t)}{\dot{x}_r^2(t) + \dot{y}_r^2(t)}
\end{align*}
$$

(2.11)-(2.13)

The arctan function in (2.11) must consider the sign of each argument in the ratio $\dot{y}_r(t)/\dot{x}_r(t)$ in order to correctly determine to which quadrant the resulting angle belongs. The two possible choices for $k$ in (2.11) allow the same trajectory to be followed either forward ($k = 0$) or backward ($k = 1$). For the trajectory tracking problem to be solvable, it is necessary and sufficient that the reference $\mathbf{q}_r = [x_r(t) \ y_r(t) \ \theta_r(t)]^T$ satisfies the equations:

$$
\begin{bmatrix}
\dot{x}_r(t) \\
\dot{y}_r(t) \\
\dot{\theta}_r(t)
\end{bmatrix} =
\begin{bmatrix}
\cos \theta_r(t) & 0 \\
\sin \theta_r(t) & 0 \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
u_1(t) \\
u_2(t)
\end{bmatrix}.
$$

(2.14)

where $\theta_r(t)$ satisfies (2.11), $u_1(t)$ and $u_2(t)$ satisfy (2.12) and (2.13) respectively.
To achieve an accurate reference tracking, Kanayama et al. (1990) proposed a set of tracking coordinates that relate the reference trajectory $q_r(t)$ and the state $q(t)$ as depicted in Figure 2.2. The associated error states $q_e = [e_x(t) \ e_y(t) \ e_\theta(t)]^T$ are formulated as

$$
\begin{bmatrix}
  e_x(t) \\
  e_y(t) \\
  e_\theta(t)
\end{bmatrix} =
\begin{bmatrix}
  \cos \theta(t) & \sin \theta(t) & 0 \\
  -\sin \theta(t) & \cos \theta(t) & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x_r(t) - x(t) \\
  y_r(t) - y(t) \\
  \theta_r(t) - \theta(t)
\end{bmatrix}.
$$

(2.15)

The error dynamics of (2.15) are given as follows:

$$
\begin{align*}
\dot{e}_x(t) &= \omega(t)e_y(t) + v_r(t)\cos e_\theta(t) - v(t) \quad (2.16a) \\
\dot{e}_y(t) &= -\omega(t)e_x(t) + v_r(t)\sin e_\theta(t) \quad (2.16b) \\
\dot{e}_\theta(t) &= \omega_r(t) - \omega(t) \quad (2.16c)
\end{align*}
$$

The trajectory tracking control problem is formulated as to find a stabilizing control law $u = [v(t) \ \omega(t)]^T$ such that the tracking error $q_e(t)$ converges to zero as time goes to infinity. Examples of trajectory tracking controllers built using the error coordinates given in (2.15) are:

- by Kanayama et al. (1990):
  $$
  \begin{align*}
  v(t) &= v_r(t)\cos e_\theta(t) + k_x e_x(t), \quad k_x > 0 \quad (2.17a) \\
  \omega(t) &= \omega_r(t) + v_r(t)\left[k_y e_y(t) + k_\theta \sin e_\theta(t)\right], \quad k_y, k_\theta > 0. \quad (2.17b)
  \end{align*}
  $$

- by Jiang and Nijmeijer (1997):
  $$
  \begin{align*}
  v(t) &= v_r(t)\cos e_\theta(t) + k_x e_x(t), \quad k_x > 0 \quad (2.18a) \\
  \omega(t) &= w_r(t) + v_r(t)e_y(t)\frac{\sin e_\theta(t)}{e_\theta(t)} + k_\theta e_\theta(t), \quad k_\theta > 0. \quad (2.18b)
  \end{align*}
  $$
Chapter 2. Preliminaries

• by Panteley et al. (1998) and further studied by Lefeber et al. (2001):

\[ v(t) = v_r(t) + k_x e_x(t) - k_y \omega_r(t) e_y(t), \quad k_x > 0, k_y > -1, \quad (2.19a) \]

\[ \omega(t) = \omega_r(t) + k_\theta e_\theta(t), \quad k_\theta > 0. \quad (2.19b) \]

• by Kostič et al. (2009):

\[ v(t) = v_r(t) \cos e_\theta(t) + \Phi_k e_x(t), \quad (2.20a) \]

\[ \omega(t) = \omega_r(t) + \frac{k_x k_y e_x(t) v_r(t) \sin e_\theta(t) + \Phi_k e_y(t)}{\sqrt{1 + (k_x e_x(t))^2 + (k_y e_y(t))^2}} \quad (2.20b) \]

where \( k_x, k_y, k_\theta, k_\Phi > 0 \), and \( \Phi_k e_x(t) \) and \( \Phi_k e_y(t) \) belong to the class of saturation functions defined in Kostič et al. (2009).

The first and last controllers are designed using a Lyapunov-based approach. The second controller is designed using a backstepping technique, whereas the third one is designed using a cascaded structure approach. Different from the rest, the fourth controller takes into account the saturation effect in the input signals. This means that the resulting control signals satisfy the relations: \( |v(t)| \leq v_{\text{max}} \) and \( |\omega(t)| \leq \omega_{\text{max}} \).

2.4 Control architecture

One important specification in designing coordination control algorithms is flexibility. In this thesis, flexibility means that the control algorithms are adaptable to system changes, e.g. addition or reduction of the number of robots in the group, changes in the group's tasks, or accommodating different control approaches.

As mentioned in Siciliano and Khatib (2008), a way to achieve flexibility is by decomposing the main control objective into modular sub-tasks. Decomposition can be done vertically, a top-down approach which results in a hierarchical structure or horizontally, a flat decomposition which results in a non-hierarchical structure or a combination of both. Figure 2.3 illustrates the three decomposition approaches.

The advantage of a vertical decomposition is the possibility to separate the control design in each layer. Changing the algorithms or strategies in one layer does not necessarily require changes in other layers. The final decision of the complete hierarchy is easier to maintain since decision in one layer is affected by the higher layers. Thus, the highest layer, if necessary can overrule all other layers below it. This may be needed, for instance, in case of an emergency. However, the dependency on the higher layer can be a disadvantage since failure to deliver a decision can cause a failure of the whole system.

On the other hand, horizontal decomposition reduces the effect of dependency between the modules (sub-objectives) since the layers are on the same level. All modules know the main control objectives so that each can make its own decision. However, the overall decision can be difficult to make if the number of modules is large.
By combining vertical and horizontal decomposition, the advantages of both approaches can be exploited.

In this thesis, the coordination control objective is mapped onto the trajectory tracking control problem. By its nature, the trajectory tracking problem requires an admissible reference trajectory, i.e., it should satisfy the nonholonomic property in the case of unicycle mobile robots, and a trajectory tracking controller. If a group of robots is considered, the coordination can be done either at the level of generating the trajectory or at the level of tracking the reference trajectory or at both levels. These coordination options are closely related to vertical decomposition. Thus, it is natural to choose the hierarchical control architecture.

Throughout this thesis, a general three-layer hierarchical coordination control architecture, depicted in Figure 2.4, is proposed. The layers and their responsibilities are:

- **Motion Planner.** This layer is responsible for generating the reference trajectories of all robots. This layer can be further expanded vertically to accommodate other functionalities related to planning. For instance, in an application where a group of mobile robots is used to realize transport in warehouses, this layer can be expanded to accommodate order generation, order distribution, task allocation, and robots’ dispatching functionalities. This layer is also called...
2.5 An automated warehouse

For the case study, the coordination control algorithm is used to regulate a group of mobile robots in an automated warehouse depicted in Figure 2.5 (Andriansyah et al., 2010). The idea is to use the robots to realize the transportation system. The automated warehouse shown in Figure 2.5 distributes slow-moving products to a number of supermarkets and is categorized as a product-to-picker warehouse (van den Berg, 1999). The warehouse consists of three main components, namely: miniloads, workstations, and conveyors. Miniloads are storage racks equipped with cranes to serve two functions: storage and retrieval of product totes. The workstations serve as the place to collect items from product totes to order totes. The conveyor is responsible to transport product totes from miniloads to workstations and vice versa.

Human operators (pickers) are responsible to complete orders. An order consists of several items. The number of items in an order is referred to as the order size.
The order size may vary significantly. For slow-moving products, the order size is typically small. A product tote contains items of the same type. At the workstation, arriving product totes create queues on buffer conveyors. Once the picker and the required product totes are available, the product totes will be moved from the buffer to the picking place where the picker stands.

In this thesis, it is investigated how the conveyor system of the automated warehouse is replaced by a group of mobile robots. As an example, Figure 2.6 shows a situation where the main conveyor that responsible for delivering items from miniloads to workstations is replaced by a group of robots. The robots are allowed to take any path. They can move freely within the area that connects the miniload and workstations. In this way, a specific desired path can also be assigned to the robots.

2.6 Performance indicators

To evaluate the performance of the coordination control algorithms, several indicators related to general control objectives in coordination between mobile robots as well as indicators related to the transport system of an automated warehouse are used.

2.6.1 Completion time ($t_{\text{complete}}$)

Completion time is the time needed to accomplish the complete transportation task in a warehouse. This indicator also gives information about the throughput of the warehouse. The lower the value of completion time, the higher the number of tasks that can be handled. This also means a higher throughput in the warehouse. The
Completion time is computed as follows:

$$t_{\text{complete}} = t_{\text{last,task}} - t_{\text{first,task}}$$ \hfill (2.21)

where $t_{\text{first,task}}$ and $t_{\text{last,task}}$ are the time for starting the first and completing the last task, respectively. Since typically robots have to start/stop from/to parking positions (not seen in Figure 2.6), the completion time is practically counted from the time the robots start to move from their parking positions, execute the picking/dropping tasks, and back to the parking positions.

### 2.6.2 Robustness ($\rho_{\text{robustness}}$)

Robustness is defined as the ability of the transport system in a warehouse to cope with uncertainties and faults during operational time. The level of robustness is quantified as the difference between completion time in normal and exceptional situations, i.e., if uncertainties and faults occur. A lower robustness value means that the system is more robust. The robustness is formulated as follows:

$$\rho_{\text{robustness}} = \frac{t_{\text{complete,exceptional}} - t_{\text{complete,normal}}}{t_{\text{complete,normal}}}$$ \hfill (2.22)

### 2.7 Summary

This chapter provided the mathematical background, the unicycle kinematic model and its trajectory tracking controller, the proposed control architecture, and the performance measures used throughout this thesis. The next chapter explains the design of the framework and the experimental setup used to validate the coordination control algorithms.
This chapter discusses the experimental multiple mobile robot setup used to implement and validate the coordination control algorithms developed in this thesis.

3.1 Introduction

In this chapter, the multi-mobile robot experimental setup used to validate, by simulation and experiments, the coordination control algorithms proposed in Chapter 4, 5, 6, and 7, is introduced. This chapter also describes the modular framework (software) design that allows quick switches between a simulation and an experimental environment.

A multiple mobile robot experimental setup was originally designed to investigate coordination control of unicycle mobile robots using a virtual structure approach at the Dynamics and Control Group, Department of Mechanical Engineering, Eindhoven University of Technology (D&C TU/e) (van den Broek, 2008). The size of the setup is 1.75 [m] × 1.28 [m] and it is able to handle up to seven robots during experiments. The software was developed using Phyton programming language.

The experimental setup used in this thesis is bigger from the one mentioned above. The maximum possible size is 3.6 [m] × 2.2 [m]. The setup is able to handle up to 15 mobile robots. The framework is developed using a combination of C++ and MATLAB programming languages. The setup has proven to be very useful to test different control algorithms for a single or a group of unicycle mobile robots, see for example: Bodden (2011), Kostić et al. (2009), Kostić et al. (2010a), Kostić et al. (2010b). The next two sections explain the hardware choices and the framework design of the experimental setup.
3.2 Hardware components

The complete experimental setup is depicted in Figure 3.1. The setup consists of a removable and resizable arena, e-puck unicycle type mobile robots (Mondada et al., 2009), a two-camera system for localization, and a PC equipped with Bluetooth communication.

![Figure 3.1: The experimental setup at D&C TU/e, assembled for a public demonstration day at the university (left) and assembled inside the lab (right).](image)

The arena

The arena is made from a collection of white-dance-floor tiles. This choice allows to easily and quickly move the setup, as well as resizing the arena. Figure 3.2 shows a single tile and how the collection is assembled.

![Figure 3.2: A single white dance floor tile (left) and an example of how a collection of tiles is assembled (right).](image)

The gap between consecutive tiles is very small so that the mobile robots do not slip
3.2 Hardware components

in the tile joints.

Mobile robot platform

The mobile robot platform selected is the e-puck (Mondada et al., 2009), depicted in Figure 3.3. The e-puck is a differential-drive unicycle-type mobile robot developed at the EPFL, Switzerland. The wheels of the robot are driven by stepper motors. The motors are actuated by sending the desired right and left wheel speeds (computed from desired forward and steering velocities). Depending on the setting, the computation of these velocities can be done in the robot’s onboard computer or in an external PC. If done in an external PC, the commands are sent to the robot via a Bluetooth communication protocol.

Figure 3.3: CAD rendering of the e-puck mobile robots (left) and a group of e-pucks in the arena (right).

The ability to communicate with a PC via Bluetooth allows the robot to be controlled using a variety of programming languages. This option also allows different ways of implementing the control algorithms, i.e. it can be implemented in centralized or decentralized/distributed way.

Two-camera localization system

The localization of the mobile robots is implemented by a two-camera system. The two cameras are needed to cope with the maximum size of the arena. Each camera is a network camera Prosilica GC1350MH as shown in Figure 3.4. The cameras are mounted on a beam that connects two foldable ladders. The cameras have to be placed between 1.5 [m] – 2 [m] above the floor to cover the maximum size of the arena. The two cameras are synchronized and connected to a PC via gigabit ethernet. Each robot is fitted with a unique marker of 7 [cm] × 7 [cm]. Examples of the markers are shown in the left side of Figure 3.5. The markers are originally designed by Caarls (2009). Out of 32 possibilities, 31 combinations are available for the mobile robots. The last one, as depicted in the right side of Figure 3.5, is used to calibrate the relative position between the cameras and the arena.
Figure 3.4: The gigabit ethernet camera (left) and the two-camera system mounted on a beam that connects two foldable ladders (right).

Figure 3.5: Example of the markers (left) and the marker for calibration and initialization (right) (Caarls, 2009).

PC equipped with Bluetooth connection

Due to the limited computing power of the onboard processor of the e-pucks, all position and orientation measurement and control signal calculations are done in an external PC. The resulting control signals are sent to the robot via the Bluetooth communication protocol. Although it is implemented in this way, the software framework also supports onboard implementation as will be explained in Section 3.3. The PC specifications are as follows:

- Processor Intel i7 920 (quad core) 2.67 GHz.
- 3 GByte memory.
- ASUSTEK Motherboard P6T Deluxe V2 equipped with two-input gigabit ethernet.
- NVIDIA 512MB Graphics card.
3.3 Framework design

The software framework is designed such that the modularity, i.e. the software component is reusable, can be achieved. This allows up scale the system easily and to try out different control algorithms.

To fulfill the specifications mentioned in the previous paragraph, the framework uses separate control modules from actual hardware via abstraction layers to allow real and simulated equipment to work together transparently. The abstraction layer provides a way to use robots in the simulation mode, and perform real-time experiments without having to change the software. All connections between modules in the framework are following the subscriber/publisher paradigm (Siciliano and Khatib, 2008) on the exchanged data. In this paradigm, a component publishes data and other components can subscribe to that data. The data is identified by a named magazine and actual data packets are called issues of the magazine.

3.3.1 Abstraction layer

The framework consists of a hardware abstraction layer and a control architecture layer. The hardware abstraction layer is shown in Figure 3.6.

![Hardware abstraction layer diagram](image)

The modules in the abstraction layer talk to the hardware in normal mode, i.e. when a real-time experiment is conducted, and talk to each other in simulation mode, i.e. when a simulation is conducted. The event scheduler module is used by all modules to abstract away from the real time, such that in simulation mode time can run faster. In principle, the hardware abstraction can be extended to allow any actuator or sensor. Currently, for the experimental setup only the mobile robots and two cameras are in use.

Figure 3.6 shows how the publisher/subscriber communication paradigm works to make the framework very flexible. The communication library used in the framework automatically takes care of the transportation of data, whether in the same
program, on the same computer, or on different computers. This way of communication makes it possible to easily change the physical (hardware) implementation of modules. In addition, each module can be implemented in the preferable programming language of the designer. Currently, MATLAB and C/C++ modules are used. The framework enables the use of proxy modules that act as gateways to other communication protocols like TCP/IP. In this way modules from any language can be made to communicate.

On top of the hardware abstraction, the second abstraction layer, called a layered control architecture, is defined. The structure of this layered control architecture is made adjustable to accommodate different control approaches. As an example, Figure 3.7 shows a layered control architecture for the trajectory tracking control of a mobile robot.

![Figure 3.7: Control abstraction layer.](image)

In this example, the control abstraction layer consists of a trajectory generator that is responsible to generate a reference trajectory, a trajectory tracking controller that is responsible for an accurate tracking of the reference, and a trajectory modifier that, if needed, is responsible for modifying the reference trajectory to guarantee a collision-free reference trajectory. In each layer, different control algorithms can be easily implemented and tested. For other applications, the control abstraction layer can be extended based on the needs. Any additional robot in the system implements the same hardware and control abstraction layers.

The layers in the control abstraction can be mapped directly to the control modules. Due to the publisher/subscriber mechanism, different ways of mapping the layers into the modules can be realized. Figure 3.8 shows the examples of centralized, decentralized, and distributed solutions for the mapping of trajectory modifier (TM) and trajectory tracking controller (TC) layers of the control abstraction layer depicted in Figure 3.7.
3.3 Framework design

For the existing experimental setup, assuming the control abstraction layer covers the complete transport task in an automated warehouse, the complete module implementation is shown in Figure 3.9.

Due to the limited power of the onboard processor of the e-pucks, all control algorithms are implemented on an external PC. The e-pucks communicate via a Bluetooth protocol with the PC. The e-puck proxy communicates wheel speeds via these virtual com-ports. Each e-puck proxy subscribes to wheel speed data for the specific e-puck it is communicating with. The magazine name is /epucks/<number>/wheel-speeds.

The two-camera system is handled by the Processing program and Pose estimator modules. The Processing program instructs the cameras to send their HD images (1360 [pix] × 1020 [pix]) at 30[Hz]. The position and orientation (poses) of all visible e-pucks, seen from their markers, are calculated with respect to a coordinate system with the z-axis pointing upwards. The coordinate system is usually represented by a large marker (as shown in the right side of Figure 3.5) in the view of both cameras. For each image, the position of each camera is computed with respect to this big marker. If the big marker is not seen, the last known position of the marker (recorded in the hard drive of the PC) is used. The Processing programs publish the calculated poses in the magazines: /epuck/cameras/<number>/poses.

Figure 3.8: Examples of solutions to the mapping of trajectory modifier and trajectory tracking controller layers onto modules.

3.3.2 Framework implementation for the existing experimental setup

For the existing experimental setup, assuming the control abstraction layer covers the complete transport task in an automated warehouse, the complete module implementation is shown in Figure 3.9.

Due to the limited power of the onboard processor of the e-pucks, all control algorithms are implemented on an external PC. The e-pucks communicate via a Bluetooth protocol with the PC. The e-puck proxy communicates wheel speeds via these virtual com-ports. Each e-puck proxy subscribes to wheel speed data for the specific e-puck it is communicating with. The magazine name is /epucks/<number>/wheel-speeds.

The two-camera system is handled by the Processing program and Pose estimator modules. The Processing program instructs the cameras to send their HD images (1360 [pix] × 1020 [pix]) at 30[Hz]. The position and orientation (poses) of all visible e-pucks, seen from their markers, are calculated with respect to a coordinate system with the z-axis pointing upwards. The coordinate system is usually represented by a large marker (as shown in the right side of Figure 3.5) in the view of both cameras. For each image, the position of each camera is computed with respect to this big marker. If the big marker is not seen, the last known position of the marker (recorded in the hard drive of the PC) is used. The Processing programs publish the calculated poses in the magazines: /epuck/cameras/<number>/poses.
The Pose estimator module subscribes to these magazines to receive marker (robot) positions from both cameras to combine data from the same time stamp. If a marker is seen by both cameras, its position is calculated as a weighted sum of both estimates with respect to the distance between the center of the marker and the center of the image in image coordinates. The e-puck position is published in the magazines: /epuck/positions.

The Trajectory generator (TG) calculates the reference trajectories of each robot from parameters that can be set via the command magazines. The maximum and nominal speed parameters are used to determine a timed trajectory. When needed, e.g. to generate collision-free trajectories for all robots, the TGs communicate to allow negotiations regarding the time entering common waypoints in the references. Currently the TG is implemented both in MATLAB and C++.

The Trajectory tracking controller (TC) subscribes to the e-puck positions and computes the necessary wheel speeds as soon as e-puck positions are received.

Figure 3.9: The implementation of the framework for experimental setup.
3.4 Summary

These wheel speeds are published for each e-puck in /epucks/<number>/wheel-speeds. If needed, Trajectory modifier(TM) can be activated to modify the reference trajectories so that no collision occurs between the robots. Currently, the TC is implemented in MATLAB. Since native MATLAB cannot subscribe/publish magazines, mex-files are developed in C++ to interface with the corresponding magazines.

3.4 Summary

In this chapter, the choice of the hardware for the experimental setup has been described. In addition, the design of the modular software/framework has been explained. The framework allows quick switches between simulation and experiments, as well as testing different control algorithms. The framework can handle up to 15 robots in real-time experiments and as many robots in simulation. The next chapters will discuss the design of coordination control algorithms for a group of mobile robots as well as how the experimental setup is used to validate the control algorithms.
This chapter discusses the design of a low-level coordination control algorithm. The control algorithm is designed using a combination of trajectory tracking control and a reactive-to-dynamic-change control concept. The applicability is demonstrated by using the control algorithm to regulate a group of mobile robots that realizes the transportation system of an automated warehouse. Simulations and experiments are conducted to validate the control algorithm. The performance of the control algorithm is evaluated in terms of completion time and robustness against failures.

4.1 Introduction

The study of coordination control has become an important aspect in multiple robotic systems. A collection of mobile robots is expected to be more robust against failures, cope with more complex tasks, and achieve a better overall group performance in comparison to a single robot. As mentioned in Section 2.4, the coordination between robots can be mapped onto a trajectory tracking control problem. Using this approach, the coordination control problem can be formulated as a hierarchical control problem, where coordination between the robots can be regulated either at the level of generating the trajectory or at the motion level of the robots.

If the coordination between the robots is implemented at the level of generating the trajectory, the motion planner has to transform the group task into a series of trajectories for each robot. On the other hand, the low-level coordination requires a trajectory tracking controller that is able to track individual references as well as to maintain the group task.

This chapter focuses on the design of a low-level coordination control which is based on a reactive-to-dynamic approach. To evaluate the performance, the algorithm is
used to coordinate a group of mobile robots that realizes the transport system of an automated warehouse.

The contribution of this chapter is twofold. First, a simple, yet powerful, low-level motion controller consisting of a tracking control and a trajectory modifier that is responsible for collision avoidance is proposed. Second, a performance analysis of the proposed control algorithm using a performance measure of transport in an automated warehouse is presented.

The remainder of this chapter is organized in the following way. Section 4.2 discusses the extension of the control architecture proposed in Section 2.4 to accommodate the full functionalities of an automated warehouse. Section 4.3 provides the design of the proposed low-level coordination control and its trajectory modifier. Section 4.4 briefly discusses how the group of mobile robots replaces the main conveyor system of an automated warehouse. Section 4.5 provides simulation and experimental results, as well as the performance analysis of the proposed control algorithm. The chapter concludes with a discussion in Section 4.6.

4.2 Extension of the control architecture

This section explains how the coordination control architecture proposed in Section 2.4 is extended to accommodate the functionalities of an automated warehouse. This also shows how modularity can be achieved within the architecture. In addition, some assumptions for the motion planner (high-level control) are given.

4.2.1 Additional layers to high-level control

The control architecture shown in Figure 4.1 needs to be extended so that the full functionalities of the transportation system in an automated warehouse can be accommodated. Examples are the entities that are responsible for customer order generation, order distribution, and mobile robot dispatching. These entities naturally act as planners that decide what kind of orders need to be generated and distributed
4.2 Extension of the control architecture

to the group of robots. Thus, the extension mostly focuses on the motion planner part of Figure 4.1. Furthermore, since the focus is on the design of a low-level control, in this chapter it is shown how the adjustable layer is shifted to the motion executor layer. Figure 4.2 shows the extended control architecture.

Figure 4.2: The extended control architecture to accommodate a number of functionalities of transport in warehouses.

**High-level Control**

- **Order Generator.** This layer is responsible for generating customer orders, including information on how many items per order are generated.
- **Picking/Dropping Generator.** This layer is responsible for deciding to which picking/dropping locations the robots have to go.
- **Trajectory Generator.** This layer is responsible for generating the reference trajectories of all robots. In this chapter, it is assumed that the resulting reference trajectories are not free of collisions. This is to show how a responsibility can be shifted between the layers.

**Low-level Control**

- **Trajectory Modifier.** This layer is responsible for modifying, if necessary, the reference trajectories so that the movements of all robots are free of collisions. In this chapter, this layer is considered as a part of the low-level control.
- **Trajectory Controller.** This layer is responsible for accurately tracking the reference trajectories. In this layer, the trajectory tracking controllers shown in Section 2.3 can be implemented.

Throughout this chapter, it is assumed that there exists a high-level controller that is responsible for generating the necessary information for all robots, i.e. orders, picking/dropping location, and reference trajectories. As for the low-level controller, it
is assumed that each robot individually implements the control algorithm. As explained in Chapter 3, the framework will easily accommodate these two choices. Although the focus of this chapter is on the design of the low-level controller, the information from high-level control must be available. The next section briefly discusses an option to generate this high-level information.

### 4.2.2 An option for the high-level control

#### Order Generators

In research about warehouses, e.g. see de Koster et al. (2007), Gu et al. (2007) or Gu et al. (2010), the number of orders and how many items in each order used in simulation case studies are either generated based on real data from a warehouse or based on a random distribution. In this thesis, the random distribution, which is sufficient for illustrative purpose, is chosen. In real automated warehouses, the number of orders are generated from real-time customer orders. This step produces order identification number, \( \text{OrderNumber} \), and the number of items per order, \( \text{NumItemsPerOrder} \):

\[
\text{OrderNumber} = 1, 2, 3, \ldots, \beta, \quad (4.1)
\]

\[
\text{NumItemsPerOrder}(i) = a_i, \quad i \in \{1, 2, 3, \ldots, \beta\}, \quad (4.2)
\]

where \( a_i \) is a random number sampled from uniform distribution. In the second step, random picking/dropping positions, where each item in (4.1)-(4.2) has to be collected, are selected. These picking/dropping positions can be anywhere in the automated warehouse layout. This step produces a Cartesian picking position \( \text{PickPosition} \), and a dropping position \( \text{DropPosition} \):

\[
\text{PickPosition}(i, j) = \left[ \begin{array}{c} \text{PickX}(i, j) \\ \text{PickY}(i, j) \end{array} \right]^T, \quad (4.3)
\]

\[
\text{DropPosition}(i, j) = \left[ \begin{array}{c} \text{DropX}(i, j) \\ \text{DropY}(i, j) \end{array} \right]^T, \quad (4.4)
\]

\( i \in \{1, 2, \ldots, \beta\}, \quad j \in \{1, 2, \ldots, a_i\} \).

In addition, the vectors \( \text{CompletePickPosition} \) and \( \text{CompleteDropPosition} \) are formulated as follows:

\[
\text{CompletePickPosition} = [\text{PickPosition}^T(1, 1) \ldots \text{PickPosition}^T(\beta, a_\beta)]^T, \quad (4.5)
\]

\[
\text{CompleteDropPosition} = [\text{DropPosition}^T(1, 1) \ldots \text{DropPosition}^T(\beta, a_\beta)]^T, \quad (4.6)
\]

### 4.2.3 Ring-task allocation

The variables formulated in (4.5)-(4.6) give a full set of transport tasks during a period of time. The next step is to divide the tasks within the group of robots. In this thesis a fixed task assignment named ring-task allocation is implemented.
4.3 Low-level coordination control

The lengths of CompletePickPosition and CompleteDropPosition are the same and are denoted \( L_{\text{complete}} \). If there are \( m \) robots in the group, the following can be computed:

\[
L_{\text{complete}} = \text{Ring} \times m + \text{Remain},
\]

\[
0 \leq \text{Remain} < m,
\]

where Ring and Remain are positive values. The tasks are allocated to each robot as follows:

1. The first CompleteDropPosition(1), \ldots, CompleteDropPosition(m) tasks are directly assigned to robot 1, 2, \ldots, \( m \) respectively.
2. The next \( m + 1 \) to 2\( m \) CompleteDropPosition tasks are assigned to robot 1, 2, \ldots, \( m \) respectively as well.
3. The process is repeated until the repetition equals Ring.
4. If Remain \neq 0, then the last Remain picking and dropping tasks are distributed to the first Remain robots.

Based on the task allocation, the trajectory generator generates the real-time reference trajectories for each robot. The trajectories also depend on the location of picking/dropping generated in (4.3)-(4.4). Without loss generality, it is assumed that the reference trajectories can be generated using a combination of lines and partial circular trajectories. In this work, the trajectory is parameterized such that it has a constant forward velocity along the path.

If \( v_i \) and \( s_i \) represent the reference forward velocity and the geometric path of robot \( i \) respectively, the reference trajectories for robot \( i \) can be formulated as follow:

\[
q_i(t) = \begin{bmatrix} x_i(s_i, v_i(t)) \\ y_i(s_i, v_i(t)) \\ \theta_i(s_i, v_i(t)) \end{bmatrix},
\]

(4.8)

Remark 4.1. The high-level approach explained above is not necessarily the optimal solution with respect to the throughput and the resource availability, i.e. the number of robots of the transport system. However, it serves the purpose of giving the necessary high-level control information for the low-level control. Other algorithms can be easily implemented within the framework. For an optimal solution, an algorithm based on the real-time dynamics of the automated warehouse must be considered.

4.3 Low-level coordination control

The main task of the low-level control is to execute the motion of the robots. In this chapter, the adjustable layer that is responsible of handling collision avoidance is embedded into the low-level control. This section explains the choice of the trajectory tracking controller used in this thesis as well as the choice for the collision avoidance algorithm.
4.3.1 Trajectory tracking controller

To execute the motion while following the assigned reference trajectories, each robot is provided with a trajectory tracking controller. Throughout this thesis, the trajectory tracking controller proposed in Jiang and Nijmeijer (1997) is used for the $i$-th robot:

$$ v_i(t) = v_{ri}(t) \cos \theta_i(t) + k_x e_x(t), \quad (4.9a) $$

$$ w_i(t) = w_{ri}(t) + k_y v_{ri}(t) \frac{\sin \theta_i(t)}{\epsilon_y(t)} + k_\theta e_\theta(t), \quad (4.9b) $$

where $k_x, k_y, k_\theta > 0$ are the control gains and $\sin \frac{\theta_i}{\gamma} = 1$ if $e_\theta_i = 0$. One of the reasons for choosing this control algorithm is the ability of the algorithm to accurately follow straight line reference trajectories, i.e. $v_{ri} \neq 0$ and $\omega_{ri} = 0$. The control gains are easy to tune, e.g. by hand or trial and error, for both simulation and real-time experiments.

4.3.2 Local collision avoidance

In addition to the trajectory tracking controller, a trajectory modifier layer is implemented to ensure collision-free movements of the robots. In an automated warehouse, a collision threat occurs when a robot has to wait for other robots to finish their tasks or a robot has to alter its path to avoid an unwanted object or a broken robot. In this case study, two low-level collision avoidance algorithms, which are based on a penalty concept function (Kostič et al., 2009) and an artificial potential function (Kostič et al., 2010a), are implemented. Each algorithm active in different situations depending on the dynamics of the transportation system.

Slowing down using the penalty function approach

A set $P_\gamma$ of continuous, monotone, and bounded penalty function indexed by a constant parameter $\gamma \in \mathbb{R}^+$ (Kostič et al., 2009) is introduced:

$$ P_\gamma = \{ \delta_\gamma : \mathbb{R} \to \mathbb{R} | \delta_\gamma \text{ is continuous monotone and} $$

$$ \quad \delta_\gamma(x) = 0 \text{ if } x < \gamma_{\text{min}}, $$

$$ \quad 0 \leq \delta_\gamma(x) \leq 1 \text{ if } \gamma_{\text{min}} \leq x \leq \gamma_{\text{max}}, $$

$$ \quad \delta_\gamma(x) = 1 \text{ if } x > \gamma_{\text{max}} \}. \quad (4.10) $$

An example of a function in $P_\gamma$ is

$$ \delta_\gamma(x) = \begin{cases} 
0, & x < \gamma_{\text{min}}, \\
\frac{1}{2} \left( x - \frac{\gamma_{\text{min}}(2\pi x / \gamma)}{2\pi} \right), & \gamma_{\text{min}} \leq x \leq \gamma_{\text{max}}, \\
1, & x > \gamma_{\text{max}}. 
\end{cases} \quad (4.11) $$
4.3 Low-level coordination control

Figure 4.3 illustrates an example of a penalty function where $x$ in (4.11) represents the distance $d_{ij}$ between robots $i$ and $j$. The value of $\gamma_{\text{min}}$ represents the minimum distance between the robots, while $\gamma_{\text{max}}$ represents the distance at which a robot must start to slow down.

![Figure 4.3: An example of a penalty function $\delta(x)$, $x = d_{ij}$.](image)

The penalty function is used to penalize the reference forward velocity so that the complete reference trajectories can be updated to accommodate the dynamic changes in the transportation system. Consider a situation where two robots almost collide as shown in Figure 4.4(a).

![Figure 4.4: a) Situation in which robot $j$ stops. Robot $i$ has to modify its path to avoid a collision; b) Situation with four robots at two junctions.](image)

In this example, robot $j$ stops to pick items. Robot $i$ has to slow down or alter its path to avoid collision. The penalty function is implemented as follows. If $\mathbf{q}_i = [x_i, y_i, \theta_i]^T$ and $\mathbf{q}_j = [x_j, y_j, \theta_j]^T$ are the position and orientation of robots $i$ and $j$ in Cartesian space, define a vector
\[ \mathbf{a}_{\mathbf{v}_{ij}} = \begin{bmatrix} (x_j - x_i) & (y_j - y_i) \end{bmatrix}^T, \]

with its magnitude, representing the distance between the centers of robots:

\[ |\mathbf{a}_{\mathbf{v}_{ij}}| = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}. \]

Define the unit vector describing the direction of travel of robot \( i \):

\[ \text{dir}_{\mathbf{a}_i} = \begin{bmatrix} \cos \theta_i \\ \sin \theta_i \end{bmatrix}^T. \]

The slowing down coefficient of robot \( i \) with respect to robot \( j \) is expressed as

\[ \sigma_{ij} = \begin{cases} 1, & \text{if } \mathbf{dir}_{\mathbf{a}_i} \cdot \mathbf{a}_{\mathbf{v}_{ij}} \leq 0 \\ \delta_{ij} \left( |\mathbf{a}_{\mathbf{v}_{ij}}| \right)^\gamma & \text{if } \mathbf{dir}_{\mathbf{a}_i} \cdot \mathbf{a}_{\mathbf{v}_{ij}} > 0 \end{cases}, \]

where the \( \cdot \) sign represents the dot product of two vectors and \( \delta_{ij} \left( |\mathbf{a}_{\mathbf{v}_{ij}}| \right)^\gamma \) is a penalty function. The computation is repeated for all \( j \neq i \), \( j \in \{1, 2, \ldots, m\} \) robots.

The reference forward velocity of each robot is penalized as follows:

\[ v_{ri} = v_{\text{des}, i} \prod_{j=1, j \neq i}^m \sigma_{ij}, \]

where \( v_{\text{des}, i} \) is the desired forward velocity of each robot. Substituting (4.15) into (4.8), the reference trajectories for robot \( i \) can be formulated as follows:

\[ \mathbf{q}_{ri} = \begin{bmatrix} x_{ri} \\ y_{ri} \\ \theta_{ri} \end{bmatrix} \begin{bmatrix} s_{ri} v_{\text{des}, i} \prod_{j=1, j \neq i}^m \sigma_{ij} \\ s_{ri} v_{\text{des}, i} \prod_{j=1, j \neq i}^m \sigma_{ij} \\ s_{ri} v_{\text{des}, i} \prod_{j=1, j \neq i}^m \sigma_{ij} \end{bmatrix}^T. \]

If \( v_{ri} = 0 \), there is no update of the reference positions, i.e. the robot stays at the last position. The coefficient computed in (4.14) indicates whether robot \( j \) is behind or in front of robot \( i \) relative to the direction of movement of robot \( i \). According to expression (4.15), if robot \( i \) is behind robot \( j \) and the other robots, then robot \( i \) will slow down its velocity to its minimum value to avoid collisions.

**Adding priority**

Consider another situation shown in Figure 4.4(b), i.e. more robots wait to enter the junctions. Applying only (4.15) may result in a deadlock, i.e. no robots move. This is because robots that enter the junction assume that they are behind each other, i.e. \( \sigma_{fg} = \sigma_{gf} \) and \( \sigma_{ij} = \sigma_{ji} \). To avoid the deadlock, the following priority rules are used:

1. **Lower number priority.** At the junction a robot with lower robot ID has higher priority. Thus, if \( i < j \) and \( g < f \):

\[ \sigma_{ij} = 1, \quad \sigma_{ji} = 0, \quad \sigma_{fg} = 0, \quad \sigma_{gf} = 1. \]
2. **Right-hand priority.** At the junction, a robot that comes from the right-hand side of the other robots has higher priority. From Figure 4.4, it can be observed that robot \( j \) is coming from the right-hand side of robot \( i \), as well as robot \( g \) with respect to robot \( f \). As a consequence:

\[
\sigma_{ij} = 1, \quad \sigma_{ji} = 0,
\]

\[
\sigma_{gf} = 0, \quad \sigma_{fg} = 1.
\]

(4.18)

The right-hand priority can be changed to the left-hand priority if at the junction a robot that comes from the left-hand side of the other robots gets higher priority.

### 4.3.3 Generating collision-free paths using artificial potential function (APF)

The algorithm explained in Section 4.3.2 is active if a robot has to wait for other robots to finish their tasks. In a different situation, it can happen that a broken down robot or an unwanted obstacle blocks the paths of other robots. In this situation, applying the slowing down algorithm can bring the system to a standstill. To avoid this situation, a collision avoidance algorithm using APF (Latombe, 1991) is implemented. Using this algorithm, a temporary collision-free path is generated so that the robot can avoid the broken robot or the unwanted obstacle.

APF is widely used in path planning algorithms. Although it can suffer from local minima, APF is very intuitive and easy to implement. In most cases, APF is used to generate a pre-computed (off-line) free-of-collision path for a robot. APF typically consists of a repulsive and an attractive function. The repulsive function makes sure the robots are pushed away from any obstacles, while the attractive function pulls the robots towards their destination. This is illustrated in Figure 4.5(a).

In this work, instead of applying APF as an off-line planner, the APF is used online to locally generate a collision-free trajectory when an obstructor (another robot or obstacle) enters the detection region (Kostić et al., 2010a). The algorithm works as follows. Each robot \( i \) has an APF of the form:

\[
A_{\text{APF}}(q_i, q_j, q_r) = \sum_{j=1, j \neq i}^{m} A_{\text{rep}}^{\text{APF}},(q_i, q_j) + \rho_{\text{at}}(q_i, q_r),
\]

(4.19)

where

\[
A_{\text{rep}}^{\text{APF}}(q_i, q_j) = \begin{cases} 
K_r e^{-\left(\frac{(x_i - x_j)^2}{\alpha} + \frac{(y_i - y_j)^2}{\beta}\right)}, & \text{if } |q_r - q_i| \leq d_r, \\
0, & \text{elsewhere,}
\end{cases}
\]

(4.20)

\[
\rho_{\text{at}}(q_i, q_r) = \frac{1}{2}K_a \left( (x_i - x_r)^2 + (y_i - y_r)^2 \right).
\]

(4.21)
Here \((x_i, y_i)\) and \((x_j, y_j)\) are the Cartesian coordinates of robot \(i\) and robot \(j\), respectively. \(K_{oi}\) is the gain of the repulsive potential function \(A_{\text{rep}}(q_i, q_j)\). \(\alpha\) and \(\beta\) are positive real numbers that can be used to determine the size of the repulsive function. \(d_{s}\) is a positive number that determines the threshold of the detection region. \(K_{ai}\) is the gain of the attractive function \(\rho_{\text{att}}(q_i, q_{ri})\) that is generated from the distance between the actual position of the robot and its original reference trajectory. This choice of APF will push the robots away from each other and attract them back to their original reference trajectories. However, a local minimum may exist that can cause the robot to a standstill. Figure 4.5(b) shows an example of the APFs of several robots in a simulation setting.

From (4.20)-(4.21), the \(A_{\text{rep}}\) becomes zero outside the detection region of robot \(i\).

Inside the detection region, the reference trajectory of robot \(i\) is modified as follows:

1. Determine the Cartesian velocities that will move a robot away from the other robots:

\[
\begin{bmatrix}
\delta v'_{x,i} \\
\delta v'_{y,i}
\end{bmatrix} = \begin{bmatrix}
\frac{\partial A_{\text{rep}}}{\partial x_i} \\
\frac{\partial A_{\text{rep}}}{\partial y_i}
\end{bmatrix} \delta v_i = \sqrt{(\delta v'_{x,i})^2 + (\delta v'_{y,i})^2}.
\] (4.22)
2. Update the collision-free reference trajectory at time-instant $t_k$

\begin{align}
  x_i(t_k) &= x_i(t_{k-1}) + (t_k - t_{k-1}) \delta v_i, \\
  y_i(t_k) &= y_i(t_{k-1}) + (t_k - t_{k-1}) \delta v_i, \\
  \theta_{ri}(t_k) &= \begin{cases} 
  \arctan \left( \frac{\delta v'_y}{\delta v'_x} \right) & \text{if } \delta v_i > 0 \\
  \theta_{ri}(t_{k-1}) & \text{if } \delta v_i = 0
  \end{cases}, \\
  v_{ri}(t_k) &= \sqrt{(\delta v'_x)^2 + (\delta v'_y)^2}, \\
  \omega_{ri}(t_k) &= \frac{\theta_{ri}(t_k) - \theta_{ri}(t_{k-1})}{(t_k - t_{k-1})}. 
\end{align}

**Remark 4.2.** The collision avoidance algorithms either using a penalty function or APF are implemented at the level of modifying the reference trajectory. In this way, the stability of the trajectory tracking controller is still guaranteed. The resulting collision-free trajectory remains continuous. However, for the APF algorithm, inside the detection region of robot $i$ an asymptotic tracking of the reference cannot be guaranteed, which can result in worse tracking performance. It is to be noted that the update of the reference is written in the discrete time as it is implemented.

### 4.4 Transport using a group of mobile robots

The proposed low-level coordination control is used to regulate a group of mobile robots that realizes the transport system of the automated warehouse shown in Section 2.5. In this particular case study, the group of robots is deployed to replace the main conveyor functionality as illustrated in Figure 4.6.

![Figure 4.6: Transport of an automated warehouse.](image)

(a) Using a conveyor.  
(b) Using a group of mobile robots.

The task of the robots is to transport product totes holding items that belong to an order from storage to the workstation buffers. At these buffers, the human picker will collect the items. These items are assumed to be available at the picking point...
when the robots are about to pick them. The transport task is finished if all individual picking/dropping tasks are completed.

The geometric path choices

Without loss of generality, two possible geometric paths, i.e. a single and multiple path geometries, illustrated in Figure 4.7 are considered. The single-path geometry resembles the geometry of the original conveyor system, whereas the multiple-paths geometry allows robots to take some fixed-short-cut from the workstation to the storage.

\[ \text{multiple-paths} \]
\[ \text{storage} \]
\[ \text{WS3} \]
\[ \text{WS2} \]
\[ \text{WS1} \]
\[ \text{single-path} \]
\[ \text{storage} \]
\[ \text{WS3} \]
\[ \text{WS2} \]
\[ \text{WS1} \]

Figure 4.7: The geometric path options for the robots.

Robot specifications

The robots satisfy the following assumptions:

- The size of the robot is 0.8 [m] \( \times \) 0.8 [m]. This equals the size of the biggest totes used in the automated warehouse.
- The nominal speed of the robot is 1 [m/s].
- In each task a robot carries a product tote containing one type of item.

4.5 Simulation and experimental results

Different simulation scenarios are considered. Variation on the number of robots (\( n_{\text{robot}} \)), priority rules, and fault status are simulated. Considering the size of the robots and the automated warehouse, \( n_{\text{robot}} \in \{2, 4, \ldots, 20\} \) are chosen.

For the fault case situation, situations where 2 robots are subject to failure are simulate. It is assumed that a broken down robot can be fixed within 30 minutes, at the position where it stops, after which it becomes operational again.
4.5 Simulation and experimental results

For comparison, the maximum capacity of the conveyor in the automated warehouse is used as benchmark, i.e. the conveyor can transport 1000 totes per hour. Performance is analyzed in terms of the completion time and the robustness measure given in Section 2.6.

The following abbreviations are used: SP-LLC: single-path, low-level control; MP-LLC-LN: multiple-paths, low-level control, low number priority; MP-LLC-LH: multiple-paths, low-level control, left-hand priority; MP-LLC-RH: multiple-paths, low-level control, right-hand priority. The conveyor capacity is identified by CS.

4.5.1 Simulation results

For the simulation, the following control parameters are used:

- Parameters for the penalty function in (4.11):
  \[ y_{\text{min}} = 0.6 \text{ [m]}, \quad y_{\text{max}} = 0.8 \text{ [m]} \]  

- Parameters for the APF in (4.20)-(4.21):
  \[ K_{oi} = 7.5, \quad K_{ai} = 7.5, \quad a = 0.7, \quad b = 0.7, \quad d_i = 0.6 \text{ [m]} \]  

- Gains for the trajectory tracking controller in (4.9):
  \[ k_{xi} = 0.25, \quad k_{yi} = 25, \quad k_{\theta i} = 0.0001 \]  

The parameters for penalty function and APF are chosen by hand and influenced by the size of the simulation environment. The parameters for trajectory tracking controller are tuned from several trials. Figure 4.8 shows the summary of \( t_{\text{complete}} \) from the simulation of different scenarios.

The results depicted in Figure 4.8 show that the flexibility and scalability of the transport system can easily be extended. A different number of robots means that the proposed control strategies can easily handle addition/subtraction of robots from the transport system. As a consequence, the throughput can easily be increased or decreased by doing so. There is no extra space needed. Thus, the proposed control strategies support the idea that by replacing conveyors with a group of mobile robots, a more flexible and scalable transportation in an automated warehouse can be obtained.

In addition, Figure 4.8 shows how the low-level control performs in different scenarios. It can be observed that the transport using a group of mobile robots, using the robot’s specification, can have similar or even better performance compared to the conveyor system. In this particular case, the scenario SP with 16 robots give the optimal throughput. Although adding more robots can increase the capacity, if the space is kept constant, this also means less space for movement. Thus, the queuing and waiting time of a robot (for the other robots) increase, especially at merging points, which results in a longer completion time. In the worst case, a deadlock, i.e. no robots move, can occur if all available space is occupied by the robots.
The deadlock situation also indicates how scalable the system is. The transportation system can be made as flexible as possible by keep adding robots. Since the space is kept constant, a deadlock indicates that adding more robots is no longer beneficial to reduce the completion time. In this case study, Figure 4.8 indicates that 16 is the appropriate choice for the number of robots if it is combined with the correct control strategy. Above 16 the transport system suffers from deadlock. If a deadlock does not occur, the completion time becomes longer than the one when 16 robots are used.

Furthermore, Figure 4.9 shows how robustness against failures can be achieved by switching between the penalty function and APF. The combination of the two algorithms manages to keep the transport system operational under faulty situation. In the examples shown in Figure 4.9, the scenarios SP and MP-LH are investigated. The ability of the APF to generate alternative paths for the normal robot makes the overall transport system still operational under faulty situation, although with lower throughput, i.e. longer completion time. This can be observed by shifting of the original solid-curves to the dash-dotted-curves in Figure 4.9. For the two scenarios, the mean values of $\rho_{\text{robustness}}$ defined in Section 2.6.2 are 0.1066 and 0.1068, respectively.

This result shows that the low-level coordination control achieves robustness against failures. The computation of $\rho_{\text{robustness}}$ from the two scenarios suggest that the robustness value has more dependency on how many times the transport system suffers from failures and how long it takes to fix each fault. The choice of the number of robots and scenarios has less influence on $\rho_{\text{robustness}}$. 

Figure 4.8: $t_{\text{complete}}$ in different scenarios. The highest values means the system suffer from deadlock.
4.5 Simulation and experimental results

4.5.2 Experimental results

The experiments are conducted using the setup and framework explained in Chapter 3. A similar, but smaller, automated warehouse layout as the one used in the simulations is implemented. The robot used in the experiments has a diameter of 0.07 m. The maximum number of robots is 14. Because of the change in size, the control parameters shown in (4.24)-(4.26) need to be modified. Using a similar way as in simulation case study, the following control parameters are used for experiments:

- Parameters for the penalty function in (4.11):
  \[ \gamma_{\text{min}} = 0.09 \, [\text{m}], \quad \gamma_{\text{max}} = 0.085 \, [\text{m}] \]. \quad (4.27)

- Parameters for the APF in (4.20)-(4.21):
  \[ K_{oi} = 20, \quad K_{ai} = 10, \quad a = 0.05, \quad b = 0.05, \quad d_s = 0.05 \, [\text{m}] \]. \quad (4.28)

- Gains for the trajectory tracking controllers in (4.9):
  \[ k_{xi} = 0.4, \quad k_{yi} = 100, \quad k_\theta = 0.5 \]. \quad (4.29)

Remark 4.3. The changes in parameters for the penalty function and APF are directly influenced by the assumption of the systems. For the simulation study, the size of the transportation system and the robots are bigger than the ones used in the experiment. Thus, parameters for simulation are higher than the ones for experiments. For the trajectory tracking control gains, the values in simulation are higher...
since the nominal-reference forward velocity is higher than the one used in experiment. When the robot moves in faster nominal forward velocity, it tends to correct its tracking errors faster. Thus, to make the controller less reactive to small errors, smaller control gains should be chosen. This reasoning also applies for the choices of $K_{ai}$ and $K_{oi}$ of the APF.

The experimental results confirm that the strategies work in a practical situation. The low-level controller can handle some uncertainties in typical real-time situation, e.g. noise in position and orientation measurements and small time-delays in sending and receiving the control signals. Figure 4.10 shows an example of a situation where robots 7, 8, 9 and 10, have to coordinate their movement to avoid collision and deadlock. Videos of experiments can be seen at www.youtube.com/adinandra98

Figure 4.11: (a) Control signals of robots 7, 8, 9, 10; (b) Mutual distances between robots 7, 8, 9, 10.

Figure 4.11 shows the control signals and distances between the robots involved in Figure 4.10 when the collision threat occurs. The results in Figure 4.11(a) illustrate how the low-level algorithm adapts the control signals to solve the collision threat.
In this example, robot 7 stops to pick an item. As a consequence, robot 8 has to slow down to avoid collision, as well as robots 9 and 10. This can be observed in Figure 4.11(a), where the control signals of the robots are zero. Once robot 7 starts to move again, robot 8 will start moving. By the left-hand priority, robot 10 gets higher priority than 9, so robot 10 will move forward followed by robot 9. Figure 4.11(b) shows that the distances between robots 7, 8, 9, and 10 are all above 0.06 [m], which is the safety distance between the robots. Thus, no collision occurs.

In addition, Figure 4.12 illustrates when robot 1 suffers from a failure and the rest of the robots, especially robots 7, 8, and 9 are using a combination of a penalty function and the APF to resolve the situation.

**Figure 4.12:** Threat of collision because robot 1 suffers from a failure. The solid arrows indicates the direction of movement of robot 7, which is partially blocked by robot 1.

**Figure 4.13:** (a) Control signals of robot 1, 7, 8, and 9; (b) Mutual distances between robots 1, 7, 8 and 9.
As shown in Figure 4.12, robot 1 is stopped for some period of time because of a failure. The APF will be active as soon as robot 7 moves closer to robot 1. This is indicated by the chattering control signal as shown by the solid-black curve of Figure 4.13(a). On the other hand, while waiting for robot 7, that has no fault status, robots 8 and 9 are still using the penalty function to slow down to avoid collisions as indicated by solid- and dashed-grey curves in Figure 4.13(a). When robot 7 activates the APF, both solid- and grey-curves show zero values which means robots 8 and 9 are using a penalty function to avoid collision. Figure 4.13(b) shows the mutual distances between robots 1, 7, 8, 9. Observation of this figure indicates that all curves never cross the dash-dotted black line that is the minimum distance to be kept. Thus, no collision occurs between the robots.

As an additional illustration, Figures 4.14 and 4.15 show some snapshots of the experiments. Figure 4.14 shows the snapshots of experiments when the threat of collisions occurs at two junctions (see Figures 4.14(a) and 4.14(b)). In this example, left-hand priority is used. As seen in Figures 4.14(c) and 4.14(d), as robots that are about to enter the same junction come closer, the ones that come from the left-hand side of other robots will get higher priority. As the threat of collision at the junctions is gone, the group of robots operates normally as depicted in Figures 4.14(e) and 4.14(f).

Finally, Figure 4.15 shows snapshots of the threat of collision because of a broken down robot. This situation is similar to the situation depicted in Figure 4.12. In this particular example, as seen in Figure 4.15(a), the robot with the red circle is the one that suffers from failure. As shown in Figures 4.15(b) and 4.15(c), the closest robot, indicated by the green circle, starts to use the APF to create a new-temporary path. The rest of the robots, in the yellow rectangle, are queuing because of the penalty function. Once the closest robot escapes, the first one in the queue also starts to use the APF as seen in Figure 4.15(d). After a while, as shown in Figure 4.15(e), the broken robot also starts to move again (indicated by the yellow circle). As a consequence, the flow of the transport system starts to go back to normal operation again as depicted in Figures 4.15(f) and 4.15(g). Finally, in Figure 4.15(h), it is shown that the transport system is operating normally.
Figure 4.14: Snapshot of experiments when there is a threat of collisions at junctions. The black lines illustrate the geometric paths that can be taken by the robots.
Figure 4.15: Snapshot of experiments when there are threat of collisions because of a broken down robot.
4.6 Conclusions

In this chapter a methodology to coordinate a group of mobile robots using a hierarchical approach, focusing on low-level control design is presented. The proposed low-level coordination algorithm is used to regulate a group of mobile robots that realize the transportation system of an automated warehouse. The simulation and experimental results show that the proposed control algorithm can easily accommodate additional robots in the system. Thus, the scalability of the transport system can easily be extended/reduced. In addition, the proposed low-level coordination control also shows some robustness against failures in the transport system. It is also shown how the proposed framework can accommodate different control strategies.

The simulation results also show that the realization of the transport system using a group of mobile robots can achieve a similar, and even better performance, depending on the choice of the number of robots and control strategies, compared to a conveyor system. Although this statement is made under some assumptions, this information from simulation is very important for either the control engineer or warehouse designer at an early design phase.

The next chapter will discuss another coordination control algorithm called a high-level coordination control where the coordination is done at the level of generating the reference trajectories for all robots.
This chapter discusses the design of a high-level coordination control algorithm. Different from the low-level control algorithm presented in Chapter 4, the proposed high-level coordination works at the level of generating the trajectory for each robot. The algorithm uses a prediction of the movement of individual robots. Similar to Chapter 4, the controller is used to regulate a group of mobile robots that realize the transportation of an automated warehouse. Both simulations and experiments are presented in addition to a performance analysis of the control algorithm. Performance comparison between the low- and high-level coordination control is presented. Furthermore, an alternative performance measure is discussed. Finally, a feasibility study on the cost of realizing the transport using a group of mobile robots is given.

5.1 Introduction

In Chapter 4, a proposed low-level coordination controller was discussed in detail. It was shown how issues in coordinating the robots are solved by the motion executor of the robots (from the perspective of the proposed control architecture). As mentioned in Section 2.4, by using the hierarchical control concept, the coordination between the robots can also be implemented at the motion planner level of the robots. In this case, the motion planner and adjustable layer shown in Figure 2.4 is seen as one entity: a high-level control. The high-level coordination control generates a series of trajectories for each of the robots so that the robots accomplish the group’s tasks.

In this chapter, a high level coordination control strategy based on a prediction method is introduced. In the particular case study of transport in a warehouse, this
means that the high-level control has to generate collision free trajectories for all robots. Using a prediction method, the high-level control estimates the movement of all robots so that when a robot has to wait for other robots, either for picking items or at a junction, no collision occurs. In Chapter 4, the task of solving the collision threats was shifted to the low level control.

The contributions of this chapter are the following. First, a high-level control algorithm that generates collision-free trajectories for all robots while maintaining the collective goal of the group is proposed. Second, a performance analysis of the proposed control algorithm using the performance measures employed in Chapter 4 is presented. In addition, a performance comparison of the high- and low-level coordination control is given. Some important aspects of using a hierarchical control concept for a group of mobile robots are addressed. Third, a feasibility study on the cost of realizing the transport in warehouses using a group of mobile robots is conducted.

The remainder of this chapter is organized in the following way. Section 5.2 provides the design of the high-level coordination control. Section 5.3 provides simulation and experimental results, as well as a performance analysis of the proposed high-level coordination control. Section 5.4 gives the performance comparison between the high-level and low-level control. Section 5.4 also provides a cost feasibility analysis of the system. In Section 5.5, an alternative performance comparison between the high- and low-level control is given. The chapter concludes with a discussion in Section 5.6.

5.2 High-level coordination control

This section describes the design of the proposed high-level coordination control. Section 5.2.1 gives an overview of the control architecture when coordination between robots is handled by the motion planner. Section 5.2.2 explains the design process of the high level control algorithm.

5.2.1 Extension of the control architecture

In Chapter 4, it is assumed that the adjustable layer shown in Figure 4.2 becomes a part of the motion executor. In this chapter, the adjustable layer became a part of the motion planner. Figure 5.1 illustrates the difference of the control architecture realization.

In the context of transport in warehouses, the extended control architecture shown in Figure 4.2 also needs to be modified. In Chapter 4, the adjustable layer that is responsible for handling collision threats was made as part of the low-level control. Due to the shifting of responsibilities, the collision avoidance is handled by the high-level control. Thus, the extended control architecture shown in Figure 4.2 needs to be modified to consider this change. Figure 5.2 shows the comparison of these two extended control architectures. The dashed-box shows the main difference between the two.
Figure 5.1: The realization of the proposed control architecture. The figure on the left shows the low-level control realization, while the figure on the right shows the high-level control realization.

As seen in Figure 5.2(b), because of the shifting of responsibilities to the high-level control, the low-level part only contains the trajectory tracking controller. The focus in this section is to design an algorithm so that the Trajectory Generator and Trajectory Modifier generate collision-free trajectories for all robots. The necessary information for the Order Generator and Picking/Dropping Generator is generated in the similar way as explained in Section 4.2.2. As for the low-level control, the similar trajectory tracking controller given in (4.10) is used.

5.2.2 High-level control design

The proposed high-level coordination control works at the Trajectory Generator level. In Chapter 4, this layer generates reference trajectories that are not collision-free. Locally, the low-level controller solves the collision threats. In this chapter, the reference trajectories of each robot are negotiated so that no robot occupies the same paths at the same time instant.

As mentioned in Chapter 4, the reference trajectories are parameterized such that they have a constant nominal velocity along a path, see (4.8). In this way, the desired timed paths are fixed and known. The actual reference velocity of each robot has to be adapted to avoid collisions with other robots and to get back on track when the robot is behind schedule. Consider a situation shown in Figure 5.3, where there are two robots about to enter a junction.

To achieve the desired non-colliding behavior, the high-level control divides the complete reference trajectories into a collection of segments. The robots communicate intended arrival times at possible collision points, i.e., junctions, as well as their entry and exit direction for each junction. Using that information, the high-level coordination algorithm tries to simultaneously maximize throughput at each junction and minimize the waiting time for each robot. Figure 5.4(a) illustrates how a junction
Chapter 5. High- vs low-level control: performance comparison

Figure 5.2: The comparison of the expanded control architecture for transport in warehouses. Figure 5.2(a) is the low-level control realization as shown in Figure 4.2 of Chapter 4. Figure 5.2(b) shows the high-level control realization.

is divided into several segments where the two robots shown in Figure 5.3 have to solve the collision threat. The segments S1 and S2 (WP1, WP3, WP3) represent the junction J1 in the figure. A threat of collision is represented by intersecting path segments.

The high-level algorithm maximizes the throughput by making the robots move fast over the junction without slowing down. As a consequence, they reduce the possibility of blocking the passage for robots wanting to cross from another direction. This means that robots should only enter a junction when they are able to leave it. The waiting time is minimized by giving a robot that is more behind schedule a higher priority to cross the junction. Therefore, this delay has to be communicated as a priority value. This method prevents starvation, i.e. keeps computing the arrival
5.2 High-level coordination control

Figure 5.3: A threat of collision of robot $i$ and $j$ in junction $J1$ parameterized by $W1, W2, \text{and } W3$.

Figure 5.4: An illustration of the high-level coordination. (a) An example with 4 waypoints and the paths of two robots; (b) Desired/negotiated trajectory of a vehicle (solid) and occupation intervals of other higher priority vehicles (dashed).

times in an infinite loop. During normal operation, robots arriving from multiple directions are allowed alternately on the junction.

To make negotiations efficient, the robots only communicate the previous and next junction they pass(ed), and only do so when needed. When a robot crosses a junction, it will communicate its intended arrival time at the next junction, and if other robots think they have priority, they will react. This starts a message stream until they all agree. The robots adapt their velocities and some may have to update their departure time for the area after the passed junction (passing time of point $A$). These messages are broadcasted to all robots going towards the involved junction, so one message reaches all interested robots. A decentralized publish/subscribe mechanism is used to keep track of interested robots.

In addition, each robot locally saves the predicted time slot for the next junction, so they can anticipate the reaction of the other robots, preventing the selection of many wrong arrival times. This keeps the number of messages to a minimum. In order to cope with errors in communication, one could set a maximum time between broadcasts of arrival times, e.g. in our simulations it is assigned 0.5 [s]. Figure
5.4(b) gives an example of the resulting arrival times for the robots involved in the situation depicted in Figure 5.4(a).

The choice of priority as the time having waited for other robots is similar to the first come first served priority scheme, which minimizes the queues in front of the junctions as well as the waiting times of the individual robots. In this way every robot is guaranteed to be allowed on the junction but never instantaneously or simultaneously, unless there are too many robots in the system. This algorithm does not prevent livelock, i.e. a situation where robots keep making negotiations in an infinite loop. This can be caused by a situation where all free segments are occupied. In another situation, this infinite-negotiation-time can be caused when a robot should wait for robots ahead and indirectly one of those has to wait for the first one. This difficult problem is not handled in this algorithm. One way to solve this problem is to create new desired paths for some robots similar to the APF concept presented in Chapter 4.

Figure 5.5 shows the summary of the algorithm in a flow chart diagram. In addition, the algorithm can be implemented using the pseudocode given in Algorithm 5.1. The pseudocode has to be applied to all segments along the paths.

![Flow diagram of the high-level coordination algorithm loop in each robot.](image)
Algorithm 5.1 (Loop for) High-level coordination control

1: Send occupation information every 0.5 [s]
2: Receive occupation intervals, store them and sorted on arrival time
3: Adapt arrival time to just not collide with robots ahead
4: Flag to broadcast own interval, when a robot behind seems to overtake
5: Receive occupation intervals, and store them sorted on arrival time
6: Adapt arrival time not to collide with robots ahead
7: Flag to broadcast own interval, when a robot behind seems to overtake
8: Set \( t_{\text{waited}} = 0 \)
9: for All robots not in the front sorted by increasing arrival time do
10: Determine relevant leaving time as:
11: if leaving at the same waypoint then
12: free space leaving time
13: else
14: junction leaving time
15: end if
16: if arrival time overlaps then
17: if other robot’s priority > \( t_{\text{waited}} \) then
18: adapt arrival time to relevant leaving time
19: update \( t_{\text{waited}} \)
20: else
21: flag that we need to broadcast own interval
22: end if
23: end if
24: if the other robots arrive after own leaving then
25: break this loop
26: end if
27: end for
28: Adapt own speed to arrive at the correct time at the junction
29: Adapt own desired speed to make up for lost time
30: if Own interval changed or when flagged as needed then
31: broadcast own interval for the next junction
32: update own free space leaving time of the previous junction
33: broadcast own interval for the previous junction
34: end if

5.3 Simulation and experimental results

Similar scenarios, robot assumptions, and an automated warehouse layout as used in Section 4.5 are considered. In this chapter, the focus is on validating the high-level coordination algorithm in normal operation status. For comparison, the maximum capacity of the conveyor in the automated warehouse is used, i.e., 1000 totes per hour. Performance analysis is focused on the completion time. The following abbreviations are used to identify the scenarios: SP-HLC: single-path, high-level control; MP-HLC: multiple-paths, high-level control; CS: conveyor system.
5.3.1 Simulation results

The high-level control works at the level of generating the trajectory. Although it produces collision-free reference trajectories, it needs a low-level controller for accurate tracking of the references. A similar trajectory tracking controller as used in Chapter 4 is implemented. For the simulation case study, similar tracking control gains as the ones given in (4.26) are used. Since the collision avoidance is handled by the high-level control, the slowing down and APF of the low-level control are not active. Thus, control parameters given in (4.24) and (4.25) are not used. For the high-level control, the control parameters that need to be set are the nominal speed and maximum allowable speed, set to 1 \([\text{m/s}]\) and 1.1 \([\text{m/s}]\) respectively.

The nominal speed value is similar to the assumption described in Section 4.4.

The summary of completion time from the simulation of different scenarios is depicted in Figure 5.6.

Figure 5.6 shows how the high-level coordination control handles the flexibility and scalability of the system in a similar way as the low-level coordination control. The high-level controller can accommodate the addition/removal of robots from the transport system. Thus, the throughput can be tuned by doing so.

From the perspective of completion time, in this particular case study, the high-level controller performs better when no shortcuts are used. This can be observed in the dashed line in Figure 5.6 where the minimum value is below the curve for CS (solid line) and MP-HLC (dotted line). This is interesting since no shortcut means that the transport system behaves like the conveyor system, only with a higher capacity. This supports the idea that using a similar transport layout as the conveyor systems, the capacity still can be increased when a group of robots is used. This cannot be achieved using a conveyor. Once the conveyor system is built, its maximum capacity is fixed.
Furthermore, the results in Figure 5.6 indicate that when the number of robots is small, e.g. below 12, the shortcuts help increase the performance. Up to 12 robots, the curve for MP-HLC is always below the curve of SP-HLC. Above 12 robots, the simulation using MP-HLC results in a livelock because the condition that is mentioned in Section 5.2.1 occurs, i.e. at least one pair of robots gets stuck in an infinite loop of negotiation.

5.3.2 Experimental results

For the experimental case study, the control gains given in (4.29) are used for the low-level (tracking) controller. For the high-level controller, the nominal speed is set to $0.09 \text{ [m/s]}$ and the maximum speed is set to $0.14 \text{ [m/s]}$. The purpose of this section is to show the main difference between how the high-level controller solves the coordination problem at the junction compared to the low-level controller.

Consider a similar situation as in Figure 4.10 now shown in Figure 5.7. In this situation, robots 1, 2, 7, and 8 are about to enter the same junction. The high-level controller needs to determine which robots need to slow down to avoid collision.

![Figure 5.7: Threat of collisions at a junction.](image)

Figures 5.8 shows the resulting reference forward velocities for robots 1, 2, 7, and 8 computed using the high-level control algorithm.

The first difference between the high- and low-level controller is the way the priority of a robot is determined. In the low-level controller a fixed priority rule is assigned. In the high-level controller the priority is determined dynamically based on the real-time situation in the system. In the example shown in Figure 5.8, the priorities are as follows: 1,7,2,8. These can be observed in Figure 5.8, robot 1 (the solid-black line), does not have to slow down at all. On the other hand, robot 8 (dashed-grey line) at some point in time has to slow down to $0.05 \text{ [m/s]}$, i.e. the slowest among others, which means it has the lowest priority. It is not always the robot that comes from
the left or right side has a higher priority. In this way, in each junction the waiting
time of individual robots can be minimized.

The second difference is the ability to increase the speed (up to the maximum) to
catch up idle time because of waiting. This can be seen in the reference velocity
profiles of robots 2, 7, and 8. The dashed-black, solid- and dashed-grey curves show
maximum values above 0.09 [m/s] (the nominal velocity). This can be achieved
since the high-level controller has access to more information of the individual robots
and negotiates among them. Thus, increasing the speed will not cause other con-
gestion or collisions to occur as long as it does not exceed $v_{\text{max}}$. The increment is
different per robot depending on the idle time. A robot with higher idle time gets
a higher speed increment. In this particular example, robot 8 has to slow down to
the lowest velocity, i.e. the highest idle time, and thus gets a "new" reference speed
which almost reaches the maximum velocity, i.e. $v_{r8} = 0.135$ [m/s].

Notes on faulty case study

The high-level control requires accurate tracking of the reference. When there is a
failure in the system, i.e. there is a broken robot that blocks other robot paths, the
high-level control will fail, since it cannot create a new path. An option to solve
this issue is by activating the APF in the low-level control. By activating the APF, the
low-level control can create a new path so that the high-level control algorithm is
still working even when there is a failure in the transportation system.
5.4 Performance comparison and cost analysis

In this section, a performance comparison between the high- and low-level coordination control is discussed. In addition, a feasibility study for using a group of mobile robots for transport in warehouses is presented.

5.4.1 Performance comparison: High- vs low-level control

Similar simulation scenarios (normal operation), control parameters, robot assumptions, and automated warehouse used in Sections 4.5 and 5.3 are investigated. The following abbreviations are used to identify the scenarios: SP-HLC: single-path, high-level control; MP-HLC: multiple-paths, high-level control; SP-LLC: single-path, low-level control; MP-LLC-LN: multiple-paths, low-level control, low number priority; MP-LLC-LH: multiple-paths, low-level control, left-hand priority; MP-LLC-RH: multiple-paths, low-level control, right-hand priority. The conveyor system is identified as CS. The performance of the different scenarios is summarized in Figure 5.9.

![Performance comparison of different $t_{\text{complete}}$ in different scenarios using both high- and low-level coordination control.](image)

Figure 5.9: Performance comparison of different $t_{\text{complete}}$ in different scenarios using both high- and low-level coordination control.

The comparison shows that in the particular case study, the scenario SP-HLC with 18 robots has the best performance, i.e., the completion time is the shortest. The shortest completion time means that it can deliver a higher throughput than the others. The comparison also suggests that for the automated warehouse being studied, the appropriate number of robots to achieve similar performance as the conveyor system is between 8–18. Lowering these values results in a lower throughput than the conveyor system, while higher values will result in livelock or deadlock. It should be remembered that although adding robots can increase throughput, if the space is
kept constant, the robots have less space for movement. Thus, the possibility of a livelock or deadlock increases.

Furthermore, the comparison also shows that using a minimal amount of information, i.e., using the low-level control strategies, a good performance can be achieved.

In this simulation study, the high-level controller yields the optimal throughput. However, the algorithm requires the robots to share their states, something which can be costly given the communication bandwidth and limited budget. On the other hand, the low-level controller does not require any information sharing and still can achieve a similar performance as the conveyor system. This suggests that in another automated warehouse transport layout, the low-level controller may outperform the high-level controller.

5.4.2 Cost analysis

In this section, a comparison between the cost of using a group of mobile robots and conveyor systems to realize the transport in warehouses is discussed. The conveyor systems consist mainly of belt conveyors and motors. Thus, the cost of the conveyor system is computed based on the cost of these two components. On the other hand, the cost of using a group of robots is mostly determined by how many robots are used in the system.

The cost is calculated as a normalized value with respect to the cost of the conveyor system, i.e., the cost of the conveyor systems equals 1, as depicted in the left hand side of Figure 5.10. The cost of the conveyor system is represented as a horizontal-grey-line equal to 1, while the cost of the mobile robots is represented by the black line. The number at the curves indicates the available robots and its related normalized cost.

![Figure 5.10: Left: normalized cost of the transport system; Right: zoom-in the comparison of different complete in different scenarios.](image)
From the left hand side of Figure 5.10 it can be observed that the cost for 12 robots, indicated by the number 12, is crossing the horizontal line that indicates the cost of conveyor system, i.e. equal 1. This means that the cost of deploying 12 robots is the same as the cost of the conveyor system. Zooming into the right-hand side of Figure 5.10, the curves for SP-HLC, MP-HLC, SP-LLC or MP-LLC-RH are all below the horizontal line for the conveyor system. These indicate that the performance when using 12 robots is better than with the conveyor system. These findings suggest that, by investing the same amount of cost as the conveyor system, using a group of robots can increase the performance of the transportation system of an automated warehouse.

Furthermore, from the right-hand side of Figure 5.10, in some scenarios, e.g. SP-HLC, SP-LLC or MP-LLC-LH, by using 10 robots a similar \( t_{\text{complete}} \) as the conveyor system can be achieved. The normalized cost of 10 robots shown in Figure 5.10 is 0.8, which is lower than the cost of the conveyor system. This suggests that, for achieving a similar performance as with the conveyor system, using a group of robots can reduce the cost. In addition, if a better performance (shorter \( t_{\text{complete}} \)) is still required, e.g. using 14, 16, or 18 robots, in scenario SP-HLC, SP-LLC, or MP-LLC-LH, a higher cost needs to be invested since the normalized cost for these choices of robots is greater than 1.

Although for a real implementation there are other costs to be considered, this simple calculation indicates that the cost of realizing the transport in a warehouse using a group of mobile robots has an acceptable cost-versus-performance margin compared to a conveyor system.

### 5.4.3 Influence of control parameters and warehouse layout on the performance

All simulations and experimental results presented in Section 4.5, 5.3, and 5.4 are tested for a specific automated warehouse layout. When another automated warehouse layout is considered the performance results can be different, i.e. the shortest \( t_{\text{complete}} \) will be obtained from a different number of robots and control algorithm. Thus, the proposed low- and high-level control does not guarantee the optimal solution for transportation of an arbitrary automated warehouse case study. In this thesis, the best solution is obtained by conducting simulations/experiments, which can give different results when the system parameters are changed. In addition, the control algorithms are formulated to solve a combination of trajectory tracking and collision avoidance. They are not aimed to find the shortest \( t_{\text{complete}} \). Thus, the control algorithm finds the optimal control signals for tracking and avoiding collisions, but not for minimizing \( t_{\text{complete}} \).

Furthermore, the simulations and experimental results are computed using a selection of control parameters. As a consequence, another choice of control parameters can result in a different \( t_{\text{complete}} \).

Take as an example the high-level control case. Suppose a set of parameters for the trajectory tracking controller and the nominal reference velocity is fixed. The high-level control has the maximum speed as the variable which can be freely chosen as
long as it is smaller or equal to the robot's speed limit. When the maximum is set higher, it can be expected that $t_{\text{complete}}$ becomes shorter since the robot can move faster to compensate the idle time. However, if the maximum is too close to the hardware limit, the robot may no longer be able to move at the desired speed.

The next example is for the low-level control case. The value of $(\gamma_{\text{min}}, \gamma_{\text{max}})$ of the penalty function determines when the robot has to start slowing down. If both values are set to small values, it means that the robot has to start slowing down and finally stop at a very short distance with respect to the other robots. This can reduce $t_{\text{complete}}$, since the robots optimize the available space for movement. However, when the distance is too short, it can increase the possibility of having collisions. If $\gamma_{\text{min}}$ is too small, a robot can fail to stop on time and hit the other robots. This can happen for example when there is a delay in computing the control signals or noise in the position measurements. When $(\gamma_{\text{min}}, \gamma_{\text{max}})$ are set to larger values, the robots will start to slow down far away from the other robots. From a safety perspective, this is beneficial. However, this setting also needs more space for movement and can increase $t_{\text{complete}}$.

A similar effect also applies for the choice of $a$ and $b$ of the APF function. The larger $a$ and $b$ means that the APF actives in a larger region which is more safe but less efficient with respect to $t_{\text{complete}}$.

### 5.5 Alternative performance measures

Although it is not discussed in great detail in this thesis, in Adinandra et al. (2010) the performance of the low- and high-level control is evaluated using different performance indicators. Still using transport in a warehouse as a case study, but in a different task, the two control algorithms are evaluated in terms of average travel time, normalized trajectory tracking errors and normalized total formation errors.

The task of the group of robots is to deliver goods along the path depicted in Figure 5.11 while moving in a convoy of seven robots. At one segment of this path, the front part of the convoy intersects with the part at the back; consequently, coordination between robots is needed to avoid collisions and keep the correct robot sequence. Keeping the sequence can be very important if the robots have to arrive in a special sequence based on the customer order requirement.

The average travel time, $t_{\text{trvl}}$, is similar to the completion time defined in Section 2.6.1. The tracking and formation errors are defined as follows:

- The normalized tracking errors of all robots

\[
e_{xy,\text{tot}} = \sum_{i=1}^{n} \frac{1}{T} \sum_{k=1}^{l} \left( (x_i(t_k) - x_i(t))^2 + (y_i(t_k) - y_i(t))^2 \right),
\]  

(5.1)

where $t_k$ is the time instant when data is collected and $l$ is the number of the data in experiments.
5.5 Alternative performance measures

- The normalized formation errors. The formation in Adinandra et al. (2010) is formulated as a time-varying Euclidean distance between the neighboring robots. For \( m \)-robots, the pattern can be described for \( i \in \{1, 2, \ldots, m\} \) and \( j = i + 1 \), as follows:

\[
\Delta_{ij}(t_k) = \sqrt{(x_{r,i}(t_k) - x_{r,j}(t_k))^2 + (y_{r,i}(t_k) - y_{r,j}(t_k))^2}. \tag{5.2}
\]

The individual formation error is defined by:

\[
\delta_{ij}(t_k) = \Delta_{ij}(t_k) - \Delta_{ij}(t_k), \tag{5.3}
\]

where \( \Delta_{ij} \) is the actual Euclidean distance between robot \( i \) and \( j \). The normalized total formation errors are given by:

\[
\delta_{ij,\text{tot}} = \frac{1}{m-1} \sum_{i=1}^{m} \sum_{j=i+1}^{m} \frac{1}{T} \sum_{k=1}^{T} (\delta_{ij}(t_k))^2. \tag{5.4}
\]

To increase the robustness in formation keeping, in Adinandra et al. (2010) coupling terms are introduced. Using a modified version of the controller proposed in Kostić
et al. (2010a), the low-level control formulated in (4.9) is modified into:

\[ v_i(t) = v_{ri}(t) \cos \theta_i(t) + k_{xi} x_i(t) \sum_{j \neq i} \frac{s_\alpha_{ij} k_{xij}}{\sqrt{1 + x_{xj}(t)^2 + x_{xj}(t)^2}} \left( x_{ri}(t) - x_{rj}(t) \right) \]

\[ w_i(t) = w_{ri}(t) + k_{yi} v_{ri}(t) \sin \theta_i(t) + \sum_{j \neq i} s_\alpha_{ij} k_{yij} \left( y_{ri}(t) - y_{rj}(t) \right) \frac{\sin \theta_{ri}(t) \sin \theta_{rj}(t)}{\theta_{ri}(t)} \]

\[ + k_{\theta i} \theta_{ri}(t) + \sum_{j \neq i} s_{\theta ij} k_{\theta ij} \frac{\theta_{ri}(t) - \theta_{rj}(t)}{\sqrt{1 + \theta_{ri}(t)^2 + \theta_{rj}(t)^2}} \]

\[(5.5a)\]

where \( k_{ij}, k_{ij}', \) and \( k_{ij}'' \) are the coupling gains and \( s_\alpha_{ij}, \alpha \in \{x, y, \theta\} \) is defined as follows:

\[ s_\alpha_{ij} = \begin{cases} 
\text{sgn}(\alpha_{ri}), & \text{if } |\alpha_{ri}| \geq |\alpha_{rj}|, \\
\text{sgn}(\alpha_{rj}), & \text{if } |\alpha_{ri}| < |\alpha_{rj}|.
\end{cases} \]

\[(5.5b)\]

Although the controller given in (5.5) is a modification of the one proposed in Kostić et al. (2010a), the stability proof follows directly from the proof in Kostić et al. (2010a).

As for the high level control, there is no modification made. A similar high-level control described in Section 5.2 is implemented.

Control parameters

The following control parameters are chosen:

\[ k_{xi} = 0.4, \ k_{yi} = 100, \ k_{\theta i} = 0.5, \]

\[ k_{xij} = 0.06, \ k_{yij} = 10, \ k_{\theta ij} = 0.000001. \]

\[(5.7)\]

\[(5.8)\]

It is to be noted that the choice in (5.7) is exactly the same as the one given in (4.29). Moreover, the choice of parameters for the APF (responsible for the collision avoidance) used in this case study is the same as the one given in (4.28).

Performance comparison analysis

The coordination control algorithms are validated in real-time experiments. For the low-level control, different ways of sharing information between the robots are considered. The difference determines whether \( k_{ij} = k_{ij}' = k_{ij}'' = 0 \) or equal the ones given in (5.8). The following indexes are used: HL: high-level control; LL-1 to LL-4: low-level control. The robots are coupled as illustrated in Figure 5.12.

The scenarios are compared in terms of total performance formulated as the summation of the three performance indicators defined previously with equal weight.
5.5 Alternative performance measures

\[
\sum_{\text{perf}} = t_{trvl} + e_{xy,\text{tot}} + \delta_{ij,\text{tot}}.
\] (5.9)

Figure 5.13 shows the computation of \(\sum_{\text{perf}}\) from the experiments with interconnection as depicted in Figure 5.12.

Figure 5.12: Options for robot interconnections. The arrows indicate from which robots the coupling information is obtained.

![Diagram of robot interconnections](image)

Figure 5.13: The mean values of \(t_{trvl}\), \(e_{xy,\text{tot}}\), \(\delta_{ij,\text{tot}}\), and \(\sum_{\text{perf}}\) from different scenarios and strategies:

- **Strategies:**
  - 1 = HL
  - 2 = LL - 1
  - 3 = LL - 2
  - 4 = LL - 3
  - 5 = LL - 4

The experimental results show that based on the values of \(\sum_{\text{perf}}\) computed from different strategies, the high-level controller is the most promising solution, followed by the low-level controller when all robots are coupled (all robots share information). The high-level control yields a better performance than the low-level thanks to the negotiation process involved in the algorithm. Using negotiation, both trajectory...
tracking, collision avoidance, and formation keeping can be completed simultaneously in an efficient way. However, since high-level control requires the robots to accurately move on their desired paths, it becomes less robust against perturbations. If a robot is not moving on its desired path, the high-level control can fail in regulating the robots, especially for avoiding collisions and keeping the formation. On the other hand, the low-level controller is more robust to perturbations since it is based on a reactive-to-dynamic control approach. When there is an unexpected situation, the low-level controller will directly react to it.

Another important result shown in Figure 5.13 is the influence of information. The high-level approach, indexed 1 in Figure 5.13, results in the most promising solution since the algorithm takes into account the states of all robots. For the second best candidate, indexed LL-2, the low-level control also requires coupling from all robots. This suggests that the more information to be shared, the better performance is to be expected. This seems logical since with more information the control algorithm can take into account more information of the robots in the group. However, if the number of robots increases rapidly, sharing information requires a larger communication bandwidth.

In this section the performance analysis conducted using alternative indicators suggests that the high-level control is better than the low-level control, but with the cost of less robustness against perturbations. This finding is similar to the performance analysis given in Section 5.4.1. While the analysis in Section 5.4.1 focuses on the global picture of the transport system, the analysis in this section focuses on the smaller-real-time characteristic of the control algorithm, e.g. the trajectory tracking errors, which in the global picture becomes less important.

**Remark 5.1.** Similar to the analysis given in Section 5.4.3, the comparison shown in Figure 5.13 is computed from a set of control parameters. Different control parameters can deliver different results. As an example, if higher coupling gains are used, the low-level control can outperform the high-level control in terms of formation keeping performance. Another important note is on the robustness as defined in Section 2.6.2. The fact that the high-level control works based on prediction, makes it less robust against perturbation compared to the low-level control. The analysis presented in Section 5.4 and 5.5, which is based on two different cases, confirms this finding.

### 5.6 Conclusions

In this chapter a methodology to coordinate a group of mobile robots using a hierarchical approach, focusing on high-level control design is presented. The proposed high-level control algorithm is used to regulate a group of mobile robots that realize the transport system of an automated warehouse. The simulation and experimental results show that the proposed control algorithm can accommodate addition/removal of robots from the system to make the transport system more scalable.

The performance comparison analysis shows that the high-level algorithm outperforms the low-level algorithm at the cost of requiring more information. However,
the comparison also suggests that, depending on the available number of robots in the transport system, the low-level control, which requires no information sharing, can have a similar performance to the high-level control. Both high-level and low-level control can have a better performance than the conveyor system.

Furthermore, in this particular case study, the simulation results and cost analysis suggest that the proposed transportation system using a group of mobile robots has an acceptable cost-versus-performance ratio considering the advantages that can be gained.

In the next chapter, a control algorithm that is able to achieve simultaneous trajectory tracking of individual robot and formation keeping with other robots is presented.
Chapter 5. High- vs low-level control: performance comparison
Simultaneous Trajectory Tracking and Formation Keeping for a Group of Unicycles

This chapter discusses the design of a control algorithm for a group of unicycles that is able to simultaneously track individual references and keep a certain formation with other unicycles. The algorithm extends the concept of individual trajectory tracking designed using dynamic feedback linearization. The extension is done by introducing coupling gains between the robots so that formation keeping can be achieved. The algorithm is validated in real-time experiments. Using a root-mean-square error-like indicator, the influence of communication topologies on the performance of the control algorithm is investigated.

6.1 Introduction

The work in this chapter is initiated by the importance of having a performance measure to evaluate formation control algorithms for a group of unicycle mobile robots. To name a few, the work of Tanner (Tanner et al. (2002a), Tanner et al. (2002b), Tanner et al. (2004), Tanner (2004)), illustrates how performance of formation control is formulated by means of the input signals of the leader and information topology between the robots. These works combine the feedback linearization technique for the control design and input-to-state stability (ISS) concept to formulate the performance measure. It is found that formulating an ISS property of a system with nonholonomic constraints is difficult. Another important finding suggests that the performance measure, although it depends on the control gain values, gives an analytical prediction on how good the group of robots maintains the formation.
In addition, the work of Sadowska et al. (2011), provides a performance index that shows the discrepancy between the desired and the actual formation. This index measures the difference between desired and actual distances between all pairs of robots in the formation. However, the index measures the geometric of the formation but with the possibility to have a rotation or a reflection (a mirror image) of the formation shape.

In this chapter a control algorithm, and its performance measure, for a group of unicycles, which is able to simultaneously track individual references and keep a certain formation with other robots is formulated. The algorithm extends the trajectory tracking controller for a single robot designed using dynamic feedback linearization as proposed by Oriolo et al. (2002). Using a theorem on interconnected systems (Section 2.2), the stability proof is analyzed.

The contributions of this chapter are the following. Firstly, a controller that is able to simultaneously track individual references and keep a certain spatial formation is proposed. Both tasks can be achieved efficiently by sharing the individual tracking errors within the group. Secondly, by means of experiments, the influence of communication topologies between the robots on the performance is analyzed using a root-mean-square error-like indicator.

The rest of this chapter is organized as follows. In Section 6.2, the design of trajectory tracking control of a single robot using dynamic feedback linearization is revisited. Section 6.3 provides the details of the extension to simultaneous tracking and formation keeping of \( m \) mobile robots. In Section 6.4, the performance measure for evaluation is explained. Section 6.5 shows experimental results and analysis of the performance. Finally, concluding remarks are presented in Section 6.6.

### 6.2 Dynamic feedback linearization of a unicycle mobile robot

Consider a group of \( m \) unicycle mobile robots. As given in Section 2.3, the nonholonomic kinematic model of robot \( i \) is:

\[
\begin{bmatrix}
\dot{x}_i \\
\dot{y}_i \\
\dot{\theta}_i
\end{bmatrix} =
\begin{bmatrix}
\cos \theta_i & 0 & u_{i1} \\
\sin \theta_i & 0 & u_{i2} \\
0 & 1 & \end{bmatrix},
\]

(6.1)

where \( u_{i1} = v_i \) is the forward/translational velocity and \( u_{i2} = \omega_i \) is the rotational/steering velocity respectively.

Suppose the position of the unicycle is chosen as the output: \( \mathbf{h}_i = [x_i, y_i]^T \). The first order derivative of \( \mathbf{h}_i \) is

\[
\begin{bmatrix}
\dot{x}_i \\
\dot{y}_i
\end{bmatrix} =
\begin{bmatrix}
-\sin \theta_i & 0 & v_i \\
\cos \theta_i & 0 & \omega_i
\end{bmatrix}.
\]

(6.2)

The first order derivative of \( \mathbf{h}_i \) given in (6.2) only depends on \( u_{i1} = v_i \). Thus, it is not statically input-output linearizable. Introducing a new state \( \varsigma_i = u_{i1} \) and new inputs
6.2 Dynamic feedback linearization of a unicycle mobile robot

\( \dot{u}_{i1} = \dot{\varsigma}_i, \quad \dot{u}_{i2} = u_{i2} \), the kinematic model in (6.1) can be transformed into:

\[
\begin{bmatrix}
\dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \\ \dot{\varsigma}_i
\end{bmatrix} =
\begin{bmatrix}
\varsigma_i \cos \theta_i \\ \varsigma_i \sin \theta_i \\ 0 \\ 0
\end{bmatrix} +
\begin{bmatrix}
0 \\ 0 \\ 0 \\ 1
\end{bmatrix} \begin{bmatrix}
0 \\ 0 \\ 1 \\tilde{u}_{i1} + \tilde{u}_{i2}.
\end{bmatrix}
\]

(6.3)

Choosing the same output \( h_i = [x_i, y_i]^T \), the second order derivative of \( h_i \) is

\[
\begin{bmatrix}
\ddot{x}_i \\ \ddot{y}_i \\ \ddot{\theta}_i \\ \ddot{\varsigma}_i
\end{bmatrix} =
\begin{bmatrix}
\cos \theta_i & -\varsigma_i \sin \theta_i \\ \sin \theta_i & \varsigma_i \cos \theta_i \\ 0 & 0 \\ 0 & 0
\end{bmatrix} \begin{bmatrix}
\dot{u}_{i1} \\ \dot{u}_{i2}
\end{bmatrix}.
\]

(6.4)

As long as \( \varsigma_i \neq 0 \) or the forward velocity is bounded away from zero, the extended-dynamic model of the unicycle given in (6.3) is input-output feedback linearizable into

\[
h_i = \begin{bmatrix}
\dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \\ \dot{\varsigma}_i
\end{bmatrix} = \begin{bmatrix}
w_{i1} \\ w_{i2}
\end{bmatrix} = w_i,
\]

(6.5)

using the control signals given by

\[
\begin{bmatrix}
\dot{u}_{i1} \\ \dot{u}_{i2}
\end{bmatrix} = \begin{bmatrix}
\cos \theta_i & \sin \theta_i \\ -\varsigma_i \sin \theta_i & \varsigma_i \cos \theta_i \\ 0 & 0 \\ 0 & 0
\end{bmatrix} \begin{bmatrix}
w_{i1} \\ w_{i2}
\end{bmatrix}.
\]

(6.6)

The resulting dynamic compensator can be formulated as follows:

\[
\dot{\varsigma}_i = w_{i1} \cos \theta_i + w_{i2} \sin \theta_i \
\]

(6.7a)

\[
v_i = \varsigma_i, 
\]

(6.7b)

\[
\omega_i = \frac{-w_{i1} \sin \theta_i + w_{i2} \cos \theta_i}{v_i}, 
\]

(6.7c)

Remark 6.1. The dynamic compensator in (6.7) has a potential singularity at \( \varsigma_i = v_i = u_{i1} = 0 \), i.e. when the unicycle is not rolling. The occurrence of a singularity in the dynamic extension process is structural for nonholonomic systems (De Luca and Di Benedetto, 1993).

Trajectory tracking controller of a single robot

Suppose the reference trajectory for unicycle \( i \) is given as \( q_{ri} = [x_{ri}, y_{ri}]^T \). The reference must satisfy the nonholonomic constraint, i.e. \(-x_{ri} \sin \theta_{ri} + y_{ri} \cos \theta_{ri} = 0 \) and must be positive, i.e. \( v_{ri} = \sqrt{x_{ri}^2 + y_{ri}^2} > 0 \), along the trajectory. It is to be noted that because of the output-feedback linearization taken in (6.4), the orientation of the robot \( \theta_i \) is automatically rendered by the controller given in (6.7).

Considering the equivalent linear and decoupled systems given in (6.5), it is straightforward to design an exponentially stabilizing feedback for the desired trajectory by...
setting $w_i$ as
\[
\begin{bmatrix}
  w_{i1} \\
  w_{i2}
\end{bmatrix} = \begin{bmatrix}
  x_i + k_{x_i}^p (x_i - x_i^o) + k_{x_i}^v (x_i - x_i^o) \\
  y_i + k_{y_i}^p (y_i - y_i^o) + k_{y_i}^v (y_i - y_i^o)
\end{bmatrix},
\]
(6.8)
with the proportional-derivative (PD) gains $k_{x_i}^p, k_{y_i}^p, k_{x_i}^v, k_{y_i}^v > 0$. Substituting (6.8) into (6.5):
\[
\begin{bmatrix}
  \dot{x}_i \\
  \dot{y}_i
\end{bmatrix} = \begin{bmatrix}
  x_i + k_{x_i}^p (x_i - x_i^o) + k_{x_i}^v (x_i - x_i^o) \\
  y_i + k_{y_i}^p (y_i - y_i^o) + k_{y_i}^v (y_i - y_i^o)
\end{bmatrix}.
\]
(6.9)
Define
\[
e_i = \begin{bmatrix}
  e_{x_i} \\
  e_{y_i}
\end{bmatrix} \equiv \begin{bmatrix}
  x_i - x_i^o \\
  y_i - y_i^o
\end{bmatrix},
\]
(6.10)
equation (6.9) can be written into
\[
\begin{bmatrix}
  \dot{e}_{x_i} + k_{x_i}^p e_{x_i} + k_{x_i}^v e_{y_i} \\
  \dot{e}_{y_i} + k_{y_i}^p e_{y_i} + k_{y_i}^v e_{x_i}
\end{bmatrix} = \begin{bmatrix}
  0 \\
  0
\end{bmatrix},
\]
(6.11)
which is the closed loop tracking error dynamics of unicycle $i$ on the equivalent linear and decoupled systems given in (6.5).

The PD controller given in (6.8) has to be fed to the dynamic compensator in (6.7) in order to obtain the actual control inputs. The dynamic compensator is valid provided that it never meets the singularity $v_i = \zeta_i = 0$. Since the reference trajectory is assumed to be positive, this situation may only happen during the initial transient of an asymptotic tracking problem. Theorem 6.1 gives sufficient conditions so that the singularity never occurs.

**Theorem 6.1.** (Oriolo et al., 2002) Let $\lambda_{11}, \lambda_{12}, \lambda_{21}, \lambda_{22}$ be, respectively, the eigenvalues of the closed loop error dynamics given in (6.11). Assume that, for $i = 1, 2$, $\lambda_{11} < \lambda_{12}$ (negative real eigenvalues) and $|\lambda_{22}|$ sufficiently small. If
\[
\min_{t \geq 0} \left\| \begin{bmatrix}
  \dot{x}_{i}(t) \\
  \dot{y}_{i}(t)
\end{bmatrix} \right\| > \left\| \begin{bmatrix}
  \dot{e}_{x_i}^0 \\
  \dot{e}_{y_i}^0
\end{bmatrix} \right\|
\]
(6.12)
with $\dot{e}_{x_i}^0 = \dot{e}_{x_i}(0) \neq 0$ and $\dot{e}_{y_i}^0 = \dot{e}_{y_i}(0) \neq 0$, then the singularity $v_i = \zeta_i = 0$ is never met.

**Proof.** See Oriolo et al. (2002). \qed

One important practical aspect of Theorem 6.1 is the following. To obtain exact trajectory tracking for a matched initial posture of the robot, i.e. $x_i^0 = x_i^o$, $y_i^0 = y_i^o$, and $\theta_i^0 = \theta_i^o$ (or $\theta_i^0 = \theta_i^o + \pi$), the dynamic compensator should be correctly initialized at $\dot{e}_{x_i}^0 = \dot{e}_{y_i}^0$ (or $\dot{e}_{x_i}^0 = -\dot{e}_{y_i}^0$).

### 6.3 Simultaneous tracking and formation keeping

Let $m$ be the number of robots in the formation. A spatial formation pattern of the group of robots is defined as
6.3 Simultaneous tracking and formation keeping

\[ \mathbf{F}(t) = [\mathbf{h}_1(t), \mathbf{h}_2(t), \ldots, \mathbf{h}_m(t)] \]

(6.13)

where \( \mathbf{h}_i(t) = [x_i(t) \ y_i(t)]^T \) are the desired Cartesian coordinates of the center of the \( i \)-th robot with respect to time.

A class of formation control problems where the group of robots is required to follow \( \mathbf{F}(t) \) while individual robots have to follow their own reference trajectories is considered. In this type of formation, the problem typically occurs when perturbations are introduced to the group. During the transition to recover from perturbations, there will be conflicting objectives. On the one hand, the group will try to recover the desired spatial pattern \( \mathbf{F}(t) \). On the other hand, by trying to recover the formation, individual robots maybe required to leave their individual reference trajectory \( \mathbf{h}_i(t) \), which may be undesirable. Define \( \xi_i \) as

\[ \xi_i \triangleq \begin{bmatrix} x_i - x \n y_i - y \n x_i - x \n y_i - y \end{bmatrix} = \begin{bmatrix} e_i \n e_i \n e_i \n e_i \end{bmatrix} \]

(6.14)

the simultaneous trajectory tracking and formation keeping is formulated as the problem to render individual tracking errors \( \xi_i \to 0 \), \( \forall i \in \{1, 2, \ldots, m\} \) as \( t \to \infty \).

To achieve the simultaneous trajectory tracking and formation keeping, the PD controller in (6.8) is modified by introducing coupling terms so that \( \mathbf{w}_i \) becomes:

\[ \mathbf{w}_i = \begin{bmatrix} x_i + k_{ixi} \dot{e}_i + k_{ix} e_i + \sum_{j \neq i} k_{ix} \dot{e}_j + k_{i} \sum_{j \neq i} e_j \n y_i + k_{iyi} \dot{e}_i + k_{iy} e_i + \sum_{j \neq i} k_{iy} \dot{e}_j + k_{i} \sum_{j \neq i} e_j \end{bmatrix} \]

(6.15)

where \( k_{i}^{\rho} \), \( \rho \in \{px, dx, py, dy\} \) are the coupling gains.

The PD controller in (6.15) basically consists of two parts. The part related to \( k_i^p \) is responsible for trajectory tracking, while the part related to \( k_i^r \) is responsible for adding robustness to formation keeping. Depending on the choices of \( k_i^p, k_i^r \), the group of robots can achieve simultaneous trajectory tracking and formation keeping, pure trajectory tracking, or pure formation keeping.

Substituting (6.15) into (6.5), we obtain

\[ \begin{bmatrix} \dot{x}_i \\
\dot{y}_i \end{bmatrix} = \begin{bmatrix} \ddot{x}_i + k_{ixi} \dot{e}_i + k_{ix} e_i + \sum_{j \neq i} k_{ix} \dot{e}_j + k_{i} \sum_{j \neq i} e_j \\
\ddot{y}_i + k_{iyi} \dot{e}_i + k_{iy} e_i + \sum_{j \neq i} k_{iy} \dot{e}_j + k_{i} \sum_{j \neq i} e_j \end{bmatrix} \]

(6.16)

Thus, the closed loop error dynamics of the simultaneous trajectory tracking and formation keeping of \( m \) unicycles can be formulated as follows:
Consider the second order dynamics of \( m \) unicycle mobile robots described in (6.14) expression (6.16) can be written as

\[
\begin{bmatrix}
\ddot{e}_{xi} + k_{dx} \dot{e}_{xi} + k_{px} e_{xi} + \sum_{j \neq i} k_{ij}^x (e_{xi} - e_{xj}) \\
\ddot{e}_{yi} + k_{dy} \dot{e}_{yi} + k_{py} e_{yi} + \sum_{j \neq i} k_{ij}^y (e_{yi} - e_{yj})
\end{bmatrix}
\]

\[ i \in \{1, 2, \ldots, m\}. \tag{6.17} \]

**Remark 6.2.** In the frame of the hierarchical control approach, the simultaneous trajectory tracking and formation keeping control problem can be presented as a hierarchical control architecture shown in Figure 6.1.

![Hierarchical Control Architecture](image)

Figure 6.1: The formation control algorithm seen as a hierarchical control structure. \( N_i(t) \) represents the number of robots in the neighborhood of, and connected to, robot \( i \).

The hierarchical control representation depicted in Figure 6.1 shows that the "high-level-control/motion planner" part is the trajectory generator. In this case study, the trajectory generator has to accurately define reference trajectories that serve both individual and collective group goals as well handling the constraint on the reference forward velocity. On the other hand, the "low-level-control/motion executor" part contains the PD-controller given in (6.15) and the dynamic compensator given in (6.7). The communication network is used by the low-level control to share the necessary information. Figure 6.1 also shows that the high-level part does not use the communication network and does not require information feedback from the low-level part.

For the closed-loop error dynamics (6.17), the following theorem holds.

**Theorem 6.2.** Consider the second order dynamics of \( m \) unicycle mobile robots described in (6.17). If \( k_{dx}^x, k_{dy}^y, k_{px}^x, k_{py}^y > 0 \) and \( k_{ij}^x, k_{ij}^y, k_{ij}^{xy}, k_{ij}^{yx} \geq 0 \) then the controller formulated in (6.15) (combined with the dynamic compensator in (6.7)) renders the origin of (6.17), \( \forall i \in \{1, 2, \ldots, m\} \) asymptotically stable.

**Proof.** The stability proof follows Theorem 2.1 regarding the stability of interconnected systems. Using \( \xi_i \) in (6.14) expression (6.16) can be written as
6.3 Simultaneous tracking and formation keeping

\[ \dot{\xi}_i = -A_i \xi_i + \sum_{j \neq i} B_{ij} \xi_j, \quad i \in \{1, 2, \ldots, m\} \]  
(6.19)

Define:

\[ f_i(\xi_i) = -A_i \xi_i, \]  
(6.20)
\[ g_i(\xi) = \sum_{j \neq i} B_{ij} \xi_j, \]  
(6.21)

Using

\[ V_i(\xi_i) = \frac{1}{2} \xi_i^T \xi_i \]  
(6.22)

as a Lyapunov function candidate for the \(i\)-th isolated subsystems, we obtain

\[ \frac{\partial V_i}{\partial \xi_i}(\xi_i) = -A_i \xi_i^T \xi_i \]
\[ \leq -\|A_i\| \|\xi_i\|^2 \]
\[ \leq -\alpha_i \varphi_i(\xi_i) \]  
(6.23)

with \(\alpha_i = \|A_i\|\), \(\varphi_i(\xi_i) = \|\xi_i\|\), and

\[ \left| \frac{\partial V_i}{\partial \xi_i} \right| \leq \|\xi_i\| \]
\[ \leq \beta_i \varphi_i(\xi_i) \]  
(6.24)

with \(\beta_i = 1\). Thus \(V_i(\xi_i)\) satisfies (2.3) and (2.4). Furthermore, the interconnection term satisfies the following inequality:
\[ \|g_i(\xi)\| \leq \sum_{j \neq i}^n \|B_{ij}\| \|\xi_j\| \leq \sum_{j \neq i}^n \gamma_{ij} \phi_j(\xi_j), \] (6.25)

Thus, \( g_i(\xi) \) satisfies (2.5) with \( \phi_j(\xi_j) = \|\xi_j\|, \gamma_{ii} = 0 \) and \( \gamma_{ij} = \|B_{ij}\| \) for \( i \neq j \). Now, matrix \( S \) can be formulated as follows:

\[ s_{ij} = \begin{cases} \|A_i\| & \text{for } i = j, \\ -\|B_{ji}\| & \text{for } i \neq j, \end{cases} \] with

\[
A_i = \begin{bmatrix}
0 & 0 & -1 & 0 \\
0 & 0 & 0 & -1 \\
(k_{ix}^t + \sum_{j \neq i} k_{ix}^t) & 0 & (k_{ix}^v + \sum_{j \neq i} k_{ix}^v) & 0 \\
0 & (k_{iy}^v + \sum_{j \neq i} k_{iy}^v) & 0 & (k_{iy}^t + \sum_{j \neq i} k_{iy}^t) \\
\end{bmatrix}
\]

\[
B_{ij} = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
k_{ix}^t & 0 & k_{ix}^t & 0 \\
k_{iy}^t & 0 & k_{iy}^t & 0 \\
\end{bmatrix}
\]

or

\[ S = \begin{bmatrix}
\|A_1\| & -\|B_{12}\| & \ldots & -\|B_{1m}\| \\
-\|B_{21}\| & \|A_2\| & \ldots & -\|B_{2m}\| \\
\vdots & \vdots & \ddots & \vdots \\
-\|B_{m1}\| & -\|B_{m2}\| & \ldots & \|A_m\| \\
\end{bmatrix}. \] (6.27)

The equilibrium point, in this case \( \xi_i = 0 \), is asymptotically stable if \( S \) is an \( M \)-matrix. The \( M \)-matrix condition can be interpreted as a requirement that the diagonal elements of \( S \) be “larger as a whole” than the off-diagonal elements. It can be shown that diagonally dominant matrices with nonpositive off-diagonal elements are \( M \)-matrices (Khalil, 1996).

In this particular case, it is straightforward to find that the \( S \) in (6.27) is an \( M \)-matrix since by the construction of matrices \( A_i \) and \( B_{ij} \), the main diagonal component of \( S \) is always larger than the sum of the off diagonal component. Thus the equilibrium point of \( \xi_i, i \in \{1, 2, \ldots, m\} \) is asymptotically stable provided that \( S \) satisfies the \( M \)-matrix condition. This means that the individual tracking errors of each robot, as well as the formation errors, are asymptotically stable provided that the interconnection with other robots satisfies the \( M \)-matrix requirement. This completes the proof. \( \square \)
Furthermore, if we formulate

\[
A_i = \begin{bmatrix}
0 & 0 & -1 & 0 \\
0 & 0 & 0 & -1 \\
k_{ij}^x & k_{ij}^x & 0 & 0 \\
k_{ij}^y & k_{ij}^y & 0 & 0
\end{bmatrix}
\quad + \quad \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\sum_{j=1}^m k_{ji}^x & 0 & \sum_{j=1}^m k_{ji}^y & 0 \\
0 & \sum_{j=1}^m k_{ji}^x & 0 & \sum_{j=1}^m k_{ji}^y
\end{bmatrix}
\]

It is straightforward to see that \( S \) in (6.27) can be written as

\[
S \leq \begin{bmatrix}
||A_{11}|| & 0 & \ldots & 0 \\
0 & ||A_{21}|| & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & ||A_{m1}||
\end{bmatrix} + \begin{bmatrix}
||A_{12}|| & -||B_{12}|| & \ldots & -||B_{1m}|| \\
-||B_{21}|| & ||A_{22}|| & \ldots & -||B_{2m}|| \\
\vdots & \vdots & \ddots & \vdots \\
-||B_{m1}|| & -||B_{m2}|| & \ldots & ||A_{m2}||
\end{bmatrix}
\]

(6.28)

From (6.29), when there is no communication between the robots, i.e. \( S_i = 0 \), the equilibrium of (6.5) is still asymptotically stable as long as \( k_{ij}^x, k_{ij}^y, k_{ij}^x, k_{ij}^y > 0 \), i.e. \( \xi_i \to 0 \) as \( t \to \infty \) because \( S_i \) is a \( M \) matrix. When coupling gains are zero, but the tracking gains are non-zero, the group of robots becomes pure trajectory tracking systems. The behavior of the group is determined by individual tracking reference capability (which from Section 6.3 has been proven to be stable under the constraint on \( v_j \)).

The formation is defined as the collection of reference trajectories from all robots, and is provided by the motion planner/high-level control. Thus, even when there is no coupling gain or no information sharing between the robots, the desired formation is still maintained under the introduction of perturbations. However, it can be achieved in the longest time. When there are more connections between the robots, i.e. \( k_{ij}^x, k_{ij}^y, k_{ij}^x, k_{ij}^y \neq 0 \), the formation keeping can be attained faster. By using information from other robots, a robot knows if perturbations occur at other robots. Thus, the perturbations can be directly compensated.

In conclusion, the simultaneous trajectory tracking and formation keeping can be achieved by means of trajectory tracking only. The introduction of coupling gains increases the robustness against perturbations in keeping the formation. Thus, it is sufficient to have \( k_{ij}^x, k_{ij}^y, k_{ij}^x, k_{ij}^y > 0 \) while the coupling gains can be chosen freely (as long as they are not negative) to achieve both tracking and formation keeping.
6.4 A performance measure for a group of unicycles

The performance of the control algorithm is evaluated using a root-mean-square error indicator. The indicator is used to measure the trade-offs between trajectory tracking and formation keeping tasks as well as the influence of communication topology.

The individual trajectory tracking performance measure is defined as

$$
P_{T_{ind}} = \sum_{i=1}^{m} \frac{1}{l} \sum_{k=1}^{l} \left( e_{x_{i}(t_{k})}^2 + e_{y_{i}(t_{k})}^2 + \dot{e}_{x_{i}(t_{k})}^2 + \dot{e}_{y_{i}(t_{k})}^2 \right),$$

(6.30)

where $m$ is the number of robots, $t_{k}$ indicates the time instant where the data is taken, and $l$ is the number of data in the simulation or experiment. The measure in (6.30) indicates how good, individually, the group of robots track their individual references. The smaller the value of $P_{T_{ind}}$, the better the performance is.

A second measure is used in order to evaluate the robustness in keeping the formation. Suppose we have different desired formation shapes as depicted in Figure 6.2.

![Figure 6.2: Some examples of formation shapes. The closed-arrows indicate the direction of individual robot movements.](image)

The examples shown in Figure 6.2 illustrate that the formation shape can be expressed as the collection of time-varying relative distances, indicated by the dashed-blue lines, between the robots. For the four robots, the shape of the formation is...
kept if the desired distances between robots 1 and 2, 1 and 3, 1 and 4, 2 and 3, 2 and 4, and 3 and 4 are kept. As a consequence, the difference between the desired and real relative distances between the robots can be used to measure how good the robots keep the desired formation.

If \( \Delta_{r_1}(t) \) and \( \Delta_{r_i}(t) \) are the distances between the reference positions and the actual positions of robot \( i \) and \( j \) respectively, the difference between the two is formulated as

\[
\delta_{ij}(t_k) = \Delta_{r_i}(t_k) - \Delta_{r_j}(t_k),
\]

which gives a measure of how good robots \( i \) and \( j \) keep the relative distance between the two. If \( \delta_{ij}(t_k) = 0 \), it means that robot \( i \) and \( j \) are keeping the desired relative distance. For \( m \)-robots in the group, the formation keeping measure is formulated as

\[
P_T^{\text{form}} = \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=i+1}^{m} \left( \delta_{ij}(t_k) \right)^2.
\]

If \( P_T^{\text{form}} = 0 \), it means that all robots maintain the desired relative distances, i.e. the formation shape is kept. It is to be noted that there is a possibility to have a formation shape that is equal to a rotation or a mirror of the desired formation shape.

The total performance measure is given by

\[
P_T^{\text{total}} = \mu_{\text{ind}} P_T^{\text{ind}} + \mu_{\text{form}} P_T^{\text{form}},
\]

where \( 0 < \mu_{\text{ind}}, \mu_{\text{form}} \leq 1 \) are two scalar weights to penalize the importance of tracking or formation keeping. The smaller the value of \( P_T^{\text{total}} \), the better the performance is.

6.5 Experimental results and performance analysis

This section presents the real-time experimental results using the setup presented in Chapter 3. An analysis of the influence of communication topologies on the formation keeping performance is given. Several experimental scenarios are investigated.

6.5.1 Experimental results

Experimental scenarios

The experiment considers a group of four robots which has to follow a rectangular-like trajectory in a platoon-like formation as depicted in Figure 6.3(a). The reference trajectories for robots 1 to 4 start from the following Cartesian positions ordered set:

\[
\text{start reference} = \{(0, 0.7), (-0.14, 0.7), (-0.28, 0.7), (-0.42, 0.7)\}.
\]
The following control gains are used:

\begin{align}
    k_{p\epsilon}^i &= 1, & k_{d\epsilon}^i &= 2, \\
    k_{p\epsilon}^{ij} &= 4, & k_{d\epsilon}^{ij} &= 1.
\end{align}

The individual trajectory tracking gains, \(k_p^i\) and \(k_d^i\) are chosen so that a (linear) second-order system (of the error dynamics given in (6.11)) with critically damped behavior (Franklin et al., 2002) is obtained. Basically, it requires the \(k_{d\epsilon}^i = \sqrt{k_{p\epsilon}^i}, \epsilon \in \{x, y\}\).
6.5 Experimental results and performance analysis

The experiments are conducted for three initial position ordered sets given as follows:

\[
\text{init set 1} = \{(0, 0.7), (-0.14, 0.7), (-0.28, 0.7), (0.42, 0.7)\}, \quad (6.37)
\]
\[
\text{init set 2} = \{(0.3, 0.725), (0.05, 0.725), (-0.3, 0.5), (-0.55, 0.5)\}, \quad (6.38)
\]
\[
\text{init set 3} = \{(-0.02, 0.5), (-0.3, 0.725), (-0.25, 0.5), (-0.55, 0.725)\}. (6.39)
\]

For the first set, the initial positions coincide with the starting references. The experiments are repeated for the six communication topologies depicted in Figure 6.3(b). If robot \( i \) receives information from robot \( j \), \( k_{ij}^m \) and \( k_{ij}^v \) equal (6.36). Otherwise, they are zero. In this way, the final controller for each robot, which is computed using (6.15), depends on the communication topology.

Furthermore, in all experiments, the forward velocity (dynamic compensator) and the derivatives of the positions of the robots are numerically initialized as follows:

\[
v_i = \zeta_i = v_{di}, \quad (6.40)
\]
\[
\dot{x}_i = v_{di} \cos \theta_i(0), \quad \dot{y}_i = v_{di} \sin \theta_i(0), \quad (6.41)
\]

where \( v_{di} = 0.09 \text{ [m/s]} \) is the desired forward velocity, and \( \theta_i(0) \) is the initial orientation of the robot.

**Remark 6.3.** It is worth noting that the dynamic compensator has a constraint, i.e. \( \zeta_i \neq 0 \), while the reference forward velocity has to be positive. The initialization given in (6.40)-(6.41), follows Theorem 6.1, are practically sufficient to avoid the singularity.

**Results**

Figure 6.4 shows examples of the resulting control signals from experiments using the second initial set. Videos of the experiments are available at www.youtube.com/adinandra98.

The results presented in Figure 6.4 show that the dynamic compensator for each robot, in any experiment, follows the (positive) reference forward velocities. The initial values of the compensators differs per robots depending on the communication topology. As later will be discussed, this situation is related to the way the group of robots react to non-zero initial conditions. For topologies \( A \) and \( F \), i.e. topologies with the least number of connection, the data shows a large overshoot during the movement (around \( t = 20 \text{ [s]} \) and afterwards). This happens when the robots turn. This finding indicates that more connections between the robots help reducing overshoot when the robots turn. Furthermore, the results shown in Figure 6.4 illustrates that the initialization given in (6.40)-(6.41) serves the purpose of avoiding the singularity.

In addition, Figure 6.5 shows examples of the resulting robots’ movements from experiments using the second initial positions set. In these examples, topologies \( A \) and \( B \) are implemented. Meanwhile, Figure 6.6 and 6.7 show the evolution of \( \xi_i \) and \( \delta_{ij} \) of the corresponding robots. The horizontal-dash-dotted lines in these two figures represent the bounds for which the steady state is achieved, i.e. \( \pm 2\% \) of zero.
Figure 6.4: The forward velocity $v_i$, $i \in \{1, 2, 3, 4\}$ from experiments using the second initial position set.

When the group of robots does not start from the same positions as the reference, in the beginning of the experiment there will be conflicting requirements between individual tracking and formation keeping. The robots can directly follow their individual references and vice versa, or accommodate both at the same time.

In this example, because topology B implies complete connection between the robots, as seen in the right hand side of Figure 6.5, the group tries to keep formation first and then follows individual reference trajectories. On the other hand, since topology A implies no connection, the robots directly follows their individual references. In this way, formation is recovered faster when topology B is used compared to topology A. As depicted in Figure 6.4, this finding can also be observed when the lowest and highest values of the compensator are obtained from topology A and B respectively.
6.5 Experimental results and performance analysis

Figure 6.5: The complete movements of the robots from experiments using the second initial position set, topology A (left) and B (right). The number indicates the starting point of each robot, while the circle indicates the beginning of the references.

Moreover, however, as a trade off, as seen in Figure 6.6 and 6.7, topology B causes larger tracking errors ($\xi_i$) compared to topology A. This also means that the formation can be recovered as fast as possible, at the cost of higher individual tracking errors.

Figure 6.6: History of $\xi_i$ and $\delta_{ij}$, $i \in \{1, 2, 3, 4\}$, $j = i + 1$, in experiments using the second initial condition set and topology A.

6.5.2 Performance analysis

Figure 6.8 and 6.9 show the individual tracking (computed using (6.30)) and formation performance keeping (computed using (6.32)) measure respectively.
Figure 6.7: History of $\xi_i$ and $\delta_{ij}, i \in \{1, 2, 3, 4\}, j = i + 1$, in experiments using the second initial condition set and topology B.

Figure 6.8: $P_{T}^{\text{ind}}$ from experiments.

The $P_{T}^{\text{ind}}$ depicted in Figure 6.8 suggests that, for the first initial set, (i.e. the robots start from the same positions as the references) the communication topology has a small influence on the performance. This is indicated by solid-black curve in Figure 6.8 which is almost flat. As for the second and third set, it can be observed that when topology A is used, the tracking performance is the best. This is indicated by the lowest value on both the dashed-black and grey curves. These two curves also suggest that the worst tracking performance is obtained when topology B is used, which is indicated by the highest values in both curves. For the other topologies, the dashed-black and grey curves indicate that if there are more connections between the robots, i.e. more information is being shared, the tracking performance tends to degrade.
As for the formation keeping, Figure 6.9 shows that, regardless of the initial condition, the best $P_f^{\text{ind}}$ is obtained when topology \( B \) is implemented. On the other hand, the worst performance is obtained when topology \( A \) is used. These findings are opposed to the results in individual tracking performance. When all robots communicate, each of them knows what happens to other robots. When there is a perturbation, in this case the non-zero initial conditions, the group tries to preserve the formation first. As a consequence, the individual tracking errors become higher as observed in Figure 6.8.

The results in Figure 6.9 also suggest that formation keeping, regardless of the initial condition, is improving if there are more connections between the robots. Moreover, if the number of connections is the same, i.e., topology \( C \) and \( D \), the connection to the "neighbor", i.e., topology \( C \), tends to yield a better formation keeping performance. The neighboring connection means that the group of robots is connected like a chain and information can be shared directly through the chain. In topology \( D \), the information can only be shared via robot 1.

Although increasing the number of connections between the robots tends to increase the formation keeping performance, it requires a higher communication bandwidth. If the number of robots is very large, this can be a disadvantage. From Figure 6.9, one way to maintain a lower bandwidth while having a meaningful formation keeping performance is by choosing topology that allows connection with the neighboring robots, e.g., topology \( C \) or \( E \).

Furthermore, using $\mu_{\text{ind}} = \mu_{\text{form}} = 1$, the total performance $P_f^{\text{total}}$ is shown in Figure 6.10. Using equal weights, it can be observed that the best total performance for each initial condition set is obtained from different topologies. This suggests that, if tracking and formation keeping are equally weighted, communication topology has less influence on the total performance. The results in Figure 6.10 suggests that to obtain a good trajectory tracking and formation keeping performance, there is less
need to share information between the robots. This is indicated by the fact that Topology B does not always give the lowest $P_{T\text{ total}}$.

![Graph showing $P_{T\text{ total}}$ from experiments.](image)

Figure 6.10: $P_{T\text{ total}}$ from experiments.

If either individual trajectory tracking or formation keeping is more important, the corresponding scalar weights need to be increased. In this way, less important measures can be suppressed by the more important ones. Thus, it can be expected that the more important measures will dominate $P_{T\text{ total}}$.

As an additional illustration, another set of experiments is conducted using the scenario depicted in Figure 6.11.

![Graph showing reference trajectories and perturbations introduced during experiments.](image)

Figure 6.11: The reference trajectories and perturbations introduced during experiments. The rectangles indicate the positions of the robots when perturbations are introduced.
In this scenario, the robots start from the same position as their individual references. During the process of completing the task, three perturbations are introduced at different times. The square signs in Figure 6.11 indicate the positions of the robots when one of them is manually positioned to the point indicated by the star. The computations of trajectory tracking, formation keeping, and total performances are given in Figure 6.12(a)-6.12(c), respectively.

![Graphs showing performance measures](attachment:image.png)

**Figure 6.12**: Performance measure from the experiments using the scenario shown in Figure 6.11.

As seen in Figure 6.12(a), the best individual tracking performance is obtained when there is no communication between the robots, i.e. topology $A$ is implemented, and vice versa. On the other hand, the best formation keeping is obtained when all robots communicate, i.e. topology $B$ is implemented, and vice versa. Furthermore, Figure 6.12(c) indicates that with equal scalar weights, the best total performance comes from topology $A$ although with a very small difference compared to topology $B, C,$ and $D$. These findings, similar to the results from the previous experimental scenarios, suggests that there is less need to share all information to achieve a good trajectory tracking and formation keeping performance.
From the perspective of the hierarchical control approach depicted in Figure 6.1, the performance analysis presented above suggests that the overall performance is mostly dictated by the high-level control as long as the low-level control accurately tracks the reference trajectories. The low-level control is also responsible for adding the robustness in formation keeping.

It is to be noted that the performance measures given in (6.30) and (6.32) are similar to the ones presented as alternative performance measures in Section 5.5.

6.6 Conclusions

In this chapter a simultaneous trajectory tracking and formation keeping control algorithm for a group of robots is presented. The control algorithm is formulated by extending a trajectory tracking controller for a single robot designed using dynamic feedback linearization approach. The stability of the closed-loop system is analyzed using an existing theorem on interconnected systems. The algorithm has been successfully validated in real-time experiments.

Using root-mean-square error-like indicators, the performance of the control algorithm is analyzed. The analysis of the experimental results suggests that the communication topology, which dictates how the robots exchange information, has more influence on the performance compared to the initial conditions/perturbations. The more information that is shared between the robots, the better formation keeping is expected. However, the opposite result is obtained for the individual trajectory tracking performance. Although more information sharing improves formation keeping performance, it requires a high communication bandwidth that can be a disadvantage. In this particular experimental case study, the best total performance with equal scalar weights result is obtained when there is no communication between the robots. This suggests that a good simultaneous trajectory tracking and formation keeping performance can be achieved with less information sharing.
This chapter discusses a model predictive control (MPC) algorithm to regulate a group of mobile robots. The MPC algorithm is based on a sequentially-decentralized approach, i.e. a single model predictive controller computes the control signals for a group of mobile robots where priority rules are used to determine which robots are handled earlier in the optimization procedure. Similar transport tasks in an automated warehouse are used as a case study. Several practical aspects are addressed so that the MPC algorithm can be validated both in simulation and real-time experiments (with 12 mobile robots). Using completion time as an indicator, the performance of the MPC algorithm is evaluated. Relevance and performance comparison to the results presented in Chapters 4 and 5 are given.

7.1 Introduction

As a complement to the hierarchical control approach, in this chapter another type of coordination control algorithm based on a Model Predictive Control (MPC) approach is proposed. MPC is a control methodology that is based on an optimization process and easily handles constraints in either the inputs or outputs. It first appeared in multivariable constrained processes in chemical industry. In the beginning, due to the combination of limited computing power and the nature of the optimization procedure, MPC was only suitable for systems with slow dynamics. However, with the increasing computing power, it is now possible to use MPC in a system with much faster dynamics such as electro-mechanical systems in general and mobile robot systems in particular.
The use of MPC to regulate a mobile robot or a group of mobile robots, especially of the unicycle type, has been extensively studied; see for example the works of Gu and Hu (2006), Xie and Fierro (2008), and Zhu and Özgüner (2008) for the control of a single robot and the works of Fukushima et al. (2005), Xie and Fierro (2007), and Voos (2009) for the control of multiple robots. Various aspects of MPC, especially the cost function choices, the optimization algorithm, and the stability analysis of the algorithms, are discussed.

In Shin and Kim (2009), three different techniques are introduced, namely centralized, decentralized, and sequentially decentralized, to implement a single model predictive controller that regulates multiple unmanned aerial vehicles. The different implementation techniques are intended to overcome the limited computing power of a single PC if it has to compute the control signals of multiple vehicles. In the work of Defoort et al. (2008), a centralized and decentralized method are used to implement an MPC that regulates a group of unicycle mobile robots so that both individual and collective goals can be achieved. The centralized implementation typically yields a better performance than the decentralized one. However, the centralized method requires large computing power while the decentralized method needs less computing power.

In this work, the focus is on the practical aspects of the MPC algorithm, so that a real-time implementation of the algorithm can be achieved. Another focus is on a large collection of mobile robots in a vastly changing environment. As a case study, the MPC algorithm will regulate a group of unicycle mobile robots for transportation in an automated warehouse. Although the currently available computing power is high, it is not straightforward to accomplish real-time MPC, as well as running the algorithm for a large number of mobile robots.

The contributions of this chapter are as follows. Firstly, a practical sequentially decentralized MPC algorithm to regulate a large collection of mobile robots is proposed. Secondly, the MPC algorithm is validated both in simulation (up to 20-mobile robots) and real time experiments (up to 12-mobile robots). The transportation system of an automated warehouse is used as a case study. The simulations show how MPC performs in regulating a group of mobile robots in a real automated warehouse. The experiments show how the algorithm works in a real-time situation in which a smaller scale of the warehouse is considered. Thirdly, the MPC algorithm is evaluated in terms of completion time. Relevance and performance comparison to the hierarchical control approach presented in Chapter 4 and 5 are given.

The remainder of this chapter is organized as follows. Section 7.2 presents brief preliminaries on the MPC algorithm, including the discrete unicycle model used in this chapter. In Section 7.3, the sequentially decentralized MPC approach is presented. Section 7.4 describes the discrete-time model used for control design. Section 7.5 discusses the choice of a cost function. In Section 7.6, the optimization algorithm is explained. Section 7.7 gives the simulation and experimental results and a detailed analysis of the performance of the MPC algorithm is given. Section 7.8 gives the relevance and performance comparison between the MPC algorithm and the hierarchical control concept presented in Chapters 4 and 5. Finally, Section 7.9 gives the conclusions of this chapter.
7.2 Preliminaries on MPC

This section describes the basic principle of MPC taken from Camacho and Bordons (2000) and Rawlings and Mayne (2009). It covers both the basic principle for systems with slow and fast dynamics.

7.2.1 General principle of MPC

The basic principle of MPC is depicted in Figure 7.1. This figure shows the past, present, and future conditions of a system. \( t = kT_s \) indicates the time instant, \( k \in \mathbb{N} \), and \( T_s \) is the sampling time. The present time is indicated by \( t = kT_s \). The past shows what has happened up to the present \( t = kT_s \). The future is shown up to the prediction horizon \( N_p \) so that \( t = (k + N_p)T_s \). The future indicates how many samples of the future behavior need to be taken into account during the optimization procedure. The output of the systems up to and including the present time \( t = kT_s \) is indicated by the curve and marker in purple.

The MPC is formulated as an optimization problem at every time instant to obtain the optimal control inputs that meet the control objectives. The optimization problem is formulated as follows:

\[
\begin{align*}
\min_I & \quad J(I) \\
\text{subject to} & \quad I_{\min} \leq I \leq I_{\max}, \\
& \quad O(I)_{\min} \leq O(I) \leq O(I)_{\max},
\end{align*}
\]

where \( J(I) \) is the cost function, \( I \) is the predicted input, i.e. the input that is computed until the prediction horizon (the red-curve in Figure 7.1), and \( O(I) \) is the predicted output (the blue-curve in Figure 7.1). This predicted input remains constant between two sampling periods as indicated by the open and full red markers. The optimization procedure will produce the predicted output shown in the blue line.
in Figure 7.1. This predicted output remains constant after the control horizon of \( N_c \) samples, where \( 1 \leq N_c \leq N_p \), has been reached.

The optimization in (7.1) minimizes the value of \( J(I) \) by adjusting the predicted input. Minimizing \( J(I) \) also means that the predicted output follows the control objectives. During the optimization procedure, constraints that exist in the system, such as on the inputs (7.2) or outputs (7.3) can easily be included in the optimization problem. This is one of the advantage of using MPC to solve a control problem.

Once the optimization procedure is completed, a set of predicted inputs that minimize the cost function is available. From this set, only the first sample of the predicted input is implemented. The optimization procedure is repeated at the next sampling time until the complete control objectives are reached. This working principle is also called receding horizon principle.

### 7.2.2 MPC for systems with fast dynamics

The description given in Section 7.2.1 is valid for a system with slow dynamics, i.e. the sampling time is much longer than the time it takes to solve the optimization. For system with fast dynamics, e.g. robotic systems, which have the opposite property, the description needs to be modified that leads to the situation shown in Figure 7.2.

![Figure 7.2: The general principle of a MPC system with fast dynamics.](image)

Due to the restriction on the sampling time, which is shorter than system with slow dynamics, it becomes very critical to make sure that optimization procedure can be completed in one sampling time. Furthermore, in this work, a delay of one sampling period is introduced. For Figure 7.2, this means that the input that is applied to the system for \( t \in [kT_s, (k+1)T_s) \) is the result of the optimization process that took place when \( t \in [(k−1)T_s, kT_s). \) In this way, the input is fixed for one sampling period while the new inputs are being calculated.

As a consequence, the output for \( t \in [kT_s, (k+1)T_s) \) is the output computed for \( [(k−1)T_s, kT_s) \) indicated with the light blue line and marker. While these inputs and outputs are held constant for \( t \in [kT_s, (k+1)T_s) \), the new control input is computed...
which will be applied at the next sampling time. Furthermore, introducing a delay of one sampling period makes the first future instant \( t = (k + 1)T_s \) uncontrollable and the control horizon \( N_c \) is restricted to \( 1 < N_c < N_p \).

### 7.3 A sequentially decentralized MPC

As used in many references, see for examples Defoort et al. (2009) and Shin and Kim (2009), centralized and decentralized MPC can be used to regulate a group of vehicles. In the centralized approach all vehicles are treated as a single entity. The equality or inequality constraints of the optimization procedure are simply the collection of individual constraints. In this way, both individual and collective goals can be directly accommodated. Suppose there are \( m \) vehicles in the system, then the MPC is formulated as given in (7.1) with \( I = \{I_1, I_2, \ldots, I_m\} \) where \( I_i \) indicates individual predicted input. The centralized approach is typically implemented in a single PC.

On the other hand, for the decentralized approach the optimization procedure is applied individually in each vehicle. The MPC is formulated as given in (7.1) but by using \( I = I_i \) only. If required to achieve a collective goal, the robots have to share the necessary information so that the goal can be accomplished. Thus, a communication channel has to be available. The decentralized approach can be implemented in each on-board processor of the vehicle or in a single PC that simultaneously runs parallel MPC algorithms for all robots. Decentralized MPC naturally accommodates individual goals.

Since the centralized approach takes into account information from all vehicles, it typically results in a better performance compared to the decentralized one. At any time instant, the optimization computes control signals that accommodate the objectives of all robots. This will ensure a better group performance. Meanwhile, in the decentralized approach, since only individual information is used and sharing information is optional, the resulting control signals mainly accommodate individual goals. Thus, the group performance becomes worse. However, the decentralized approach requires less computing power compared to the centralized one. This is result from the fact that centralized MPC takes into account information from all robots when computing the control signals.

A way to combine the advantages of both methods is to implement a sequentially decentralized method (Shin and Kim, 2009). In this method, each vehicle uses the future prediction information from other vehicles whose optimization process has already been performed. This method basically requires communication between the vehicles based on priority rules. The information to be shared is limited only to vehicles with earlier optimization procedures. In this way, the computation load of the optimization procedure can be kept low. The priority can be used to maintain the group goal since it can determine which vehicle is handled earlier in the optimization procedure so that the group goals can be maintained.

In this work, the focus is on the real time implementation and large scale systems. Thus, the centralized version is not a good option because of the high computing power requirement. The limitation to achieve the group goals and limited computa-
tion capabilities of the experimental setup (See Chapter 3) make the decentralized MPC unviable.

In this thesis, a sequentially decentralized MPC is chosen. This method serves our need in accommodating individual and group goals at the same time. Furthermore, it is also suitable with the experimental setup where a PC has to compute the required control signals of all robots. A nonlinear cost function that accommodates trajectory tracking and collision avoidance between the robots is proposed. A combination of the steepest descent method and line search is implemented to solve the optimization problem.

For comparison purposes, the centralized approach is also implemented. Later in Section 7.6 the limitation of the centralized approach will be discussed. The next sections discuss the design of each component of the proposed sequentially decentralized MPC (and the necessary modification for the centralized approach).

### 7.4 Discrete-time model of unicycle mobile robots

This section discusses the discrete-time model of unicycle mobile robots, which is needed to compute the predicted input and output signals.

#### 7.4.1 A discrete time kinematic model of unicycle mobile robots

Since MPC tries to solve the control problem at every sample period, it is more natural to work in the discrete time domain. Thus, for the sake of control design, different from the continuous time unicycle model used in Chapters 4 to 6, in this chapter a discrete time kinematic model of the unicycle mobile robot is considered. The model proposed by Niño-Suárez et al. (2006) is used. The discretization process uses the assumption that the inputs, \( v \) and \( \omega \), of the (continuous time) unicycle model as given in (2.10) remain constant on the interval between two sampling periods \( t_k \) defined as

\[
    t_k \equiv \{ kT_s, (k+1)T_s \}, \quad (7.4)
\]

where \( T_s \) denotes the sampling period. The discrete-time model of unicycle \( i \) is given by:

\[
    \begin{bmatrix}
        x_i((k+1)T_s) \\
        y_i((k+1)T_s) \\
        \theta_i((k+1)T_s)
    \end{bmatrix} =
    \begin{bmatrix}
        x_i(kT_s) \\
        y_i(kT_s) \\
        \theta_i(kT_s)
    \end{bmatrix} + \int_{kT_s}^{(k+1)T_s} \begin{bmatrix}
        \cos \theta_i(\lambda) & 0 \\
        \sin \theta_i(\lambda) & 0 \\
        0 & 1
    \end{bmatrix} \begin{bmatrix}
        v_i(kT_s) \\
        \omega_i(kT_s)
    \end{bmatrix} d\lambda .
\]

After solving the integral (see Niño-Suárez et al. (2006) for details), the discrete-
time model is formulated as follows:
where \( \gamma_i(\omega_{kT_i}) \) is defined as

\[
\gamma_i(\omega_{kT_i}) = \begin{cases} \frac{\dot{\omega}_i(\omega_{kT_i})}{\omega_i(\omega_{kT_i})}, & \text{if } \omega_i(\omega_{kT_i}) \neq 0, \\ \frac{2}{\pi}, & \text{if } \omega_i(\omega_{kT_i}) = 0. \end{cases}
\] (7.7)

### 7.4.2 Obtaining predicted outputs

As mentioned in Section 7.2, MPC uses the predicted output to compute the cost function values. In this unicycle case, the predicted output equals the state of the unicycle.

Using (7.6) and (7.7), the predicted input of unicycle \( i \) is given as follows:

\[
I_i(kT_i) = \begin{bmatrix} \frac{v_i(kT_i)}{\omega_i(kT_i)} & v_i((k+1)T_i) & \cdots & v_i((k+N_p-1)T_i) \\ \omega_i(kT_i) & \omega_i((k+1)T_i) & \cdots & \omega_i((k+N_p-1)T_i) \end{bmatrix}^T.
\] (7.8)

The inputs below the braces are the first inputs that are calculated while \( t \in [(k-1)T_i,kT_i). \) As soon as \( t = kT_i, \) these inputs are sent to the unicycles and they become fixed. Substituting (7.8) into (7.6), the predicted output matrix \( X_i(kT_i) \) is formulated as

\[
X_i(I_i(kT_i)) = \begin{bmatrix} x_i((k+2)T_i) & \gamma_i((k+2)T_i) & \theta_i((k+2)T_i) \\ x_i((k+3)T_i) & \gamma_i((k+3)T_i) & \theta_i((k+3)T_i) \\ \vdots & \vdots & \vdots \\ x_i((k+N_p-1)T_i) & \gamma_i((k+N_p-1)T_i) & \theta_i((k+N_p-1)T_i) \end{bmatrix}.
\] (7.9)

The states at \( t = (k+1)T_i \) are excluded since they are fixed due to the delay of one sample caused by the controller.

**Remark 7.1.** For centralized MPC, the predicted input \((m \text{ mobile robots})\) is given by

\[
I_i(kT_i) = \begin{bmatrix} (I_1(kT_i))^T & (I_2(kT_i))^T & \cdots & (I_m(kT_i))^T \end{bmatrix}^T.
\] (7.10)

while the predicted output for centralized MPC of \((m \text{ mobile robots})\) can be formulated as

\[
X_i(I_i(kT_i)) = \begin{bmatrix} (X_1(I_1(kT_i)))^T & (X_2(I_2(kT_i)))^T & \cdots & (X_m(I_m(kT_i)))^T \end{bmatrix}^T. \] (7.11)
Chapter 7. MPC for a group of unicycles

7.5 Cost function

MPC requires a cost function that resembles the control objectives and is used to measure how far the control objectives have been achieved. In this work, a cost function that consists of two parts is implemented. The first part deals with the trajectory tracking problem and the second corresponds to the problem of avoiding collision with other unicycles.

The cost function has to be defined from \( t = (k + 2)T_s \) to \( t = (k + N_p)T_s \). The control objective are not specified at \( t = (k + 1)T_s \) since this future time instant is uncontrollable due to the fact that the input and output of the system are held for \( t \in [kT_s, (k + 1)T_s) \).

7.5.1 Trajectory tracking problem

Suppose that the reference trajectories for the unicycle are given as

\[
R_i(t) = \begin{bmatrix} x_i(t) \\ y_i(t) \\ \theta_i(t) \end{bmatrix}, \quad i \in \{1, 2, \ldots, m\}, \quad (7.12)
\]

where \( m \) robots are available. \( R_i(t) \) represents the desired position and orientation of unicycle \( i \). Any arbitrary geometric path can be assigned to \( R_i(t) \) as long as the nonholonomic constraint is satisfied. One advantage from this property is that point-to-point movements can be formulated as a trajectory tracking problem.

Consider a situation as depicted in Figure 7.3 (note the color representation used in Section 7.2). In this situation, the robot has to follow the given reference trajectories shown by the green marker. The distances between the actual position and orientation and their reference at a future sampling time \( nT_s \), \( D'_i(nT_s) \) and \( A'_i(nT_s) \), respectively are given as

\[
D'_i(nT_s) = \sqrt{(x_i(nT_s) - x_i(nT_s))^2 + (y_i(nT_s) - y_i(nT_s))^2}, \quad (7.13)
\]

\[
A'_i(nT_s) = (\theta_i(nT_s) - \theta_i(nT_s))^2. \quad (7.14)
\]
The cost function of the trajectory tracking problem is formulated as follows:

\[ J_R^i(kT_s) = \sum_{n=k+2}^{k+N_p} \left( P_R^i D_r^i(nT_s) + Q_R^i A_r^i(nT_s) \right), \quad (7.15) \]

where \( P_R^i \) and \( Q_R^i \) are two scalar positive weights that determine the relative importance between tracking the desired position or the desired angle.

### 7.5.2 Collision avoidance with other unicycles

It is assumed that the unicycle has a circular shape. This choice is also motivated by the fact that the robots used in experiments have a circular shape. As explained in Section 7.3, using the sequentially decentralized method, robots can plan their trajectories in an order that results from their respective priorities. A unicycle needs to avoid lower priority unicycles only at the first controllable future instant \( t = (k+2)T_s \). The higher priority robots need to be avoided at every controllable future instant.

Consider a situation where a unicycle is driving in the presence of another unicycle, as depicted in Figure 7.4. The collision avoidance between the unicycles is solved by using the distance between the centers of the robots.

\[
D_r^i(nT_s) = \sqrt{\left( x_i(nT_s) - x_j(nT_s) \right)^2 + \left( y_i(nT_s) - y_j(nT_s) \right)^2}. \quad (7.16)
\]
The occurrence of collisions at future sampling time $nT_s$ can be checked as follows:

$$C_j^i(nT_s) = \begin{cases} 0, & \text{if } D_j^i(nT_s) - r_i - d_{ij}^{extr} - r_j > 0 \text{ (no collision)}, \\ 1, & \text{if } D_j^i(nT_s) - r_i - d_{ij}^{extr} - r_j \leq 0 \text{ (collision)}, \end{cases}$$

(7.17)

where $r_i$ and $r_j$ are the diameters of unicycle $i$ and $j$, respectively and $d_{ij}^{extr} \geq 0$ is an extra safety distance.

The cost of a unicycle approaching another unicycle, or colliding with it, is formulated as follows:

$$J_j^i(nT_s) = \begin{cases} \frac{1}{D_j^i(nT_s) - r_i - d_{ij}^{extr} - r_j}, & \text{if } C_j^i(nT_s) = 0 \text{ (no collision)}, \\ P_{jcol, i} D_j^i(nT_s), & \text{if } C_j^i(nT_s) = 1 \text{ (collision)}, \end{cases}$$

(7.18)

where $P_{jcol, i}$ is a scalar weight. Depending on the priority between robot $i$ and $j$, for prediction horizon $N_p$, expression (7.18) can be written as:

$$J_j^i(kT_s) = \sum_{n=0}^{\tilde{n}} \left( \frac{1 + C_j^i(nT_s)(P_{jcol, i} - 1)}{D_j^i(nT_s) - (1 - C_j^i(nT_s))(r_i + d_{ij}^{extr} + r_j)} \right),$$

(7.19)

with

$$\tilde{n} = \begin{cases} k + 2, & \text{if } P_i(kT_s) > P_j(kT_s), \\ k + N_p, & \text{if } P_i(kT_s) < P_j(kT_s), \end{cases}$$

(7.20)

where $P_i(kT_s)$ and $P_j(kT_s)$ is the priority of robot $i$ and $j$ respectively.

### 7.5.3 The complete cost function

The complete cost function is the summation of the cost function for the trajectory tracking and avoiding collision between the robots. For sequentially decentralized MPC, it is formulated as

$$J_{To}^i(kT_s) = J_{R}^i(kT_s) + \sum_{j=1, j \neq i}^{m} J_{j}^i(kT_s), \quad i \in \{1, 2, \ldots, m\}.$$  (7.21)

The cost function in (7.21) is a scalar expression that is evaluated using the input vector $I_i(kT_s)$ given in (7.8). As for centralized MPC, the cost function is formulated as

$$J_{To}^i(kT_s) = \sum_{j=1}^{m} \left( J_{R}^j(kT_s) + \sum_{j=1, j \neq i}^{m} J_{j}^i(kT_s) \right).$$  (7.22)

The cost function in (7.22) is evaluated using the input vector $I_i(kT_s)$ given in (7.10).

**Remark 7.2.** It is worth noting that the control objectives of trajectory tracking and collision-avoidance with other unicycles are similar to the control objectives considered in Chapters 4 and 5.
7.6 Optimization algorithm

7.5.4 Priority rules

For sequentially-decentralized MPC, priority $P_i(kT_s)$ is required to determine which robot is handled earlier in the optimization procedure. In this work, two priorities are investigated, namely:

1. Fixed priority. In this option, a robot has a priority that is fixed every sampling time. Examples are lower number priority, i.e. a lower robot ID gets higher priority:

$$P_i(kT_s) = i, \quad i \in \{1, 2, \ldots, l\}, \quad \forall k,$$

(7.23)
or higher number priority, i.e. a higher robot ID gets higher priority:

$$P_i(kT_s) = l - i + 1, \quad i \in \{1, 2, \ldots, l\}, \quad \forall k,$$

(7.24)

2. Non-fixed priority. In this option, a robot can receive different priorities every sampling time. An example is by using the value of the trajectory tracking cost function:

$$P_i(kT_s) = J^R_i(kT_s).$$

(7.25)

The idea is that the robot that is far from accomplishing its control objective, i.e. has higher $J^R_i(kT_s)$, is more important than the robot that almost or already completed its task.

To avoid back and forth priorities switching, e.g. which can occur while robots try to avoid each other, priorities are only allowed to increase or stay the same as they were at the previous sampling time. Only if a priority level drops below the base priority level, which is set at a fraction (e.g. 1%) of the highest priority value that has ever occurred with all unicycles, the priority value becomes zero. This base priority level indicates that a unicycle has fulfilled its control objectives. When this strategy is applied, the unicycles will be able to resolve the situation when priorities comes back and forth.

Remark 7.3. It is to be noted that the fixed priority choices are similar to the priority choices used in Chapter 4 (low-level control), while the non-fixed priority concept is similar to the priority concept used in Chapter 5 (high-level control).

7.6 Optimization algorithm

The optimization problem for unicycle $i$ can be formulated as follows:

$$\min_{I_i(kT_s)} J^R_i(I_i(kT_s))$$

subject to

$$v_{i,\text{min}} \leq v_i \leq v_{i,\text{max}},$$
$$\omega_{i,\text{min}} \leq \omega_i \leq \omega_{i,\text{max}},$$
where $J^m_i(k_{Ts})$ is the cost function given in (7.21) and $I_i(k_{Ts})$ is the input vector given in (7.8). The optimization will be executed per unicycle and depends on its priority. For centralized MPC, the problem formulation remains the same, but $J^m_i(k_{Ts})$ is replaced by $J^f_i(k_{Ts})$ given in (7.22) and $I_i(k_{Ts})$ is replaced by $I_c(k_{Ts})$ formulated in (7.10).

Since the cost function is nonlinear and there are constraints on the inputs of the unicycles, it is impossible to find an explicit solution for an optimal predicted input sequence. One way to overcome this problem is to use a local exploration method (Papalambrso and Wilde, 2000).

In this work, a combination of the steepest descent and line search optimization methods is implemented. The steepest descent method has the advantage that only the gradient of the cost function needs to be available. This is very beneficial to reduce the computation load. A disadvantage of the steepest descent method is that when the input vector approaches a local optimum (where the input constraints are not active) the progress towards the optimum becomes slow because the gradient decreases in size. In addition, when an input vector is far from a local optimum it can occur that a much too large step in the search direction is taken. This leads to a higher cost function value. To resolve these disadvantages, the steepest descent method is used in combination with a line search optimization.

Without loss of generality, the description given in the next sections considers the situation of sequentially decentralized MPC. For the centralized MPC, the description is still valid with a proper modification on the cost function, i.e. from $J^m_i$ to $J^f_i$, and input vector, i.e from $I_i(k_{Ts})$ to $I_c(k_{Ts})$.

### 7.6.1 Steepest descent optimization method

The gradient of the cost function is formulated as

$$g_i(I_i(k_{Ts})) = \frac{\partial J_i(k_{Ts})}{\partial I_i(k_{Ts})}.$$  \hspace{1cm} (7.26)

The steepest descent optimization method uses a first order approximation of the cost function to determine a lower cost function value. This approximation is given by

$$J_i^{p+1}(k_{Ts}) = J_i^p(k_{Ts}) + g_i^p(I_i^p(k_{Ts})) \cdot \partial \left( I_i^p(k_{Ts}) \right),$$  \hspace{1cm} (7.27)

where, $p$ indicates the current iteration, and

$$\partial \left( I_i^p(k_{Ts}) \right) = I_i^{p+1}(k_{Ts}) - I_i^p(k_{Ts}).$$  \hspace{1cm} (7.28)

A search direction that results in a lower cost function value can be found in the negative direction of the gradient of the cost function:

$$\partial \left( I_i^p(k_{Ts}) \right) = -g_i(I_i^p(k_{Ts})).$$  \hspace{1cm} (7.29)

Combining (7.28) and (7.29), an iterative procedure can be formulated as follows:

$$I_i^{p+1}(k_{Ts}) = I_i^p(k_{Ts}) - g_i(I_i^p(k_{Ts})).$$  \hspace{1cm} (7.30)
7.6.2 Line search optimization

Line search optimization is a method that searches for a minimum in a descent direction that is determined in advance. This gives more control over the step size that is taken and the gradient does not have to be evaluated at every iteration, leading to shorter computation times. Using (7.30), a descent direction \( s^p \) can be computed using

\[
s^p \left( I^p(kT) \right) = -g, \left( I^p(kT) \right). \tag{7.31}
\]

Introducing a step size of length \( \alpha^p \), a new iterative optimization procedure can be created as follows:

\[
I^{p+1} \left( kT \right) = I^p \left( kT \right) + \alpha^p s^p \left( I^p(kT) \right). \tag{7.32}
\]

To limit the variation in the search direction, the descent direction vector \( s^p \) is scaled so that the largest size of a component of the gradient equals to one. Thus,

\[
s^p \left( I^p(kT) \right) = s^p \left( I^p(kT) \right) \| s^p \left( I^p(kT) \right) \|_\infty. \tag{7.33}
\]

By keeping the initial step size \( \alpha^0 \) small, e.g. \( \alpha^0 = 10\% \) of the range of the input, large changes in inputs can no longer occur in one iteration.

To iteratively obtain a lower cost function value using the search direction, the following steps are used:

1. Determine an initial input vector \( I^0 \), the initial step size \( \alpha^0 \), the minimum allowed step size \( \alpha_{\text{min}} \), and the maximum number of function evaluations \( \text{NP}_{\text{max}} \).
2. Compute the search direction using (7.31) and scale the search direction using (7.33).
3. Keep taking steps of size \( \alpha^p \) in the search direction given by (7.32) and increasing the iteration index \( p \) until \( J^p \left( I^p(kT) \right) > J^p \left( I^{p-1}(kT) \right) \).
4. Use \( I^{p-1}(kT) \) as the result of the line search of step 3, and if the first step taken with a new \( \alpha^p \) already results in a larger cost function value, reduce the step size of \( \alpha^0 \).
5. If \( \alpha^p \geq \alpha_{\text{min}} \), and if \( p \leq \text{NP}_{\text{max}} \), return to step 2.

If the computation of (7.32) in step 3 results in an input vector that exceeds the constraint, the input vector \( I^p(kT) \) is bounded to its constraint value and used in step 2. After the search direction is recalculated, it is checked if the direction of the input(s) that are at their bounds can lead to another constraint violation. If this is so, the corresponding part of the search direction is set to zero. In addition, it can occur that the optimization results in a path for an unicycle that will collide with
another unicycle. If this occurs, and if \(0 \in [v_{i,\text{min}}, v_{i,\text{max}}]\), the unicycle is stopped at the next future sampling time. In this way, collisions can be avoided. It is expected that at the next sampling time the optimization results in a collision free path.

### 7.6.3 Implementation: computation time reduction.

The optimization algorithm is implemented in MATLAB at the experimental setup PC described in Section 3.1. The actual computation of the gradient of the cost function is performed using the Symbolic Toolbox of MATLAB once before the overall optimization procedure is started. However, direct use of the toolbox results in a large computational load that is not beneficial, especially for experimental purposes.

To solve this problem, the computed cost function from the Symbolic Toolbox is manipulated by gathering and grouping the terms in the gradient that occur multiple times. This ensures that a similar term is only calculated once which reduces the computation time.

The method works by first gathering every sine and cosine term that occurs in the expression. If a new term is encountered, it is stored in a database. If the same term is encountered again, it is replaced by the entry from the database. After that, the modified terms without sines and cosines are further simplified. Terms that are connected with a plus or minus sign are separated, starting with the terms that are between the most inner brackets.

An example of this simplification is shown in Table 7.1 where the expression \((\sin 5x + \cos 2x)^2 + 3x (\sin 5x + \cos 2x)\) is simplified using the proposed method. While the expression is simplified, its components are stored in a database which is shown in column three of Table 7.1. Any subsequent expression that is simplified, has access to the terms that are already in the database, and can add new terms.

<table>
<thead>
<tr>
<th>Step</th>
<th>Expression</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>((D_1 + \cos 2x)^2 + 3x (D_1 + \cos 2x))</td>
<td>(D_1 = \sin 5x)</td>
</tr>
<tr>
<td>2</td>
<td>((D_1 + D_2)^2 + 3x (D_1 + D_2))</td>
<td>(D_2 = \cos 2x)</td>
</tr>
<tr>
<td>3</td>
<td>(D_1^2 + 3xD_3)</td>
<td>(D_1 = D_1 + D_2)</td>
</tr>
<tr>
<td>4</td>
<td>(D_4 + 3xD_5)</td>
<td>(D_2 = D_2^2)</td>
</tr>
<tr>
<td>5</td>
<td>(D_6 + 3xD_5)</td>
<td>(D_5 = 3xD_3)</td>
</tr>
</tbody>
</table>

### 7.7 Simulation and experimental results

For both simulation and experimental results, the same transportation system of an automated warehouse used in Chapters 4 and 5 is investigated. Similar scenarios as in Chapters 4 and 5 are used. The MPC algorithm is evaluated for a different numbers of robots, priorities, and prediction horizon. The performance is analyzed in terms of completion time \(t_{\text{complete}}\).
In Chapters 4 and 5, two possible fixed paths that can be taken by the robots depicted in Figure 7.5(a) are tested. In this chapter there is no specific geometric path assigned to the robots. The robots are assigned to points, i.e. sets of picking/dropping points, that they have to go as, illustrated in Figure 7.5(b). The MPC algorithm finds the optimal paths for each robot while at the same time it guarantees that no collision occurs. Similar sets of picking/dropping points as in Chapters 4 and 5 are implemented.

![Figure 7.5](image)

Figure 7.5: a) The fixed geometric path options used in Chapter 4 and 5; b) The non-fixed geometric paths used in this chapter.

### 7.7.1 Simulation results

For the simulation case study, the group of unicycles has to deliver 8577 picking/dropping tasks. The task division among the robots follows the algorithm given in Section 4.2.3. The number of robots is given as \( m = \{6, 8, 10, 12, 14, 16, 18, 20\} \). Both fixed (7.23) and non-fixed priorities (7.25) are tested. The following MPC parameters are used:

\[
T_s = \frac{1}{6} \text{[s]}, \quad N_r \in \{3, 6, 9\}, \quad N_i = 2, \\
P_h = 10^6, \quad Q_h = 10^2, \\
P_{i.e} = 10^4, \quad Q_i = 10^6, \quad R_i = 10^6, \\
V_{\text{min}} = 0 \text{[m/s]}, \quad V_{\text{max}} = 1.1 \text{[m/s]}, \\
\omega_{\text{min}} = -2.5 \text{[rad/s]}, \quad \omega_{\text{max}} = 2.5 \text{[rad/s]}.
\]

The choice of (7.37) and (7.38) are influenced by robot assumptions given in Section 4.4. The sampling time is chosen so that the same values can be used in real-time experiment. Parameters in (7.35) and (7.36) are chosen by manual tuning. The summary of \( t_{\text{complete}} \) from simulations with different scenarios and parameters is depicted in Figure 7.6.

Figure 7.6 shows that the MPC can handle a large variation of the number of robots in the system. Although the number of robots used is at most 20, the proposed sequentially decentralized MPC still can handle a higher number of robots. Figure 7.6 indicates that in this particular case study the addition of robots reduces the completion time, which is desired.
Comparing the values of the red and black curves of Figure 7.6, either the solid or dashed one, it can be observed that increasing the prediction horizon tends to shorten the completion time. With a longer prediction horizon, the optimization procedure takes into account longer predicted outputs of the robot that improves the resulted optimal control signals. This situation can also be observed from the black curves. When the number of robots increases, a lower prediction horizon tends to lengthen the simulation time. It can be seen that the values of $t_{\text{complete}}$ at the black curves increase when the number of robots equals 14 or 16. In the worst case, a lower prediction horizon can cause deadlock, i.e. the robots try to reach their own goals but cannot find the paths that every robots agrees. As a consequence, no robots move, e.g. this happens when $m = 20$ and $N_p = 3$.

However, comparing the values of the red and the green curves, it can be observed that making the prediction horizon too long can be inefficient. This is because the future predicted outputs that are taken into account during the optimization, are no longer beneficial. Furthermore, a longer prediction horizon requires higher computation load. Since there is no definite rule to determine the "best" prediction horizon, a simulation tool like the one proposed in this work constitutes an ideal option to tune the MPC parameters.

The results in Figure 7.6 also show that for these picking/dropping tasks, the non-fixed priority outperforms the fixed priority regardless of the choice of prediction horizon. The non-fixed priority is more adaptive to the real-time situation compared to the fixed priority. Thus, it handles the dynamic changes of the transport system in a better way. However, it can be observed that the difference is quite small. This suggests that the prediction horizon has more influence on the completion time com-
It is to be noted that not all simulations are finished faster than the completion time, especially for $N_p = 9$. Figure 7.7 shows the ratio between the time needed to finish the simulation and $t_{\text{complete}}$, denoted as $\tau$.

If $\tau < 1$, it means the simulation time is faster than the completion time and vice versa. Typically, it is desirable that $\tau \leq 1$ because simulation results can be obtained quickly. Also, if $\tau \leq 1$, experiments can be conducted.

It can be observed from Figure 7.7, that if the number of robots increases, the ratio also increases. This is because MPC has to compute control signals for more robots. Similarly, if prediction horizon becomes longer, $\tau$ also increases. If $\tau > 1$, the simulation takes longer than the real completion time. It takes longer time to obtain the results. For the experimental case study, it is required that $\tau \leq 1$.

**Remark 7.4.** It has been mentioned that for comparison purposes the centralized version is implemented. However, in this particular choice of tasks, the combination that is expected to have the lowest computation load, i.e. $m = 6, N_c = 2, N_p = 3$, does not deliver the expected results. After some time steps of running the algorithm, the PC is out of memory. This shows that the centralized MPC, although is expected to have the best outcome, may be not be beneficial in a large system with a longer set of tasks.

### 7.7.2 Experimental results

For the real-time experiments, a smaller set of tasks is considered. The group of unicycles has to deliver 110 picking/dropping tasks. We choose $m \in \{6, 8, 10, 12\}$. 
Both fixed (7.23) and non-fixed priorities (7.25) are tested. Due to the smaller set of tasks and less number of robots in the system, it is possible to conduct limited experiments using a centralized MPC algorithm.

The following MPC parameters are used in the experiments:

\[
T_s = \frac{1}{6} \text{[s]}, \quad N_p \in \{3, 9, 14, 19\}, \quad N_c = 2, \quad (7.39)
\]

\[
P_{k_1} = 10^4, \quad Q_{k_1} = 10^6, \quad (7.40)
\]

\[
P_{\text{col}} = 10^4, \quad Q'_i = 10^6, \quad R'_i = 10^6, \quad (7.41)
\]

\[
v_{\text{ini}} = 0.01 \text{[m/s]}, \quad v_{\text{ini}} = 0.13 \text{[m/s]}, \quad (7.42)
\]

\[
\omega_{\text{ini}} = -2 \text{[rad/s]}, \quad \omega_{\text{ini}} = 2 \text{[rad/s]}. \quad (7.43)
\]

The parameters given in (7.39)-(7.43) are used both in sequentially decentralized and centralized MPC. The summary of \( t_{\text{complete}} \) from experiments is depicted in Figure 7.8. Examples of videos of the experiments are available at www.youtube.com/adinandra98.

**Remark 7.5.** For the centralized MPC, the options: \( m = 12, N_p \in \{3, 9\} \) and \( m \in \{6, 8, 10, 12\}, N_p = 14 \), the data is collected from simulation. Experiments are no longer possible since the time to successfully complete the optimization procedure is longer than one sampling time. This results in undesired real-time behavior. The experimental results confirm that the proposed sequentially decentralized MPC performs well in a real-time situation. The summary in Figure 7.8 suggests that increasing the number of robots tends to shorten the completion time, both for the centralized and sequentially decentralized approaches. This finding had already been noted in the simulations.
Another similarity with the simulation results is regarding the influence of the prediction horizon. The shorter the prediction horizon, the worse performance is expected. During the (real-time) experiments it can be seen that when the prediction horizon is shorter, the movements of the robots are less smooth compared to the experiments with longer prediction horizon. With a longer prediction horizon, the collision avoidance is also handled in a better way. However, the experiments also suggest that if the prediction horizon is too long, for instance \( N_p = 19 \), the completion time increases (see www.youtube.com/adinandra98 for details). In this particular case study, the experimental results suggest that a good prediction horizon choice is either 9 or 14.

For sequentially decentralized MPC, in almost any choice for the prediction horizon, non-fixed priority outperforms fixed priority although with small differences. This suggests that the choice of priority has less influence on the performance compared to the choice of prediction horizon. This finding is similar to the simulation results.

Considering the shorter set of tasks, the PC is able to successfully compute the optimal control signals using a centralized MPC algorithm. The experimental results suggest that centralized MPC has a better performance, i.e. shorter \( t_{\text{complete}} \), compared to sequentially decentralized MPC. This is as expected since the centralized MPC takes into account information from all robots in the optimization procedure. In this way, the planned path in each sampling time can be optimized. During the experiments, it can be observed that when a centralized approach is used, the movements of the robots are more efficient compared to when the sequentially decentralized approach is used (see www.youtube.com/adinandra98 for details).

However, improving performance of the centralized MPC requires a higher computational load. As mentioned in Remark 7.5, the experiments using a centralized MPC can only be conducted in limited scenarios, i.e. this approach can only handle 10 robots with a prediction horizon not longer than 9. Above 10 robots, the centralized MPC fails to deliver the required control signals. Furthermore, as discussed in Section 7.7.1, for the longer set of tasks, the centralized MPC fails to deliver any results. Thus, it can be concluded that for real-time experiment with long sets of tasks, the centralized MPC is not a good option.

**Remark 7.6.** The results presented in Figures 7.6 and 7.8 do not include variation in any of the scalar weights, i.e. \( P_R, Q_i, P_{\text{col}}, Q^j_i, R^j_i \), the control horizon \( N_c \), as well as different minimum/maximum \( v_i \) and \( \omega_i \). By varying the penalty terms, a different tracking reference behavior can be obtained that results in different completion time. By making the control horizon longer, a performance improvement can be expected. However, similar to the choices of \( N_p \), this results in higher computational load. In this simulation and experimental case study, \( N_c = 2 \) seems to be a good option.

Different \( v_{\min}, v_{\max} \) and \( \omega_{\min}, \omega_{\min} \) choices also affect the completion time. If \( v_{\min} \) is made smaller and \( v_{\max} \) is made higher, the completion time is expected to be shorter and vice versa. This is because the robots are allowed to move in a larger speed range. The choice of sampling time \( T_s \) is also an important factor. Larger \( T_s \) gives more time for the optimization procedure and allows a higher computational load. However, this may not be applicable in a real-time experiment where the choice of the sampling time is constrained by the hardware specification.
Chapter 7. MPC for a group of unicycles

7.8 Comparison with the hierarchical control approach

This section describes the relevance of the MPC algorithm and the hierarchical control approach presented in Chapters 4 and 5. A performance comparison of all control algorithms proposed in this thesis is given.

7.8.1 Relevance to the hierarchical control approach

The proposed sequentially decentralized (and centralized) MPC presented in Sections 7.3 - 7.3 can be seen as a way to combine two or more layers in the hierarchical concept. In the hierarchical concept, the trajectory tracking and collision avoidance problems are solved in two separate layers. In the proposed MPC algorithm, the problems are solved as one problem represented by the cost function (although it is possible to construct a hierarchical MPC). In this way, there is no more freedom to separately design the trajectory tracking controller or the collision avoidance algorithm.

Furthermore, comparing the MPC cost function for trajectory tracking given in (7.13)-(7.15) and the performance measure formulated in (5.1) shows that there is a similarity in the use of \((x_{ri} - x_i)\) and \((y_{ri} - y_i)\), i.e. the Cartesian position errors. Since both the cost function and the performance measure are used as indicators of how good the control algorithm is, this shows that both MPC and hierarchical control approaches try to minimize the position errors. This is also supported by the fact that both control approaches consider a trajectory tracking problem.

In addition, the cost function for avoiding collisions between the robots formulated in (7.18)-(7.19) has a similar functionality as the APF given in (4.19)-(4.21). In both cases, it results in a high value when there are two or more robots that are close to each other. This shows another similarity between the high-/low-level control and the MPC approach.

7.8.2 Performance comparison for the automated warehouse case study

In this section, a performance comparison between the low-level, high-level control and the MPC algorithms in terms of \(t_{\text{total}}\) is given. First, the simulation case study is investigated.

In the simulation case study, the low-level controller uses the control parameters given in (4.24)-(4.26), the high-level controller uses the control parameters described in Section 5.3.1, and the MPC uses the control parameters given in (7.34)-(7.38).

The following abbreviations are used for the low- and high-level control simulation results: SP-HLC: single-path, high-level control; MP-HLC: multiple-paths, high-level control; SP-LLC: single-path, low-level control; MP-LLC-LN: multiple-paths, low-level control, low number priority; MP-LLC-LH: multiple-paths, low-level control, left-hand priority; MP-LLC-RH: multiple-paths, low-level control, right-hand priority. The
Comparison with the hierarchical control approach

The conveyor system is identified by CS. Figure 7.9 recaps the summary of $t_{\text{complete}}$ from the simulations.

![Graph showing comparison of $t_{\text{complete}}$](image)

**Figure 7.9:** Comparison of $t_{\text{complete}}$ from simulation results using the low-level, high-level, and MPC algorithms. The highest value means the transport system suffers from deadlock.

The summary presented in Figure 7.9 suggests that the high-level controller outperforms the low-level controller and the MPC algorithm. This is caused by two main reasons. Firstly, the high-level controller uses information from all robots in determining the optimal reference forward velocity profiles, which serve as the control input, for each robot. Secondly, the high-level controller uses a kind of negotiation procedure to compute the velocity profiles so that the waiting time can be minimized. In this way, the high-level controller delivers the best result.

Both the low-level controller and sequentially decentralized MPC only use limited information from other robots in computing the control signals. The low-level controller only uses information from the robots in the neighborhood, while the sequentially decentralized MPC only uses information from the robot whose optimization has been completed earlier. The use of limited information yields worse performance, indicated by a longer $t_{\text{complete}}$. Furthermore, for any number of robots chosen, the sequentially decentralized MPC outperforms the low-level control algorithm. This is because the MPC algorithm uses point-to-point movements so that it may take shorter paths compared to the fixed paths that are assigned to the low-level control.

As mentioned in Remark 7.4 for the simulation setting, i.e. long set of tasks, the centralized MPC cannot deliver any result. Thus, its performance cannot be compared to other control algorithms. However, since centralized MPC uses information from all robots, it can be argued that, similar if not better performance compared to the high-level control should be obtained.

Furthermore, a second comparison is provided using experimental results. For the experiments, the low-level controller uses the parameters given in (4.27)-(4.29),
the high-level controller uses the parameters as described in Section 5.3.2, and the MPC uses the parameters given in (7.39)-(7.43). Figure 7.10 shows the summary of $t_{\text{complete}}$ from the experiments.

![Figure 7.10: Comparison of $t_{\text{complete}}$ from experimental results using the low-level, high-level, and MPC algorithms.](image)

The results in Figure 7.10 show that the centralized MPC outperforms the other control algorithms. Similar to the performance of the high-level controller in the simulation case study, this result is influenced by the use of information from all robots when computing the control signals. This fact suggests that if the centralized MPC is able to deliver results in the simulation case study, its performance will be closer to the performance of the high-level controller. If the prediction horizon is chosen properly, i.e., 9 or 14, in this particular experimental case study, the sequentially decentralized MPC outperforms the low-level controller. The reason is similar to the finding from the simulation results.

Both summaries shown in Figure 7.9 and 7.10 illustrate how the trajectory tracking and collision avoidance problem for a group of unicycle mobile robots, applied for transportation system of an automated warehouse, can be solved using different control approaches. One important message is that, regardless of the choice, using more information from other robots when computing the control signals can improve performance in terms of completion time. However, as a trade-off this may require a higher computational load and larger communication bandwidth, or less robustness against perturbations.
7.9 Conclusions

In this chapter a sequentially decentralized MPC algorithm to regulate a group of mobile robots is proposed. Focusing on the practical aspect of the algorithm, the controller is successfully validated in simulations and experiments with a large number of robots using transportation system of an automated warehouse as a case study. The validation shows that, increasing a number of robots combined with a proper prediction horizon choice, increase the performance of the transport system. This is indicated by a shorter completion time.

For comparison purposes, a centralized MPC algorithm is also implemented. The comparison shows that the centralized MPC can perform better than the sequentially decentralized MPC. However, the centralized MPC can only be applied in a short set of tasks and for limited number of robots.

This chapter also gives the relevance of the MPC algorithm with the hierarchical control approach. In the hierarchical control approach, the trajectory tracking and collision avoidance problems are solved as two separate problems that can be shifted to different control layers. In MPC, these issues are solved as one unique problem represented in a single cost function.

From the perspective of the information used to compute the control signals, the simulation and experimental results suggest that regardless the control algorithm chosen, more information from the robots can improve the performance. However, as a trade off, this requires higher computational load in MPC and less robust against perturbation in the hierarchical approach.

In the next section, important findings and conclusions of the thesis are presented. Recommendations for future works are given.
This chapter recapitulates the main ideas of the thesis. The conclusions of this thesis are summarized and recommendations for future work are presented.

8.1 Conclusions

In this thesis, the problem of coordination control for a group of mobile robots has been considered. As stated in Section 1.3, there is ample room to exploit the benefits of multi mobile robotic systems. Tools to compare the performance of different coordination control algorithms, mathematically or experimentally, are lacking. Furthermore, transport systems of future automated warehouses provide an ideal environment to validate coordination control algorithms in a large and complex situation. Consequently, the main objective of this thesis as stated in Section 1.3 is formulated as "To perform analysis on coordination control algorithms for a group of mobile robots." This main goal is further divided into three sub-objectives: formulate coordination control algorithms based on a hierarchical control approach, develop a framework to facilitate validation of different control algorithms, and conduct performance analysis and comparison of the control algorithms using the transportation systems of an automated warehouse as a case study.

The main results of this thesis are twofold. The first is a generic three-layer coordination control architecture which is able to accommodate different control strategies implementation as well as enlarging the scale of the system. The second is a performance analysis and comparison of different coordination control algorithms which are designed both at the motion planning and motion execution of the robots. These main results are further elaborated as follows.
In Chapter 3, to answer the need for a framework to facilitate validation of coordination control algorithms, an experimental setup equipped with modular software is presented. The setup/framework allows validation of different coordination control algorithms in a short time. The framework handles addition or removal of robots in the system easily. The setup/framework has been successfully used to provide the necessary evaluation of coordination control algorithms proposed in Chapter 4, 5, 6, and 7 respectively. This shows how the setup is flexible to system changes and requirements. In addition, the setup has also been used to validate control algorithms for a group of mobile robots in different applications such as coordination control that considers delays in sharing information between the robots.

In Section 2.4, a general three-layer hierarchical control architecture is presented. The architecture expands the concept of trajectory tracking control for a single robot. The architecture consists of a motion planner (high-level control) to generate the reference trajectories, a motion executor (low-level control) to accurately track the references, and an adjustable layer to accommodate any functionality that can be shared or shifted between the planner and executor layers. Two algorithms are implemented in both planner and executor layers. A collision avoidance algorithm is embedded in the adjustable layer. For validation, the control algorithms are used to regulate a group of mobile robots that realize the transportation system of an automated warehouse.

In the first implementation, presented in Chapter 4, the motion planner assigns references trajectories that are not collisions-free. The adjustable layer is unified with the motion executor. The low-level controller consists of a trajectory tracking controller and a local collision avoidance which is based on a combination of penalty function concept and artificial potential fields (APF). Simulation and experimental results show that the low-level controller achieves flexibility and scalability of the transportation system as indicated by the ability of handling different numbers of robots in the system. Using completion time as indicator, the low-level controller can achieve similar/better performance compared to the conventional conveyor transport system. Furthermore, simulation and experimental results also show the robustness of the low-level controller against perturbations in the system. Even when there is a robot that blocks other robot’s paths, the transportation system is still operational although with lower throughput. This property cannot be achieved if conveyor systems are used.

For the second implementation, presented in Chapter 5, the adjustable layer is unified with the motion planner. The motion executor consists only of trajectory tracking controllers. The high-level controller solves the collision avoidance problem at the level of generating the reference trajectories by dividing the paths into segments. Based on a prediction method, the movements of the robots are coordinated so that no robots occupy the same segments at the same time. In this way, collision between robots can be avoided. The task of the motion executor is to accurately track collision free reference trajectories. Simulation and experimental results show that the high-level control achieves flexibility and scalability similar to the low-level control. The high-level control also achieves similar/better performance compared to the conveyor systems.
The performance comparison of the high- and low-level coordination control, explained in Section 5.4.1, shows that the high-level control outperforms the low-level control. One of the reasons is because the high-level control uses information from all robots in predicting the correct arrival times of the robots at each segment. The low-level control only uses information from the robots in the neighborhood of each robot. However, since high-level control uses a prediction method, it is less robust against perturbations. When there is a robot that blocks other robots' paths, the high-level control cannot deliver the collision-free trajectories. Although the completion time of the low-level control is longer than the high-level control, it has a better robustness against perturbations in the systems.

A cost analysis of using a group of mobile robots is presented in Section 5.4.2. Based on the normalized cost of a conveyor system, it is shown that the cost of realizing the transportation system of an automated warehouse using a group of mobile robots is acceptable compared to the cost of a conveyor system. Although not all components of both alternatives are taken into account, the cost analysis gives an important indication in the early design phase of which number of robots and control strategies are the "best" with respect to the available budget.

As a complement to the hierarchical control approach, a coordination control algorithm based on Model Predictive Control (MPC) is implemented. The focus is on the real-time aspects and the ability to regulate a large number of mobile robots. A sequentially decentralized MPC, i.e. a single MPC computes the control signals for the robots where priority rules are used to determine which robots are handled earlier in the optimization procedure, is proposed. The nonlinear optimization is solved using a combination of steepest descent and line search techniques. The steepest descent is convenient from the practical point of view since it only requires the first derivative of the cost function. The line search prohibits exhaustive search in finding the optimal solution. For comparison purposes, centralized MPC, i.e. a single MPC computes the control signals for the robots simultaneously, is also implemented.

Using a similar case study as in the hierarchical control approach, simulation and experimental results in Section 7.7 show that the MPC can handle increments and decrements in the number of robots in the transportation system. This shows that the sequentially decentralized MPC achieves flexibility and scalability. However, since the nonlinear optimization procedure has to be completed within one sampling time, the variation of the number of robots is limited compared to the hierarchical approach. The analysis of the completion time suggests that similar/better performance as the conveyor systems can be achieved. In a limited choice of tasks and number of robots, the centralized MPC outperforms the sequentially decentralized MPC. One of the reasons is because the centralized MPC uses information from all robots in computing the optimal control signals. Thus, a shorter completion time is expected compared to the sequentially decentralized MPC. However, the centralized MPC requires a very intensive computational load and works only for a limited number of robots.

The performance comparison of hierarchical control and MPC approaches suggests that using either a high-level controller or centralized MPC tends to give a shorter completion time. The two algorithms have a similarity in the use of information from all robots when computing the control signals. However, as trade-offs, they require
higher computing power in the case of centralized MPC and less robustness in the case of high-level control. On the other hand, although it results in a longer completion time, the low-level control is more robust against perturbations, while the sequentially decentralized MPC does not require an intensive computational load. The performance comparison presented in this thesis can be used as a tool to decide which control algorithm is the best according to the needs and available resources. For example, if perturbations are assumed to be a rare occurrence, the high-level control should be chosen over the low-level control.

In a different application, presented in Chapter 6, the hierarchical control architecture is applied to solve the problem of simultaneous individual trajectory tracking and formation keeping for a group of mobile robots. A control algorithm that solves the problem is designed using dynamic feedback linearization. By introducing coupling between the robots, tracking and formation keeping can be achieved simultaneously. The stability of the controller is analyzed using a theorem of interconnected systems. The analysis of the experimental results using root-mean-square error indicators shows that the communication topology has more influence on the performance compared to the initial conditions or perturbations. More information sharing between the robots tends to increase formation keeping performance indicated by small formation keeping errors. However, the opposite result is obtained for the individual trajectory tracking performance. In addition, more information sharing also requires a larger communication bandwidth, which can be a disadvantage. For the overall performance, if both individual tracking and formation keeping are equally weighted, the best result can be obtained without the need to share all information between the robots.

It is to be noted that the performance analysis is conducted using a certain set of control parameters for each control algorithm. The simulation and experimental results presented in this thesis show some trends in the performances of the coordination control algorithms under several requirements and assumptions. Although different scenarios are investigated, not all possibilities are covered.

### 8.2 Recommendations

The proposed control algorithms in this thesis are all implemented in the setup/framework in a centralized way (see Figure 3.8), i.e. a single PC computes the control signals of all robots, including a localization process to determine the position and orientation of the robots. For experiments, this way of implementation has a disadvantage when the localization process fails to deliver the correct measurements. If the position and orientation (poses) measurement is wrong, for example because of a lighting problem at the camera or a delay in delivering the results, the control algorithm will fail to deliver the correct signals. Another disadvantage is the dependency on the Bluetooth communication protocol. When the delay in sending the control signals to the robots is too high, the robots are no longer using the correct signals. To overcome these disadvantages, in the future a decentralized/distributed implementation needs to be considered. To overcome the limited computing power of the e-puck, a combination of centralized-
decentralized computation can be used. While the PC still computes the necessary signals, locally each robot also computes its poses by using the odometry, as well as the control signals. An interpolation method can be used to compute the final poses and control signals.

The performance comparison of the coordination control algorithms presented in Chapters 4, 5, and 7 is given using a certain set of control parameters and one automated warehouse layout. As mentioned in Remarks 4.3 and 7.6, and Section 5.4.3, a change in the control parameters can yield a different performance. Furthermore, this thesis provides a framework that allows a comparison to be made by simulation and experiments. Although performance comparison results are obtained, this thesis is still lacking a formal mathematical analysis. Dependency in simulation or experiments becomes a big disadvantage if a wide range of control parameters needs to be investigated. The results that cover all possible options and scenarios cannot be obtained in a short time. For future research, the availability of a formal mathematical tool that is able to predict accurately the performance of the coordination control algorithms is crucial. The formal tool can reduce the time to evaluate the performance, even if there are many control parameters and scenarios to be investigated. It will also help combining the advantages of different control algorithms. In this thesis, there is no attempt to combine the advantages of the control algorithms proposed in Chapters 4, 5, and 7.

The proposed control algorithm presented in Chapter 6 has a limitation in the forward velocity, i.e. it is not allowed to be zero. In an application that requires the robot to stop, e.g. transport in an automated warehouse, this limitation becomes a disadvantage. Therefore, the problem of simultaneous trajectory tracking and formation keeping needs to be solved using other control approaches, e.g. using a Lyapunov based approach. Using the indicators proposed in Chapter 6, the performance of the “new” control algorithm needs to be investigated to see if similar conclusions as in Chapter 6 regarding the influence of the communication topology are obtained. Specific for the transport system of an automated warehouse case study, there is a need to investigate the influence of other motion planning algorithms, e.g. algorithms for task division among the robots. Furthermore, in this thesis it is assumed that a robot delivers a product tote that contains one item. In reality, a combination of single and multiple items in one product tote needs to be implemented. Another important factor is battery management. To move around the warehouse, the robots need batteries for their power source. Consequently, after some hours of movement, the batteries need to be charged. During this period, extra control strategies need to be implemented to avoid drastic throughput decrement.
Bibliography


Samson, C., 1995. Control of chained systems application to path following and
time-varying point-stabilization of mobile robots. IEEE Transaction on Automatic
Control 40 (1), 64–77.

Shin, J., Kim, H., 2009. Nonlinear model predictive formation flight. IEEE Trans-
1116–1125.

Siciliano, B., Khatib, O. (Eds.), 2008. Springer Handbook of Robotics. Springer-
Verlag Berlin.

Transaction on Automatic Control 41 (3), 349–357.


Tanner, H., Kumar, V., Pappas, G., 2002a. The effect of feedback and feedforward on
formation iss. In: Proceedings of IEEE International Conference on Robotics and

Tanner, H., Pappas, G., Kumar, V., 2002b. Input-to-state stability on formation graphs.

Tanner, H., Pappas, G., Kumar, V., 2004. Leader-to-formation stability. IEEE Transac-

van den Berg, J., 1999. A literature survey on planning and control of warehousing
systems. IIE Transactions 31, 751–762.


van den Broek, T., van de Wouw, N., Nijmeijer, H., 2009. Formation control of
unicycle mobile robots: A virtual structure approach. In: Proceedings of the
48th IEEE Conference on Decision and Control/28th Chinese Control Conference
(CDC/CCC). pp. 8328–8333.

Vis, I., 2006. Survey of research on the design and control of automated guided

Voos, H., 2009. Model predictive collaborative motion planning and control of mo-
bile robots including safety aspects. In: Proceeding of International Coference on

Robotics and Autonomous Systems 23, 125–152.

pp. 1199–1204.


# Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPC</td>
<td>Model Predictive Control</td>
</tr>
<tr>
<td>APF</td>
<td>Artificial Potential Function</td>
</tr>
<tr>
<td>CS</td>
<td>Conveyor systems</td>
</tr>
<tr>
<td>SP-LLC</td>
<td>Single-path, low-level control</td>
</tr>
<tr>
<td>MP-LLC-LN</td>
<td>Multiple-paths, low-level control, low number priority</td>
</tr>
<tr>
<td>MP-LLC-LH</td>
<td>Multiple-paths, low-level control, left-hand priority</td>
</tr>
<tr>
<td>MP-LLC-RH</td>
<td>Multiple-paths, low-level control, right-hand priority</td>
</tr>
<tr>
<td>SP-HLC</td>
<td>Single path, high-level control</td>
</tr>
<tr>
<td>MP-HLC</td>
<td>Multiple-paths, high-level control</td>
</tr>
</tbody>
</table>
Samenvatting

In het laatste decennium hebben robotsystemen de menselijke wereld meer bin-
nengedrongen dan een mens zich kan voorstellen. In het bijzonder zijn dit multi-
mobiele robotsystemen, wat te wijten is aan het feit dat door de inzet van een
grote groep mobiele robots het totale systeem een hoge redundantie heeft en de
mogelijkheid biedt tot het afhandelen van meer complexe taken. Een groep mobiele
robots verhoogt de robuustheid tegen verstoringen en biedt flexibiliteit tegen veran-
deringen in het systeem. Om tot een volledig autonome werking te komen, wordt het
regelalgoritme om de robots te coördineren steeds belangrijker en doorslaggevend.

De belangrijkste innovaties van dit proefschrift zijn een algemene drielaags
hiërarchische coördinatie-regelarchitectuur en de prestatie-evaluatie van de
coördinatie-regelalgoritmes voor een groep van mobiele robots.

In het eerste deel van dit proefschrift wordt een modulair raamwerk voor simu-
latie en experimenten van de coördinatie-regeling van mobiele robots gepresenteerd.
Alle verbindingen tussen de modules volgen het abonnee/uitgever paradigma voor
de uitgewisselde data, dat wil zeggen, een component publiceert data en andere
cOMPONENTEN kunnen zich abonneren op deze data. De data wordt geïdentificeerd
door een tijdschrift en actuele datapaketten heten nummers van dit tijdschrift. Door
middel van simulatie en experimenten is beschreven hoe de modulariteit van het
raamwerk eenvoudige aanpassingen van systeemconfiguraties, ontwerpparameters
en regelalgoritmen mogelijk maakt.

In het tweede deel wordt een hiërarchische coördinatie-regeling van mobiele robots
gepresenteerd. De hiërarchie bestaat uit drie lagen: een high-level regeling voor be-
wegingsplanning van de robots, een low-level regeling voor bewegingsuitvoering, en
een flexibele laag om te voorzien in verschuivingen van verantwoordelijkheden. Een
voordeel van een hierarchische aanpak is de isolatie van een regelaarontwerp in elke
laag. Veranderingen in de regelstrategie van één laag vereist niet noodzakelijkerwijs
een aanpassing in andere lagen.

De voorgestelde regelalgoritmen worden gebruikt om een groep mobiele
eenwielerrobots te coördineren, welke het transportssysteem van een automatisch
magazijn realiseert. Met gebruik van het raamwerk zijn een aantal simulaties en real-
time experimenten uitgevoerd, beide met verschillende ontwerpparameters. Ver-
volgens is een prestatie-analyse van de resultaten uitgevoerd. Er wordt aange-
toond dat de voorgestelde algoritmen flexibel zijn tegen systeemveranderingen en
schaalbaar zijn tegen de variatie in eisen van een magazijn. De algoritmen zijn
robuust tegen verstoringen in de transportsystemen. Het is gebleken dat een al-
goritme, welke aangeduid is als high-level regelaar, met een hogere doorvoersnel-
heid meer informatie-uitwisseling tussen de robots vereist. Dit algoritme is minder
robust tegen fouten ten opzichte van een algoritme, welke aangeduid is als low-level regelaar en een lagere doorvoersnelheid oplevert. Bovendien is een kostenanalyse van de voorgestelde regelalgoritmen gegeven. Het is onderzocht of de kosten van het realiseren van een transport met een groep mobiele robots, gecoördineerd met het voorgestelde algoritme, een vergelijkbare prestatie-kosten ratio heeft in vergelijking met transportbanden. Deze conclusie is zeer veelbelovend indachtig met het feit dat het voorgestelde regelalgoritme niet noodzakelijk het optimale is.

In het derde deel, wordt het probleem van het gelijktijdig volgen van individuele referentiepaden en het behoud van formatie voor een groep van mobiele robots onderzocht. Het regelalgoritme is ontwikkeld met behulp van dynamisch teruggekoppelde lineairisatie. Het stabiliteitsbewijs wordt geanalyseerd met behulp van de stelling van gekoppelde systemen. Een root-mean-square-achtige indicator analyseert de afweging tussen het individueel volgen en het behoud van formatie, alsmede de invloed van de communicatietopologie. Uit de analyse van de real-time experimentele resultaten blijkt dat het beste formatiebehoud wordt verkregen wanneer alle robots communiceren. Als afweging vereist dit een grotere communicatiebandbreedte en levert dit grote individuele volgfouten. Bovendien blijkt uit de analyse dat, voor het tegelijk verkrijgen van optimale individuele volging en behoud van formatie, er minder noodzaak is om informatie tussen de robots te delen.

In het vierde deel, als aanvulling op de hiërarchische regelstrategie, is een coördinatieregelalgoritme ontworpen welke gebaseerd is op Model Predictive Control (MPC). Gericht op de praktische aspecten is een sequentieel gedecentraliseerde MPC-algoritme ontworpen. Dat wil zeggen, een enkel MPC-algoritme berekent de stuursignalen van alle robots waar de voorrangsgewichten worden gebruikt om te bepalen welke robots eerder behandeld worden in de optimalisatieprocedure. Met behulp van een soortgelijke, geautomatiseerde magazijnomgeving, wordt het sequentieel gedecentraliseerde MPC-algoritme gevalideerd. Ter vergelijking is een gecentraliseerd MPC-algoritme ontworpen. Dat wil zeggen, een enkel MPC-algoritme berekent de stuursignalen van alle robots tegelijkertijd. Met doorlooptijd als indicator zijn de invloeden van de MPC parameters onderzocht. Het blijkt dat het gecentraliseerde MPC-algoritme beter is dan het sequentieel gedecentraliseerde MPC-algoritme, echter, ten koste van een hoge rekentijd en een beperkt aantal robots in een real-time toepassing. De relevantie en de vergelijking van prestaties tussen de MPC-methode en de hiërarchische regelstrategie zijn gepresenteerd. Het blijkt dat, onafhankelijk van de keuze van het regelalgoritme, met betrekking tot de doorlooptijd, betere resultaten worden bereikt wanneer meer informatie wordt gebruikt om de regelaar te berekenen. Als afweging wordt het systeem minder robust tegen verstoringen en is een grotere communicatiebandbreedte vereist.
Acknowledgment

After more than 4 years battling with textbooks, papers, experimental setup, and laptop, finally I can finish this book which marks the completion of my PhD. Finishing, more importantly enjoying, this PhD would not be possible without the contributions of a very large number of people. To all of them, I would like to express my gratitude.

First and foremost, I would like to thank my promotor Henk Nijmeijer for the giving me the opportunity of having a PhD position his Dynamic and Control Group. His constant support, encouragement, confidence, guidance, and long-lasting patience have helped me survive during the good and bad time of my PhD. I have felt grateful and privileged to be able to work with him.

I would like to thank the committee members of my promotion, Henri Huijberts (Queen Mary University of London), Bart De Schutter (TU Delft), Pieter Jonker (TU Delft/TU/e), Ivo Adan (TU/e), and Jacques Verriet (ESI) for the time and efforts which they spent to thoroughly read my thesis and giving me such valuable feedback and comments.

I would also like to thank Dragan Kostić for being my co-promotor in two years period. Although we cannot finish the collaboration, I really enjoyed the productive time in discussion or sitting in the lab that results in some publications. I also want to thank Jurjen Caarls for the time, effort, and a nice piece of work on the experimental setup so that I can conduct simulation and experiment in an easy way. To Alejandro, my thanks for all discussion about unicycle mobile robot and formation control, proof-reading of my thesis, and providing information about life in Eindhoven during my first PhD year. My thanks to Thijs van den Broek for the knowledge about e-puck and the setup at Vision Lab. I also want to thank Anna Sadowska for the discussion about theoretical aspect of formation control.

To Roel Pieters, thank you for being a very nice roommate for 3.5 years. Thank you for creating a wonderful working environment, the time for discussion, proof reading my thesis, and borrowing me the cymbals. To Jonatan, thank you for all serious and relaxing time we spend and the help on proof reading the thesis. I really enjoyed all of them, especially the biking time although we did not manage to bike from Eindhoven to Amsterdam. To Alper, thank you for the (sometimes not so) funny jokes. To Zhenyu, Mark, and Carlos, thank you for all nice conversation we had, especially during the lunch. To Nick and Robert, thank you for the basketball time and we should spend more time on court. I would also like to thank all other colleagues at Dynamics and Control, Control Systems Technology, and Networked Control Systems groups for providing a competitive academic and research atmosphere.
I would also like to thank Geertje Janssen-Dols, Petra Aspers, and Lia Neervort at the secretariat for helping me handling all administration issues, and also to Hetty van Neerven and Thea Weijers at DPO for all the help in arranging the resident permit.

To all Indonesian friends in Eindhoven, who are too many to be mentioned individually, thank you for the wonderful time we spend together. Your presence and hospitality has helped me and my family feel less lonely in this city far away from Indonesia. Without all you, passing 4.5 years in Eindhoven will be felt like it is more than it should be. To PPI/e basketball team and Angklung Eindhoven, thank you for letting me became a member. I really had a good time in all competitions and performances.

None of this would have been possible without the everlasting love, prayers, and support of my mother and father at Banteng, my sister at Bekasi, and my parents in law at Suronatan. Knowing that everything is okay back there always meant a lot to me. This is one achievement that your son/brother/son in law has made and I hope all of you can be proud of it. May Allah bless you all.

To my son Arkan Aluna, it is unbelievable to know how fast you grow and how much new things you have learned. It is just like yesterday that I hold you in my arm and now you are already a meter tall. Your dad's time in school is over and it is your time now. Go spread your wings in this world.

Finally, to the one that accompany me since the first day I started my PhD. She who always stand besides me throughout the good and bad time. For my lovely and beautiful wife Lukluk Noor Affah, no words will suffice. Thank you for everything and let's continue our journey altogether to the future.

Alhamdulillahirabbil’aalamiin

Sisdarmanto Adinandra,
August, 2012.
Curriculum Vitae

Sisdarmanto Adinandra was born on November 19, 1979 in Yogyakarta, Indonesia. He completed the Mathematics and Natural Science Program in 1998 at 3 Senior High School in Yogyakarta. Afterwards, he studied Electrical Engineering, with specialization in Control and Instrumentation, at University of Gadjah Mada also in Yogyakarta where he graduated cum laude in 2002.

In September 2005, he received StuNed scholarship from the Dutch Government via Nuffic Neso Indonesia to join the master program Systems and Control at Delft University Technology (TUD) in Delft, The Netherlands. Upon his graduation in August 2007, he did an internship at General Electric Wind Energy Salzbergen, Germany and carried out a thesis entitled "Control for Mechanical Braking of Wind Turbines".

Since February 2008, he started a PhD Project in the Dynamics and Control Group at Department of Mechanical Engineering of the Eindhoven University of Technology (TU/e) in Eindhoven, The Netherlands. The project focuses on flexible transportation in automated warehouses, which is part of the FALCON (Flexible and Autonomous Logistic CONcept) project. The results of his PhD Project, entitled "Hierarchical Coordination Control of Mobile Robots", are presented in this thesis.