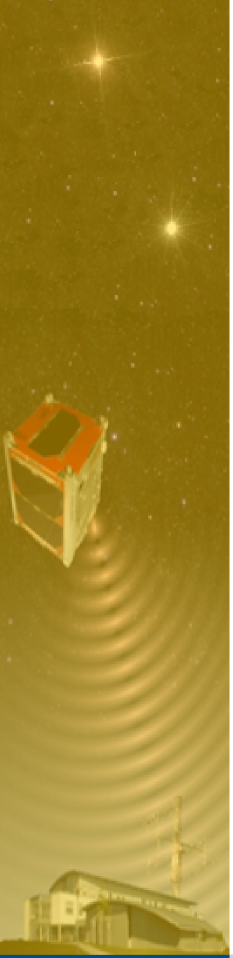


Institut für Informatik
Lehrstuhl für Robotik und Telematik
Prof. Dr. K. Schilling

Würzburger Forschungsberichte in Robotik und Telematik

Uni Wuerzburg Research Notes in Robotics and Telematics

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Band 3

Martin Saska

Trajectory planning and optimal control for formations of autonomous robots

Die Schriftenreihe

wird vom Lehrstuhl für Informatik VII: Robotik und Telematik der Universität Würzburg herausgegeben und präsentiert innovative Forschung aus den Bereichen der Robotik und der Telematik.

Die Kombination fortgeschrittener Informationsverarbeitungsmethoden mit Verfahren der Regelungstechnik eröffnet hier interessante Forschungs- und Anwendungsperspektiven. Es werden dabei folgende interdisziplinäre Aufgabenschwerpunkte bearbeitet:

- Robotik und Mechatronik: Kombination von Informatik, Elektronik, Mechanik, Sensorik, Regelungs- und Steuerungstechnik, um Roboter adaptiv und flexibel ihrer Arbeitsumgebung anzupassen.
- Telematik: Integration von Telekommunikation, Informatik und Steuerungstechnik, um Dienstleistungen an entfernten Standorten zu erbringen.

Anwendungsschwerpunkte sind u.a. mobile Roboter, Tele-Robotik, Raumfahrtsysteme und Medizin-Robotik.

Lehrstuhl Informatik VII
Robotik und Telematik
Am Hubland
D-97074 Wuerzburg

Tel.: +49 (0) 931 - 31 - 86678
Fax: +49 (0) 931 - 31 - 86679

schi@informatik.uni-wuerzburg.de
http://www7.informatik.uni-wuerzburg.de

ISSN (Internet): 1868-7474
ISSN (Print): 1868-7466
eISBN: 978-3-923959-56-3

Zitation dieser Publikation

SASKA, M. (2009). Trajectory planning and optimal control for formations of autonomous robots. Schriftenreihe Würzburger Forschungsberichte in Robotik und Telematik, Band 3. Würzburg: Universität Würzburg.

Würzburger Forschungsberichte in Robotik und Telematik

Uni Würzburg Research Notes in Robotics and Telematics

Herausgeber

Prof. Dr. K. Schilling

Universität Würzburg

Institut für Informatik

Lehrstuhl für Informatik VII: Robotik und Telematik

Am Hubland

D-97074 Würzburg

Tel.: +49-931-31-86647

Fax.: +49-931-31-86679

email: schi@informatik.uni-wuerzburg.de

ISSN (Internet): 1868-7474
ISSN (Print): 1868-7466

Trajectory planning and optimal control for formations of autonomous robots

Dissertation zur Erlangung
des naturwissenschaftlichen Doktorgrades
der Julius-Maximilians-Universität Würzburg

vorgelegt von

Martin Saska

aus

Kladno, Tschechische Republik

Würzburg, November 2009

Eingereicht am: 25.11.2009
bei der Fakultät für Mathematik und Informatik
1. Gutachter: Prof. Dr. Klaus Schilling
2. Gutachter: Prof. Dr. Hans Josef Pesch
Tag der mündlichen Prüfung: 29.11.2010

Acknowledgements

First of all, I would like to express my deepest gratitude to my wife, Romana, whose support gave me strength for writing this dissertation, but mainly during my studies in foreign country. She spent three long years without her husband to enable me to fulfill my ambitions and dreams. She is the love of my life and I want to dedicate this thesis to her.

I would like to thank my Ph.D. advisor Prof. Dr. Klaus Schilling for providing me the opportunity to joint his team. I appreciate the guidance and support he gave me. I would also like to thank Prof. Dr. Hans Josef Pesch for his important comments and discussions during reviewing of this thesis. His suggestions significantly improved scientific quality of my work.

My special thanks go to Martin Hess with whom I shared the office and "party Schrank" in Wuerzburg. I closely cooperated with him for the whole time and he influenced my research results a lot. I also want to thank Martin Macas from Czech Technical University (CTU) for inspirational discussions and scientific cooperation at the beginning of my Ph.D. studies. Besides, I would like to thank colleagues at CTU for helping me in the final period of writing this thesis.

My thanks also go to Prof. Stipanovic for the time he gave me during my stay in his group at the University of Illinois at Urbana-Champaign (UIUC). He provided me numerous suggestions and discussions about my research topics. The three months I spent at UIUC have changed the course of my research significantly. Regarding this stay, I would like to thank all my colleagues from UIUC to help me with all administrative as well as research matters. Namely, I want to thank Juan Mejia for introducing me to the Receding Horizon Control problematic, but mainly for plenty of social activities he invited me for. I thank all Columbian friends who allowed me to be a member of their community.

My Ph.D. studies have been supported by the international doctorate program "Identification, Optimization and Control with Applications in Modern Technologies" within the Elite Network of Bavaria (ENB). This support opened me the possibility of studying in Germany, but furthermore it enabled me participating in numerous international conferences and staying three months at UIUC in United States. I am very proud that I can be member of the big ENB family. I would like to thank the coordinator of the doctorate program, Prof. Dr. Gunter Leugering, and the other professors in the program for their inspiring comments on my work during our meetings and winter/summer schools. I am grateful to my ENB colleagues Nils Altmuller, Michael Blume, Izabella Ferenczi, Tobias Gradl, Falk Hante, Anna von Heusinger, Indra Kurniawan, Marcus von Lossow, Thorsten Spickenreuther, Alexander Thekale, Slobodan Veljovic, StefanWendl and Dr. Erik Kropat. They enlarged my view on research in various disciplines, cooperated with me on research tasks and mainly formed a nice companionship.

At the chair of Computer Science VII: Robotics and Telematics, I would like to express my gratitude to all my colleagues, especially to Laszlo Lemmer, Adam Cseh, Zhongyang Wu, Radu Barza, Pablo Baier, Lakshminarasimhan Srinivasan, Zhihao Xu, Christian Herrmann, Rajesh Shankar Priya, Daniel Eck, Marco Schmidt, Markus Sauer, Florian Zeiger, Dr. Frauke Driewer, Dr. Lei Ma, Dr. Peter Hokayem, Florian Leutert, Kaipeng Sun, Dieter Ziegler, Heidi Frankenberger, Edith Reuter, and Heidi Schaber. I enjoyed day-to-day collaboration and a lot of private activities with them.

Finally, I would like to thank my mother for her unlimited support and help, and all members of my family and my family in-law for their patience.

To my wife, Romana.

Contents

1

Introduction

In this thesis, we present novel approaches for formation driving of nonholonomic robots and optimal trajectory planning to reach a target region. The methods consider a static known map of the environment as well as unknown and dynamic obstacles detected by sensors of the formation. The algorithms are based on leader following techniques, where the formation of car-like robots is maintained in a shape determined by curvilinear coordinates. Beyond this, the general methods of formation driving are specialized and extended for an application of airport snow shoveling. Detailed descriptions of the algorithms complemented by relevant stability and convergence studies will be provided in the following chapters. Furthermore, discussions of the applicability will be verified by various simulations in existing robotic environments and also by a hardware experiment.

1.1 Motivation

The motivation of the research presented in this thesis is the application of multi-robot formations driving in workspaces with dynamic obstacles. We will be focussing on applications requiring variations of distance between the vehicles of the team or even dynamic allocation of the robots to separate sub-formations during the mission. A temporary deformation of the group's shape enforced by the environment will be enabled without any failure of the desired task. We would like to point out applications of *3D cooperative mapping*, *cloud of toxic gas determination*, and *airport snow shoveling* as examples of such scenarios.

The idea of 3D mapping with multiple robots is motivated by utilization of camera sensors, which are interesting due to their wide usage and low price. Two single robots equipped by a monocular camera can form a stereo pair with optional baseline length as is demonstrated in Fig. 1.1. Any other added robot allows to increase the robustness of such a system, while it provides additional information and allows to adjust the established correspondences of features extracted from obtained images. Such a configuration enables to set the optimal distance and viewing angles between cameras with respect to the observed environment and desired properties of the final map. A too narrow baseline generates a less precise map, whereas too wide means

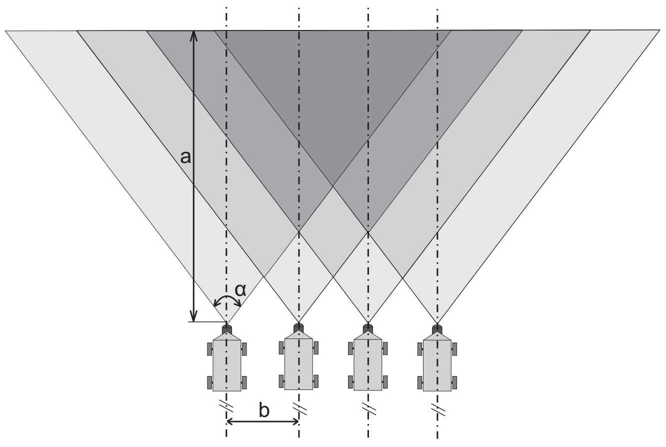


Figure 1.1: Demonstration of 3D mapping using formation of mobile robots. The grey intensity grows with the number of robots sensing the area.

less correspondences, less precise localization and thus lower map quality and reliability. In case of larger groups of robots, one should also consider the advantages of splitting and the subsequent merging of the groups e.g. in case of obstacles dividing the explored environment. This application has been motivated by formation flying of spacecrafts, which is utilized for the visual 3-D mapping, the time-varying gravity field measurements, the magnetosphere and the radiation measurements as well as for long baseline spacecraft interferometers in deep space (for more detailed description of the applications see [175]). By sharing the individual measurements, the resolution obtained using the spacecraft formation is potentially much higher than the resolution of single spacecraft. Additionally, such a system is more reliable and fault tolerant due to possible redundancy of measurement. The initial research in this field can be found in [71; 184] and some more advanced methods are available in [15; 150]. Unfortunately, these techniques applying a virtual structure approach for formation stabilization are hard to modify for the car-like robots due to different spacecraft kinematics.

The second scenario is a part of the protection of civilians around possibly dangerous factories. Nowadays, industrial zones are surrounded by a ring of stationary sensors detecting escape of pollutants that are dangerous mainly to inhabitants in the close neighborhood. For obtaining better information on the position of the poisson, such a system could be supplemented by a group of autonomous mobile measurement unites moving along the borders of the dangerous area. While a single vehicle can only determine if the point of measurement lies in the "forbidden" zone, a formation of several robots can continuously follow the borders keeping one robot inside and one robot outside the cloud. Furthermore, additional vehicles could provide an estimation of the gas concentration gradient, which is important for the future formation movement determination. Similarly as in the previous example, the distance between the rovers can influence the functionality of the system. An approach with more compact formations deals with the problem of keeping the desired threshold of concentration within the formation and on the contrary bigger distances between the robots decrease accuracy of the solution. A proposed schedule for such scenario is depicted in Fig. 1.2.

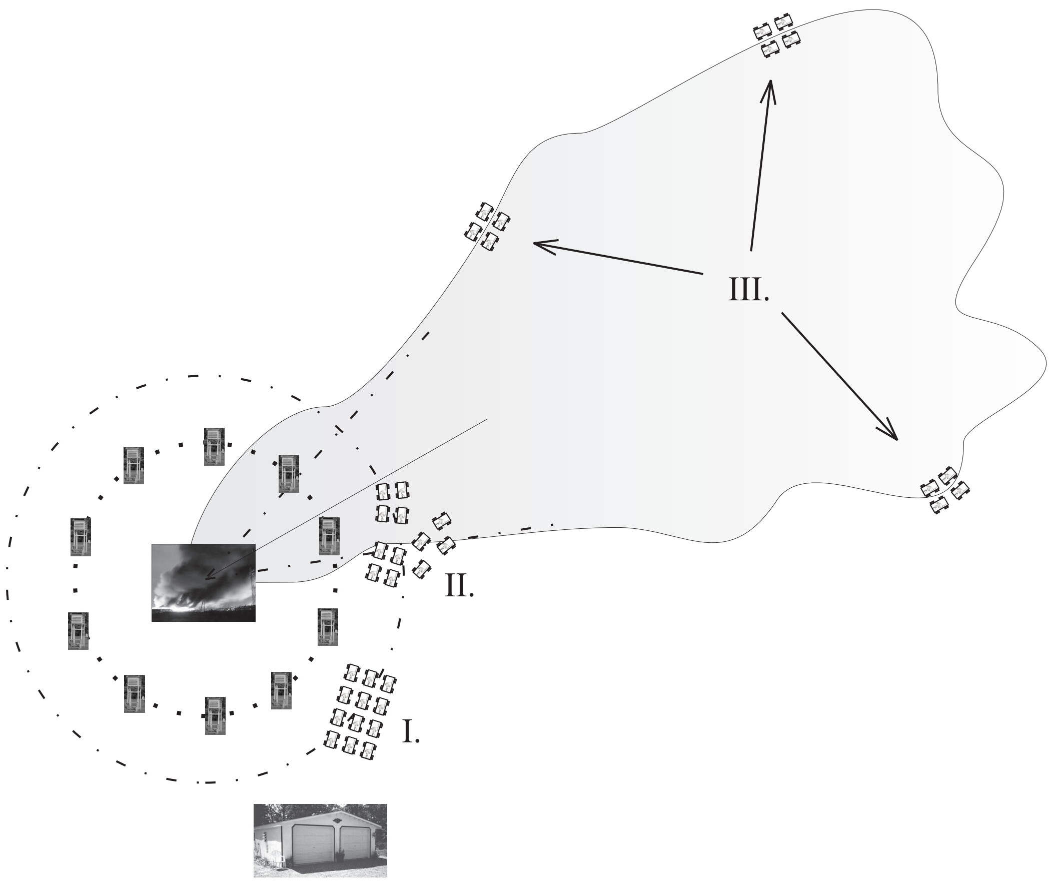


Figure 1.2: Three composed snapshots of gas-cloud position detection scenario.

In the step I., the complete group has to move as fast as possible from a car-shed to the initial position given by the stationary nodes of the system. This is followed by splitting to several sub-formations (denoted by II.) utilized for the measuring in the phase III. The existing approaches relevant to this task are aimed mainly at the source of pollution searching by following the gradient of gas concentration [35; 43; 128; 154]. This problem is simpler in terms of differences between the usual static source and the highly dynamic cloud of gas. The few studies of the environmental hazard perimeter mapping are focused on determination of the desired position of particles representing the robots while the vehicles' dynamics are neglected [74] or on building gas concentration grid-maps in indoor windless environment [114]. A comprehensive survey of existing odor sensing methods is available in [154].

The third scenario, *airport snow shoveling* by groups of autonomous ploughs, has been chosen for a closer investigation in this thesis. The application includes formation stabilization into variable shape depending on the width of runways, splitting and coupling of formations for cleaning smaller auxiliary roads surrounding main runways as well as specific maneuvers as is for example formation turning at the end of blind routes. This project is the reason why we have adapted all methods in this thesis for the kinematics of car-like robots. The snow-ploughs currently used at the airport are ordinary trucks with the same kinematic model. A more detailed description of this application followed by an overview of existing relevant approaches and proposed novel solutions are introduced in Section 5.

1.2 Literature survey

Multi-robot systems have been the subject of major interest over the last decades as can be seen in this full-range survey of initial works [141] and they are still a hot topic in leading robotics and control journals nowadays (see e.g.: [19; 48; 63; 88; 92; 144]). Multi-robot systems are intensively studied to deal with complex tasks, which can be solved faster and more robustly in comparison with utilization of a single powerful robot [143]. The broad research is covering large systems employing 1000's of autonomous robots [37; 41] as well as small groups maintaining precisely defined formations [65; 174].

The large systems can be additionally represented by work published in [144] where collision free trajectories are designed for a 100 robots. The method uses graph and spanning tree representation and is developed for utilization in underground mine environment. In the next example [96], a large swarm of robots is controlled using a hierarchical abstraction, where inter-robot collision avoidance and environment containment are guaranteed by the application of centralized communication architecture. The work published in [140] offers a study of the problem of heterogeneity in teams of more than hundred mobile robots. Relevant practical experiments including exploration, mapping, deployment and detection are described in [80]. In [58], a method for decentralized information exchange between vehicles in large formations is proposed. Furthermore, you can see how the topology of the information flow affects the stability and performance of the system. The work presented in [126] applies a Stochastic Hybrid Automation model for modelling and control of multi-agent population composed of a large number of agents. A probabilistic description of the task allocation as well as distribution of the population over the workspace is considered in this method. Also traffic coordination should be mentioned as an example of common multi-robots application with large number of vehicles. A decentralized approach using traffic rules for control of tens of vehicles is presented in [138]. The method enables dynamic adding and removing of the vehicles and is based only on a local communication which makes the algorithm scalable.

This thesis is focussed on the formation driving of autonomous robots. In the classical literature, formation driving approaches are divided into the three main groups: virtual structure, behavioral techniques, and leader-follower methods.

In the virtual structure, the entire formation is regarded as a single structure where a set of controls for following the desired trajectory of the formation as a rigid body is given to each vehicle [15; 102; 150]. Virtual structures are often applied in spacecraft formation flying due to the ability to maintain the fixed formation accurately [89; 151]. The high precision leads to using this method also in tasks of multi-robot box pushing [112] or load carrying [178] where the object itself naturally forms the shape of the formation. For example in [19], differentially-driven wheeled mobile robots are moving in the virtual structure for ultimate deployment in cooperative payload transport tasks. In [108], formation and maneuver controls for multiple rigid bodies are obtained by decomposition of the group dynamics of the multiple agents into two decoupled systems: the shape system representing internal formation shapes, and the locked system abstracting the overall group maneuver as a complex unit. A time-varying global output-feedback controller that solves simultaneously path tracking and formation stabilization for unicycle-type mobile robots at the torque level is presented in [50]. This work was extended in [47; 48] where the virtual structure based method is introduced for mobile agents with limited sensor range keeping guarantee of no collision between the robots. The last contribution of this team we will mention is a novel interlaced observer employed to estimate the robot velocities of each robot [49]. This approach uses only position measurements of the robot itself and the robots within its communication range.

In behavior based methods, the desired behavior is designated for each agent and the final control is derived as a weighted sum with respect to the importance of each task. The utilization of the behavior techniques is usually inspired by nature, e.g. by flocks of birds [111] or molecules forming crystals [11]. Primarily, the behavior approach has been applied for coordination of multi-robot systems [103; 142]. From these basic techniques, formation driving algorithms have been derived using desired patterns for maintaining of shape of formations [10; 106]. The advantages of the behavioral approaches are an explicit feedback to the formation from each robot and a low bandwidth communication enabled by simple decentralization. Nevertheless, a convergence/stability proof of such a collective behavior is complicated in general and has to be focused on concrete application. Typically, only the equilibria that correspond to simple behaviors (e.g., constant heading [84]) are provable. Therefore, these approaches are not easily adaptable for complicated formations and maneuvers (e.g., time-varying formation). We should mention that the ability of the precise formation keeping is limited mainly for car-like robots which is crucial in the airport snow shoveling being our target application.

In the leader-follower approaches, a robot or even several robots are designated as the leaders. The followers maintain their position in the formation relative to the leader and the state of the leading vehicle needs to be distributed within the team [38; 153]. Utilization of the leader-follower approach is often motivated by the heterogeneity of the multi-robot system with few well equipped vehicles designated as the leaders and several cheap and simple followers [82]. Examples of the approach employing multiple leaders are presented in [64] and [172] where a limited range of communication is applied and the followers are led by their closest neighbors. The movement of a reference point is then propagated to the members of the formation subsequently. The standard scheme with one leader was applied e.g. in [42] or [82]. An improvement of these standard methods is presented in [60] where a hybrid control switching between *Discrete-state formation control* is employed to achieve desired positions and *Continuous-state robot control* to maintain robots in the formation during the leader following task. In [113], Neural Networks are utilized for stabilization of formations of differential drive robots following predefined trajectories in an environment without obstacles. The same task is solved in [31], where the control inputs of robots are forced to satisfy suitable constraints that restrict the set of the leader's possible paths and admissible positions of the followers with respect to the leader. A combination of the virtual structure and the leader-follower methods is presented in [192]. The formation feedback in [192] is applied for stabilization during prescribed formation maneuvers along a desired trajectory. The feedback from the followers to the leader can avoid the breakaway of a robot that is temporarily slowed-down from the formation. On the contrary, one not fully functional follower can delay the complete formation that is unwanted in most of the time-critical applications. As the last example dealing with the leader-follower approach we would like to mention a theoretical work published in [179]. The article offers an interesting study of stability (*Leader-to-Formation Stability*), which is used to characterize how the motion of the leader can affect the motion of the group. Nevertheless, all these results are focused on the following of the leader's trajectory, which is assumed as an input. It is supposed that the trajectory is designed by a human operator or by a standard path planning method modified for requirements of the formation driving. In literature, there is no adequate method providing flexible control inputs for the followers as well as for the leader of the formation responding to the dynamic environment and trying to solve the optimality as well as the stability of the *leader-to-goal* and the *followers-in-formation* tasks together. In this thesis, we rely on the leader-follower method that is most often used in applications of the car-like robots and we consider all the specifications discussed above.

At the end of this survey, we will mention a few application focused papers to show possible utilization of multi-robot systems. An interesting and already existing project of the autonomous formations is the continuous observation of planetary surface using cameras attached to a number of spacecrafts flying in the leader-follower constellation [115]. In the future a swarm of cheap "nano" or "pico" satellites could provide the same performance in cooperative mapping as one well equipped and expensive robot. Additionally, such systems should provide an increase of robustness and some additional features done by possibility of formation size changing [169]. Another broad area of applications includes aircraft formation flying, which is usually motivated by airborne refueling, energy saving from vortex forces or fuel efficiency via induced drag reduction [69; 197]. Systems employing autonomous formation can be found also afloat (autonomous boats maintained in formations under sliding mode in [56]) or under water (submarines in the project: Autonomous oceanographic sampling networks [36]). The mobile oceanographic sensing networks, including underwater vehicles together with buoys and nodes of the stationary network at bottom, are autonomously measuring temperature and currents. The application of the formation driving within this project is presented in [134; 135]. A stable coordination strategy for formation translation, rotation, expansion and contraction is proposed in [134]. The entire group of submarines is considered as a web of virtual leaders that move together like a rigid body. In [135], the method is extended using artificial potentials for gradient climbing missions in which the mobile sensor network seeks for local extremes in the ocean environment. Other frequent applications of formations, which should be mentioned, are collective harvesting [72], military projects for defending vulnerable vehicles by armored robots distributed around (for details see[11]) or automated highway systems (existing methods are summarized in [18]) using the leader-follower approach for reduction of space between the vehicles moving in one lane [176; 177]. Frequently referred applications are also box pushing [181] or load transport [178], where the carried object is too heavy or large for a single robot and the shape of the load determines the structure of the formation. A biologically inspired approach was presented in [193]. The method uses a modified shunting neural network for automated path generation and a self-organization map for multi-robot coordination in cooperative object pushing using swarms of nano-robots. The last example of the applications, which is intensively investigated nowadays, is cooperative localization using formations of autonomous rovers. This task is important for missions in unknown environment without any external positioning system deployed in advance. These can be search and rescue scenarios and especially fire-fighter support by robots in partly collapsed buildings (see e.g. PeLoTe project [51; 166; 168]). The task of cooperative localization was theoretically studied in [129], where a problem of resource allocation is addressed. This approach provides the sensing frequencies, for each sensor on every robot, required in order to maximize the positioning accuracy of the group. This work is extended in [130] by a performance analysis providing upper bound on the robots' expected positioning uncertainty, which is determined as a function of the sensors' noise covariance and relative position measurements.

1.3 Developed methods overview

The core of the thesis is an approach employing Receding Horizon Control (RHC) for the leader trajectory planning to a desired goal area and for the followers stabilization in desired positions behind the virtual leader. The method enables to obtain optimal control inputs for the robots in near future as well as a complete trajectory of the virtual leader to reach the target region in one entire optimization step. The commonly used approaches obtaining optimal control inputs by the simple following of a desired trajectory, that was computed separately beforehand, can only provide a suboptimal solution. Our method can continuously respond to changes in the environment of the robots while the cohesion of the immediate control inputs with direction of the formation movement in future is kept. A similar concept that is used in the virtual leader's case is also applied for the stabilization of followers. Here, the RHC is employed to maintain the desired position of the followers behind the virtual leader as well as to prevent collisions within the team. The approach also enables to avoid collisions with dynamic obstacles that could not be considered in the planned trajectory of the virtual leader. The complete system is designed in a way that the computational tasks are distributed to the appropriate robots with minimal required communication. Such an approach increases robustness of the method in comparison to techniques utilizing a calculating centrum carried by a well equipped robot that is sensitive to failures.

An interesting extension of the proposed structure is a method providing formation stabilization and trajectory planning during complicated maneuvers in limited workspace. To our best knowledge, there is no general solution of the trajectory planning for formations of car-like robots turning on the spot or passing curves with a radius smaller than a minimal turning radius of the robots in the literature. Our method based on switching between two virtual leaders can provide a collision free solution of these problems even in the environment with dynamic obstacles.

The Receding Horizon Control (also known as Model Predictive Control) utilized in this thesis is an optimization based method developed for stabilization of nonlinear systems. This approach can achieve a desired system performance with handling system constraints at the same time. As an appropriate optimization technique used in the frame of RHC for the nonholonomic robots, the Sequential Quadratic Programming (SQP)[1] is recommended. The disadvantage of the SQP technique, which is a generalization of Newton's method, is missing ability to overcome local extremes in the cost function. The solution of this problem could be the utilization of an artificial uncertainty during the optimization process or applying a global optimization method. Unfortunately both approaches increase the computational time necessary for one iteration of the planning loop that needs to be as short as possible to respond to changes in dynamic environment.

The second approach of the formation driving presented in the thesis deals with this problem. This method, based on path planning for the virtual leader of the formation in space of multinominals, was developed to reduce high complexity of the optimization process in the previous algorithm. Furthermore, the design of the method is adjusted for simple utilization of the existing global optimization methods that can give an alternative to the local SQP. From the broad offer of optimization approaches we implemented and compared one deterministic and one stochastic algorithm. Global optimization based on sparse grids modified for robotic applications in [155] was chosen as an example of the deterministic method. The results obtained by this technique were compared with a commonly used stochastic method, Particle Swarm Optimization (PSO).

In the theoretical part of the thesis, the description of these formation driving methods is supplemented by proofs of convergence with specified restriction of practical utilization. Beyond this, two approaches are proposed for better initialization of the optimization methods applicable in both versions of the leader planning. These methods should significantly reduce the high computational complexity of the optimization and partly avoid the problem of multiple local extremes of the cost function. In the first algorithm, the solution is initialized in the areas close to estimated promising solutions of the problem. This approach reduces computational time but also the risk that the optimization process gets stuck in a local minimum of the cost function. The second algorithm is based on a hierarchical decomposition of the trajectory planning. The more complicated parts of the trajectory are sequentially added in the regions with more complicated structure of the environment. This process is done automatically only by comparison values of the cost function in the appropriate segments with a threshold. Beyond the computational time reduction, the dimension of the space of solutions can be easily determined with this process which is a crucial problem of common trajectory planning algorithms based on optimization.

In the last part of the thesis, an approach for airport snow shoveling using formations of autonomous ploughs is introduced as an application of the presented methods. In contrast to the general trajectory planning methods, here the plan for the robots should be not only feasible and short, but it should also be optimal with respect to maximal coverage of the cleaning roads. Furthermore, the structure of the airport environment

[1]One can find an overview of optimization methods recommended for MPC in [20; 149].

is too complicated and it is not efficient to find a complete feasible solution covering all runways by the general trajectory planning approaches. We propose to decompose the cleaning of the airport to sub-tasks of single road shovelling. The formation driving method using RHC is then utilized only to guide the formation between the way-points corresponding with crossroads of the cleaning roads while the desired sequences of the roads are obtained by a task allocation algorithm.

We developed two different approaches of the sub-task allocation to the ploughs to propose a complete snow shoveling system. In the first method, a big formation of ploughs used for shoveling of main runways is splitted to two static smaller formations for cleaning auxiliary roads. The plan for the sub-formations is obtained using a thoroughly explored tree of solutions. The final result is time optimal, but the applicability of the method is restricted to smaller airports and the algorithm runs off-line without any possibility of on-line reaction to failures of ploughs or to changes in the environment. The second approach is more robust and can be applied on-line even for the biggest airports. The method is based on assignment of the cleaning tasks to temporarily formed groups of the ploughs. These teams can be repeatedly splitted or merged to formations with arbitrary sizes depending on the width of the cleaning road. The method can provide only a local optimal solution but due to the bigger flexibility in the formations assembling the total shoveling time is usually shorter than in the case of the first approach.

The arbitrary splitting and reuniting of the formations require a general approach of the lower level control for the ploughs during these specific tasks. The formation splitting is relatively easy. The virtual leader of the original formation has to be multiplied for each sub-formation in an appropriate point as is presented in an extension of the formation driving method. A bit more complicated situation can occur during the formation merging, where the sub-formations have to be synchronized before the coupling. We propose an approach of iterative adding of sub-formations to the rising formation with determination of intervals where the leaders' commands are fused together for the guidance of the group as a compact unit. All these abilities have been verified by various simulations using a map of Frankfurt airport as well as by simple hardware experiment in the laboratory environment.

1.4 Contribution

The main contributions of our work to mobile robotics and control research are summarized in this section. As the most conspicuous contribution, we offer a full-scale approach for trajectory planning (enforced by a convergence constraint) and stabilization of the formations of nonholonomic car-like robots under the leader-follower concept. The method is general enough to consider static as well as dynamic obstacles of arbitrary shapes, but also collisions and dangerous proximity of neighboring vehicles within the formation which increases robustness of the system in case of robots' failures. The unique approach is the integration of momentary control together with the trajectory planning to reach a desired target region in future. It allows to obtain a solution by fusion both the image of local environment and the complex structure of workspace.

The utilization of RHC with variable differences $\Delta t(\cdot)$ between adjacent transition points collected in a supplementary time interval describing the future prediction of the formation movement is the biggest contribution from the control perspective. In comparison to the classical method involving only shot time interval with constant $\Delta t(\cdot)$ [53; 63; 70; 190], the values of $\Delta t(\cdot)$ are in our approach obtained as a part of an entire optimization vector. This provides an appropriate distribution of the transition points in regions with higher density of obstacles or where the structure of the environment requires complicated maneuvers. Furthermore, there is a "bridge" connecting the short interval of conventional model predictive control with the target region. The approach gives also a meaningful estimation of the total time to the goal and it enables to clearly specify assumptions of convergence according to Lyapunov's second theorem.

The extension of the RHC method with variable $\Delta t(\cdot)$ providing autonomously designed maneuvers of the formation is another contribution of this thesis. This is a unique system suitable for the kinematics of the formation of car-like robots offering autonomous turning on the spot or other complicated driving including polarity of speed changing in an environment with obstacles. The proposed method is based on alternations of two virtual leaders, when the optimal plan for both leaders is obtained in one optimization process which guarantees convergence to the target region. The number and the time of alternations of leading roles are determined automatically during the optimization process. The application is possible in a workspace with static as well as dynamic obstacles and without any restrictions on the shape of the formation or even number of the followers that can be dynamically changed.

Another novel algorithm presented in this thesis is the method based on path planning for the virtual leader of the formation in the space of multinominals. Using this method, the path and the control inputs can be optimized separately via two segregate planning loops. Usually, the information about the environment close to the robots is updated by onboard sensors more often than the rest of the map that is more or less static. Therefore, the rate of the path planning loop capturing mainly the progression of the movement can be slower in the future. The price we pay for this option is only a suboptimal solution due to process decomposition.

Both variations of the formation driving have been extended by an algorithm based on a hierarchical decomposition of the optimization process. The contribution of such an approach is the possibility to determine the optimal number of the transition points forming the trajectory (resp. number of multinominals in the path). The dimension of the space of solutions can be iteratively increased in areas of workspace where the current dimension is insufficient. Furthermore, only the segments of solution covering the part of environment with a detected dynamic obstacle can be corrected by replanning. This together with reduced complexity of the optimization due to the decomposition speed up the response to unexpected events.

From the application perspective, the biggest contribution of this thesis is the adapting of general approaches for the purpose of airport snow shoveling applying formations of autonomous ploughs which has been never closely investigated in literature. We offer a complete multi-robotic system including a task allocation module as the highest reasoning level, followed by the path planning optimal for runways' coverage in the middle level and by the formation stabilization providing control inputs for each vehicle in the lowest level. The last contribution we would like to highlight, is a novel general approach for continuous merging and splitting of the formations of car-like robots which is optimal regarding the maximal coverage of runways. This approach, much like an extended algorithm for formation turning on the road with similar width as is the width of the formation, is applicable in an arbitrary workspace with static as well as dynamic obstacles.

2

Formation coordination in dynamic environment using receding horizon control

In this chapter, we will describe an approach for multiple car-like robots driving in a desired formation to reach a target region in a dynamic environment. The method is based on a receding horizon concept, which has been initially formulated in [147] and for the control application extended as Model Predictive Control (MPC) in [152]. The MPC approach has been used in the industry since the end of the 1970s and currently only in Germany, there is more than 9000 industrial MPC applications in operation (counted in [46]). A worldwide overview of commercially available MPC technologies is provided in [148] while a survey of MPC theoretical works dealing with stability and optimality issues is presented in [121].

An example of the utilization of MPC in robotics can be found in [101] where the RHC method was applied to obtain local control for a single differential drive robot. The global information about the static environment is added using a visibility graph technique. Another method for tracking control of nonholonomic mobile robots with a given time varying reference trajectory is proposed in [70]. The stability of the method using RHC is guaranteed by adding a Lyapunov function to the cost function as the terminal-state penalty. An algorithm for autonomous trajectory tracking of aircrafts is presented in [124]. The planes there follow straight lines given by the air traffic control and RHC is utilized for safe collision conflict resolution.

Development of RHC for applications of the formation driving is intensively studied by the research community. An approach employing distributed RHC for stabilization of robots in desired positions within the formation is presented in [52]. This paper, which is extended in [53] by a stability analysis, is focused on reaching the formation and the switching between two different shapes of the formation in obstacle-free environments. Another approach is used in [23] for spacecraft formation flying where RHC is applied to reduce effects of the noise of sensors and to increase robustness. The work introduced in [75] is solving the problem of formation control to a predefined relative position with limited communication. A formation control with obstacle avoidance is introduced in [190]. The approach assumes a desired path predefined by a higher level planning and the RHC technique is used to avoid detected static obstacles blocking this path. The entire team is treated as a compact object which enables to keep the robots in the given formation and automatically avoid collisions within the team. A method of cooperative control of a team of distributed agents with decoupled nonlinear discrete-time dynamics is addressed in [63]. The problem is formulated in a receding-horizon framework, where the control laws depend on the local state variables and on delayed information gathered from cooperating neighboring agents. A technique for the coordination of autonomous organic air vehicles is proposed in [92]. The presented scheme employs decentralized receding horizon controllers that reside on each vehicle to achieve coordination among team members. The information about neighbors is used to predict their behavior and plan conflict-free trajectories that maintain coordination in a short future horizon.

In most of the previously mentioned works, the RHC is utilized for the optimal formation driving with respect to the local environment. The information about the global structure of workspace is unused or added by employing an additional global path planning method producing a path that is followed by the formation. The method we present offers an extension of these classical approaches by adding the global information directly to the optimization process under RHC. The method considers static obstacles with arbitrary shapes as well as dynamic obstacles but also a nonlinear kinematic model for car-like robots. We should mention that a shortened description of the work presented in this chapter has been published in [163].

2.1 Preliminaries and problem definitions

The proposed approach for solving the *formation to target zone problem* is formulated using RHC methodologies for both: i) the trajectory planning and control of the virtual leader[1] of formation $\mathcal{F}$, and ii) the control of n_r followers where each individual robot performs the trajectory tracking of a specific path generated using the travelled trajectory of the virtual leader. The results presented here are general enough to not accommodate any assumptions regarding the homogeneity of the group of robots with respect to geometric parameters as well as kinematic constraints.

2.1.1 Configuration space of robots

Let $\psi_L(t) = \{x_L(t), y_L(t), \theta_L(t)\} \in \mathcal{C}$ denote the configuration of a leader R_L at time t, and $\psi_i(t) = \{x_i(t), y_i(t), \theta_i(t)\} \in \mathcal{C}$, with $i \in \{1, \ldots, n_r\}$, denote the configuration of each of the n_r followers R_i at time t, where $\mathcal{C} = \mathbb{R}^2 \times [0, 2\pi)$ is the configuration space. The Cartesian coordinates $x_j(t)$ and $y_j(t)$, where $j \in \{1, \ldots, n_r, L\}$, for an arbitrary configuration $\psi_j(t) \in \mathcal{C}$, define the position $\bar{p}_j(t)$ of a robot R_j and $\theta_j(t)$ denotes its heading. Let us assume that the environment of the robots contains a finite number n_0 of compact obstacles collected in a set of regions $\mathcal{O}_{obs}$. This set is

[1]Leaders are considered as a virtual point placed in front of the formation in this thesis. In case of a heterogenous formation, where a better equipped robot is placed on the position of the virtual leader, this robot is considered as another follower led by a virtual leader in the same position.

composed of self-intersecting polygons (static obstacles) and circles (radially bounded dynamic obstacles). The configuration space $\mathcal{C}$ can be then divided into two segments, $\mathcal{C}_{obs}$ representing the configurations of a robot colliding with an obstacle and $\mathcal{C}_{free}$ representing subspace of feasible configurations as $\mathcal{C}_{free} = \mathcal{C} \backslash \mathcal{C}_{obs}$.

2.1.2 Kinematic model and constraints

The kinematics for any robots R_j, where $j \in \{1, \ldots, n_r, L\}$, is described by the simple nonholonomic kinematic model:

$$\begin{aligned} \dot{x}_j(t) &= v_j(t) \cos \theta_j(t) \\ \dot{y}_j(t) &= v_j(t) \sin \theta_j(t) \\ \dot{\theta}_j(t) &= K_j(t) v_j(t), \end{aligned} \tag{2.1}$$

where velocity $v_j(t)$ and curvature $K_j(t)$ represent control inputs $\bar{u}_j(t) = \{v_j(t), K_j(t)\} \in \mathbb{R}^2$.

2.1.2.1 Model and controller parametrization

Let us define a time interval $\langle t_0, t_{N+M} \rangle$ containing a finite sequence with $N+M+1$ elements of nondecreasing times $\mathcal{T}(t_0, t_{N+M}) := \{t_0, t_1, \ldots, t_{N-1}, t_N, \ldots, t_{N+M-1}, t_{N+M}\}$, such that $t_0 < t_1 < \ldots < t_{N-1} < t_N < \ldots < t_{N+M-1} < t_{N+M}$.[1] Also, let us define a controller for a robot R_j starting from a configuration $\psi_j(t_0)$ by $\mathcal{U}_j(t_0, t_{N+M}, \mathcal{T}(\cdot)) := \{\bar{u}_j(t_0; t_1 - t_0), \bar{u}_j(t_1; t_2 - t_1), \ldots, \bar{u}_j(t_{N+M-1}; t_{N+M} - t_{N+M-1})\}$. Each element $\bar{u}_j(t_k; t_{k+1} - t_k)$, where $k \in \{0, \ldots, N+M-1\}$, of the finite sequence $\mathcal{U}_j(t_0, t_{N+M}, \mathcal{T}(\cdot))$ will be held constant during the time interval $\langle t_k, t_{k+1})$ with length $t_{k+1} - t_k$ (not necessarily uniform). In this spirit we notice how over a time interval $\langle t_0, t_{N+M} \rangle$ a controller can be parametrized with a minimal amount of information contained in the sequences of constant control values $\mathcal{U}_j(\cdot)$ and switching times $\mathcal{T}(\cdot)$. In order to simplify our notation, the relation between $\mathcal{U}_j(\cdot)$ and $\mathcal{T}(\cdot)$ becomes implicit such that $\mathcal{U}_j(t_0) \equiv \mathcal{U}_j(t_0, t_{N+M}, \mathcal{T}(\cdot))$ and $\mathcal{T}(t_0) \equiv \mathcal{T}(t_0, t_{N+M})$.

Let us integrate the model in (2.1) over a given interval $\langle t_0, t_{N+M} \rangle$ with constant control inputs from $\mathcal{U}_j(t_0)$ in each time interval $\langle t_k, t_{k+1})$, where t_{k+1}, $t_k \in \mathcal{T}(t_0)$ and $k \in \{0, 1, \ldots, N+M-1\}$ (from this point we may refer to t_k using its index k). By this integrating, we can obtain the following model for the *transition points* at which control inputs change:

$$\begin{aligned} x_j(k+1) &= \begin{cases} x_j(k) + \frac{1}{K_j(k+1)} \left[\sin\left(\theta_j(k) + \ K_j(k+1) v_j(k+1) \Delta t(k+1)\right) - \right. \\ \left. \sin\left(\theta_j(k)\right) \right], \text{if} K_j(k+1) \neq 0; \\ x_j(k) + v_j(k+1) \cos\left(\theta_j(k)\right) \Delta t(k+1), \text{if} K_j(k+1) = 0 \end{cases} \\ y_j(k+1) &= \begin{cases} y_j(k) - \frac{1}{K_j(k+1)} \left[\cos\left(\theta_j(k) + \ K_j(k+1) v_j(k+1) \Delta t(k+1)\right) - \right. \\ \left. \cos\left(\theta_j(k)\right) \right], \text{if} K_j(k+1) \neq 0; \\ y_j(k) + v_j(k+1) \sin\left(\theta_j(k)\right) \Delta t(k+1), \text{if} K_j(k+1) = 0 \end{cases} \\ \theta_j(k+1) &= \theta_j(k) + K_j(k+1) v_j(k+1) \Delta t(k+1), \end{aligned} \tag{2.2}$$

[1]The meaning of constants N and M will be explained in Section 2.3.2.

where $x_j(k)$ and $y_j(k)$ are the rectangular coordinates and $\theta_j(k)$ the heading angle for the configuration $\psi_j(k)$ at the transition point with index k. Control inputs $v_j(k+1)$ and $K_j(k+1)$ are extracted from $\bar{u}_j(k+1) := \bar{u}_j(t_k; t_{k+1} - t_k)$ at time index $k+1$, and $\Delta t(k+1) := (t_{k+1} - t_k)$ is the sampling time. For simplification, we will gather the control inputs $\bar{u}_j(k)$, for $k \in \{1, \dots, N+M\}$, under vector $\mathcal{U}_{j,NM} \in R^{2(N+M)}$ and values $\Delta t(k)$, for $k \in \{1, \dots, N+M\}$, under vector $\mathcal{T}^{\Delta}_{j,NM} \in R^{N+M}$.

This notation allows us to describe the long trajectories exactly using a minimal amount of information such as: i) the initial configuration $\psi_j(t_0)$, ii) the sequence of switching times $\mathcal{T}^{\Delta}_{j,NM}$, and iii) the sequence of control actions $\mathcal{U}_{j,NM}$.

2.1.2.2 Controller constraints

In applications, the control inputs are limited by vehicle mechanical capabilities (i.e., chassis and engine). These constraints can be taken into account for each robot R_j limiting their control inputs by the following inequalities:

$$\begin{aligned} v_{min,j} \leq v_j(k) \leq v_{max,j} \\ |K_j(k)| \leq K_{max,j}, \end{aligned} \tag{2.3}$$

where $v_{max,j}$ is the maximal forward velocity of the j-th vehicle, $v_{min,j}$ is the limit on the backward velocity and $K_{max,j}$ is the maximal control curvature. These values can be different for each robot R_j besides the additional introduction of a time dependency as will be shown in Section 2.3.1. Notice that restrictions in equation (2.3) represent a convex compact set $\mathbb{U}_j$ for each vehicle j. We will denote the set of admissible controls for each follower R_i and the virtual leader R_L by $\mathbb{U}_i$ and $\mathbb{U}_L$, accordingly.

2.1.3 Problem Formulation

The control for a formation $\mathcal{F}$ of n_r nonholonomic car-like mobile robots reaching a target zone is defined as our main goal. To accomplish such a task the target zone must be clearly defined by a higher control entity or planner, so as the task allocation approach presented in Section 5.3.

Definition 2.1.1. (Target Region) A target region S_F is a convex compact region such that for any robot R_j, $j \in \{1, \dots, n_r, L\}$, with position $\bar{p}_j(\cdot) \in S_F$, $\psi_j(\cdot) \in \mathcal{C}_{free}$.

Now, we need the following assumption which is not restrictive in order to develop the methodology.

Assumption 1. (Desired Reachability) The virtual leader R_L can get from any initial configuration $\psi_L(t_0) \in \mathcal{C}_{free}$ at time t_0 to any other configuration $\psi_L(t_f)$, $t_f > t_0$, inside a defined compact ball $\mathbb{B}(\bar{p}_f, \epsilon)$ in some finite time $t_f - t_0$. The ball $\mathbb{B}(\bar{p}_f, \epsilon)$ is centered at the point $\bar{p}_f \in \text{Proj}_{\bar{p}}(\mathcal{C}_{free})$ with radius $\epsilon > r_{min} > 0$, such $\{\mathbb{B}(\bar{p}_f, \epsilon) \cap \text{Proj}_{\bar{p}}(\mathcal{C}_{obs})\} = \emptyset$. $\text{Proj}_{\bar{p}}(\cdot)$ is a projection operator on the Cartesian coordinates of a given set. The radius bound r_{min} is chosen based on some of the constraints associated with the maximal control curvature $K_{max,j}$ of the robots.

Remark 2.1.2. A target region S_F can be reassigned to another place in environment only by a higher control entity once $\bar{p}_L(\cdot) \in S_F$. The procedure of reassigning S_F once $\bar{p}_L(\cdot) \in S_F$ allows the robots to complete higher complexity missions by solving simpler tasks. Such an approach is utilized in the airport snow shoveling presented in Chapter 5 where the complete task is splitted to subtasks of separated runway cleaning.

2.1.4 Receding Horizon Control concept

The main idea of the receding horizon control is to solve a finite horizon optimization control problem for a system (represented by a dynamic model) starting from current states or configuration $\psi(t_0)$ over the time interval $\langle t_0, t_0 + N\Delta t\rangle$ under a set of constraints on the system states and control inputs. In this framework, the duration $N\Delta t$ of the time interval $\langle t_0, t_0 + N\Delta t\rangle$ is known as the control horizon (for simplification later called T_N). Δt is a constant sampling time, and N is number of transition points in the control horizon. After a solution from the optimization problem is obtained on a control horizon, a portion of the computed control actions is applied on the interval $\langle t_0, t_0 + n\Delta t\rangle$, known as the receding step. Parameter n is the number of transition points applied in one receding step. This process is then repeated on the interval $\langle t_0 + n\Delta t, t_0 + N\Delta t + n\Delta t\rangle$ as the finite horizon moves by *time steps* $n\Delta t$, yielding a state feedback control scheme strategy. Advantages of the RHC scheme become evident in terms of adaptation to new events such as obstacles appearing in an environment. These characteristics make the receding horizon control scheme appealing for the robot control and trajectory planning in a feedback fashion.

In this thesis, we propose a modified version of the standard approach to receding finite time horizon with constant sampling time. In our approach, the horizon is divided into two segments: i) the first segment that has a constant sampling rate used for obtaining a refined immediate control input for the virtual leader who generates desired trajectories for the followers, and ii) the second segment where the length of the time intervals between instances where control inputs change are also variables taking part of the planning problem to reach the target region. Details on such construction are presented in Section 2.3. Regarding the RHC of the followers a standard approach of the trajectory tracking problem is taken [97].

2.2 Description of general approach and proof of convergence

2.2.1 Finite time horizon optimal control of virtual leaders

Here, we need to denote the formation to target zone problem starting at time t_0 as $\mathcal{P}(t_0)$. The cost of the required solution of $\mathcal{P}(t_0)$ designed for the virtual leader R_L can be specified as:

$$J_L(\psi_L(t_0), \mathcal{U}_L^\circ(t_0); T)^\circ = \min_{\mathcal{U}_L} \left\{ \int_{t_0}^{t_0+T} L(\psi_L(s), \mathcal{U}_L(s), s) ds \right\}. \tag{2.4}$$

In this formulation, $\mathcal{U}_L^\circ(t_0) \in \mathbb{U}_L$ denotes the optimal controller that generates optimal predicted states $\psi_L^\circ(\cdot)$. The minimization of the predicted cost $J(\cdot)$ is subjected to a set of equality constraints $h(\cdot) = 0$ representing the system model in (2.2) over a finite time horizon of length T (initially chosen arbitrary large at t_0) and a set of inequality constraints $g(\cdot) \leq 0$ that impose system's state and control input constraints[1] as well as artificially introduced constraints to guarantee stability properties (for more details see [121]). The proposed stability constraint, $g_{S_F}(\cdot) \leq 0$, represents the condition that the last position state of the horizon enters S_F.

Function $L(\psi_L(\cdot), \mathcal{U}_L(\cdot), \cdot)$ penalizes elapsed time for the leader until reaching S_F and local proximity of $\psi_L(\cdot)$ with unsafe regions in the environment (proximity penalties are only active in vicinity of static or dynamic obstacles). $L(\psi_L(\cdot), \mathcal{U}_L(\cdot), \cdot)$ is chosen as a time-invariant, positive definite function. Making this choice is a key element in the developing stability properties for the leader-follower control scheme. Now, for regularity purposes we need the following trivial assumption.

Assumption 2. Once $\psi_L(\cdot)$ enters S_F at some time $\bar{t} \in \langle t_0, \infty)$, no additional cost will be acquired. In other words, the desired target equilibrium is not a point instead it is the region S_F. Then, we can assume that $L(\psi_L(\cdot), \mathcal{U}_L(\cdot), \cdot) = 0$ for all $\bar{p}_L(\cdot) \in S_F$, while $L(\psi_L(\cdot), \mathcal{U}_L(\cdot), \cdot) > 0$ for all $\bar{p}_L(\cdot) \notin S_F$.

Remark 2.2.1. Once the first optimization problem has been solved, the time $\bar{t}$ is identified as the minimum time to reach S_F.

2.2.2 Proof of convergence

In this section we present proof of convergence for the virtual leader receding horizon scheme based on the presented framework.

Theorem 2.2.2. *Under Assumptions 1-2, given a target region S_F, and a feasible solution of $\mathcal{P}(t_0)$, the receding time horizon control scheme iteratively solves the optimal control problem in equation (2.4) to obtain control inputs $\mathcal{U}_L^\circ(\cdot)$ for the virtual leader. These control inputs stabilize and guide R_L toward the target region S_F if perturbations on $J(\cdot)^\circ$, due to obstacles penalties for $L(\cdot)$, denoted by $D(k)$, satisfy $D(k) < \int\limits_{\tau_1}^{\tau_2} L^\circ(\psi_L(s), \mathcal{U}_L(s), s)ds$, between any two times $\tau_1 = kn\Delta t + t_0$ and $\tau_2 = (k+1)n\Delta t + t_0$, for $k \in \mathbb{Z}^+$ where $k < (\bar{t} - t_0)/n\Delta t$.*

Proof. The convergence of R_L to the target region S_F can be proven according to Lyapunov's second stability theorem [93] by choosing the optimal cost $J_L(\cdot)^\circ$ as a candidate Lyapunov function. The desired equilibrium, which is needed for the stability, is considered as the whole S_F where the $J_L(\cdot)^\circ$ is equal to zero. Within this region a local controller working in an obstacle free environment can be applied to reach a state, which becomes a new equilibrium for such a controller, if it is desired. In this thesis we

[1]Detailed description of the constraints $h(\cdot)$ and $g(\cdot)$ is provided in Section 2.3.2

are focused on the task to reach S_F. We are not considering the inner local controller and stabilization of the system inside S_F. The only property that has to be shown to prove the convergence of the method is the decrease of $J_L(\cdot)^\circ$. This can be transformed to the following inequality that needs to be verified:

$$J_L(\psi_L(n\Delta t+t_0),\mathcal{U}_L^\circ(n\Delta t+t_0);T_2)^\circ - J_L(\psi_L(t_0),\mathcal{U}_L^\circ(t_0);T_1)^\circ < 0, \forall \bar{p}_L(\cdot) \notin S_F. \tag{2.5}$$

The second part of the left hand side, $J_L(\psi_L(t_0),\mathcal{U}_L^\circ(t_0);T_1)^\circ$, is the cost of the optimal solution of $\mathcal{P}(t_0)$. We will assume that the solution is feasible and therefore $T_1 \geq \bar{t}-t_0$. The first term of the left hand side, $J_L(\psi_L(n\Delta t+t_0),\mathcal{U}_L^\circ(n\Delta t+t_0);T_2)^\circ$, is the optimal cost obtained solving problem $\mathcal{P}(n\Delta t+t_0)$ again by starting from the initial conditions $\psi_L(n\Delta t+t_0)$ yet. The initial state $\psi_L(n\Delta t+t_0)$ was reached after applying the first n elements of the sequence $\mathcal{U}_L^\circ(t_0)$ during $n\Delta t$ units of time. Again we can assume that the solution is feasible and therefore $T_2 \geq \bar{t}-n\Delta t-t_0$.

The term $J_L(\psi_L(t_0),\mathcal{U}_L^\circ(t_0);T_1)^\circ$ can be rewritten, according to principle of optimality [8] as

$$\begin{aligned} J_L(\psi_L(t_0),\mathcal{U}_L^\circ(t_0);T_1)^\circ &= J_L(\psi_L(t_0),\mathcal{U}_L^\circ(t_0);\bar{t}-t_0)^\circ \\ &\quad + J_L(\psi_L(\bar{t}),\mathcal{U}_L^\circ(\bar{t});T_1-\bar{t}+t_0)^\circ. \end{aligned} \tag{2.6}$$

Similarly, the term $J_L(\psi_L(n\Delta t+t_0),\mathcal{U}_L^\circ(n\Delta t+t_0);T_2)^\circ$ can be rewritten as

$$\begin{aligned} J_L(\psi_L(n\Delta t+t_0),\mathcal{U}_L^\circ(n\Delta t+t_0);T_2)^\circ &= J_L(\psi_L(n\Delta t+t_0),\mathcal{U}_L^\circ(n\Delta t+t_0);\bar{t}-n\Delta t-t_0)^\circ \\ &\quad + J_L(\psi_L(\bar{t}),\mathcal{U}_L^\circ(\bar{t});T_2-\bar{t}+n\Delta t+t_0)^\circ. \end{aligned} \tag{2.7}$$

Now we can apply Assumption 2 to get equalities

$$J_L(\psi_L(\bar{t}),\mathcal{U}_L^\circ(\bar{t});T_2-\bar{t}+n\Delta t+t_0)^\circ = 0 \tag{2.8}$$

and

$$J_L(\psi_L(\bar{t}),\mathcal{U}_L^\circ(\bar{t});T_1-\bar{t}+t_0)^\circ = 0. \tag{2.9}$$

Considering these observations together with equations (2.7) and (2.6) we arrive to equality

$$\begin{aligned} &J_L(\psi_L(n\Delta t+t_0),\mathcal{U}_L^\circ(n\Delta t+t_0);T_2)^\circ - J_L(\psi_L(t_0),\mathcal{U}_L^\circ(t_0);T_1)^\circ = \\ &= J_L(\psi_L(n\Delta t+t_0),\mathcal{U}_L^\circ(n\Delta t+t_0);\bar{t}-n\Delta t-t_0)^\circ - J_L(\psi_L(t_0),\mathcal{U}_L^\circ(t_0);\bar{t}-t_0)^\circ. \end{aligned} \tag{2.10}$$

The terms on the right hand side of the last equality can be subtracted in the case of an unchanged environment and missing disturbances as

$$\begin{aligned} &J_L(\psi_L(n\Delta t+t_0),\mathcal{U}_L^\circ(n\Delta t+t_0);\bar{t}-n\Delta t-t_0)^\circ - J_L(\psi_L(t_0),\mathcal{U}_L^\circ(t_0);\bar{t}-t_0)^\circ = \\ &= -\int_{t_0}^{t_0+n\Delta t} L(\psi_L(s),\mathcal{U}_L^\circ(s),s)ds. \end{aligned} \tag{2.11}$$

Collecting the equations (2.10) and (2.11) we get

$$\begin{aligned} & J_L(\psi_L(n\Delta t + t_0), \mathcal{U}_L^\circ(n\Delta t + t_0); T_2)^\circ - J_L(\psi_L(t_0), \mathcal{U}_L^\circ(t_0); T_1)^\circ = \\ = \; - & \int\limits_{t_0}^{t_0+n\Delta t} L(\psi_L(s), \mathcal{U}_L^\circ(s), s)ds. \end{aligned} \tag{2.12}$$

Comparing the equations (2.5) and (2.12), and considering the part of Assumption 2, $L(\psi_L(\cdot), \mathcal{U}_L(\cdot), \cdot) > 0, \forall \bar{p}_L(\cdot) \notin S_F$, one can easily see that the equation (2.5) is satisfied.

□

2.3 Implementation details

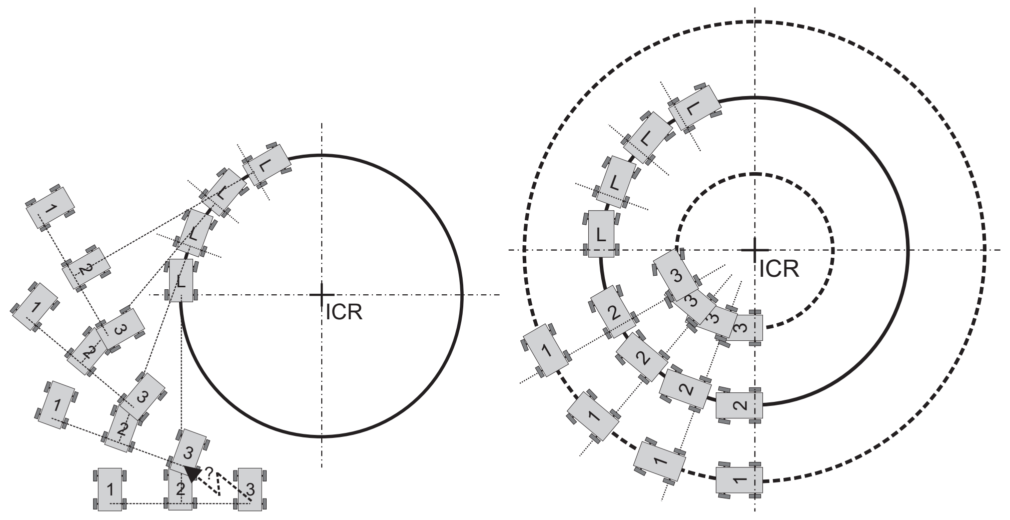
The previously introduced general framework will be adjusted and implemented to the control of the virtual leader as well as of the followers in this section. The applied optimization methods can be decentralized and computed independently on board of the appropriate vehicles. In the case of the virtual leader we expect that a follower will be equipped by an additional computational power for computing both, the leader plan as well as its own control inputs. This robot can also be carrying better sensors for obstacle detections or a device for fusion data from formation $\mathcal{F}$. First of all, it is necessary to describe a concept in which the whole formation will be kept.

2.3.1 Formation driving concept

The formation driving methods described in this thesis are based on a leader-follower approach, in which the followers R_i, where $i \in \{1, \dots, n_r\}$, track the leader's trajectory. For a better explanation the trajectory will be divided into two parts according to an actual position of the virtual leader.

Definition 2.3.1. (Leader's trajectory) Let us define the part of the trajectory that was followed by the virtual leader in the past as $\overleftarrow{\Psi}_L(t) := \{\psi_L(\hat{t}) : t_0 \leq \hat{t} \leq t\}$ and similarly the actual plan that represents the part of the trajectory that should be followed by the virtual leader in the future as $\overrightarrow{\Psi}_L(t) := \{\psi_L(\hat{t}) : t < \hat{t} \leq t_f\}$. Then, the complete leader's trajectory is simply the union $\Psi(t) := \{\overleftarrow{\Psi}_L(t) \cup \overrightarrow{\Psi}_L(t)\}$.

The shape of the formation and consequently states of the followers relative to the state of the virtual leader are usually determined by specific applications. Furthermore, most of the applications require a dynamically changing shape of formations during missions to respond to changes in the environment or in the tasks (for details see Section 1.1). The approach presented in this thesis enables the utilization of formations with generally varying shape with restrictions given only by kinematic and dynamic limitations of the car-like robots. It is evident that the position of a robot within the formation cannot be changed suddenly and such transition must be feasible for the robot. An exact analysis and determination of the feasible and unfeasible changes of



(a) Cartesian coordinates. (b) Curvilinear coordinates.

Figure 2.1: Four subsequent snapshots of the formation with fixed relative positions between the followers in (a) Cartesian and (b) curvilinear coordinates. Solid lines denote paths of the virtual leader while paths of the followers are denoted by dashed lines.

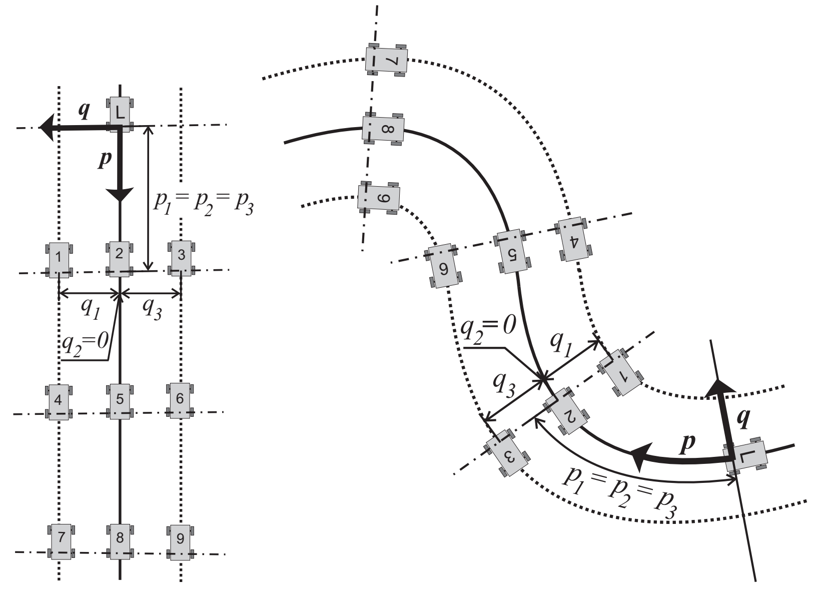
the formation that depend on the leader's trajectory will not be addressed in this thesis and can be found in [157].

A less evident, but more important problem for formation driving of car-like robots is caused by the impossibility to change heading of the robot on the spot. That is why the formations with fixed relative distance in Cartesian coordinates cannot be used. Such structure makes smooth movement of the followers impossible as is shown in a simple example presented in Fig. 2.1. To solve this problem, we utilized an approach in which the followers are maintained in relative distance to the virtual leader in curvilinear coordinates with two axes p and q, where p traces $\Psi_L(t)$ and q is perpendicular to p as is demonstrated in Fig. 2.2. The positive direction of p is defined from R_L back to the origin of the movement R_L and the positive direction of q is defined in the left half plane from the robots perspective.

The shape of the formation is then uniquely determined by states $\psi_L(t_{p_i(t)})$, where $i \in \{1, \dots, n_r\}$, in *travelled distance* $p_i(t)$ from R_L along $\overleftarrow{\Psi}_L(t)$ and by *offset distance* $q_i(t_{p_i(t)})$ between $\bar{p}_L(t_{p_i(t)})$ and $\bar{p}_i(t)$ in perpendicular direction from $\overleftarrow{\Psi}_L(t)$. $t_{p_i(t)}$ is the time when the virtual leader was at the *travelled distance* $p_i(t)$ behind the actual position. The parameters $p_i(t)$ and $q_i(t)$, defined for each follower i, can vary during the mission.

To convert the state of the followers in curvilinear coordinates to the state in rectangular coordinates, the following equations can be applied:

$$\begin{aligned} x_i(t) &= x_L(t_{p_i(t)}) - q_i(t_{p_i(t)}) \sin(\theta_L(t_{p_i(t)})) \\ y_i(t) &= y_L(t_{p_i(t)}) + q_i(t_{p_i(t)}) \cos(\theta_L(t_{p_i(t)})) \\ \theta_i(t) &= \theta_L(t_{p_i(t)}), \end{aligned} \tag{2.13}$$



(a) Constant curvature. (b) Variable curvature.

Figure 2.2: Equivalent formations with time invariant curvilinear coordinates p and q, following different trajectories.

where $\psi_L(t_{p_i(t)}) = \{x_L(t_{p_i(t)}), y_L(t_{p_i(t)}), \theta_L(t_{p_i(t)})\}$ is state of the virtual leader at time $t_{p_i(t)}$.

Remark 2.3.2. Positions of R_i, where $i \in \{1, \ldots, n_r\}$, can be determined only if $p_i(t) \geq 0$, $t_0 \leq t \leq t_f$, since $\overleftarrow{\Psi}_L(t)$ that is necessary for defining R_i is defined only for the past time.

By applying the leader-follower approach using $p_i(t)$ and $q_i(t)$ coordinates we can easily determine admissible controls sets $\mathbb{U}_L$ based on

$$\begin{aligned}
K_{max,L}(t) &= \min_{i=1,\ldots,n_r} \left(\frac{K_{max,i}}{1 + q_i(t)K_{max,i}} \right) \\
K_{min,L}(t) &= \max_{i=1,\ldots,n_r} \left(\frac{-K_{max,i}}{1 - q_i(t)K_{max,i}} \right) \\
v_{max,L}(t) &= \min_{i=1,\ldots,n_r} \left(\frac{v_{max,i}}{1 + q_i(t)K_L(t)} \right) \\
v_{min,L}(t) &= \max_{i=1,\ldots,n_r} \left(\frac{v_{min,i}}{1 + q_i(t)K_L(t)} \right).
\end{aligned} \tag{2.14}$$

These restrictions must be applied to satisfy the different curvatures and speeds of the robots in different positions in $\mathcal{F}$ during turning. From the Fig. 2.1(b), it may be clear that the robot following the inner track should go slower but with a bigger curvature than the robot further from the center of turning.[1] Control inputs of the leading robot

[1]For an explanation of motivation of equations (2.14), let us suppose that followers 1, 2 and 3 in

$v_L(t)$ and $K_L(t)$ will then be bounded by the inequalities:

$$\begin{aligned} K_{min,L}(t) \leq K_L(t) \leq K_{max,L}(t) \\ v_{min,L}(t) \leq v_L(t) \leq v_{max,L}(t). \end{aligned} \tag{2.15}$$

Similarly, the obstacle avoidance behavior of the virtual leader can differ from the behavior of the follower robots. Firstly, we should define a circular detection boundary with radius r_s and a circular avoidance boundary with radius r_a for a single robot. Single robots should not respond to obstacles detected outside the region with radius r_s. On the contrary, the distance between the robots and obstacles less than r_a is considered as inadmissable. Since the leader's trajectory has to be collision free for the virtual leader but also for the followers, the shape of the formation should be included to the avoidance behavior of the virtual leader. The extended detection and unsafe zones for the virtual leader can be expressed as

$$\begin{aligned} r_{s,L}(t) &= r_s + \max_{i \in \{1,\ldots,n_r\}} |q_i(t)| \\ r_{a,L}(t) &= r_a + \max_{i \in \{1,\ldots,n_r\}} |q_i(t)|, \end{aligned} \tag{2.16}$$

where $r_{s,L}(t)$ is the detection zone of R_L and $r_{a,L}(t)$ is the unsafe zone of R_L. These concepts are based on the concept of avoidance control [32; 33; 109; 110] and will be used in this thesis for the collision avoidance guaranties.

In order to simplify our presentation and without any particular loss of generality, we will assume that the regions are time invariant, that is,

$$\begin{aligned} r_{s,L} &= \max_{0<t<t_f} r_{s,L}(t) \\ r_{a,L} &= \max_{0<t<t_f} r_{a,L}(t). \end{aligned} \tag{2.17}$$

Generally, the detection and unsafe zones of the virtual leader can be asymmetric in the case of an asymmetric formation [173]. In this thesis, this problem is solved using the approach with a virtual leader that is always positioned at the axe of the formation. Furthermore the utilization of the virtual leader can handle the case $p_i(t) \leq 0$ forbidden in *Remark* 2.3.2 by placing the leader in front of the formation. The virtual leader R_L should be during the mission situated within the formation in such a position that the following equations are satisfied for all $i \in \{1, \ldots, n_r\}$:

$$\begin{aligned} \max_{i=1,\ldots,n_r} q_i(t) = - \min_{i=1,\ldots,n_r} q_i(t) \\ p_i(t) \geq 0. \end{aligned} \tag{2.18}$$

Fig. 2.1(b) have the same controller constraints. During the turning right, the maximal curvature of the leader, $K_{max,L}(t)$, must be lower than the maximal curvature of robot 3, $K_{max,3}$, otherwise robot 3 may not be able to follow a leader's movement and to stay in its desired position within the formation. The maximal velocity $v_{max,L}(t)$ must be lower than $v_{max,1}$ (resp. the minimal velocity $v_{min,L}(t)$ must be higher than $v_{min,3}$) otherwise follower 1 gets behind the formation (resp. follower 3 foreruns the formation). If the formation is turning left, the minimum curvature $K_{min,L}(t)$ must be higher than $-K_{max,1}$ and the roles of robots 1 and 3 are swopped for the leader's velocity limitations. All these effects grows with the distance of followers from the leader in q direction and furthermore the the leader's velocity limits depend on the actual curvature of leader $K_L(t)$.

2.3.2 Leader trajectory planning and control

In the presented approach we propose to solve two usually separated problems: collision free trajectory planning and a computation of control sequence, in one optimization step. The aim of the method is to find a control sequence which could control the virtual leader to the target region by minimizing a given cost function. The main idea of the approach is to divide such a sequence into two finite time intervals T_N for $k \in \{1, \ldots, N\}$ and T_M for $k \in \{N+1, \ldots, N+M\}$. The first time interval T_N should provide control inputs for R_L regarding local environment of the formation. By applying this portion of the control sequence, the group should be able to respond to changes in workspace that can be dynamic or newly detected static obstacles. The difference $\Delta t(k+1) = t_{k+1} - t_k$ is kept constant (later denoted only Δt) in this time interval. Value of Δt should satisfy the requirements of the classical receding horizon control scheme, because this part of control will be directly used as an input of the vehicles.

The second interval T_M takes into account information about global characteristics of the environment to navigate the formation to the goal optimally. The transition points in this part of $\overrightarrow{\Psi}_L(t)$ can be distributed irregularly to effectively cover the environment. During the optimization process, more points should be automatically allocated in the regions with higher density of obstacles or in places where a complicated maneuver of the formation is needed. This is enabled due to the varying values of $\Delta t(k+1) = t_{k+1} - t_k$ that will be for the compact description collected into the vector $\mathcal{T}^{\Delta}_{L,M} := \{\Delta t(N+1), \ldots, \Delta t(N+M)\}$. The total time from actual position of the robot to the target region will be then

$$t_f - t = N\Delta t + \sum_{k=N+1}^{N+M} \Delta t(k). \tag{2.19}$$

To define the trajectory planning problem with two time intervals in a compact form we need to gather states $\psi_L(k)$, where $k \in \{1, \ldots, N\}$, and $\psi_L(k)$, where $k \in \{N+1, \ldots, N+M\}$, into vectors $\Psi_{L,N} \in \mathbb{R}^{3N}$ and $\Psi_{L,M} \in \mathbb{R}^{3M}$. Similarly the control inputs $\bar{u}_L(k)$, where $k \in \{1, \ldots, N\}$, and $\bar{u}_L(k)$, where $k \in \{N+1, \ldots, N+M\}$, can be gathered into vectors $\mathcal{U}_{L,N} \in \mathbb{R}^{2N}$ and $\mathcal{U}_{L,M} \in \mathbb{R}^{2M}$. All variables describing the complete trajectory from the actual position of the virtual leader until target region are collected into the optimization vector $\Omega_L = [\Psi_{L,N}, \mathcal{U}_{L,N}, \Psi_{L,M}, \mathcal{U}_{L,M}, \mathcal{T}^{\Delta}_{L,M}] \in \mathbb{R}^{5N+6M}$.

2.3.2.1 Objective function and constraints

The trajectory planning and the static as well as dynamic obstacle avoidance problem for the virtual leader can be transformed to the minimization of cost function $J_L(\cdot)$ subject to sets of equality constraints $h_{T_N}(\cdot), h_{T_M}(\cdot)$ and inequality constraints $g_{T_N}(\cdot)$, $g_{T_M}(\cdot)$, $g_{S_F}(\cdot)$, $g_{r_{a,L}}(\cdot)$, that is

$$\min J_L(\Omega_L) \tag{2.20}$$

$$\begin{aligned} \text{s.t. } h_{T_N}(k) &= 0, \forall k \in \{0, \dots, N-1\} \\ h_{T_M}(k) &= 0, \forall k \in \{N, \dots, N+M-1\} \\ g_{T_N}(k) &\le 0, \forall k \in \{1, \dots, N\} \\ g_{T_M}(k) &\le 0, \forall k \in \{N+1, \dots, N+M\} \\ g_{S_F}(\psi_L(N+M)) &\le 0 \\ g_{r_{a,L}}(\Omega_L, \mathcal{O}_{obs}) &\le 0. \end{aligned} \tag{2.21}$$

The cost function $J_L(\cdot)$ is given by

$$J_L(\Omega_L) = \sum_{k=N+1}^{N+M} \Delta t(k) + \alpha \sum_{j=1}^{n_0} \left(\min \left\{ 0, \frac{dist_j(\Omega_L, \mathcal{O}_{obs}) - r_{s,L}}{dist_j(\Omega_L, \mathcal{O}_{obs}) - r_{a,L}} \right\} \right)^2, \tag{2.22}$$

where the endeavor of the trajectory planning to reach a desired goal as soon as possible is expressed in the first part of $J_L(\cdot)$ and the influence of the environment on the final solution is added to the cost function in the second term. This second part of $J_L(\cdot)$, which is the sum of modified avoidance functions, contributes to the final cost when an obstacle from $\mathcal{O}_{obs}$ (static or dynamic) is closer to the trajectory than $r_{s,L}$ and it will approach infinity if distance $r_{a,L}$ to the obstacle is reached. Function $dist_j(\Omega_L, \mathcal{O}_{obs})$ providing Euclidean distance between obstacle j and $\overrightarrow{\Psi}_L(t)$ cannot influence optimization if the obstacle is sufficiently far. The utilization of these avoidance functions is motivated by [45] and [173], where similar approaches have been used for a cooperative collision avoidance in multi-agent systems.

The influence of both parts of the cost function is adjusted by constant α. An increase of α results in longer trajectories with larger distances from obstacles which can be in highly dynamic environments applications more safety. On the contrary, even zero value of α does not lead to collisions due to the constrain $g_{r_{a,L}}$. One can find a more detailed description of the influence of α in our previous works [161; 164]

The kinematic model (2.2) with initial conditions given by the actual state of the virtual leader is represented using equality constraints $h_{T_N}(k)$, where $k \in \{0, \dots, N-1\}$, and $h_{T_M}(k)$, where $k \in \{N, \dots, N+M-1\}$, from (4.7) as

$$h_{T_N}(k) := h_{T_N}(\psi_L(k), \psi_L(k+1), \bar{u}_L(k+1)) = \begin{bmatrix} h^x(k) \\ h^y(k) \\ h^\theta(k) \end{bmatrix} \tag{2.23}$$

and

$$h_{T_M}(k) := h_{T_M}(\psi_L(k), \psi_L(k+1), \bar{u}_L(k+1), \Delta t(k+1)) = \begin{bmatrix} h^x(k) \\ h^y(k) \\ h^\theta(k) \end{bmatrix}. \tag{2.24}$$

The first set of constraints $h_{T_N}(k)$, where $k \in \{0, \dots, N-1\}$, is applied for the first part of the trajectory with constant Δt, while the set $h_{T_M}(k)$, where $k \in \{N, \dots, N+M-1\}$, should be satisfied for the second part where $\Delta t(k)$, where $k \in \{N+1, \dots, N+M\}$, is a variable.

Similarly, for the intervals T_N and T_M, we define different sets of inequality constraints $g_{T_N}(k)$, where $k \in \{1, \ldots, N\}$, and $g_{T_M}(k)$, where $k \in \{N+1, \ldots, N+M\}$, as

$$g_{T_N}(k) := g_{T_N}(\psi_L(k), \bar{u}_L(k)) = \begin{bmatrix} g^{v_{min,l}}(k) \\ g^{v_{max,l}}(k) \\ g^{K_{min,l}}(k) \\ g^{K_{max,l}}(k) \end{bmatrix} \tag{2.25}$$

and

$$g_{T_M}(k) := g_{T_M}(\psi_L(k), \bar{u}_L(k), \Delta t(k)) = \begin{bmatrix} g^{v_{min,l}}(k) \\ g^{v_{max,l}}(k) \\ g^{K_{min,l}}(k) \\ g^{K_{max,l}}(k) \\ g^{\Delta t_{min}}(k) \end{bmatrix}. \tag{2.26}$$

In both sets the constraints $g^{v_{min,l}}(\cdot)$, $g^{v_{max,l}}(\cdot)$, $g^{K_{min,l}}(\cdot)$ and $g^{K_{max,l}}(\cdot)$ characterize bounds on the velocity and curvature given by (2.14). The constraints $g^{\Delta t_{min}}(k)$ in $g_{T_M}(k)$ ensure that inequalities $\Delta t(k) \geq 0$, where $k \in \{N+1, \ldots, N+M\}$, are satisfied.

The avoidance inequality constraints $g_{r_{a,L}}(\Omega_L, \mathcal{O}_{obs})$ that characterize safety avoidance regions are defined as:

$$g_{r_{a,L}}(\Omega_L, \mathcal{O}_{obs}) := r_{a,L}^2 - dist_j(\Omega_L, \mathcal{O}_{obs})^2, j \in \{1, \ldots, n_0\}. \tag{2.27}$$

Finally, $g_{S_F}(\psi_L(N+M))$ is a stability constraint guaranteeing that $\overrightarrow{\Psi}_L(t)$ will enter target region S_F. For simplification it is supposed that the target region is a circle with radius r_{S_F} and center C_{S_F}. The stability constraint is then given by

$$g_{S_F}(\psi_L(N+M)) := r_{S_F} - \|\bar{p}_L(N+M) - C_{S_F}\|. \tag{2.28}$$

2.3.2.2 Initialization

The experimental results presented in this chapter were obtained using sequential quadratic programming, which is a generalization of Newton's method [139]. The outcome of such a local optimization process with the cost function that was introduced in Section 2.3.2.1 is strongly dependent on the initialization of the optimizer. The cost function contains a lot of local extremes in which the process can easily get stuck.

A solution of such a phenomena could be the utilization of a global optimization method or the addition of a stochastic part to the local optimization which can help to disengage from a local minima. For both approaches, it is difficult to prove an ability to find the global minimum and in addition the optimization is significantly slower which cannot be accepted for our purposes. Thus, we suggest an approach based on an appropriate initialization of the optimization. The idea is to initialize the planning process as close to the global optimum as possible in the first optimization run. We propose a path planning approach using Voronoi Diagrams as an appropriate method for finding an estimation of the position of the global minimum. The proposed algorithm is based on a distribution of the vectors $\Psi_{L,N}$ and $\Psi_{L,M}$ along the shortest feasible Voronoi paths. A detailed description of this approach, which is used only for the first initialization of the planning loop, is presented in Section 4.

In this section we will focus on the re-initialization of the planning method applied in the next steps of the loop which is important for a fast response to problems appearing suddenly. Once we have the optimal solution of the task, it is useful for the decreasing of the computational rate to initialize the next run using this result. In a static environment without disturbances, we can even easily prove, using the principle of optimality as shown in Section 2.2.2, that the optimal plan from the new position of the robot will be part of the previous plan. In real applications, the solution should be adapted for the new situation but usually the optimal solution needs to be changed only slightly and the previous result can be successfully used for the re-initialization.

Based on the receding horizon control used in this thesis, only part of the optimal solution $\Omega_L^\circ = [\Psi_{L,N}^\circ, \mathcal{U}_{L,N}^\circ, \Psi_{L,M}^\circ, \mathcal{U}_{L,M}^\circ, \mathcal{T}_{L,M}^{\Delta\circ}]$ can be utilized for the re-initialization without any modification. As we have explained in Section 2.1.4, the sequence $\Psi_{L,N}^\circ$ (resp. $\mathcal{U}_{L,N}^\circ$) can be divided into two parts. In the first part, we have $\psi_L^\circ(k)$, for $k \in \{1, \ldots, n\}$, (resp. $\bar{u}_L^\circ(k)$, for $k \in \{1, \ldots, n\}$) and in the second part, we have $\psi_L^\circ(k)$, for $k \in \{n+1, \ldots, N\}$, (resp. $\bar{u}_L^\circ(k)$, for $k \in \{n+1, \ldots, N\}$). Only the inputs from the first part, $\bar{u}_L^\circ(k)$, for $k \in \{1, \ldots, n\}$, will be used as the immediate control. This is why $\psi_L^\circ(k)$, where $k \in \{n+1, \ldots, N\}$, (resp. $\bar{u}_L^\circ(k)$, where $k \in \{n+1, \ldots, N\}$) can be utilized directly as the beginning of the new initialization as $\psi_{L,init}(k) := \psi_L^\circ(k+n)$, for $k \in \{1, \ldots, N-n\}$, (resp. $\bar{u}_{L,init}(k) := \bar{u}_L^\circ(k+n)$, for $k \in \{1, \ldots, N-n\}$). Now we have to initialize states $\psi_{L,init}(k)$, where $k \in \{N-n+1, \ldots, N\}$, and control inputs $\bar{u}_{L,init}(k)$, where $k \in \{N-n+1, \ldots, N\}$. Here we must ensure that the trajectory is continuous in both intervals, T_N and T_M. Therefore, this part of the initialization should be extracted from the optimal plan to target region capturing the states and control inputs at times $t + (k+n)\Delta t$, where $k \in \{N-n+1, \ldots, N\}$. Finally, we have to create the vectors $\Psi_{L,M,init}$, $\mathcal{U}_{L,M,init}$ and $\mathcal{T}_{L,M,init}^{\Delta}$. The suitable initialization is to use the vectors $\Psi_{L,M}^\circ$, $\mathcal{U}_{L,M}^\circ$ and $\mathcal{T}_{L,M}^{\Delta\circ}$ with a modification necessary due to the overlapping of the trajectory that is described by vectors $\Psi_{L,N,init}$ and $\mathcal{U}_{L,N,init}$, and the trajectory that is described by vectors $\Psi_{L,M}^\circ$, $\mathcal{U}_{L,M}^\circ$ and $\mathcal{T}_{L,M}^{\Delta\circ}$. This overlapping is caused by the concept of RHC. Therefore, the first k_e transition points must be from the vectors $\Psi_{L,M}^\circ$, $\mathcal{U}_{L,M}^\circ$ and $\mathcal{T}_{L,M}^{\Delta\circ}$ extracted and the first element of such obtained vector $\mathcal{T}_{L,M,init}^{\Delta}$ must appropriately be shortened. Number k_e is the maximal integer satisfying inequality $n\Delta t > \sum_{i=N}^{N+k_e} \Delta t^\circ(i)$. Then the remaining part of the initialization is $\psi_{L,init}(k) := \psi_L^\circ(k+k_e)$, where $k \in \{N+1, \ldots, N+M-k_e\}$, $\bar{u}_{L,init}(k) := \bar{u}_L^\circ(k+k_e)$, where $k \in \{N+1, \ldots, N+M-k_e\}$, $\Delta t_{init}(N+1) := \sum_{i=N}^{N+k_e} \Delta t^\circ(i) - n\Delta t$ and $\Delta t_{init}(k) := \Delta t^\circ(k+k_e)$, where $k \in \{N+2, \ldots, N+M-k_e\}$. The complete initialization vector will be composed as $\Omega_{L,init} = [\Psi_{L,N,init}, \mathcal{U}_{L,N,init}, \Psi_{L,M,init}, \mathcal{U}_{L,M,init}, \mathcal{T}_{L,M,init}^{\Delta}]$ after this step.

Finally, it is important to mention that the repeated usage of the previous plan as the initialization of the new optimization is not always appropriate. Unseemly moving obstacles crossing the trajectory of the formation can push the formation away from the desired target region under special condition. In such a case the proof of the optimality presented in Section 2.2.2 can fail, because the cost of the new plan will be bigger then the previous one. A similar situation can also happen, if the map of the environment is changed significantly which can produce new local extremes. Fortunately, both examples can be simply detected by the increasing of the cost and the easiest solution of this problem is to initialize the optimization from the beginning with the approach using the Voronoi Diagram.

2.3.2.3 Remark on convergence

The proof of convergence of formation $\mathcal{F}$ to target region S_F presented in Section 2.2.2 can be utilized for an arbitrary cost function satisfying the Assumption 2 (such can be e.g. a function describing the fuel consumption, the minimum operation time or the optimal coverage). In the case of the cost function described in equation (2.22), the satisfaction of the Assumption 2 is not explicitly described but it results from the structure of the cost function. Considering the first term of equation (2.22) and the convergence constraint $g_{S_F}(\cdot)$, we can assume that the equality $T_1 = \bar{t} - t_0$ (resp. $T_2 = \bar{t} - n\Delta t - t_0$) is held for the total time of the optimal solution from state $\psi_L(t_0)$ (resp. $\psi_L^\circ(n\Delta t + t_0)$). In other words, a feasible plan that contains a part of the trajectory inside the target region cannot be optimal, because such a plan can be divided to two parts: the part reaching the border of S_F and the remainder. Let us suppose a shortened plan that consists from the first part of the complete one. Such a shortened plan is cheaper than the previous plan (the value of the first term of the cost function is decreased and value of the second term is decreased or unchanged) and as a part of the feasible plan also satisfies all the constraints. Therefore, the Assumption 2 does not need to be included into the cost function.

2.3.3 Trajectory tracking for followers

The trajectory $\Psi_L^\circ(\cdot)$ computed for the virtual leader of the formation as the result of the previous section can be directly used via the equations (2.13) for navigation of the followers to S_F. Unfortunately, this plan is not able to respond to events behind the actual position of the leader. In terms of applications, it cannot be assumed that the environment stays static until the last member passes a sector in which previously collision free plan was created by the virtual leader moving at the head of the group. We also cannot expect that each follower will follow its optimal plan faultlessly. Incorrect driving direction or velocity can be dangerous for the neighbors, which should be able to avoid possible collisions.

The idea of our approach is to use the leader's trajectory, which consists from $\overleftarrow{\Psi}_L(\cdot)$ and from the part of $\overrightarrow{\Psi}_L^\circ(\cdot)$ on interval T_N, as an input for the formation driving approach described in Section 2.3.1. The desired states for each of the followers can be obtained as $\psi_{d,i}(k) := FormationDriving_i(t + k\Delta t)$, for $k \in \{1, \ldots, N\}$ and $i \in \{1, \ldots, n_r\}$, where $FormationDriving_i(\hat{t})$ represents equations (2.13) applied on state $\psi_L(t_{p_i(\hat{t})})$ with coordinates $p_i(\hat{t})$ and $q_i(t_{p_i(\hat{t})})$. The states $\psi_{d,i}(k) = (\bar{p}_{d,i}(k), \theta_{d,i}(k))$, for $k \in \{1, \ldots, N\}$ and $i \in \{1, \ldots, n_r\}$, will be utilized for trajectory tracking algorithm with obstacle avoidance functions in this section.

The states and the control vectors $\psi_i(k)$, where $k \in \{1, \ldots, N\}$, and $\bar{u}_i(k)$, where $k \in \{1, \ldots, N\}$, can be gathered as vectors $\Psi_i \in \mathbb{R}^{3N}$ and $\mathcal{U}_i \in \mathbb{R}^{2N}$ for each follower i similarly to the leader planning in Section 2.3.2. These vectors can be collected into a unique optimization vector $\Omega_i = [\Psi_i, \mathcal{U}_i] \in \mathbb{R}^{5N}$. The vector Ω_i will be used to represent the dynamic behavior of the discrete trajectory tracking with collision avoidance and to capture it as a static optimization process under the receding horizon scheme described in Section 2.1.4.

2.3.3.1 Objective function and constraints

Discrete-time trajectory tracking for followers R_i, where $i \in \{1, \dots, n_r\}$, is in this section transformed to an optimization problem with cost function $J_i(\cdot)$ subject to a number of equality constraints $h_i(\cdot)$ and inequality constraints $g_i(\cdot)$, $g_{r_a}(\cdot)$, and $g_{ra,i}(\cdot)$. The optimization process for each follower is decentralized. The only necessary communication within the group is used for propagation of the appropriate part of the leader's trajectory and for sharing the positions of the obstacles detected by the members of the team. The complete optimization scheme can be presented in the form:

$$\min J_i(\Omega_i), i \in \{1, \dots, n_r\} \tag{2.29}$$

$$\begin{aligned} \text{s.t. } h_i(k) &= 0, \forall k \in \{0, \dots, N-1\} \\ g_i(k) &\leq 0, \forall k \in \{1, \dots, N\} \\ g_{r_a}(\Omega_i, \mathcal{O}_{obs}) &\leq 0 \\ g_{r_{a,i}}(\Omega_i, \Omega_j^\circ) &\leq 0, \forall j \in \bar{n}_n, \end{aligned} \tag{2.30}$$

where

$$\begin{aligned} J_i(\Omega_i) = \sum_{k=1}^{N} \|(\bar{p}_{d,i}(k) - \bar{p}_i(k))\|^2 &+ \alpha \sum_{j=1}^{n_0} \left(\min \left\{ 0, \frac{dist_j(\Omega_i, \mathcal{O}_{obs}) - r_s}{dist_j(\Omega_i, \mathcal{O}_{obs}) - r_a} \right\} \right)^2 \\ &+ \beta \sum_{j \in \bar{n}_n} \left(\min \left\{ 0, \frac{d_{i,j}(\Omega_i, \Omega_j^\circ) - r_{s,i}}{d_{i,j}(\Omega_i, \Omega_j^\circ) - r_{a,i}} \right\} \right)^2, \end{aligned} \tag{2.31}$$

$$h_i(k) := h_i(\psi_i(k), \psi_i(k+1), \bar{u}_i(k+1)) = \begin{bmatrix} h^x(k) \\ h^y(k) \\ h^\theta(k) \end{bmatrix}, \forall k \in \{0, \dots, N-1\} \tag{2.32}$$

and

$$g_i(k) := g_i(\psi_i(k), \bar{u}_i(k)) = \begin{bmatrix} g^{v_{min,l}}(k) \\ g^{v_{max,l}}(k) \\ g^{K_{min,l}}(k) \\ g^{K_{max,l}}(k) \end{bmatrix}, \forall k \in \{1, \dots, N\}. \tag{2.33}$$

The proposed cost function $J_i(\cdot)$ consists of three components with their influence adjusted by constants α and β. The first component represents deviations of the position $\bar{p}_i(k)$, for $k \in \{1, \dots, N\}$, from the desired position $\bar{p}_{d,i}(k)$, for $k \in \{1, \dots, N\}$, and so it is crucial for the effort of the trajectory tracking. The second summation in $J_i(\cdot)$ is equivalent to the second term used in (2.22) and should ensure that dynamic or lately detected obstacles will be avoided. Only difference is that the avoidance regions for single robots can be smaller than the avoidance regions for the virtual leader as was explained in Section 2.3.1. The third component of $J_i(\cdot)$ is the sum of avoidance functions in which the other members of the team are considered also as dynamic obstacles. This part has to protect the robots in cases of an unexpected behavior of defective neighbors. Function $d_{i,j}(\Omega_i, \Omega_j^\circ)$ is the minimal distance between the planned trajectory of R_i and the actually exercised plan of R_j, for $j \in \bar{n}_n$, where $\bar{n}_n = \{1, \ldots, i-1, i+1, \ldots, n_r\}$. It is evident from the idea of the leader following motion planning that the planning loops of the followers are synchronized by the time at which the plan of the virtual leader is prepared. Then, all robots are computing their controls concurrently and only the previously computed plans Ω_j° can be utilized to obtain the value of $d_{i,j}(\Omega_i, \Omega_j^\circ)$ as

$$d_{i,j}(\Omega_i, \Omega_j^\circ) := \min_{k \in \{1,\ldots,N-n\}} \left\| \bar{p}_j^\circ(k+n) - \bar{p}_i(k) \right\|, \tag{2.34}$$

where $\bar{p}_j^\circ(\cdot)$ is an assumed position of the robot R_j if the plan stays unchanged. The detection radius $r_{s,i}$ is usually smaller than basic r_s, because the follower should not try to avoid a close neighbor if both are at the desired position. The constant $r_{s,i}$ then can be defined as

$$r_{s,i} := \min\left(r_s, \min_{j \in \bar{n}_n} \left(\min_{k \in \{1,\ldots,N\}} \| \bar{p}_{d,j}(k) - \bar{p}_{d,i}(k) \| \right) \right). \tag{2.35}$$

The avoidance radius $r_{a,i}$ is simply defined as $r_{a,i} = \min(r_{s,i}, r_a)$ in our case. In general the independently moving obstacles can be considered as more dangerous and difference between $r_{a,i}$ and r_a can be increased appropriately.

The equality constraints $h_i(k)$, defined in (2.32), are identical to the equality constraints $h_{T_N}(k)$, given in (2.23), where $k \in \{0, \ldots, N-1\}$. Similarly, the inequality constraints $g_i(k)$, defined in (2.33), are identical to the inequality constraints $g_{T_N}(k)$, given in (2.25), where $k \in \{1, \ldots, N\}$, and finally the avoidance inequality constraints $g_{r_a}(\Omega_i, \mathcal{O}_{obs})$ and $g_{r_{a,i}}(\Omega_i, \Omega_j^\circ)$ are given by equations

$$\begin{aligned} g_{r_a}(\Omega_i, \mathcal{O}_{obs}) &:= r_a^2 - dist_j(\Omega_i, \mathcal{O}_{obs})^2, j \in \{1, \ldots, n_0\} \\ g_{r_{a,i}}(\Omega_i, \Omega_j^\circ) &:= r_{a,i}^2 - d_{i,j}(\Omega_i, \Omega_j^\circ)^2, j \in \bar{n}_n, \end{aligned} \tag{2.36}$$

similarly as $g_{r_{a,L}}(\Omega_L, \mathcal{O}_{obs})$ in equation (2.27).

2.3.3.2 Initialization

The advantage of the leader-follower method described in Section 2.3.1 consists in the possibility to compute $\mathcal{U}_i^\circ$, where $i \in \{1, \ldots, n_r\}$, analytically directly from Ω_L°. Such a solution would be obtained faster than by the optimization method presented in the previous section, but these results can be applied only in an obstacle free environment and with ideal robots without failures and disturbances.

We propose a combination of both approaches as an ideal solution. The analytically obtained result can be utilized in the first step as an efficient and fast initialization of the optimization based method. In the second step of the planning approach, the optimization technique is applied to solve the problems with obstacles, failures of robots and disturbances of sensors. Such an architecture of the system can significantly decrease the total computational time and furthermore it can reduce the problem of the local minima of the cost function.

The initialization of the vector $\Psi_{i,init}$ directly results from the concept of the trajectory tracking and we can use $\psi_{i,init}(k) := \psi_{d,i}(k)$, for $k \in \{1, \ldots, N\}$. The initialization of the second part of $\Omega_{i,init}$, the vector $\mathcal{U}_{i,init}$, can be simply computed only if parameters $p_i(t)$ and $q_i(t)$ are time invariant. A more advanced formation driving method must be applied for applications with variable shapes of the formation.

The initial work in this field has been described by Barfoot and Clark in [14] and [13]. Their approach, developed only for trajectories with piecewise constant curvatures, has been later generalized by Hess *et al.* in [78]. The idea of their method is similar to the approach introduced in Section 2.3.1, where the actual $\psi_i(t)$ is computed using $\psi_L(t_{p_i(t)})$ in the *travelled distance* $p_i(t)$ backwards. They propose that $\bar{u}_i(t)$ is obtained from $\bar{u}_L(t_{p_i(t)})$ which was applied when the virtual leader was at distance $p_i(t)$ from the actual position along $\overleftarrow{\Psi}_L(t)$. In general, if the shape of the formation is changing, the control inputs $\bar{u}_i(t) = \{v_i(t), K_i(t)\}$ can be computed according to [78] using

$$\begin{aligned} v_i(t) &= Q_i(t_{p_i(t)}) v_L(t_{p_i(t)}) \\ K_i(t) &= \frac{1}{Q_i(t_{p_i(t)})} \left(K_L(t_{p_i(t)}) + \frac{K_L(t_{p_i(t)}) \left(\frac{dq_i}{dt}(t_{p_i(t)}) \right)^2}{Q_i(t_{p_i(t)})^2} \right. \\ &\quad \left. + \frac{\left(1 - q_i(t_{p_i(t)}) K_L(t_{p_i(t)})\right) \frac{d^2 q_i}{dt^2}(t_{p_i(t)}) + q_i(t_{p_i(t)}) \frac{dq_i}{dt}(t_{p_i(t)}) \frac{dK_L}{dt}(t_{p_i(t)})}{Q_i(t_{p_i(t)})^2} \right), \end{aligned} \tag{2.37}$$

where

$$Q_i(t_{p_i(t)}) = \sqrt{\left(\frac{dq_i}{dt}(t_{p_i(t)}) \right)^2 + \left(1 - q_i(t_{p_i(t)}) K_L(t_{p_i(t)})\right)^2}.$$

The initial control inputs for the first planning step gathered under the vector $\mathcal{U}_{i,N,init}$ can be then computed as $\bar{u}_{i,init}(k) := Hess_i(t + k\Delta t)$, for $k \in \{1, \ldots, N\}$, where $Hess_i(\hat{t})$ represents equations (2.37) applied on control inputs $\bar{u}_L(t_{p_i(\hat{t})})$ with coordinates $p_i(\hat{t})$ and $q_i(t_{p_i(\hat{t})})$.

At the end, we will describe the re-initialization of $\Omega_{i,init}$ which can differ from the initialization for the first step of the optimization described above. Due to the RHC method applied for the driving of the followers, a part of the vector Ω_i is computed repeatedly. We can assume that the environment is changed only slightly between two iterations of the planning loop. Then the appropriate approach is to re-initialize the beginning of the optimization vector using the values obtained in the previous iteration as the optimal solution. Concretely $\psi_{i,init}(k) := \psi_i^\circ(k + n)$ and $\bar{u}_{i,init}(k) := \bar{u}_i^\circ(k + n)$, for $k \in \{1, \ldots, N - n\}$. The rest of the vectors must be obtained using the same approach as for the initialization of the first step: $\psi_{i,init}(k) := \psi_{d,i}(k)$ and $\bar{u}_{i,init}(k) := Hess_i(t + k\Delta t)$, for $k \in \{N - n + 1, \ldots, N\}$.

2.3.3.3 Notes on stability

The convergence results proposed in Section 2.2.2 for the virtual leader trajectory planning and control should be completed by the proof of stability of the formation behind the leader. One can find several studies showing stability properties of the trajectory tracking using RHC in literature and therefore the development of a new concept would be useless. We will just briefly refer to a few of them and then select an approach suitable for our algorithm.

Most of the approaches assume that the optimal cost function over the finite horizon is a valid Lyapunov function or elaborate on idea that the last state in the horizon N must reach the equilibrium. In [90; 122], an equality constraint is applied to guarantee that the last state of horizon N reaches the equilibrium. This is achieved by the utilization of sufficiently big N, which is due to the high computational demand hardly practicable. A terminal region (similar to our target region S_F) is introduced in a neighborhood of the required origin in [40; 125]. A complete trajectory as the input and a local linear control law K_L, which is applied once the system enters the terminal region, are assumed in this approach. These assumptions together with the necessity to extend N to reach the region are omitted in the work published in [123] where the idea of a static terminal region is extended to the variable set $E_F(t)$ contracting with time. Furthermore an additional horizon of length $P = n$ is utilized in the paper. This horizon, used first in [61], is called *prediction horizon* and it is used to describe the future behavior of the plant. This "virtual" horizon is considered only for the proof of stability and the implementation of the method can be used with the standard two horizons.

In **Theorem 1.** of [123] and in the attached proof, an asymptotical stability of the system, which is equivalent to our system described in equations (2.1), is shown. Two necessary assumptions are applied in the proof: the state feedback of the system is available at all sampling instances and no disturbances are considered in the system environment. The second assumption cannot be always satisfied in our application due to dynamic and unknown obstacles. Similarly as in the **Proposition 2.2.2** for the virtual leader, we have to accommodate a restriction on the perturbations caused by the obstacles. In practical application a threshold of the cost function has to be settled. If the cost for a follower exceeds the threshold, the virtual leader of formation should interrupt the mission or the follower has to be removed from the group.

2.3.4 System overview

A general architecture of the overall system is presented in this section to highlight the relations between independent modules as well as the necessary communication within the team of robots.

The entire system can be logically divided into two blocks as you can see in the scheme depicted in Fig. 2.3. The first block, called *Virtual Leader*, can be physically placed onboard a sufficiently equipped follower within the formation. Such an approach increases the robustness of the system, because in case of the failure of such a robot, the same package can be initialized in a backup robot and the formation can continue to accomplish the task. In the *Virtual Leader* part, the formation to target zone problem (see Section 2.3.2) and the formation driving task (see Section 2.3.1) are iteratively solved. Concretely, the *Trajectory Planning* block should provide control inputs for the virtual leader but also the complete trajectory to the target zone which is collision free for the whole formation. The trajectory is described by a sequence of configurations of the virtual leader $\psi_L(k)$ and by constant control inputs $\bar{u}_L(k)$, where $k \in \{1, \ldots, N +$

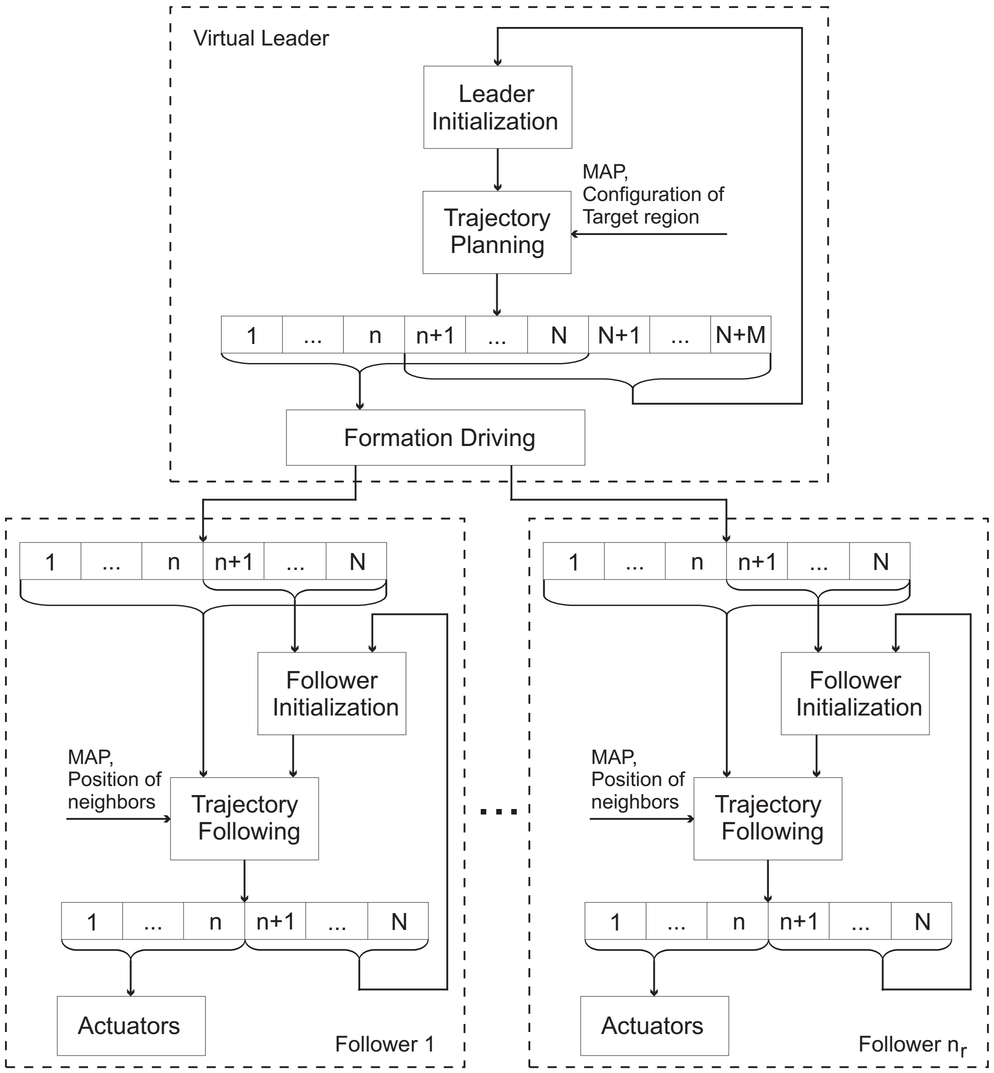


Figure 2.3: Diagram of the entire planning system with denoted communication channels.

$M\}$, that are applied in between the transition points. The output of the *Trajectory Planning* block is used as an input for *Leader Initialization* and *Formation Driving* modules. In the *Formation Driving* module, the first N components of the leader plan are transformed to the desired configurations of the followers. This planning process is then repeated under the RHC concept, where only the first n samples of the solution are really implemented in the controller of the robots. Nevertheless, the rest of the solution can be used for the initialization of the planning task in the next step via the *Leader Initialization* module. The plan will be significantly altered only in case of the changed topology of the environment. For details of the initialization see Section 2.3.2.2.

The core of the second main block, which is multiplied for each of the followers, is the *Trajectory Following* module designing appropriate collision free control inputs for the vehicles (see Section 2.3.3). This part is responsible for the avoiding of impending collisions with obstacles or other members of the team and it should correct deviations from the desired trajectory provided by the virtual leader. Due to the RHC concept again only the first n components of the optimal solution will be applied to the real system and the rest can be recycled in the *Follower Initialization* module. In the initialization, this part of the final solution is combined with part of the new desired trajectory designed by the *Formation Driving* module which enables to obtain the new solution faster (for details see Section 2.3.3.2).

No explicit feedback from the followers to the virtual leader planning is depicted in the scheme from Fig. 2.3. Nevertheless, we should mention that the map of the environment, which is used in *Trajectory Planning*, is updated by the sensors on board of the followers. Based on this, the positions of the newly detected obstacles as well as the positions of the team members need to be shared within the formation. Beyond this, the plan provided by the *Formation Driving* module is the only necessary communication in the presented approach. A direct feedback from the followers that could enable the leader to wait for delayed or broken robots is not considered. The aim of the system presented in this chapter is to reach the target region as soon as possible even in the case of the robots' failures.

2.4 Experimental results of formation to desired goal region problem

2.4.1 Parameters tuning

The crucial problem of the methods using the RHC schema is to find an appropriate number of samples (variables n and N) used in the optimization. By the increasing of n, the planning loop is enlarged and the time interval usable for computing of the optimal solution is longer, but the ability to respond to changes in the dynamic environment is logically reduced. With the growing value of N, the necessary computational time of optimization is increased and contrariwise a too small N can cause undesirable oscillation as shown in Section 5.4.2 where the parameters n and N are tuned for the application of airport snow shoveling.

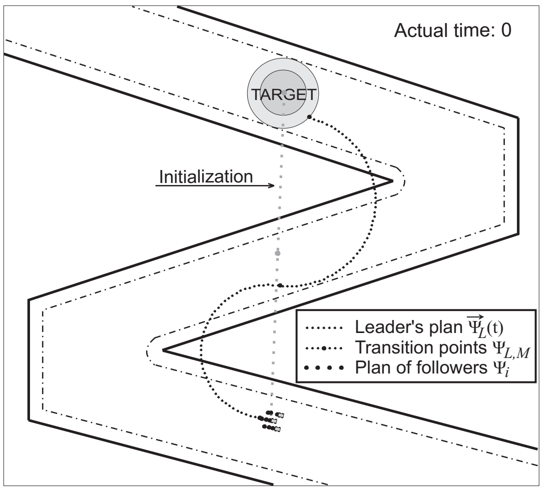
This subsection will be focused on the proper setting of parameter M, which is employed in the novel incorporation of the trajectory planning and the optimal control

Table 2.1: Values of the cost function and required computational times for one planning step of the algorithm using a different value of parameter M and fixed constants $N = 4$, $n = 2$. First row: Time needed for the first step. Second row: Maximum time needed for the other steps. Third row: Cost of the final solution obtained in the first step. The results have been obtained by computer with Pentium 4 CPU 3.2GHz using function *fmincon* of Matlab [62].

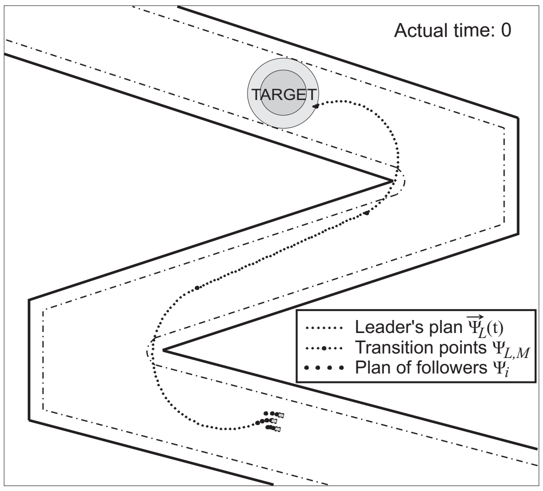
M [-]	1	2	3	4	5	6	7	8
time [s] (1.step)	17.8	34.7	34.9	43.4	54.7	68.4	82.5	99.3
max. time [s] ($\geq$ 2.steps)	0.15	0.28	0.29	0.36	0.49	0.61	0.75	0.93
cost function [-] (1.step)	∞	∞	38.8	27.8	21.1	15.7	13.1	11.6

under the receding horizon concept. Similarly as the constant N, also the value of M has significant influence on the computational time needed for the optimization as shown in Table 2.1. There are presented results of the situation from Fig. 2.4 which has been solved by the algorithm with different values of parameter M. The first row of the table shows the time at which the result was obtained in the first step of the planning loop. In the next steps of the planning loop, the computational time is notably decreased. The reduction is more than a hundred times in the case of static obstacles as shown in the second row of the table. These values are maximum times that are needed for the computation of the planning steps with a sequential number bigger than 1. The decrease of the task complexity is achieved by the utilized initialization, where the major part of the solution is reused for the new optimization and the modified result only responds to changes in the environment and to disturbances of the sensors. The values presented in the second row of the table together with product $n\Delta t$ determine an upper bound of M. The optimization process must be prepared before the formation accomplishes the part of the previous plan described by the first n transition points. It is clear, that in the case of a significant change of workspace, the new plan must be created from scratch. Then the optimization process is longer and the formation has to interrupt its movement similarly as the system has to wait at the beginning for the first plan.

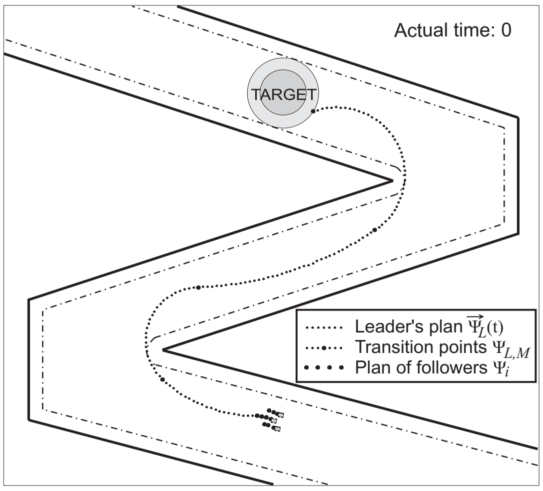
The third row of the Table 2.1 presents costs of the final solutions in the first iteration obtained using the function introduced in equation (2.22). As expected the quality of the results increases with the increase of parameter M. This effect can be simply observed in Fig. 2.4 presenting solutions obtained by the algorithm with different values of M. Using the first setting, $M = 2$, the space of solutions is insufficient and the obtained optimal trajectory is colliding with borders of the road (Fig. 2.4(a)). The optimal trajectory obtained using $M = 3$, presented in Fig. 2.4(b), is feasible for the virtual leader but is to close to the roadside for the formation whose shape would need to be deformed. The first feasible solution for the entire formation was achieved with $M = 4$ (Fig. 2.4(c)) but the trajectories designed by the algorithm using a bigger M (e.g. $M = 7$ in Fig. 2.4(d)) lead the formation to the target region faster.


Actual time: 0
TARGET
Initialization
Leader's plan $\overrightarrow{\Psi}_L(t)$
Transition points $\Psi_{L,M}$
Plan of followers Ψ_i

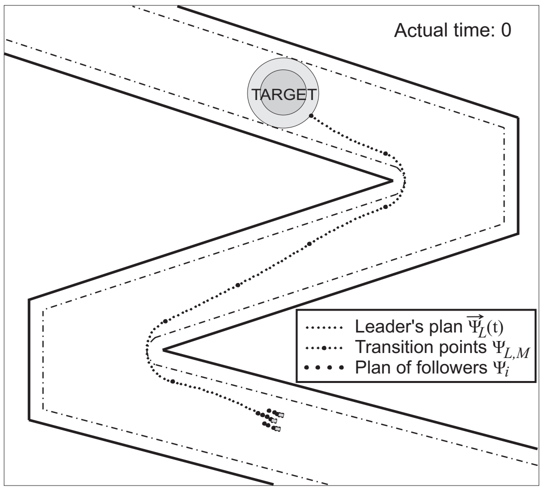
(a) $M = 2$.


Actual time: 0
TARGET
Leader's plan $\overrightarrow{\Psi}_L(t)$
Transition points $\Psi_{L,M}$
Plan of followers Ψ_i

(b) $M = 3$.


Actual time: 0
TARGET
Leader's plan $\overrightarrow{\Psi}_L(t)$
Transition points $\Psi_{L,M}$
Plan of followers Ψ_i

(c) $M = 4$.


Actual time: 0
TARGET
Leader's plan $\overrightarrow{\Psi}_L(t)$
Transition points $\Psi_{L,M}$
Plan of followers Ψ_i

(d) $M = 7$.

Figure 2.4: Solution of the *formation to target zone problem* with different setting of parameter M.

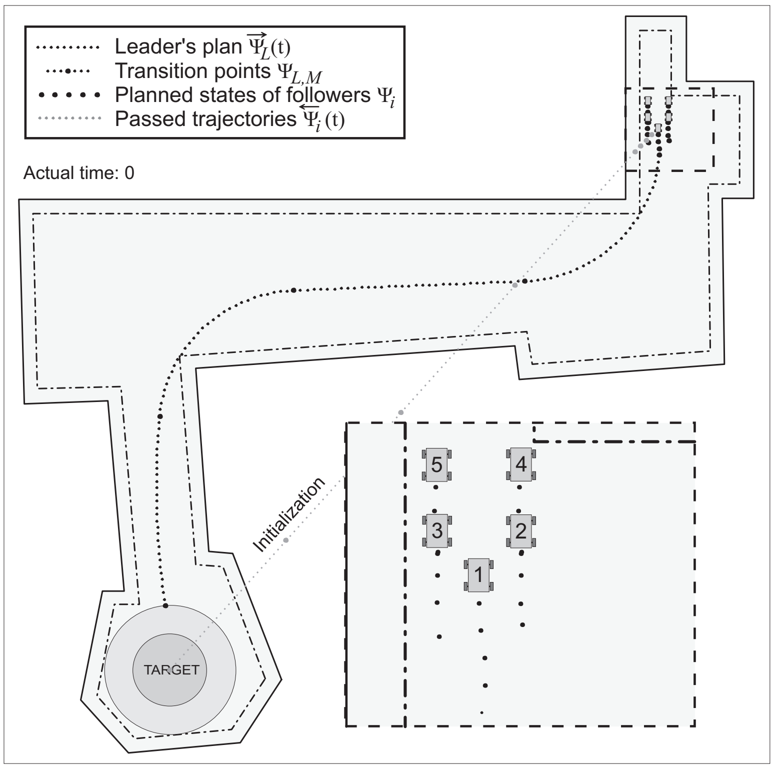

Leader's plan $\overrightarrow{\Psi_L}(t)$
Transition points $\Psi_{L,M}$
Planned states of followers Ψ_i
Passed trajectories $\overleftarrow{\Psi_i}(t)$
Actual time: 0
Initialization
TARGET
1
2
3
4
5

Figure 2.5: A plan of the leader and the followers to reach the desired target zone S_F from initial positions with delineated initialization of the trajectory planning for leader.

A speculation about a sufficient value of M is possible only in such a well structured environment like the one we used in this section. We can expect that a trivial sequence like: go straight, turn right, go straight, turn left, go straight will be feasible and therefore also any plan with $M \geq 5$ is feasible. In most of the examples this approach is too pessimistic. Even in our simple case the possibility of the feasible solution using $M = 4$ cannot be found by our simple speculation. In more complicated environments, such a better estimation is difficult. Therefore the parameter M has to be adjusted close to the upper bound determined by the computational resources, except simple environments or specific applications where the estimation of the appropriate length of the optimization vector is possible. Nevertheless, this will not guarantee a feasible result. The approach based on a hierarchical decomposition of the optimization task, which will be introduced in Section 4.2, could be a solution of this problem. The basic idea of the method is to initialize the planning into a small space of solutions, similarly as in Fig. 2.4(a) or Fig. 2.4(b). In the case of no collision the result can be followed by the team. Contrariwise, if a collision is detected, the appropriate part of the solution is replaced by a more detailed plan. This procedure can be iteratively repeated until a final feasible solution is reached and the knowledge of the variable M is not needed beforehand.

2.4.2 Simulation with dynamic obstacles

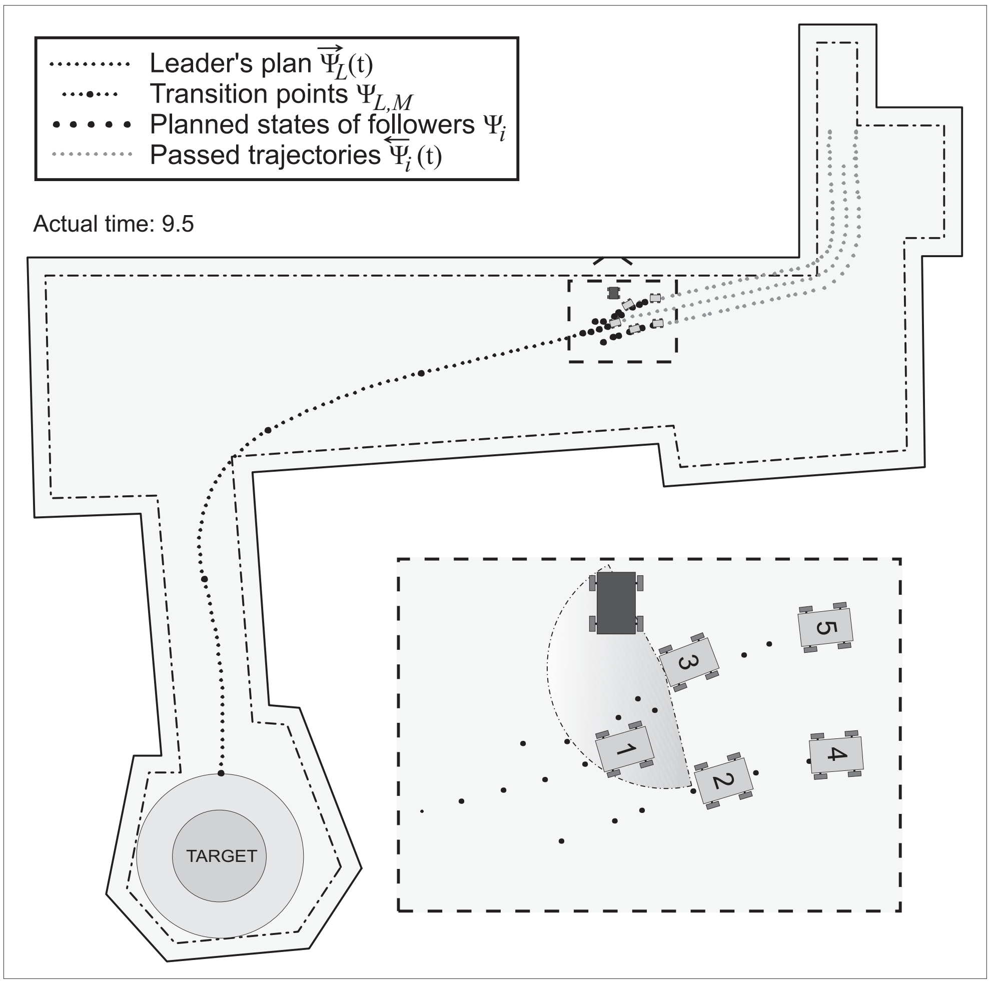
A simulation of the proposed method applied to the formation movement in a dynamically changing environment is provided in this section. The example has been designed to illustrate the approach and to present features and abilities of the algorithm in multiple car-like robotic scenarios. As the workspace for the robots, the map of the Computer Science building at the University of Wuerzburg, Germany, was chosen. This map, depicted in Fig. 2.5, is known by the robots at the beginning of the mission, whereas inner obstacles (static as well as dynamic) are unknown and will be detected during the mission. The range of the leader's sensors will be wider than the range of the followers' sensors (as is shown by dot-dash cones in Fig. 2.6 and Fig. 2.9)[1] to follow the idea of the heterogeneously equipped robots. The information obtained from different sensors will be fused and shared within the formation. Four followers (robots 2-5) led by one leader (robot 1) will be used in the experiment. Nevertheless, due to the architecture of the trajectory planning, robot 1 will be considered as a follower with distance $q_1 = 0$ and $p_1 = 0$ from the virtual leader. The parameters for the algorithm defined in Section 2.3.2 are given by $N = 4$, $M = 4$, $n = 2$ and $\Delta t = 0.25s$.

The task of the robots in the following simulation is to achieve the desired target region, while maintaining predefined position with respect to the virtual leader. The shape of the formation can be temporarily changed only in the case of an imminent collision with obstacles. The dot-dash map of the building indicates original borders dilated by the radius of the formation. Then, the feasible trajectory of the virtual leader should be within this polygon. A straight line connecting actual position of the virtual leader and the center of the target region was used as the initialization of the Sequential Quadratic Programming (SQP) which was applied as optimization method. Such a trajectory is intersecting with walls of the building and therefore it is unfeasible. Nevertheless, the plan presented in the first snapshot of the simulation (see Fig. 2.5) is applicable which shows robustness of the algorithm. The plans sequentially obtained for the followers are highlighted in the zoomed area. The optimality of the obtained plans is restricted by the utilized optimization method. Generally, we cannot prove that the SQP algorithm found the desired optimum of the utilized cost functions but we can easily check using the value of the cost function whether the solution is feasible for the formation or not.

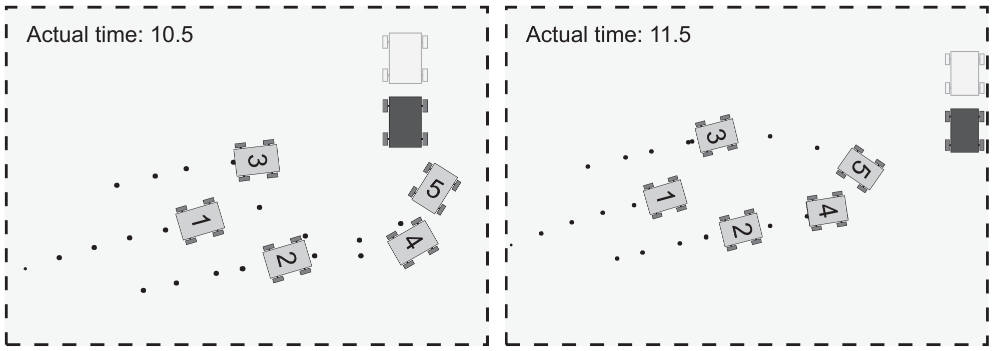
In the second snapshot of the simulation (zoomed part of the snapshot in Fig. 2.6(a)), a moving obstacle is coming from a neighboring room in a colliding course behind the virtual leader that already cannot appropriately react. The ability of the followers to avoid such a collision is shown in the following zoomed snapshot in Fig. 2.6(b). The followers close to the obstacle (robots 3 and 5) are leaving desired trajectories to avoid the crash, but also plans of the other robots are slightly changed to keep sufficient distances within the team.

In Fig. 2.7, where the formation is again brought back to the desired shape, a new obstacle is detected by the leader on the previously planned trajectory and the plan

[1]It is assumed that a set of sensors can completely cover the region around the robots with radius $r_{s,L}$ resp. r_s. For simplification, only the area covered by the sensor that detected an obstacle are depicted in the visualization.

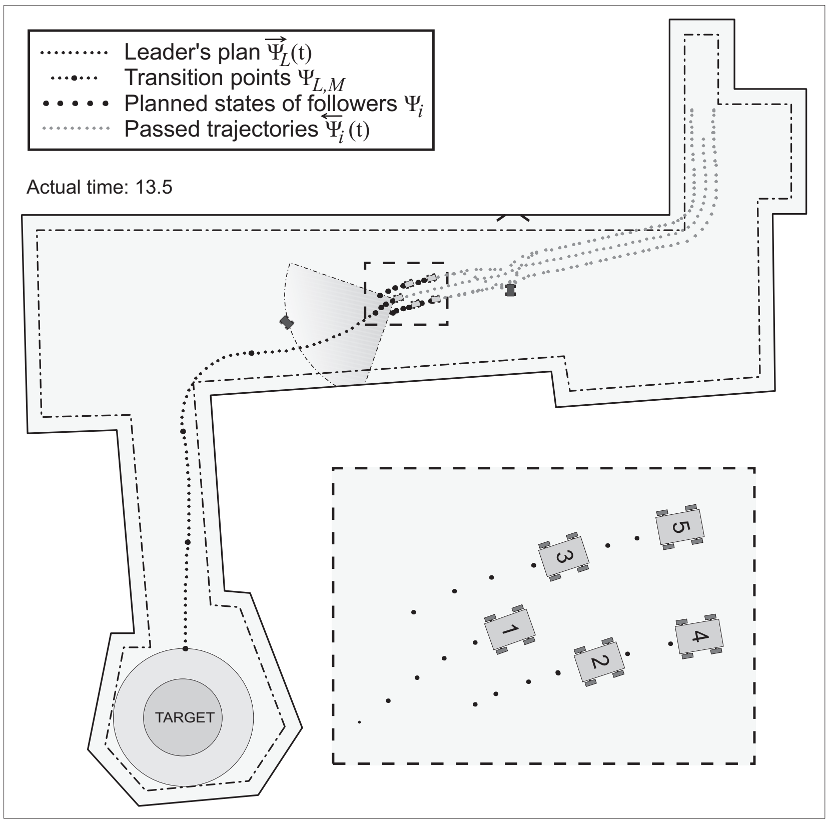

Leader's plan $\overrightarrow{\Psi}_L(t)$
Transition points $\Psi_{L,M}$
Planned states of followers Ψ_i
Passed trajectories $\overleftarrow{\Psi}_i(t)$
Actual time: 9.5
TARGET
1
2
3
4
5

(a) Adapted plan as a response to the moving obstacle detected by the followers.



(b) Zoomed details of the dynamic obstacle avoidance manoeuvre. Shaded contour denotes position of the obstacle one second ago.

Figure 2.6: Snapshots of the formation deformation as a reaction to the observed robot with colliding course.



Leader's plan $\overrightarrow{\Psi_L}(t)$
Transition points $\Psi_{L,M}$
Planned states of followers Ψ_i
Passed trajectories $\overleftarrow{\Psi_i}(t)$
Actual time: 13.5
TARGET
1
2
3
4
5

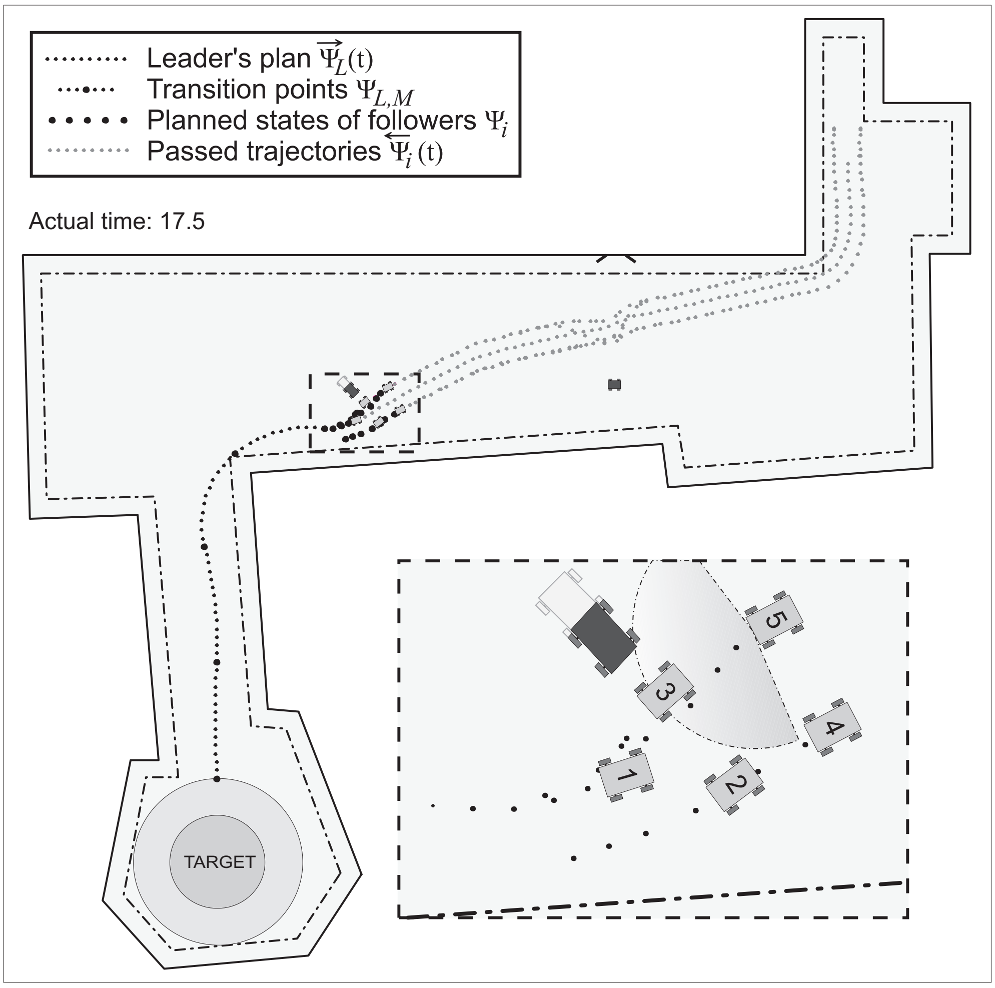
Figure 2.7: Leader's plan is adaptively changed with respect to the newly detected obstacle situated on the previously planned trajectory.

is modified for the following step. The new trajectory is feasible for all the robots until the obstacle stays static. In Fig. 2.8(a), a movement of the object was detected (previous static position is denoted by the shadow contour) which forces the followers to an avoidance maneuver. In the following snapshots (see Fig. 2.8(b)), the moving obstacle is avoided without collisions.

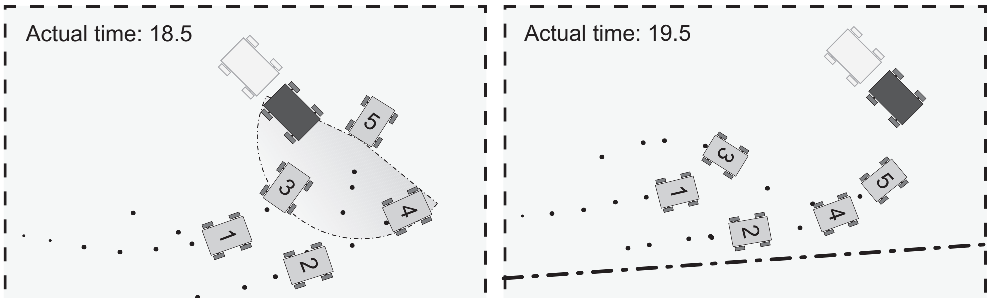
The last obstacle (group of robots detected in Fig. 2.9) is blocking the way to goal almost completely. The small passage along the wall is too narrow for the formation with desired shape. Nevertheless, the plan of the followers can be changed to avoid the obstacles at the price of a temporary contracting the formation as can be seen in the trajectories depicted in Fig. 2.10 where the desired task is accomplished.

The history of values of the cost function for the virtual leader is shown in Fig. 2.11. The perpendicular dot-dash lines denote times of the snapshots presented in Figures 2.5-2.10. The course of the value is smoothly decreasing which matches the theoretical results except for the points where the plan of the virtual leader was changed to avoid a newly detected obstacle. In the first case, when the obstacle is observed at time $t = 13.5s$ (see Fig. 2.7), the trajectory was changed significantly. Nevertheless, the increase of time to goal was smaller than the step between two iterations of the planning loop and so the inequality (2.5) holds.

A slight increase of the time to goal can be seen also at the time of the multiple

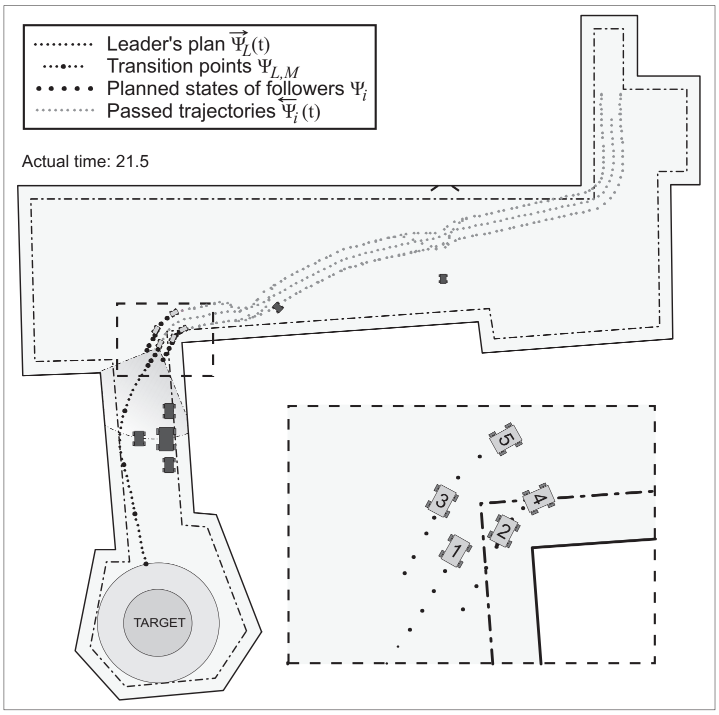

Leader's plan $\overrightarrow{\Psi}_L(t)$
Transition points $\Psi_{L,M}$
Planned states of followers Ψ_i
Passed trajectories $\overleftarrow{\Psi}_i(t)$
Actual time: 17.5
TARGET
5
3
4
1
2

(a) Beginning of the maneuver to avoid a dynamic obstacle. Shaded contour denotes position of the obstacle in Fig. 2.7.



(b) Zoomed details of the manoeuvre fulfilment. Shaded contour denotes position of the obstacle one second ago.

Figure 2.8: Snapshots of the reaction to unexpected movement of an obstacle that could not be avoided by the leader's plan.



Leader's plan $\overrightarrow{\Psi_L}(t)$
Transition points $\Psi_{L,M}$
Planned states of followers Ψ_i
Passed trajectories $\overleftarrow{\Psi_i}(t)$
Actual time: 21.5
TARGET
1
2
3
4
5

Figure 2.9: Solution of the situation with partly blocked corridor by a group of obstacles.

obstacle detection in Fig. 2.9. However in this case, mainly the second term in equation (2.22) is contributing which creates the peak in the course of final values. This effect, which is caused by the narrow passage between the obstacles and borders, does not influence practical implementation of the method yet it was not included in the theoretical part of this chapter.

Values of the cost function evaluating solutions computed by the followers are depicted in Fig. 2.12. It can be observed that in the cases of formation deformations caused by a dynamic obstacle, the value of the cost function increases rapidly which is followed by the fast convergence to the desired position. Let us analyze the first maneuver beginning at time $t = 9.5s$. At first, robot 3 is threatened by the obstacle which results in the first peak. In the next step, also the following robot 5 is forced to a jink. Here, the corresponding peak in the cost values is much higher. The obstacle continues in the collision course and so the deviation of robot 5 from the desired plan which is penalized by the first term in function (2.31) becomes bigger. The small peak in the cost values of robot 4 is contribution of the third term in cost function (2.31) penalizing the approach of robot 5. The same courses can be found during the second avoidance maneuver beginning at time $t = 17.5s$. The third interesting place that needs to be explained is the area of two peaks around $t = 22s$. At this time the formation is passing the narrow corridor between the group of obstacles and the wall of corridor. Although the formation is guided in the axis of the passage, only the followers closer to the detected obstacles (robots 2 and 4) are forced to keep bigger spacing. This is based on an assumption that the position of the newly detected objects could be determined

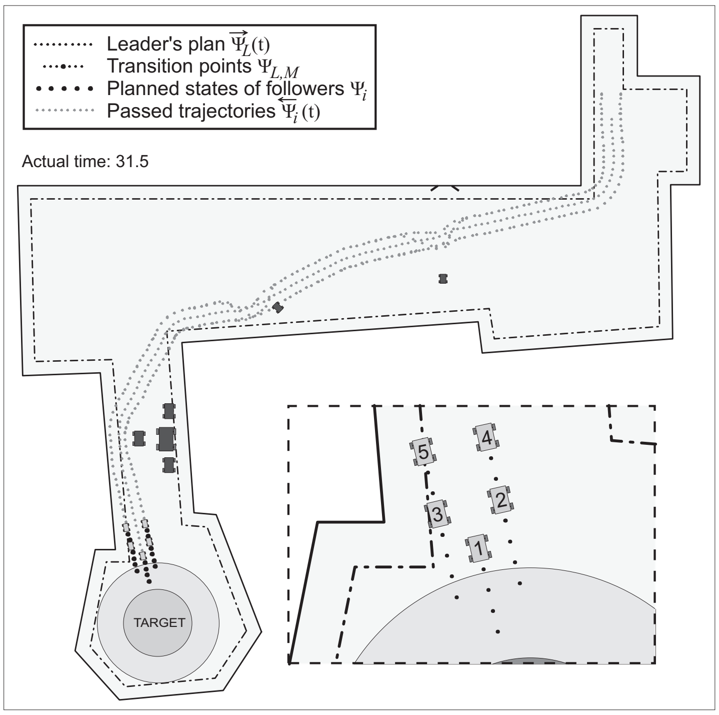

Leader's plan $\overrightarrow{\Psi}_L(t)$
Transition points $\Psi_{L,M}$
Planned states of followers Ψ_i
Passed trajectories $\overleftarrow{\Psi}_i(t)$
Actual time: 31.5
TARGET
1
2
3
4
5

Figure 2.10: Accomplished task with denoted trajectories of the robots.

imprecisely or that the obstacles are not static. This feature is optional and can be utilized according to the application.

Finally, we should analyze the costs of solutions obtained for robot 1. This follower is placed on the position of the virtual leader and its plan should deviate from the desired one only in the case of an endangerment by another member of the team. External obstacles are already avoided by the plan of the virtual leader and they are not contributing to the cost which remains close to the x-axis in Fig. 2.12 during the complete simulation.

2.5 Complicated maneuvers of formations

The experimental scenarios presented in the previous section have been solvable without the necessity to reverse the movement of the formation. To offer a complete and general solution of the *formation to target zone problem*, we should also consider the situation where the backward movement of the formation is needful.

The optimization problem that we have defined in Section 2.3.2 is general enough to provide a complete plan for the virtual leader including changes of the polarity of velocity $v_L(\cdot)$ without any modification. Furthermore, the negative velocity of the leader is also allowed by the controller constraints from equation (2.3). Unfortunately, such a plan, which is feasible for the virtual leader, can be applied only for the followers with $p_i(t) = 0$, $t_0 \leq t \leq t_f$. The problem with the followers situated behind the leader comes from the principle of the formation driving method as we have defined

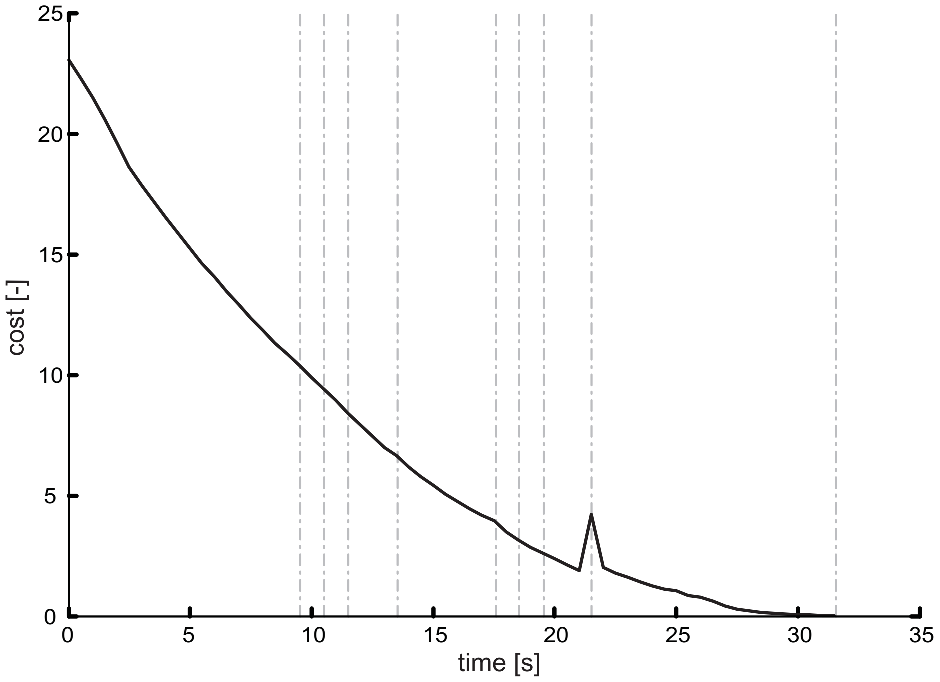


Figure 2.11: Values of the cost function of solutions obtained for the virtual leader. Dot-dash vertical lines denote the times of snapshots presented in Figures 2.5-2.10.

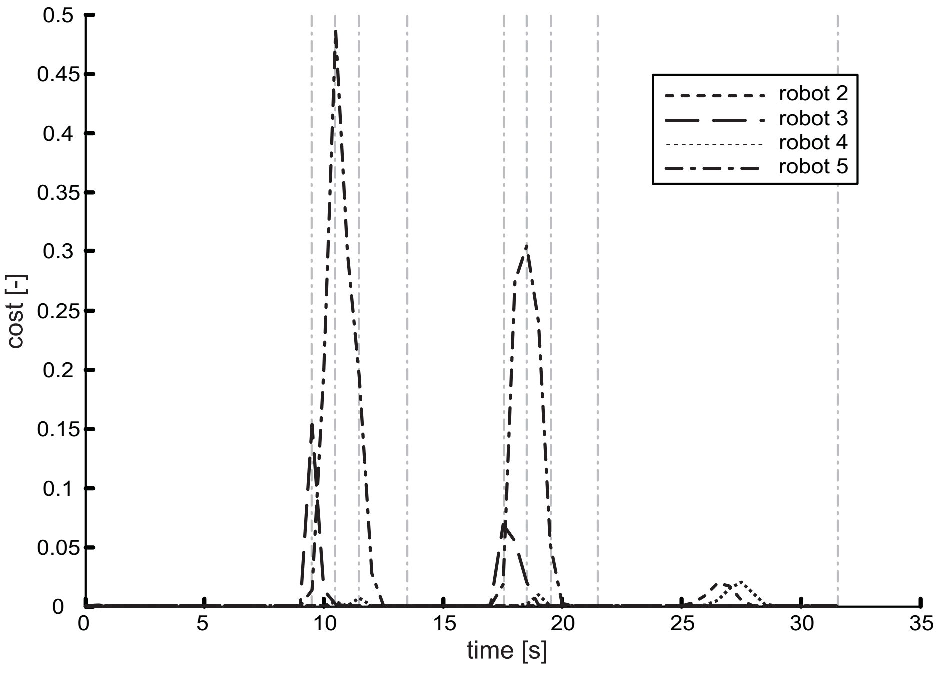
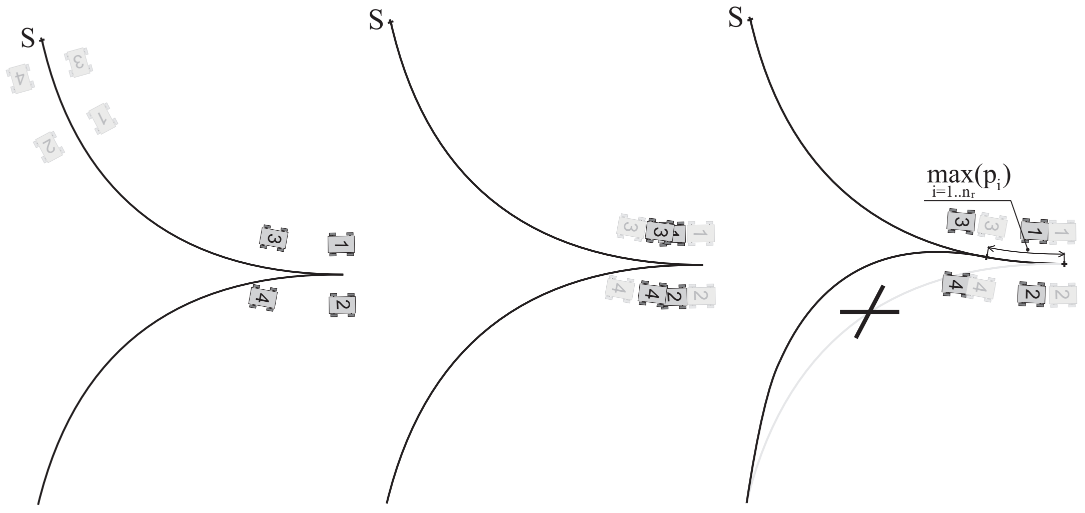


Figure 2.12: Values of the cost function of solutions obtained for the followers. Dot-dash vertical lines denote the times of snapshots presented in Figures 2.5-2.10.



(a) The formation in the initial position (shaded contours) and in the position where the sign of the leader's velocity is changed.

(b) Collisions of the vehicles that are simply following the desired path. The shaded contours denote the final position from Fig. 2.13(a).

(c) Feasible solution using modified desired path. The shaded contours denote the final position from Fig. 2.13(a).

Figure 2.13: Motivation for development of the 2 virtual leaders approach.

in Section 2.3.1. The trajectory of the followers is determined by states $\psi_L(t_{p_i(t)})$ in *travelled distance* $p_i(t)$ from $R_L(\psi_L(t))$ along $\overleftarrow{\Psi}_L(t)$ (for details see equation (2.13)). It means that the followers behind the leader continue with forward movement until the state where the leader changed the polarity of its velocity has been reached. The natural conception of the formation movement supposes that the entire group keeps a compact shape and therefore all members should change the polarity of their velocity in the same moment. This is not described in the formation driving concept published in Section 2.3.1, because each follower with different value of variable $p_i(t)$ approaches the state where the leader changed the polarity of its velocity in a different time. In addition the application of this concept could cause collisions or unacceptable disordered motion as demonstrated in the sequence of snapshots in Fig. 2.13(a)-2.13(b). An idea of a proposed solution of this problem is shown in Fig. 2.13(c). The desired path there has been modified in such a way that the formation can smoothly continue with a backward movement while keeping the desired shape. Nevertheless, such subsequence changes of the optimal leader's trajectory can be difficult in an environment with obstacles and the feasibility of the obtained solution is not guaranteed.

2.5.1 Concept of two alternating virtual leaders

To prevent the problems specified above, the basic method for the virtual leader has been extended to an approach employing two virtual leaders, one for forward movement and one for backward movement. Their leading role is always switched when the sign of the leader's velocity is changed and the virtual leader that is suspended temporarily becomes a virtual follower. The virtual follower traces the virtual leader similarly as the other followers to be able to undertake his leading duties at the time of the next switching.

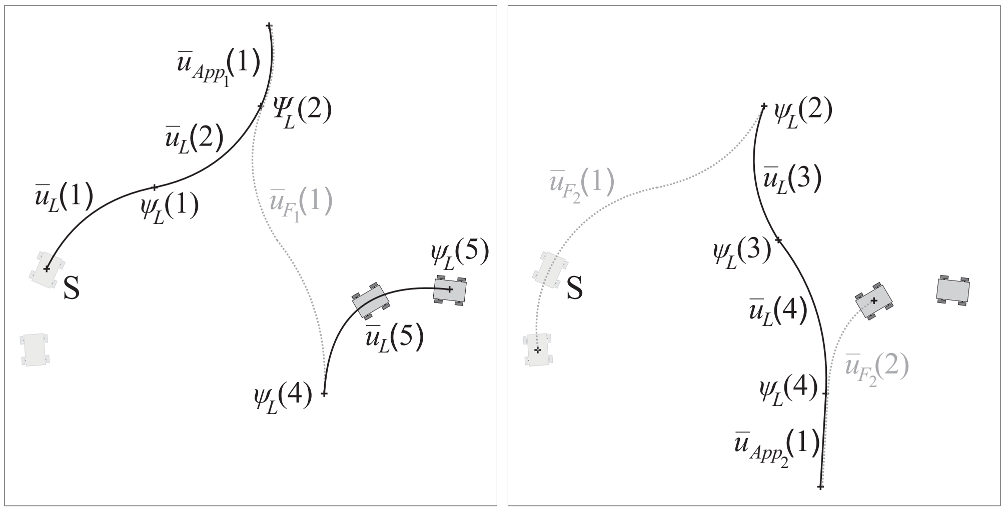
The plan for both leaders should be designed in one optimization step to guarantee that the solution of the *formation to target zone problem* is consistent and feasible. The easiest way is to employ a similar approach like the one we have presented in Section 2.3.2. The sequences of control inputs and states $\mathcal{U}_{L,N}$, $\mathcal{U}_{L,M}$, $\Psi_{L,N}$, $\Psi_{L,M}$ as well as the vector $\mathcal{T}^{\Delta}_{L,M}$ can contain necessary information for both leaders. The decision, which part of the vectors will be assigned to which leader can be simply done using the value of $v_L(\cdot)$ that can be collected in the vector $\mathcal{U}_{L,MN} = [\mathcal{U}_{L,N}, \mathcal{U}_{L,M}] \in \mathbb{R}^{2(N+M)}$. Let us define an ordered set of indexes of samples where the polarity of velocity is changed as $I_{sw} := [i : v_L(i)v_L(i+1) < 0]$, where $i \in \{1, \ldots, N+M-1\}$. The control inputs used for the first virtual leader can then be collected as

$$\begin{aligned}
\mathcal{U}_{L_1,MN} &= [\bar{u}_L(1), \bar{u}_L(2), \ldots, \bar{u}_L(I_{sw}(1)), \bar{u}_{App_1}(1), \bar{u}_{F_1}(1), \bar{u}_L(I_{sw}(2)+1), \\
&\quad \bar{u}_L(I_{sw}(2)+2), \ldots, \bar{u}_L(I_{sw}(3)), \bar{u}_{App_1}(2), \bar{u}_{F_1}(2), \ldots, \bar{u}_L(I_{sw}(2n_{sw_1}-2)+1), \\
&\quad \bar{u}_L(I_{sw}(2n_{sw_1}-2)+2), \ldots, \bar{u}_L(I_{sw}(2n_{sw_1}-1)), \bar{u}_{App_1}(n_{sw_2}), \bar{u}_{F_1}(n_{sw_2})], \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{if } n_{sw_1} = n_{sw_2} \\
\mathcal{U}_{L_1,MN} &= [\bar{u}_L(1), \bar{u}_L(2), \ldots, \bar{u}_L(I_{sw}(1)), \bar{u}_{App_1}(1), \bar{u}_{F_1}(1), \bar{u}_L(I_{sw}(2)+1), \\
&\quad \bar{u}_L(I_{sw}(2)+2), \ldots, \bar{u}_L(I_{sw}(3)), \bar{u}_{App_1}(2), \bar{u}_{F_1}(2), \ldots, \bar{u}_{App_1}(n_{sw_2}), \bar{u}_{F_1}(n_{sw_2}), \\
&\quad \bar{u}_L(I_{sw}(2n_{sw_1}-2)+1), \bar{u}_L(I_{sw}(2n_{sw_1}-2)+2), \ldots, \bar{u}_L(N+M)], \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{if } n_{sw_1} \neq n_{sw_2}
\end{aligned} \tag{2.38}$$

and the control inputs for the second virtual leader as

$$\begin{aligned}
\mathcal{U}_{L_2,MN} &= [\bar{u}_{F_2}(1), \bar{u}_L(I_{sw}(1)+1), \bar{u}_L(I_{sw}(1)+2), \ldots, \bar{u}_L(I_{sw}(2)), \bar{u}_{App_2}(1), \\
&\quad \bar{u}_{F_2}(2), \bar{u}_L(I_{sw}(3)+1), \bar{u}_L(I_{sw}(3)+2), \ldots, \bar{u}_L(I_{sw}(4)), \bar{u}_{App_2}(2), \bar{u}_{F_2}(3), \ldots, \\
&\quad \bar{u}_{App_2}(n_{sw_1}-1), \bar{u}_{F_2}(n_{sw_1}), \bar{u}_L(I_{sw}(2n_{sw_2}-1)+1), \bar{u}_L(I_{sw}(2n_{sw_2}-1)+2), \\
&\quad \ldots, \bar{u}_L(N+M)], \qquad\qquad\qquad\qquad\qquad\qquad \text{if } n_{sw_1} = n_{sw_2} \\
\mathcal{U}_{L_2,MN} &= [\bar{u}_{F_2}(1), \bar{u}_L(I_{sw}(1)+1), \bar{u}_L(I_{sw}(1)+2), \ldots, \bar{u}_L(I_{sw}(2)), \bar{u}_{App_2}(1), \\
&\quad \bar{u}_{F_2}(2), \bar{u}_L(I_{sw}(3)+1), \bar{u}_L(I_{sw}(3)+2), \ldots, \bar{u}_L(I_{sw}(4)), \bar{u}_{App_2}(2), \bar{u}_{F_2}(3), \\
&\quad \ldots, \bar{u}_L(I_{sw}(2n_{sw_2}-1)+1), \bar{u}_L(I_{sw}(2n_{sw_2}-1)+2), \ldots, \bar{u}_L(I_{sw}(2n_{sw_2})), \\
&\quad \bar{u}_{App_2}(n_{sw_1}-1), \bar{u}_{F_2}(n_{sw_1})], \qquad\qquad\qquad\qquad \text{if } n_{sw_1} \neq n_{sw_2},
\end{aligned} \tag{2.39}$$

where n_{sw_1} (resp. n_{sw_2}) is the number of time intervals in which the first (resp. the second) virtual leader has the leadership. This value can be computed as $n_{sw_1} = floor\,(l_I/2+1)$ (resp. $n_{sw_2} = floor\,((l_I+1)/2)$). The function $floor(X)$ rounds the value of X to the nearest integer towards minus infinity and constant l_I is length of the vector I_{sw}. Control inputs $\bar{u}_{F_1}(j)$, where $j \in \{1, \ldots, n_{sw_2}\}$, (resp. $\bar{u}_{F_2}(j)$, $j \in \{1, \ldots, n_{sw_1}\}$) are collected in the vector $\mathcal{U}_{F_1}$ (resp. $\mathcal{U}_{F_2}$). This control scheme is obtained by the formation driving method when the first (resp. the second) virtual leader is the virtual follower. The control inputs $\bar{u}_{App_1}(j)$, where $j \in \{1, \ldots, n_{sw_1}\}$,



(a) Trajectory of the first virtual leader. (b) Trajectory of the second virtual leader.

Figure 2.14: Trajectories and appropriate control inputs of the two virtual leaders going from the initial point S. The solid black curves denote trajectories where the virtual leader is leading the formation while the dotted gray curves denote the trajectories where the virtual leader is the virtual follower. The initial positions of the robots are depicted by gray shadows.

(resp. $\bar{u}_{App_2}(j)$, where $j \in \{1, \dots, n_{sw_2} - 1\}$) gathered in vector $\mathcal{U}_{App_1}$ (resp. $\mathcal{U}_{App_2}$) are applied for the formation driving in the appendixes to the continuous path. This movement is important to satisfy constraints (2.23) and (2.24) applied for the original vectors $\mathcal{U}_{L,N}$, $\mathcal{U}_{L,M}$, $\Psi_{L,N}$, $\Psi_{L,M}$, $\mathcal{T}^{\Delta}_{L,M}$. This means that the formation has to travel forward from the states $\psi_L(I_{sw}(j))$, where $j \in \{1, \dots, n_{sw_1} + n_{sw_2} - 1\}$, to the distance $d_{App} = \max_{i \in n_r}(|p_i|)$ and thereby to shift the new leader to the position of the old one. An example illustrating this approach is presented in Fig. 2.14 where the parameters and outputs of the optimization are: $N = 0$, $M = 5$, $I_{sw} = \{2, 4\}$, $\mathcal{U}_{L_1} = [\bar{u}_L(1), \bar{u}_L(2), \bar{u}_{App_1}(1), \bar{u}_{F_1}(1), \bar{u}_L(5)]$, $\mathcal{U}_{L_2} = [\bar{u}_{F_2}(1), \bar{u}_L(3), \bar{u}_L(4), \bar{u}_{App_2}(1), \bar{u}_{F_2}(2)]$.[1]

The shape of the appendixes, which are introduced above, does not influence the total driving time. Nevertheless, in the environment with obstacles also the movement of the formation along such a part of the trajectory must be collision free which has to be considered in the cost function during the optimization. The minimal length of this additional trajectory is defined by the size of the formation and therefore only the curvature of the leading robot is not uniquely determined. The extended optimization vector describing the complete trajectory from the actual position of the first virtual leader to the target region is then defined as $\Omega_L = [\Psi_{L,N}, \mathcal{U}_{L,N}, \Psi_{L,M}, \mathcal{U}_{L,M}, \mathcal{T}^{\Delta}_{L,M}, \mathcal{K}_{App}] \in \mathbb{R}^{5N+6M+l_I}$. The variables $K_{App_1}(j)$, where $j \in \{1, \dots, n_{sw_2}\}$, and $K_{App_2}(j)$, where $j \in \{1, \dots, n_{sw_1} - 1\}$, applied for the movement of the virtual leaders in the appendixes, are collected in the vector $\mathcal{K}_{App} = [\mathcal{K}_{App_1}, \mathcal{K}_{App_2}]$. The remaining control inputs $v_{App_1}(j)$,

[1]The new vector $\mathcal{T}^{\Delta}_{L_1,MN}$ (resp. $\mathcal{T}^{\Delta}_{L_2,MN}$) will be compiled similar as the vector $\mathcal{U}_{L_1,MN}$ (resp. $\mathcal{U}_{L_2,MN}$), because the variables $\bar{u}(\cdot)$ and $\Delta t(\cdot)$ are inseparably joined.

where $j \in \{1, \ldots, n_{sw_2}\}$, and $v_{App_2}(j)$, where $j \in \{1, \ldots, n_{sw_1} - 1\}$, as well as the values $\Delta t_{App_1}(j)$, where $j \in \{1, \ldots, n_{sw_2}\}$, and $\Delta t_{App_2}(j)$, where $j \in \{1, \ldots, n_{sw_1} - 1\}$, are uniquely determined by the distance d_{App}, controller constraints, and by the effort to minimize the total performance time.

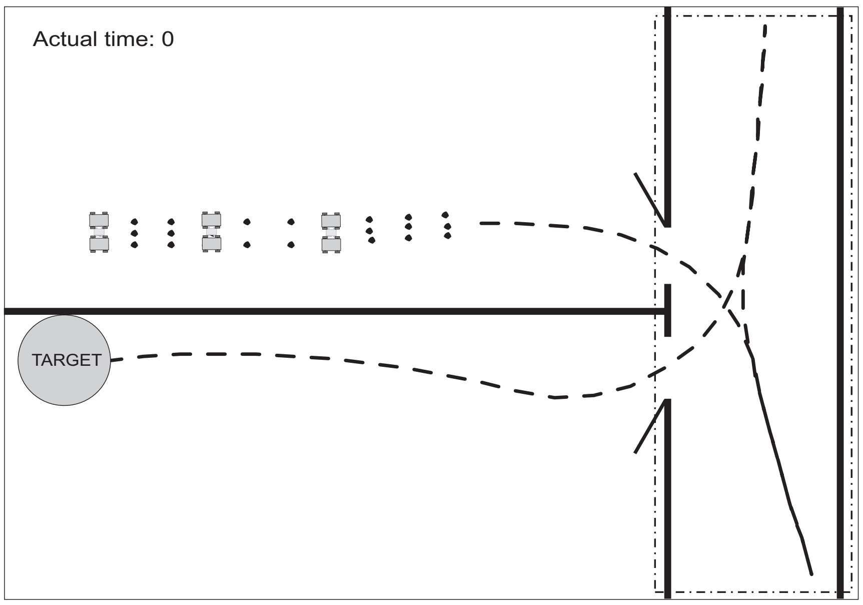
Remark 2.5.1. The length of such an optimization vector can vary during the optimization because of the last component $\mathcal{K}_{App}$ whose size depends on variable l_I. This requires the utilization of an optimization method with variable length of the solution or setting of l_I as $l_I = N+M-1$, which is the maximal amount of possible changes. The unused variables in the case of a smaller number of switching will be not contributing to the cost function.

2.5.2 Experimental results

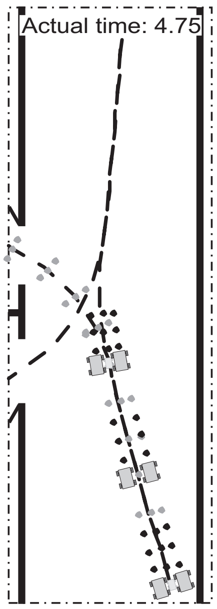
2.5.2.1 Reverse driving

A passage of sharp curves on narrow roads is one of the most frequent examples of the car-like robot driving which is solvable by the utilization of the reverse driving. This phenomenon is even more obvious in applications of the formation driving where the maximum turning radius is increased as shown in equations (2.14). Two such situations extracted from scenarios of indoor mobile robotics have been chosen to verify the functionality, robustness and general applicability of the proposed method. The first task is to navigate a formation from one room to the neighboring room through a narrow corridor. In the scenario, the relative position of entrance doors of the rooms and the width of the corridor do not allow to simply turn the formation and a more complicated movement is necessary. In the first snapshot of the simulation presented in Fig. 2.15(a), a solution of the formation to the target zone problem is shown. As highlighted in the following sequences of the formation movement, the formation is led by the first virtual leader to safely pass trough the first door. The heading of the robots is optimally oriented to reach the second door after a sequence of backward and forward movements. In Fig. 2.15(b), the leading role is switched from the first leader to the second one and the formation heads backwards with approximately half speed. In Fig. 2.15(c), the movement is reversed again and the group is guided by the first leader through the door to the second room. A final snapshot with denoted trajectories of the followers is presented in Fig. 2.15(d).

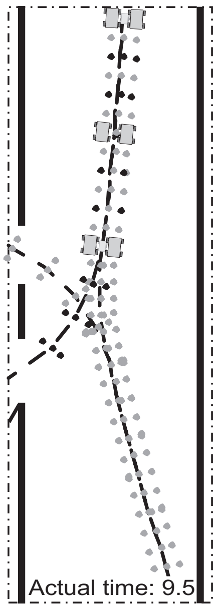
In Fig. 2.16, values of the heading of all followers during the movement are plotted. It provides, together with the perpendicular dot-dash lines denoting times of snapshots in Fig. 2.15(b)-2.15(c), complete information about the entire maneuver of the formation. One can see that the plotted curves conspicuously form groups accordant with belonging to particular rows (followers with identical parameter $p_i(t)$) of the formation. In each row the followers should keep the same heading at the same time. The previous fact arises from the basic fundamentals of the formation driving explained in Section 2.3.1. In reality, the formation driving is influenced by all factors collected in cost function (2.31) and the values of heading are more or less deviating. In the picture, one can also see the shift of the grouped curves which depends on the space in between of the rows. Finally, the short intervals with constant heading or even with



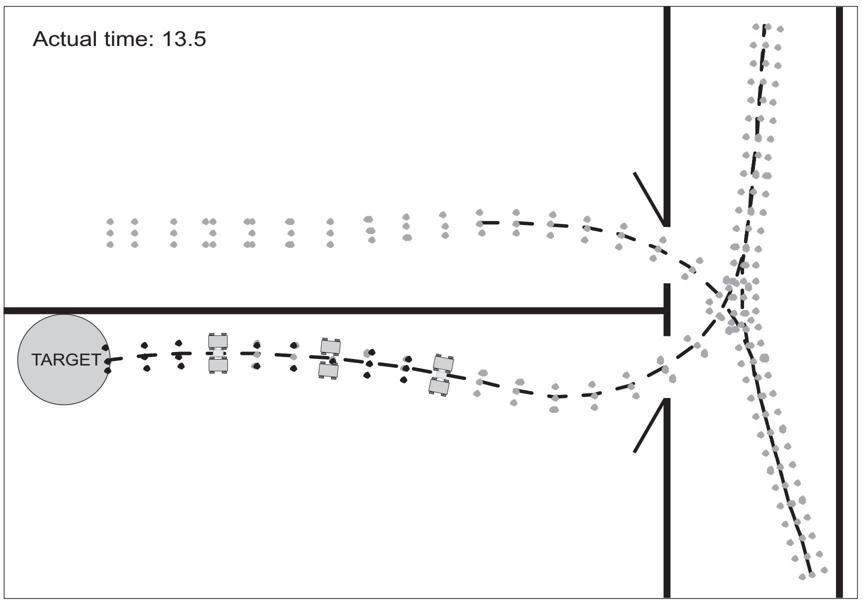
(a) The initialization of the task with denoted complete plan.



(b)



(c)



(d) Final snapshots of the accomplished mission.

Figure 2.15: Snapshots of simulations of the formation driving in an office environment. The task requires the switching of leading rules two times. Positions of the vehicles in time of switching is shown in (b) and (c).

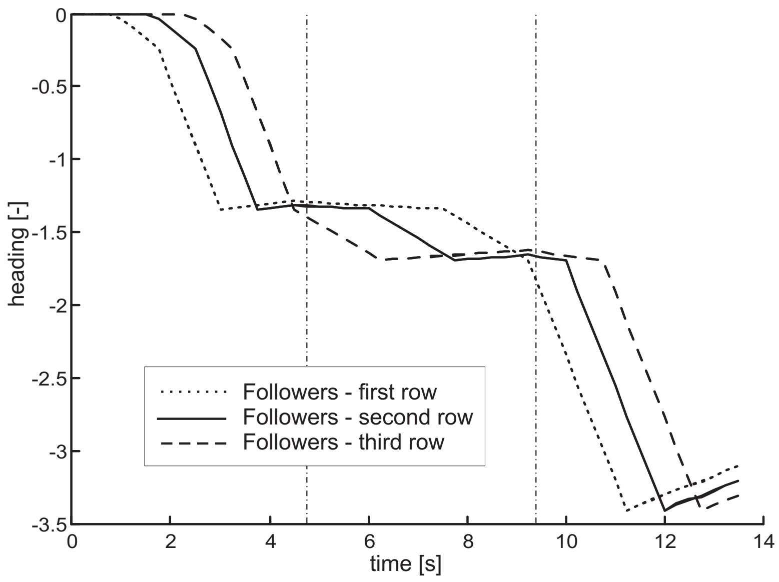


Figure 2.16: The values of the followers' heading during the simulation presented in Fig. 2.15.

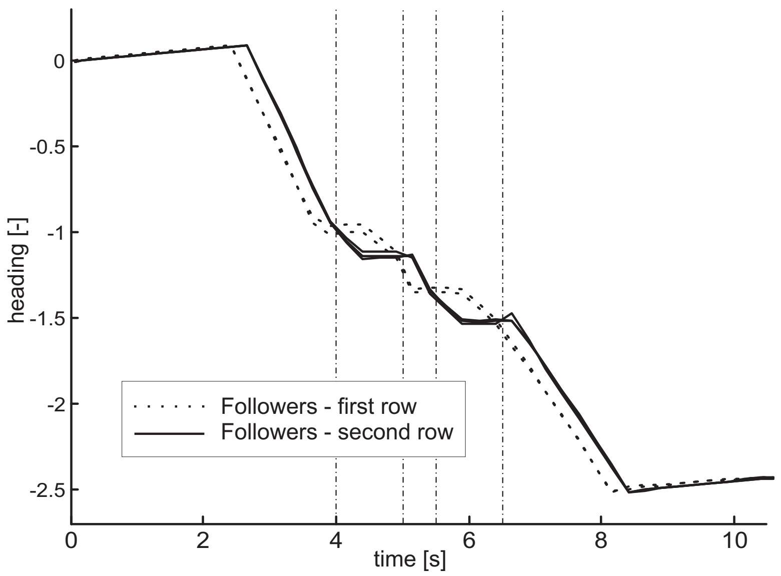


Figure 2.17: The values of the followers' heading during the simulation presented in Fig. 2.18.

slightly growing values of heading are caused by the movement of the followers in the appendixes added to the trajectory.

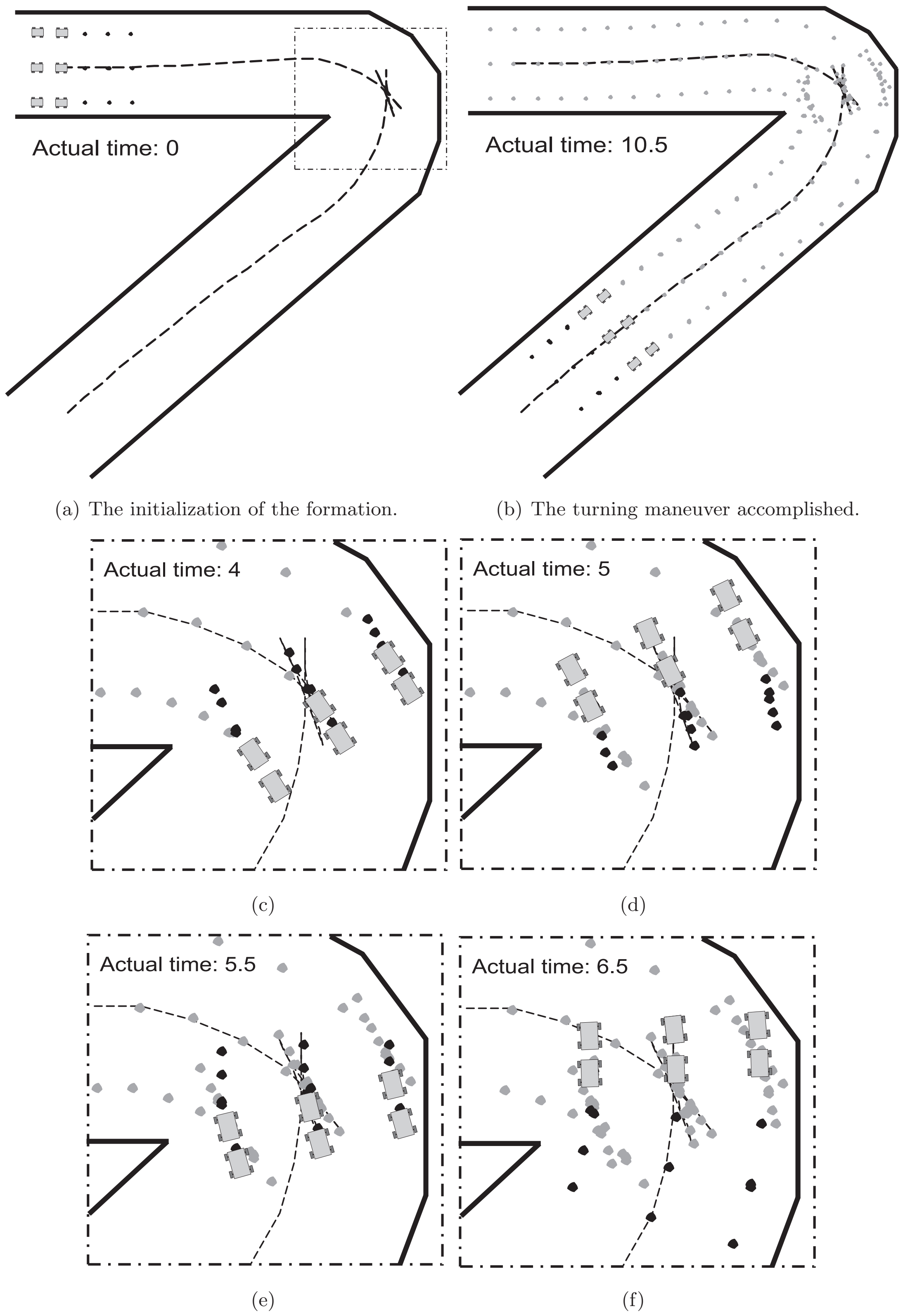
The second example of the composed maneuver, which was completely designed in one optimization step, is shown in Fig. 2.18. The goal of the mission is to follow a narrow corridor with a sharp curve and to keep the compact shape of the formation. The optimal solution of this problem contains a four times changed velocity of the leader which is enforced by the structure of the workspace. The initial position of the formation and a schematic plan of the movement is presented in Fig. 2.18(a). The following snapshots show zoomed parts of the simulation at times when the leadership was changed (see Fig. 2.18(c)-2.18(f)). The last snapshot of the simulation with plotted passed trajectories of the followers is presented in Fig. 2.18(b). Finally, the sequential change of the followers' heading from the initial values $\theta_j(t_0) = 0$ to the desired values $\theta_j(t_f) = -3\pi/4$, where $j \in \{1, \ldots, 6\}$, with denoted times of snapshots from Fig. 2.18 is plotted in Fig. 2.17.

2.5.2.2 Turning 180 degrees

Turning 180 degrees (also called a U-turn) is one of the most challenging tasks in the utilization of car-like robots and mainly formations of car-like robots that cannot simply turn on the spot and require a more complicated maneuver. To our best knowledge, there is no general method providing an optimal solution in environment with arbitrary static as well as dynamic obstacles available in literature since now.

In the scenario presented in Fig. 2.19 a formation of 4 robots is aimed to reach the target region, which is situated behind the group. In accordance with real application, the maximum forward and backward velocities of the virtual leader of the formation were set unsymmetrically: $v_{min,L} = -v_{max,L}/2$. Putting the final region sufficiently far behind the formation, the optimal solution of the *formation to target zone problem*,as is defined in this thesis, is to turn the formation and then continue forward to reach the desired area. Such a maneuver, which contains two switching between the virtual leaders, is denoted by dashed curves in Fig. 2.19(a). The movement of the rovers following the obtained plan is presented in snapshots of the simulation in Fig. 2.19(b) and in Fig. 2.19(c), while the complete trajectories passed during the turning can be seen in Fig. 2.19(d).

The same situation that was solved in the previous simulation is modified in Fig. 2.20 using obstacles with known positions. The new solution of the task keeps the formation outside the obstacles during the whole maneuver. Beyond this, the ability to avoid dynamic obstacles with a collision course detected by the rear sensors of the robots during the moving backwards is demonstrated in the simulation (see snapshots in Fig. 2.20(b) and in Fig. 2.20(c)). The simulation was interrupted when the vehicles reached the same distance from the target zone like in the simulation from Fig. 2.19. By the comparison of the snapshots presented in Fig. 2.19(d) and Fig. 2.20(d), one can see that the total time of the solution in the environment with obstacles was increased only slightly.

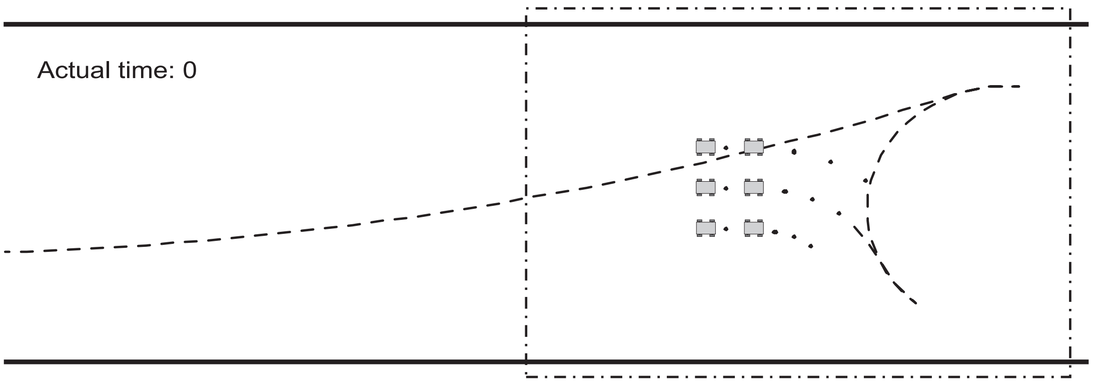


(a) The initialization of the formation. (b) The turning maneuver accomplished.

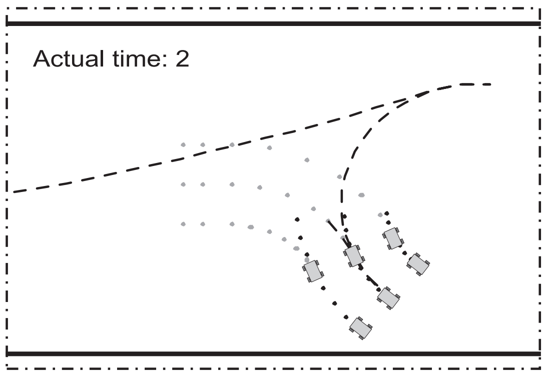
(c) (d)

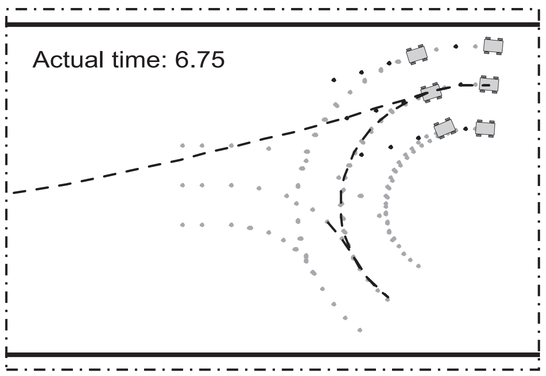
(e) (f)

Figure 2.18: The formation turning in a sharp curve. The snapshots of the leadership switching between the virtual leaders are depicted in (c)-(f).



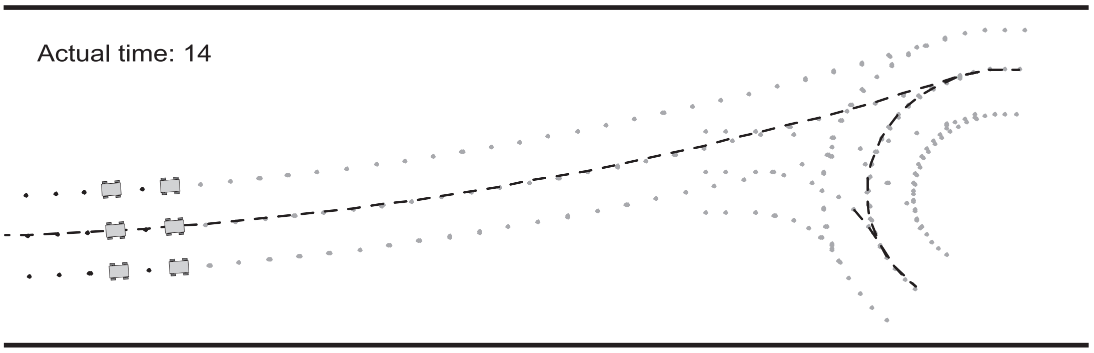
(a) The initial position of the formation with a complete plan for both virtual leaders denoted by a dashed curve.





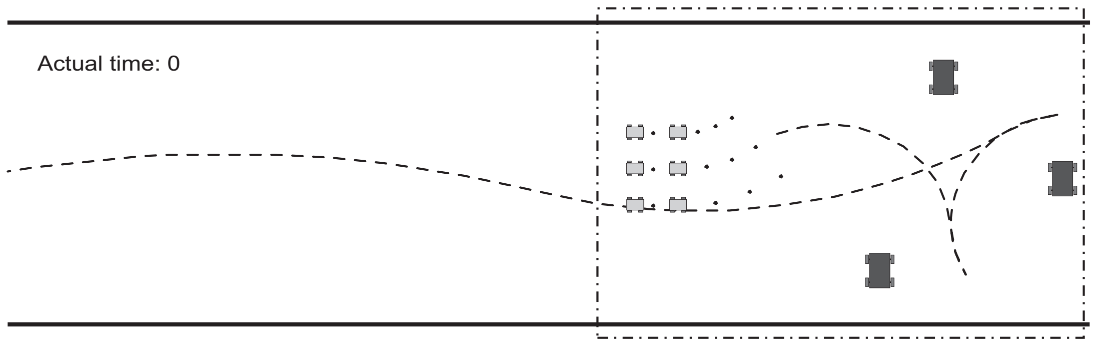
(b) The second virtual leader undertakes the leadership.

(c) The leadership is returned to the first virtual leader.

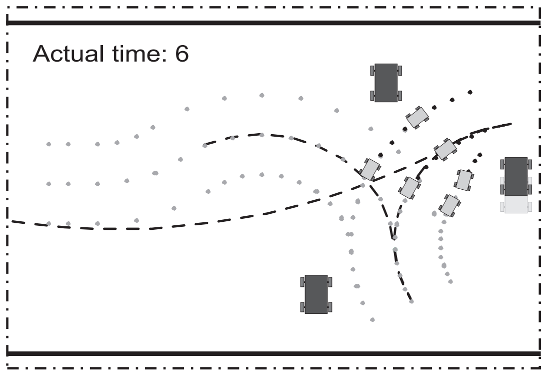


(d) The accomplished task with denoted trajectories of the robots.

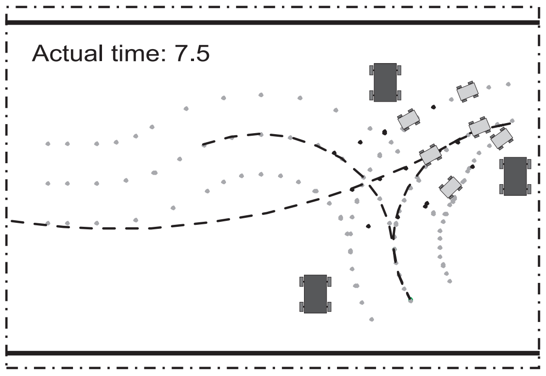
Figure 2.19: Snapshots of turning 180 degrees on the obstacle free road.



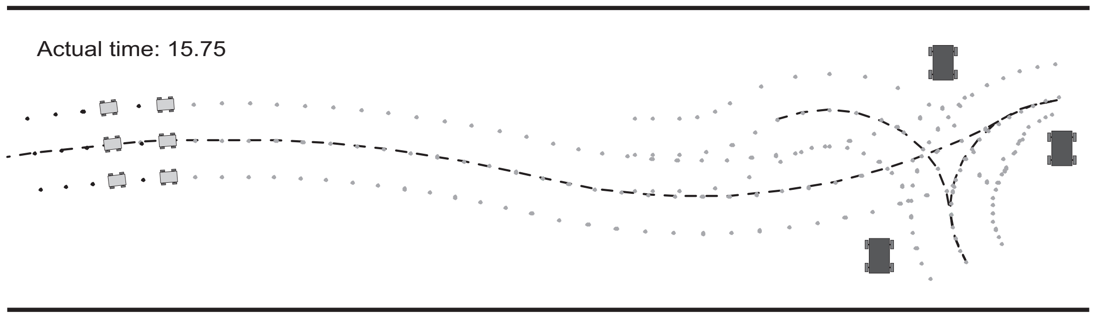
(a) The initial position of the formation with a complete plan for both virtual leaders denoted by a dashed curve.



(b) The detection of the movement of one obstacle and the consequential reaction of the followers.



(c) The followers successfully passed by the obstacle and they are going back to their positions within the formation.



(d) The accomplished task with denoted trajectories of the robots.

Figure 2.20: Snapshots of turning 180 degrees with the same initialization like in Fig. 2.19 but in an environment with static as well as dynamic obstacles.

3

Formation coordination with path planning in space of multinomials

In this chapter we will describe a method developed to simplify the complexity of the problem solved in Chapter 2 with a great number of variables that need to be optimized. The overall trajectory of formation $\mathfrak{F}$ will not be defined by the states and control inputs of the leader, but by a string of multinomials that are determined by a reduced amount of information. The equality constraints $h_{T_N}(\cdot)$ and $h_{T_M}(\cdot)$ from equation (4.7) that are difficult to satisfy for most of the optimization methods are excluded in the proposed approach and the controller constraints defined in Section 2.1.2 are penalized within the cost function. Therefore in this method, only absolute values of the optimized variables will be constrained, which extends the group of optimization techniques that are applicable for the *formation to target zone problem.* Finally, this approach provides a general frame that can be later utilized for the airport snow shoveling which is our target application.

3.1 Literature review

The core of the formation driving method presented in this chapter is a path planning algorithm for the virtual leader of the formation. Before the detailed description of the method, we will provide a short survey of existing path planning approaches that have been our guidance during the development. A comprehensive overview of standard path planning and obstacle avoidance methods can be found in [104] or in the first part of [105]. Algorithms presented there can be divided into two main groups: local methods and global methods.

Local approaches only find an optimal direction from the actual position using an image of the local area of the robot. While the limited amount of information processed by these approaches leads to a low computational time, generated trajectories are not guaranteed to be globally optimal. Moreover, the unavailability of a full path from the beginning can cause problems to higher-level strategy planning methods that can require an estimation of time to achieve the goal for their decision process. The most widely used local path planning method is Potential Field (PF) [94]. This approach is based on the description of the scenario using a function that consists of two parts: one inspired with the repulsive force and one inspired with the attractive force. The repulsive part describes the influence of the obstacles in the workspace, while the attractive part expresses the intention to go to the desired point. In other words, the repulsive function has a maximum (usually infinity) in the collision distance to each obstacle and decreases with the growing distance from the obstacles. The attractive function has a global minimum in the goal of the robot and uniformly grows with the distance from the goal. The optimal direction of the movement is then chosen in the direction of the gradient descent of the sum of these functions. The biggest problem of this basic approach is that the movement can get stuck in a local minimum and also a global optimal path is not guaranteed to be found. In the formation driving techniques proposed in this thesis, modifications of the repulsive part of PF, which is employed for obstacle avoidance behavior, are included in the trajectory planning. The second local technique we would like to mention, is a so-called Vector Field Histogram (VFH) [21], which was originally developed for obstacle avoidance of robots equipped with a sonar. VFH uses a histogram of distances to the closest obstacle in each direction much like the rotating sonar exploring a space in 360 range. The directions whose distance to the closest obstacle is smaller than a threshold are selected from this histogram. The heading with the smallest declination to the line connecting of the actual and desired positions of the robot is chosen from this selection. VFH cannot solve situations with a high density of obstacles and with U-shape obstacles. Both types of the workspace configurations cause movement oscillations.

The global path planning approaches construct a complete path from the actual position to the goal position of the robot. They utilize the all available information about the workspace that is gathered to the known map as their input. The processing of the data on the map and the searching in a generally bigger space of solutions increase the total computational time in comparison with the local path planning techniques. We selected three the most important representatives from the global path planning methods. All of them are based on a planar graph and on a searching for the shortest path connecting the actual position of the robot with the desired one. In the first approach, Visibility Graph (VG) [136], the nodes of the graph are vertices of polygons representing obstacles and the edges are lines connecting each pair of the nodes whenever it is possible without any intersection with obstacles. It means that two vertices are connected if there is a direct visibility between them. The path found by this algorithm is typically close to the obstacles, which can raise the probability of collisions. The second approach, Occupancy Grid (OG) [22], divides the workspace into disjoint cells that cover the environment completely. The nodes of the graph are the centers of the cells that are not overlapping with any obstacle and the edges of the graph connect the nodes that are next to each other in the grid. As the third example we have chosen Voronoi Diagram (VD) method [9]. In this approach, the workspace is divided into disjoint cells too. Here, each cell represents a region around an obstacle. For the points within the region, it counts that the Euclidean distance from the point to the obstacle is smaller than the distance to the other obstacles. The edges of the graph are then the borders of the cells and the nodes are the crossings of the borders. The VD method will be utilized for the advanced initialization of the formation driving algorithms presented in Section 4.1.

One can find several extensions and combinations of these standard algorithms as well as their modifications for special applications. For example in [28], authors proposed a method for finding a Euclidean shortest path between two distinct locations in a planar environment. In the paper, OG and VD approaches are combined within a framed-quadtree. This method provides the accuracy of high resolution grid-based path planning methods and the efficiency of quadtree-based techniques together. An approach combining a modified probabilistic road-map method with a potential based variational technique is applied for a path planning of hyper-redundant manipulators in [39]. The optimal path planning algorithm presented in [86] is based on a higher geometry maze routing algorithm. It applies a pixel-based plane for the navigation of a single differential drive robot. In [185], a grid-based method is presented. This algorithm is designed for real-time path planning in dynamic environments applicable to situations in which targets and barriers are permitted to move. The robot's workspace is represented by a topologically organized map where each grid point has only local connections to its neighboring grid points from which it receives information in real time. The approach is extended in [186] by incorporated safety margins around obstacles using local penalty functions. In [81], a global path planning in a dynamic partly known manufacturing environment is proposed. The approach uses a deterministic cost for the known part of the robot's workspace, and an uncertainty cost dynamically updated by available sensor data for the unknown part of the workspace. Finally, a collision-free path planning for diamond-shaped robots based on the retraction of free space onto the VD is presented in [180].

Another frequent direction of research of the path planning methods is a tendency to employ nature inspired algorithms for the inclusion of artificial intelligence to autonomous decision making. The proposed algorithm enables learning from a previous experience (Neural Networks (NN) techniques) as well as global searching in a usually large space of solutions that contains local extremes (Evolutionary Algorithms (EA)). A combination of both approaches is shown in [132]. Here, a NN method is employed to identify unstationary robot's dynamics, and a Genetic Algorithm (GA) is utilized to search in a space of solutions for a suboptimal path planning of an agricultural mobile robot. In [7], a path planning algorithm using GA for mobile robots operating in cluttered environments is described. A path planning method using a dynamic wave expansion NN is proposed in [107]. This algorithm is developed for mobile robots as well as for robotic manipulators working in a dynamic environment. An approach using an advanced EA called the memetic algorithm is presented in [171]. The method offers an optimal collision free path for mobile robots. The paper [194] focuses on the problem of point-to-point trajectory planning for flexible redundant robot manipulators in joint space.

A logical extension of the 2-D path planning methods is required for the utilization of autonomous systems in a 3-D environment. An approach for Autonomous Underwater Vehicles (AUVs) is proposed in [25]. The method uses a precomputed look up table of 3D-routes that are modified according to the current situation. A more advanced algorithm for AUVs using a GA for path planning in an ocean environment with strong currents is presented in [6]. The third and the most up-to-date AUVs' path planning method is described in [145]. The approach uses a continuous version of the $A^\star$ algorithm to find an optimal solution for nonholonomic torpedo-like vehicles in anisotropic environments with currents. The application of path planning in a 3-D environment is not limited to UAVs but the utilization of Unmanned Aerial Vehicles (UAVs) is even more common. The present state of the art of path planning algorithms for UAVs can be found for example in [95; 131]. These approaches must take into account not only the maximum but also the minimum possible speed determined by the aerodynamic of the aircraft which sets bounds to robots' maneuverability. Two real-time path planning schemes employing limited information for fully autonomous UAVs in a hostile environment are offered in [95]. In both algorithms, a convergence to a given target point is proven and series of safe waypoints whose risk is always less than a given threshold value are produced. Finally, a differential EA to produce 3-D B-Spline trajectories for the off-line as well as the on-line cooperative UAVs path planning is applied in [131].

Several studies are interesting not only in the optimal path of the robot, but they produces also an optimal velocity profile along the path. These trajectory planning approaches usually differ with a different type of robot. For example in the algorithm presented in [187], a time optimal trajectory consisting from lines and circles has been designed for differential drive robots. Besides this the outputs of the method are limits of velocity and bounds on acceleration depending on path features in the consideration of robots' dynamics. An approach of a collision-free, optimal trajectory planning problem for a point-robot moving between two configurations inside a convex polygon is presented in [2]. Another method for the time-optimal trajectory planning of omnidirectional vehicles in environment without obstacles can be found in [12]. The last example of a time-optimal trajectory planning algorithm developed for a high-speed cable-based parallel manipulators is presented in [16].

Unfortunately, the approaches mentioned above are not directly applicable for the path planning of the car-like robots' formations with a time variant shape. Such a method has to produce a collision free path with characteristics feasible for the kinematics of the formation (e.g.: continuous second derivative, radius satisfying the condition (2.15)). In addition it should provide an optimal output even if a feasible path for the complete formation does not exist. This ability requires to incorporate the skill that enables a shrink of the formation directly to the path planning. The temporary decrease of $\max_{i\in\{1\ldots n_r\}}(q_i(t))$ weakens the condition (2.15) and enables to pass passages that are too narrow for the wide-spread formation in the desired shape. Furthermore even a short, "slightly" unfeasible path can be preferable in some applications to a feasible, but longer solution (e.g. shortcut using a narrow tunnel through a hill where the formation needs to be shrinked). Our proposed algorithm incorporates all these requirements.

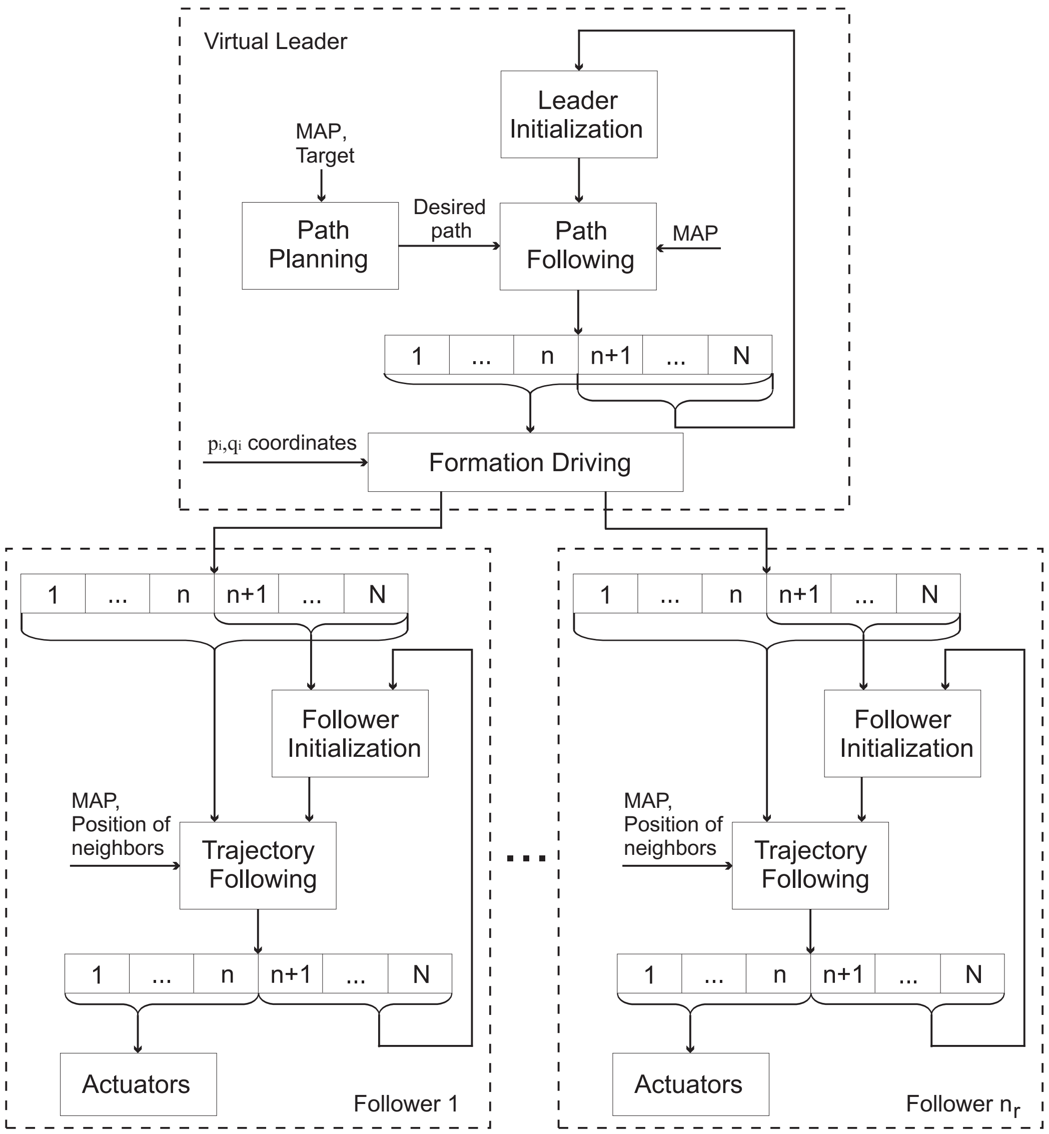


Figure 3.1: A diagram of the entire formation driving system based on the following of a preplanned path.

3.2 System overview

In the following section, the differences between the presented approach using path planning in a space of multinomials and the system described by the diagram in Fig. 2.3 will be highlighted. It is obvious from the Fig. 2.3 and Fig. 3.1, that the entire block which is multiplied for each follower is equivalent in both methods and so the only divergence lies in the first block, *Virtual Leader*. In the first approach introduced in Chapter 2, the trajectory planning and the optimal control design for the virtual leader is solved together in one optimization step. The obtained leader's trajectory consists of two main parts: i) $\Psi_{L,N}, \mathcal{U}_{L,N}$ used for the optimal control, and ii) $\Psi_{L,M}, \mathcal{U}_{L,M}, \mathcal{T}^{\Delta}_{L,M}$ used for the path planing to the desired target region. In the algorithm proposed in this chapter, the formation coordination is splitted into two separate modules: *Path Planning* and *Path Following*. This enables to reduce the complexity of the optimization but such a structure can only provide suboptimal solutions, because the direct inclusion of the kinematic model to the path planning is missing.

In general, the module *Path Planning* can be realized using any path planning method feasible for the formation of car-like robots. In this thesis, two original path planning approaches will be proposed. The first algorithm presented in Section 3.3.1 offers an optimal solution for a formation going to desired state in scenarios with various obstacles. The second path planning approach is adapted to the specific application of the airport snow shoveling, which will be described in Chapter 5.

The module *Path Following*, described in Section 3.4, is based on the RHC concept derived from the approach presented in Chapter 2. Here, the obtained leader's trajectory is formed only by the sequences $\Psi_{L,N}, \mathcal{U}_{L,N}$ that are used for the *Formation Driving* module and partly recycled in the *Leader Initialization* module similarly as in the previous method from Fig. 2.3.

3.3 Path planning for virtual leaders

3.3.1 Method description

The path planning approach presented in this section is based on a searching through a space of positions $\bar{p}_L(\cdot)$. This is the biggest difference from the approach presented in Section 2 where the space contains the complete states of the virtual leader $\psi_L(\cdot)$ but also the control inputs $\bar{u}_L(\cdot)$. Here, the final path of the robot is determined by transitions $\bar{p}_L(k-1) \overset{\varphi(k,\cdot)}{\rightarrow} \bar{p}_L(k)$, where $k \in \{1, \ldots, \tilde{n}\}$, between each pair of neighboring positions. The paths $\varphi(\cdot,\cdot)$ have been chosen as $\varphi(k,\cdot) \in C^1$, for $k \in \{1, \ldots, \tilde{n}\}$, where C^1 is the set of cubic splines with a smooth first derivation and $\tilde{n}$ denotes the amount of splines in the string.

In the same way as in the approach introduced in Section 2, the first position $\bar{p}_L(0)$ is equal to the actual position of the robot. Furthermore, here the last position $\bar{p}_L(\tilde{n})$ is fixed in a desired position of the virtual leader of formation. Usually, it is the center of the target region S_F or a point on the border of S_F. Therefore, the desired position of the formation needs to be determined as an input of the method.

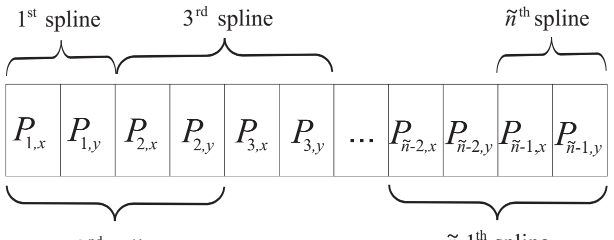
The mathematic notation of the k-th cubic spline in the string is

$$\varphi(k,s) \quad = \quad A_k s^3 + B_k s^2 + C_k s + D_k, \tag{3.1}$$

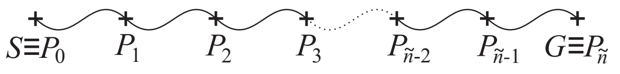
where s is within the interval $\langle 0, 1\rangle$ and the constants A_k, B_k, C_k, D_k can be computed according to [191] as

$$\begin{aligned} A_k &= 2P_{k-1} - 2P_k + P'_{k-1} + P'_k \\ B_k &= -3P_{k-1} + 32P_k - 2P'_{k-1} - P'_k + 1 \\ C_k &= P'_{k-1} \\ D_k &= P'_k. \end{aligned} \tag{3.2}$$

The control points of the splines P_k are defined as $P_k := \bar{p}_L(k)$, where $k \in \{0, \dots, \tilde{n}\}$. In 2D case, the whole string of the splines is uniquely identified by $8\tilde{n}$ variables as $\varphi(k,\cdot) = \{\varphi_x(k,\cdot), \varphi_y(k,\cdot)\}$, $k \in \{1, \dots, \tilde{n}\}$, but some of the variables are determined by the path planning requirements. 8 variables, P_0, P'_0, $P_{\tilde{n}}$ and $P'_{\tilde{n}}$ (x and y coordinates for each position) are determined due to known initial and desired states of the formation[1]. The continuity of the first and the second derivative in the whole path is guaranteed by $6(\tilde{n}-1)$ equations. Therefore, only $8\tilde{n}-8-6(\tilde{n}-1) = 2(\tilde{n}-1)$ degrees of freedom define the whole path. This conforms to the positions of the points P_k, where $k \in \{1, \dots, \tilde{n}-1\}$, in the spline connections that need to be used as variables in the optimization method. The optimization vector describing the complete path will later be denoted $\overrightarrow{x}$. The whole path representation used in our method is shown in Fig. 3.2.


1st spline
3rd spline
ñth spline
$P_{1,x}$ $P_{1,y}$ $P_{2,x}$ $P_{2,y}$ $P_{3,x}$ $P_{3,y}$ ... $P_{\tilde{n}-2,x}$ $P_{\tilde{n}-2,y}$ $P_{\tilde{n}-1,x}$ $P_{\tilde{n}-1,y}$
2nd spline
ñ-1th spline

(a) Sequence of variables obtained as a result of optimization process.



(b) Path composed from $\tilde{n}$ multinomials.

Figure 3.2: The representation of the solution with a corresponding path.

Having defined the optimization vector, we can transform the path planning problem to the minimization of cost function $f(\overrightarrow{x})$, which is

$$\min f(\overrightarrow{x}), \tag{3.3}$$

[1]The desired state is determined as $\bar{p}_L(\tilde{n})$, which was introduced above, and an arbitrary desired heading $\theta_L(\tilde{n})$. This is usually specified as a direction of the next movement from the desired state in case of complex missions composed from simple path planning tasks (see Chapter 5) for details.

without any equality or inequality constraints.

The cost function should penalize unfeasible solutions too close to the obstacles similarly as in the previous approach. Furthermore, also the required feasibility of the final path for formation $\mathcal{F}$ which supposes a limited minimal radius of turning will be included to the cost function here. The third objective included in the presented cost function should penalize the length of the path which is correlating with the time to the goal.[1] This leads to the path planning with multiple objectives which was first investigated in [66] as an inclusion of three requirements: safety, expected time required to traverse the path and energy consumption to terrain navigation. Inspired by the approach in [66] we propose the cost function including all requirements of the path planning for formations as

$$f(\overrightarrow{x}) = f_{length}(\overrightarrow{x}) + \alpha f_{distance}(\overrightarrow{x}) + \beta f_{radius}(\overrightarrow{x}), \tag{3.4}$$

where their influence is adjusted by constants α and β.[2] The part $f_{length}(\overrightarrow{x})$ in the cost function matches the length of the path that can be analytically computed by

$$f_{length}(\overrightarrow{x}) = \sum_{k=1}^{\tilde{n}} \int_0^1 \sqrt{(\varphi'_x(k,s))^2 + (\varphi'_y(k,s))^2} ds. \tag{3.5}$$

The component $f_{distance}(\overrightarrow{x})$ (see Fig. 3.3(a)) penalizes paths close to an obstacle and it is defined by the equation

$$f_{distance}(\overrightarrow{x}) = \begin{cases} d(\varphi(\cdot,\cdot), \mathcal{O}_{obs})^{-2}, & \text{if } r_{a,L} < d(\varphi(\cdot,\cdot), \mathcal{O}_{obs}) \\ d(\varphi(\cdot,\cdot), \mathcal{O}_{obs})^{-2} + p_{df}, & \text{if } r_a < d(\varphi(\cdot,\cdot), \mathcal{O}_{obs}) < r_{a,L} \\ d(\varphi(\cdot,\cdot), \mathcal{O}_{obs})^{-2} + p_{df} + p_{dr}, & \text{if } d(\varphi(\cdot,\cdot), \mathcal{O}_{obs}) < r_a, \end{cases} \tag{3.6}$$

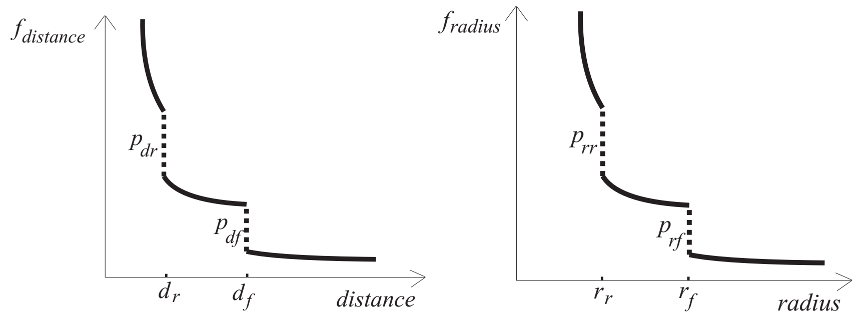
where p_{df} penalizes solutions leading to a collision of the followers with obstacles that can be avoided by a change in the formation and p_{dr} penalizes paths leading to a collision of the virtual leader. The solutions penalize by p_{dr} are unfeasible even for a single robot and cannot be repaired by a change of the shape of the formation. Function $d(\varphi(\cdot,\cdot), \mathcal{O}_{obs})$ is providing the minimal distance of the path to the closest obstacle and a guideline for its cheap computation will be given in the following subsection.

The part of the cost function $f_{radius}(\overrightarrow{x})$ (see Fig. 3.3(b)), which is necessary because of the usage of car-like robots as well as due to the presented formation driving approach, is computed likewise:

$$f_{radius}(\overrightarrow{x}) = \begin{cases} r(\varphi(\cdot,\cdot))^{-2}, & \text{if } r_f < r(\varphi(\cdot,\cdot)) \\ r(\varphi(\cdot,\cdot))^{-2} + p_{rf}, & \text{if } r_r < r(\varphi(\cdot,\cdot)) < r_f \\ r(\varphi(\cdot,\cdot))^{-2} + p_{rf} + p_{rr}, & \text{if } r(\varphi(\cdot,\cdot)) < r_r, \end{cases} \tag{3.7}$$

[1]Here the total time to goal is difficult to estimate due to unknown control inputs.

[2]The appropriate values of the constants have to be experimentally identified. One can find an example of such a study in our papers [161; 164]. We used the setting: $\alpha = 0.1$, $\beta = 10$ for the results presented in this thesis.



(a) Function $f_{distance}(\cdot)$. (b) Function $f_{radius}(\cdot)$.

Figure 3.3: An illustration of components of cost function with denoted penalization.

where

$$r_r = \max_{i=\{1,\ldots,n_r\}} K^{-1}_{max,i} \tag{3.8}$$

and

$$r_f = \max\left(\max_{\tau\in\langle t,t_f\rangle} K^{-1}_{max,L}(\tau), -\min_{\tau\in\langle t,t_f\rangle} K^{-1}_{min,L}(\tau)\right). \tag{3.9}$$

Solutions penalized only by p_{rf} can be repaired by a change of the shape of the formation, while paths with a radius smaller than r_r do not even meet the criteria of the equation (2.3) for a single robot.[1] Function $r(\varphi(\cdot,\cdot))$ is providing minimal radius along the whole path and it is defined by

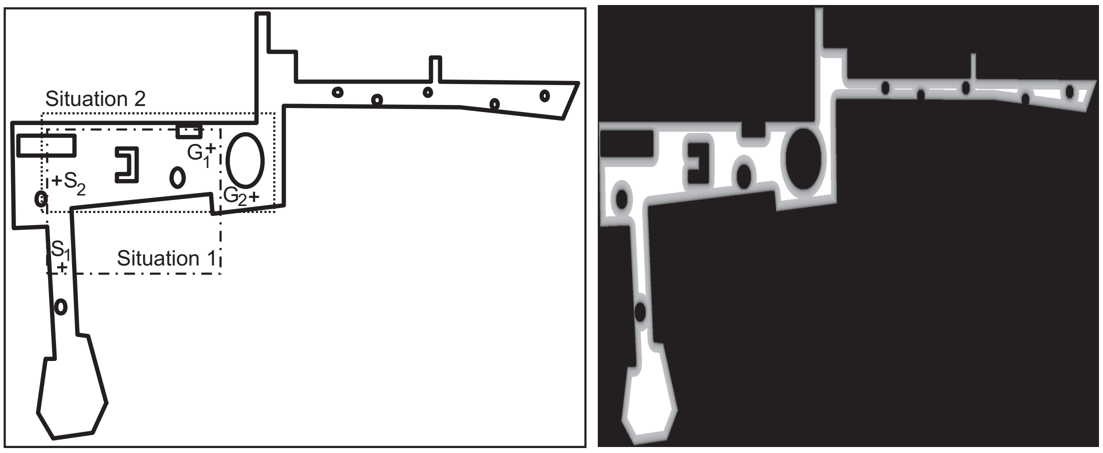
$$r(\varphi(\cdot,\cdot)) = \min_{k\in\{1,\ldots,\tilde{n}\}} \min_{s\in<0;1>} \frac{\begin{vmatrix} \varphi'_x(k,s) & \varphi'_y(k,s) \\ \varphi''_x(k,s) & \varphi''_y(k,s) \end{vmatrix}}{(\varphi'_x(k,s)^2 + \varphi'_y(k,s)^2)^{\frac{3}{2}}}. \tag{3.10}$$

The crucial problem of the method is to find a result with a minimum or nearly minimum value of the cost function usually in the vast set of solutions.. The complexity of the task grows with the parameter $\tilde{n}$, which has similar meaning to the variable M in the method presented in Chapter 2. Solutions with small $\tilde{n}$ can be found faster but a collision free path may not exist in such a limited space if the workspace of the robot is too complicated.

3.3.2 Implementation details

A great number of evaluations is required by available optimization methods and therefore the computational complexity of the cost function is a key factor for real time applications. The most calculation-intensive part of the equation (3.4) is $f_{distance}(\overrightarrow{x})$. It is done by an amount of obstacles from which the distance needs to be computed. In the presented approach, a distance grid map of the environment is precomputed to

[1]Here, we should mention that the optimization is not sensitive to exact values of the penalizations. The only assumptions that need to be satisfied are $p_{rf} < p_{rr}$ and $p_{df} < p_{dr}$. We used values: $p_{rf} = 10$, $p_{dr} = 100$, $p_{df} = 10$ and $p_{rr} = 100$ in the experiments.



(a) Schematic map of robot workspace with denoted zoomed areas of the scenarios: Situation 1 and Situation 2.

(b) Distance map used for computing of the cost function.

Figure 3.4: Illustration of compilation of data structure to time consuming cost function.

deal with this problem. Each cell in such a matrix denotes the minimum distance of the relevant place to the closest obstacle according to equation (3.6). A big advantage of such an approach is the possibility to use obstacles with an arbitrarily complicated shape, which is often provided by autonomous mapping techniques. An occupancy grid that is obtained by a range finder can be used as well. In our approach, a partly known map is assumed, where only new or moving obstacles will be added during the path planning.

In this chapter we will again use the computer science building in Wuerzburg, which was used for the experiment in Chapter 2, as an example of the robot workspace with static obstacles. The map of the building with two highlighted scenarios (Situation 1 and Situation 2), that are applied for the verification of the methods in this section, is depicted in Fig. 3.4(a) and the appropriate distance map is drawn in the Fig. 3.4(b).

3.3.3 Experimental results

At the beginning of this section, we would like to present a set of experiments describing different behavior of different optimization methods used for finding the global optimal solution of cost function (3.13). We have chosen a technique combining the Sparse Grids (SG) method in the global part of the optimization together with the Nelder-Mead method in the local part of the optimization as an appropriate approach representing deterministic algorithms. An overall description of the algorithm is offered in Appendix A.1. The second global optimization method, employed for obtaining results in this section, is stochastic, a nature inspired evolutionary technique called Partical Swarm Optimization (PSO), which is described in Appendix A.2. The parameters of the PSO method were adjusted in agreement with [162], where the algorithm was used in a similar application. We should mention that the following study is only a small portion of results we have obtained. More comprehensive tests of available optimization methods including the comparison of several evolutionary as well as deterministic methods have been published in a chapter of book [120] and in conference papers [155; 162].

Both optimization techniques here will be confronted by solving Situation 1. In the first experiment, the initial and the desired states of the robot are fixed. The deterministic SG algorithm offers a reproducable solution in static environments, while the comparable results of the stochastic PSO method can be obtained only through a statistic set of runs[1]. In Fig. 3.5, mean objective values of feasible solutions obtained by PSO technique are compared with the progress of the solution obtained by the SG method. Infeasible solutions that are highly penalized by the cost function make in statistic steps preclusive of an interpretation and they are excluded from the statistics. The fluctuations at the beginning of the mean values in Fig. 3.5 are caused by an insufficient number of feasible solutions that were added to the statistics.

The progress of the mean cost of the solutions reflects a faster convergence of the evolutionary method at the beginning of the optimization while the deterministic approach offers better solutions with the increasing number of evaluations. This is caused by the utilization of the Nelder-Mead method for the local optimization initialized in the results of global SG. The feasible solution was obtained by SG after about 1450 evaluations (it is approximately 0.6 second using Pentium 4, 3.2GHz, 504 MB), which enables a real application only in such a simple scenario.

A set of 200 randomly generated initial and desired positions was prepared for the second experiment. Each situation was once solved by PSO and once by SG. The mean costs of feasible solutions are compared in Fig. 3.6. The results are highly influenced by the adding of only feasible solutions to the statistics which makes the unsmooth curves. Nevertheless, the behavior of both algorithms is similar to the previous experiment and we can conclude that the PSO method is useful in situations where the fast response is preferred to the perfect solutions.

The second aim of this section is to highlight the problem of local extremes in the cost functions used in both formation driving algorithms that are presented in this thesis. To deal with this phenomena is one of the crucial tasks, because available optimization methods with a sufficiently fast response can easily get stuck in a minima that matches with an unfeasible solution. The result presented in this part has been obtained by the PSO technique, but the same behavior has been observed during the utilization of the SG method too.

Situation 1 and Situation 2 from Fig. 3.4(a), which contain several local extremes corresponding to feasible as well as unfeasible paths for the virtual leader, have been chosen as the test scenario. In Fig. 3.7, two solutions of the Situation 1 were presented. The path evaluated by cost $f = 13.02$ is feasible for the formation maintaining the fixed shape. Contrariwise the second path ($f = 28.71$) is feasible only for a single robot. This means that the shape of formation $\mathfrak{F}$ must be temporarily changed during the passage around the obstacles as well as in the loop replacing a sharp unfeasible curve next to the corner of the room. We should note that the loop was created automatically by the path planning method. Such maneuvers could be useful for example in crossroads of narrow

[1]1000 runs of PSO were used in the experiment.

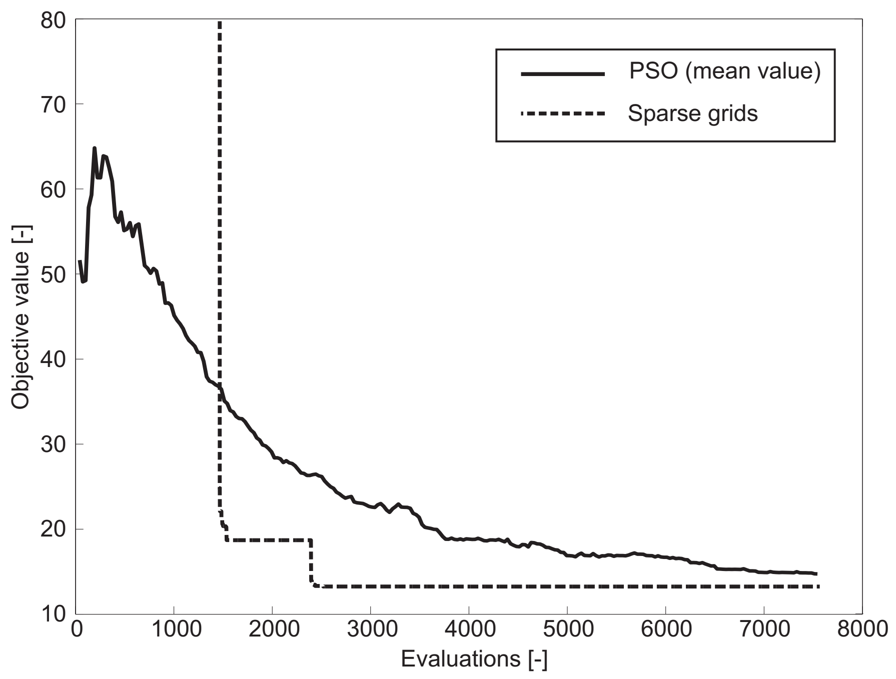


Figure 3.5: Objective values. (fixed initial and desired position of the formation)

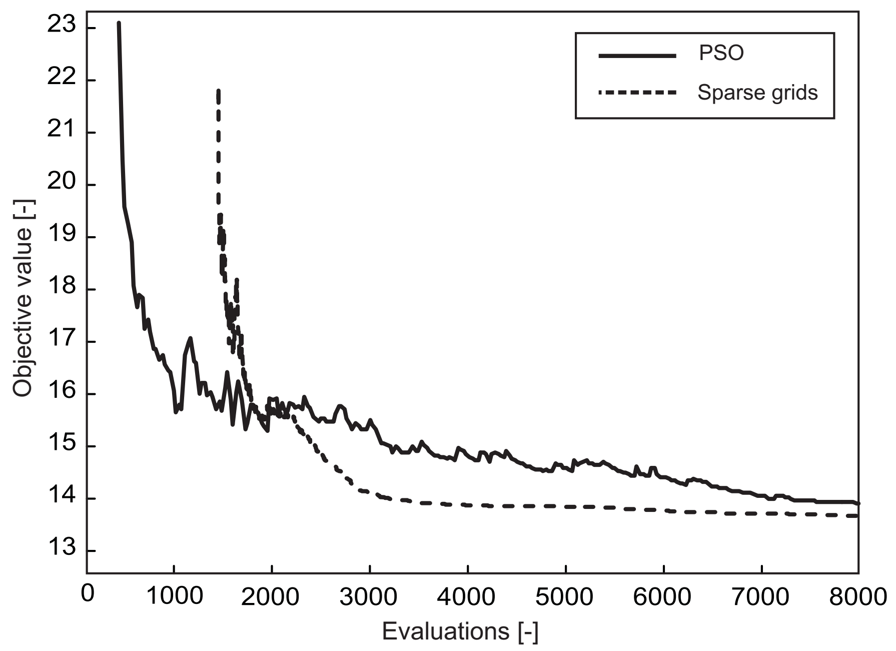


Figure 3.6: Objective values. (variable initial and desired position of the formation)

corridors where the straightforward movement is impossible due to the restriction of the turning radius.

Results of the second scenario, Situation 2, are shown in Fig. 3.8. The solution with $f = 13.82$ is feasible for the complete formation whereas the second solution ($f = 18.31$) requires small changes of the positions of outer followers. The second path is shorter than the first one and it could be preferred in the application where the shape of $\mathcal{F}$ can easily be modified.

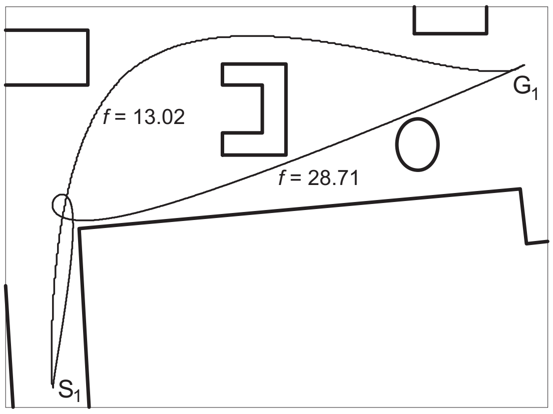


Figure 3.7: Two different solutions of the Situation 1 obtained by PSO.

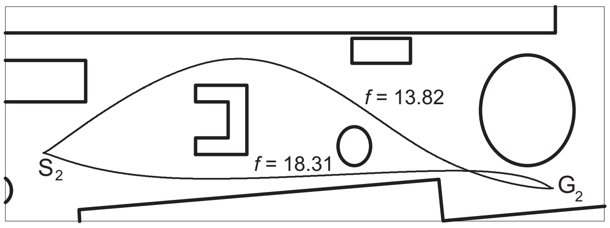


Figure 3.8: Two different solutions of the Situation 2 obtained by PSO.

3.4 Formation driving

3.4.1 Method description

As announced in Section 3.2, the formation driving method presented in this chapter is equivalent to the approach from Chapter 2 except for the leader's control. Here the second time interval T_M, considering the global character of the environment, is not necessary, because the information about the future direction of the formation is included in the designed path and all variables describing the trajectory can be collected in an shorter optimization vector, $\Omega_{L,2} = [\Psi_{L,N}, \mathcal{U}_{L,N}] \in \mathbb{R}^{5N}$.

The trajectory planning for the virtual leader is then transformed to the minimization of cost function $J_{L,2}$ subject to sets of equality constraints $h_{T_N}(\cdot)$ and inequality constraints $g_{T_N}(\cdot)$, $g_{r_{a,L}}(\cdot)$, that is

$$\min J_{L,2}(\Omega_{L,2}) \tag{3.11}$$

$$\begin{aligned} \text{s.t. } h_{T_N}(k) = 0, \forall k \in \{0, \dots, N-1\} \\ g_{T_N}(k) \leq 0, \forall k \in \{1, \dots, N\} \\ g_{r_{a,L}}(\Omega_L, \mathcal{O}_{obs}) \leq 0. \end{aligned} \tag{3.12}$$

The constraints $h_{T_N}(\cdot)$, $g_{T_N}(\cdot)$ and $g_{r_{a,L}}(\cdot)$ can be used in the same way as proposed in equations (2.23), (2.25) and (2.27). The cost function $J_{L,2}(\Omega_{L,2})$ is presented as

$$\begin{aligned} J_{L,2}(\Omega_{L,2}) = & \sum_{k=1}^{N} d\left(\varphi(\cdot,\cdot), \bar{p}_L(k)\right)^2 \\ & + \alpha \sum_{j=1}^{n_0} \left(\min \left\{ 0, \frac{dist_j(\Omega_{L,2}, \mathcal{O}_{obs}) - r_{s,L}}{dist_j(\Omega_{L,2}, \mathcal{O}_{obs}) - r_{a,L}} \right\} \right)^2 \\ & + \beta \left(\int_{ind_s}^{1} \sqrt{(\varphi_x'(ind_k, s))^2 + (\varphi_y'(ind_k, s))^2} ds \right. \\ & \left. + \sum_{k=ind_k+1}^{\tilde{n}} \int_{0}^{1} \sqrt{(\varphi_x'(k, s))^2 + (\varphi_y'(k, s))^2} ds \right)^{-1} \\ & + \gamma \sum_{k=1}^{N} \left(v_L^2(k) + K_L^2(k) \right) \left\| \bar{p}_L(k) - \bar{p}_L(k-1) \right\|^2, \end{aligned} \tag{3.13}$$

where the first term penalizes solutions with states deviated from the desired trajectory and the second term is identical to the second part of equation (2.22). The function $d\left(\varphi(\cdot,\cdot), \bar{p}_L(k)\right)$ is providing the minimal distance between the path $\varphi(\cdot,\cdot)$ and the position $\bar{p}_L(k)$. The third term of the objective function utilized in this approach is important for the convergence of the method. It replaces the inequality convergence constraint in equation (2.28) which due to missing time interval T_M cannot be applied here. This part of equation (3.13), which is inversely proportional to the length of the path $\varphi(\cdot,\cdot)$ between the closest point on $\varphi(\cdot,\cdot)$ to the last state $\psi_L(N)$ and the desired end of $\varphi(\cdot,\cdot)$, "pulls" via the constraints $h_{T_N}(k)$, where $k \in \{0, \dots, N-1\}$, all states $\psi_L(k)$, where $k \in \{1, \dots, N\}$, along $\varphi(\cdot,\cdot)$ to the end of $\varphi(\cdot,\cdot)$. The variables ind_k and ind_s indexing the closest point on $\varphi(\cdot,\cdot)$ can be determined by

$$\begin{aligned} (ind_k, ind_s) := & \left\{ (k, s) : dd(\varphi(k, s), \bar{p}_L(N))^2 = d(\varphi(\cdot,\cdot), \bar{p}_L(N))^2 \right\}, \\ & \forall k \in \{1, \dots, \tilde{n}\}, \forall s \in \langle 0; 1 \rangle, \end{aligned} \tag{3.14}$$

where the function $dd(\varphi(k, s), \bar{p}_L(N))$ is providing distance between the point $\varphi(k, s)$ and the position $\bar{p}_L(N)$. The last term of equation (3.13) penalizes aggressive control inputs and is analogous to the minimization of energy consumption. The influence of this part on the stabilization of the plant will be shown in Section 5.4.2. Finally, the constants α, β and γ are utilized for the balancing of frequently antagonistic endeavors:

i) closely follow the desired path, ii) avoid dynamic obstacles, iii) reach the desired goal as soon as possible and iv) save energy. The weights of the cost function are $\alpha = \beta = 1$, $\gamma = 0.4$ in the experiments presented in Section 3.4.3 and in Chapter 5, if it is not explicitly changed.

Remark 3.4.1. Since the result of the path planning method presented in this chapter is feasible for the formation of car-like robots, the leader's control based on RHC can be replaced by the work of Barfoot and Clark [13; 14]. The states $\psi_L(k)$, for $k \in \{1, \ldots, N\}$, are uniquely defined by the solution of the path planning and so the vector $\Omega_{L,2}$ can be directly obtained using equations (2.13) and (2.37). The advantage of the RHC based method, as we have presented in this section, consists of the possibility to run the more time-consuming path planning in a slower loop or only if it is required by a significant change of workspace. The ordinary dynamic obstacle avoidance in between the path planning interventions is provided by the RHC method, concretely by the contribution of the third term in the cost function in equation (3.13). Furthermore, the proposed RHC schema can deal with unfeasible paths as we will demonstrate in Chapter 5.

3.4.2 Proof of convergence

In this section, we will present proof of convergence to the desired state ψ_G for the virtual leader robot following the predefined path under the receding horizon scheme. To show the convergence properties in a compact way, we should first rewrite the optimization problem from equation (3.11) in accordance with equation (2.4) in the form

$$\mathcal{P}_2(t_0): \quad J_{L,2}(\psi_L(t_0), \mathcal{U}_L^\circ(t_0); N\Delta t)^\circ = \min_{\mathcal{U}_L} \left\{ \int_{t_0}^{t_0+N\Delta t} L_2(\psi_L(s), \mathcal{U}_L(s), s) ds \right\} + F(\psi_L^\circ(t_0 + N\Delta t | t_0), \psi_G), \tag{3.15}$$

where the integral is a running cost and the component $F(\psi_L^\circ(t_0 + N\Delta t|t_0), \psi_G)$ is a final cost between the last state of horizon T_N and the desired final state ψ_G. The state $\psi_L^\circ(t_0 + N\Delta t|t_0)$ is an optimal state of the virtual leader in time $t_0 + N\Delta t$ obtained as a result of the optimization problem $\mathcal{P}_2(t_0)$, which has been applied in time t_0. The final cost is expressed using the third part of the cost function in the equation (3.13). Generally, the physical meaning of the term $F(\psi_1, \psi_2)$ is the length of $\varphi(\cdot, \cdot)$ in between the closest points on $\varphi(\cdot, \cdot)$ to states ψ_1 and ψ_2.

Theorem 3.4.2. *Having the feasible preplanned path $\varphi(k, \cdot)$, where $k \in \{1, \ldots, \tilde{n}\}$, with target state ψ_G, and a feasible solution of $\mathcal{P}_2(t_0)$, the receding time horizon control scheme, which iteratively solves the optimal control problem to obtain control inputs $\mathcal{U}_L^\circ(\cdot)$ for the virtual leader, converges and guides $R_L(\psi_L(\cdot))$ toward the target state ψ_G. The claim holds as long as the perturbations on $J_{L,2}(\cdot)^\circ$, denoted by $D_2(k)$, satisfy*

$D_2(k) < F(\psi_L^\circ(t_0{+}(kn{+}N)\Delta t|t_0{+}(k{+}1)n\Delta t), \psi_L^\circ(t_0{+}((k{+}1)n{+}N)\Delta t|t_0{+}(k{+}1)n\Delta t))$, *for all* $k \in \mathbb{Z}^+$ *where* $k < (\bar{t} - t_0 - N\Delta t)/n\Delta t$. *The perturbations are caused by unexpected obstacles or system disturbances (acting on proximity penalties for* $L_2(\cdot)$*).*

Proof. Let us assume that the preplanned path $\varphi(k, \cdot)$, where $k \in \{1, \dots, \tilde{n}\}$, is time invariant for $t \in \langle t_0, \bar{t} \rangle$. To satisfy the assumption in the theorem on feasibility of $\varphi(\cdot,\cdot)$ with respect to a sufficient space between the path and the obstacles in the environment, we must presume the utilization of a static environment for the proof. The difficulties with choosing the dynamic environment will be mentioned at the end of the convergence analysis. Similarly as in the proof in Section 2.2.2, we can choose the optimal cost $J_{L,2}(\cdot)^\circ$ as a candidate Lyapunov function and the state ψ_G as an equilibrium.

Subtracting the optimal costs obtained solving the problem $\mathcal{P}_2(t_0)$ with initial condition $\psi_L(t_0)$ and the problem $\mathcal{P}_2(t_0 + n\Delta t)$ with initial condition $\psi_L(t_0 + n\Delta t)$, which has been reached after applying the first n elements of $\mathcal{U}_L^\circ(t_0)$, the following equality can be written

$$\begin{aligned} &J_{L,2}(\psi_L(t_0 + n\Delta t), \mathcal{U}_L^\circ(t_0 + n\Delta t); N\Delta t)^\circ - J_{L,2}(\psi_L(t_0), \mathcal{U}_L^\circ(t_0); N\Delta t)^\circ = \\ &= \int_{t_0+n\Delta t}^{t_0+(N+n)\Delta t} L_2(\psi_L(s), \mathcal{U}_L^\circ(s), s)ds - \int_{t_0}^{t_0+N\Delta t} L_2(\psi_L(s), \mathcal{U}_L^\circ(s), s)ds \\ &+F(\psi_L^\circ(t_0 + (N+n)\Delta t|t_0 + n\Delta t), \psi_G) - F(\psi_L^\circ(t_0 + N\Delta t|t_0), \psi_G). \end{aligned} \tag{3.16}$$

In the static environment without disturbances, we can suppose that the part of the optimal solution $\psi_L^\circ(\tau|t_0)$, where $\tau \in \langle t_0 + n\Delta t; t_0 + N\Delta t \rangle$, which is repeated in the next iteration, stays unchanged between two steps of the planning loop. Therefore we can rewrite the last term in the right hand side of (3.16) as

$$F(\psi_L^\circ(t_0 + N\Delta t|t_0), \psi_G) = F(\psi_L^\circ(t_0 + N\Delta t|t_0 + n\Delta t), \psi_G). \tag{3.17}$$

As we have mentioned above, the running cost $F(\psi_L^\circ(t_0 + N\Delta t|t_0 + n\Delta t), \psi_G)$ is the length of $\varphi(\cdot,\cdot)$ between the points on $\varphi(\cdot,\cdot)$ that have minimum distance to states $\psi_L^\circ(t_0 + N\Delta t|t_0 + n\Delta t)$ and ψ_G. Let us now split the path $\varphi(\cdot,\cdot)$ to two parts in a point on $\varphi(\cdot,\cdot)$ that is closest to $\psi_L^\circ(t_0 + (N+n)\Delta t|t_0 + n\Delta t)$. The length $F(\psi_L^\circ(t_0 + N\Delta t|t_0 + n\Delta t), \psi_G)$ can then be obtained as a sum of lengths of the splitted parts:

$$\begin{aligned} F(\psi_L^\circ(t_0 + N\Delta t|t_0 + n\Delta t), \psi_G) &= F(\psi_L^\circ(t_0 + (N+n)\Delta t|t_0 + n\Delta t), \psi_G) \\ &+F\left(\psi_L^\circ\left(t_0 + N\Delta t|t_0 + n\Delta t\right), \psi_L^\circ\left(t_0 + \left(N+n\right)\Delta t|t_0 + n\Delta t\right)\right). \end{aligned} \tag{3.18}$$

Including the observations (3.17) and (3.18) together with the substitution

$$\int_{t_0+n\Delta t}^{t_0+(N+n)\Delta t} L_2(\psi_L(s), \mathcal{U}_L^\circ(s), s)ds - \int_{t_0}^{t_0+N\Delta t} L_2(\psi_L(s), \mathcal{U}_L^\circ(s), s)ds = D_2(0) \tag{3.19}$$

into the equation (3.16) we arrive to an equality:

$$\begin{aligned} &J_{L,2}(\psi_L(t_0+n\Delta t),\mathcal{U}_L^\circ(t_0+n\Delta t);N\Delta t)^\circ - J_{L,2}(\psi_L(t_0),\mathcal{U}_L^\circ(t_0);N\Delta t)^\circ = \\ &= D_2(0) - F\left(\psi_L^\circ\left(t_0+N\Delta t|t_0+n\Delta t\right),\psi_L^\circ\left(t_0+\left(N+n\right)\Delta t|t_0+n\Delta t\right)\right). \end{aligned} \tag{3.20}$$

By choosing the optimal cost as a candidate Lyapunov function and according to Lyapunov's second theorem of stability [93], the following relation must be held to obtain a convergence to the desired state ψ_G:

$$D_2(0) < F\left(\psi_L^\circ\left(t_0+N\Delta t|t_0+n\Delta t\right),\psi_L^\circ\left(t_0+\left(N+n\right)\Delta t|t_0+n\Delta t\right)\right), \tag{3.21}$$

for the first step of the planning loop and

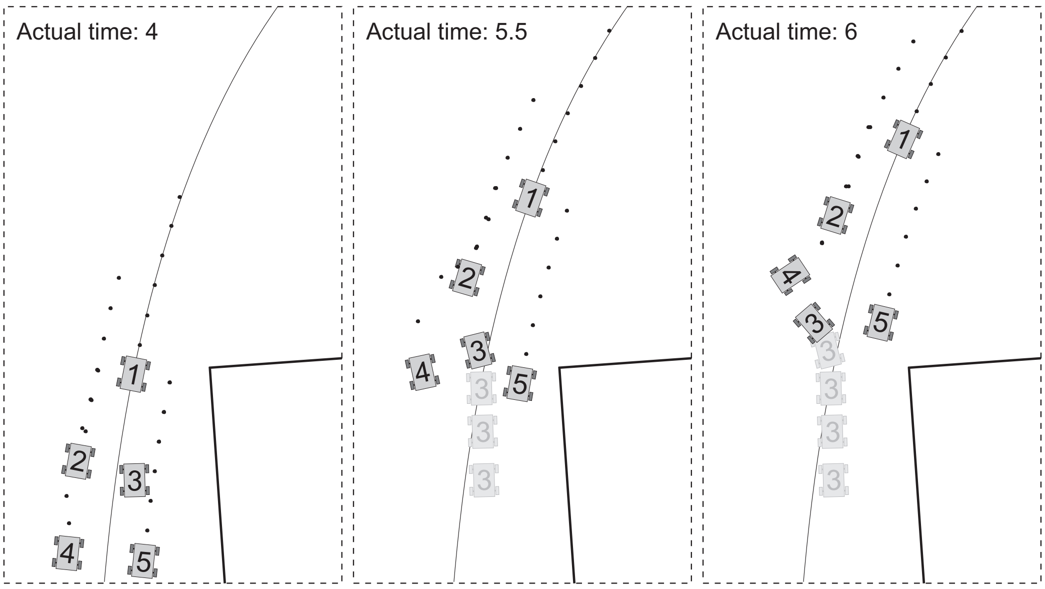
$$\begin{aligned} D_2(k) < F\left(\psi_L^\circ\left(t_0+(nk+N)\Delta t|t_0+(k+1)n\Delta t\right),\psi_L^\circ\left(t_0+\left(nk+N+n\right)\Delta t|\right.\right. \\ \left.\left. t_0+(k+1)n\Delta t\right)\right),\forall\psi_L^\circ\left(t_0+\left(nk+N+n\right)\Delta t|t_0+(k+1)n\Delta t\right)\in\breve{C} \end{aligned} \tag{3.22}$$

in general. $\breve{C} \subset \mathcal{C}$ is a set of states $\breve{C} = \{\psi_L(t) \in \mathcal{C} | F\left(\psi_L\left(t\right),\psi_G\right) > \breve{\varepsilon}\}$, where constant $\breve{\varepsilon}$ has to satisfy relation $\breve{\varepsilon} > D_2(k)$ during the whole planning process. Once the optimal solution $\psi_L^\circ(\cdot)$ enters the region surrounding the position of ψ_G with radius $\breve{\varepsilon}$ the value N is shortened as $N := \min(n, N-n)$ and the convergence can then be easily proven using the principle of optimality [8].

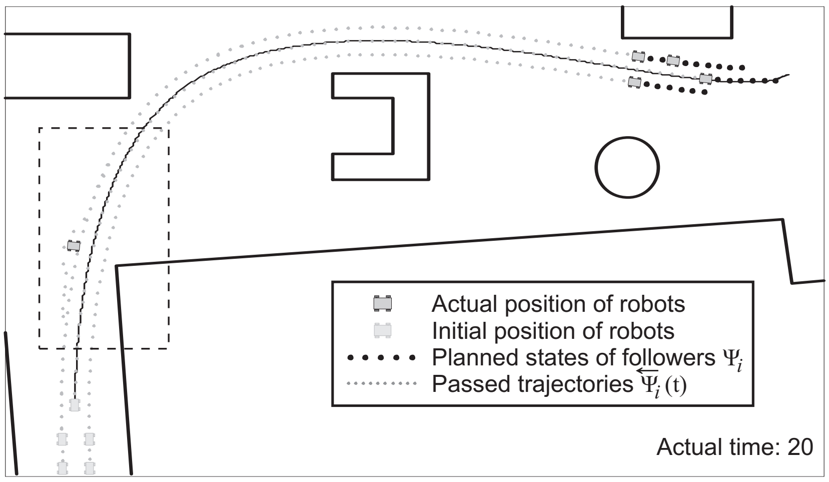
To choose the value of $\breve{\varepsilon}$ correctly let us analyze the equation (3.19) where the physical meaning of $D_2(0)$ is shown. The running cost $\int\limits_{t_0}^{t_0+N\Delta t} L_2(\psi_L(s),\mathcal{U}_L^\circ(s),s)ds$, which keeps the virtual leader on the desired path, is in a discrete form composed from the first two sums in equation (3.13). The second sum penalizes positions of formation $\mathcal{F}$ that are too close to obstacles. This function contributes to the cost of the optimal solution only in the case of an unfeasible path $\varphi(\cdot,\cdot)$ or a change of environment. Both possibilities have been excluded by the initial assumptions. The contribution of the first sum is influenced mainly by disturbances of unprecise actuators and sensors for position determination which forces the virtual leader to deviate from the desired path. The value of $\breve{\varepsilon}$ is therefore influenced by these disturbances, which must be identified experimentally.

Considering dynamic and unexpected obstacles or an unfeasible path $\varphi(\cdot,\cdot)$ we can claim, similarly as in the **Proposition 2.2.2**, that the optimality holds as long as the difference of the perturbations of the running cost on intervals $\langle t_0+N\Delta t, t_0+(N+n)\Delta t\rangle$ and $\langle t_0, t_0+n\Delta t\rangle$, denoted $D_2(0)$, satisfy the equation (3.22). Contrariwise, the violated inequality (3.22) enforces the restarting of the path planning process in an updated environment.

$\square$



(a) The beginning of the follower 3 failure. (b) Robot 4 is avoiding the problematic robot 3. The history of the movement of robot 3 is depicted by shaded contours for each planning step. (c) Robot 3 has been successively avoided by robot 4, which is going back to desired position within $\mathcal{F}$.


Actual position of robots
Initial position of robots
Planned states of followers Ψ_i
Passed trajectories $\overleftarrow{\Psi}_i(t)$
Actual time: 20

(d) The accomplished task with denoted trajectories of the incomplete formation. The dashed square designates the region zoomed in Fig. 3.9(a)-3.9(c)

Figure 3.9: A sequence of snapshots presenting the failure tolerance of the system during the following of a path that consists of three smoothly connected multinomials.

3.4.3 Simulation of failures tolerance

The feasible solution of Situation 1 (path in Fig. 3.7 denoted by $f = 13.02$) has been chosen as an example verifying the functionality of the method developed for the following predefined path by the formation $\mathcal{F}$ with a fixed desired shape. Furthermore, during the movement of the robots a failure of one follower has been simulated to demonstrate the ability of collision avoidance within $\mathcal{F}$. The beginning of the robot's problem is shown in Fig. 3.9(a). Follower 3 with blocked steering has slowly left its desired position in the formation which has detected robot 4. Due to the contribution of the third term in equation (2.31) the plan of robot 4 has been slightly deviated to avoid the defective vehicle. In the second snapshot in Fig. 3.9(b), rover 3 has started to slip leftwards because one of the robot's wheel has been stopped. This behavior has forced robot 4 to increase the avoidance maneuver which has been successfully completed in Fig. 3.9(c). For a better overview of the failure tolerance, the history of the movement of robot 3, which is considered as an obstacle, is denoted by the shaded contours. The incomplete formation at the end of the task is depicted with the trajectories from the initial positions of the robots in Fig. 3.9(d).

4

Advanced planning algorithms for virtual leaders

The common denominator of the approaches introduced in this section is an ability to reduce the computational time of the optimization methods applied within the algorithms presented in Chapters 2 and 3. In addition these approaches are able to solve the problem of a large quantity of local extremes as well as to avoid the necessary knowledge of the length of the optimal solution which is already required during the initialization of the trajectory planning. To shorten the description, the algorithms in this chapter are only described for the usage in the approach introduced in Chapter 3. Nevertheless, the same ideas can be applied for the method from Chapter 2 without any significant changes as you will be able to find in Section 4.2.4.

4.1 Voronoi Strains

An algorithm developed to solve the problem of a large number of unfeasible solutions in the initial population of the optimization methods is proposed in this section. The approach is based on a combination of the VD method and a method inspired by strains of bacteria. Strains whose evolution is stuck in a dead end die out because of the influence of strains whose progress is more successful. Only the strain located close to the global optimum or close to the best known local optimum will survive the evolutionary process.

4.1.1 Method description

Generally, the space of feasible solutions occupies a small part of the space of all possible solutions of the planning approaches introduced in Chapters 2 and 3. Almost a 100% of randomly initialized solutions of the simple path planning problem with few static obstacles is unfeasible as shown in the experimental comparison of the different optimization methods in Section 3.3.3.

We propose an algorithm in which the planning for the virtual leader is initialized as close as possible to the expected optimal solution. This approach increases the probability to reach the feasible space. A classical path planning method can be utilized as a tool providing such an estimation of the solution. Such a path planning algorithm should provide a fast response but the solution may not be optimal or even feasible for the kinematics of formations because this problem is solved using our method.

As an appropriate path planning method for the initialization, the algorithm based on finding the shortest path in Voronoi Diagram (VD) [9] was chosen. This technique has been introduced in the literature review in Section 3.2. The advantage of this approach is that the obtained path is situated far away from the obstacles and it may be assumed that the initialization in such a denoted space is feasible. It is important to mention that the space of paths in VD is not equal to the space of smoothly connected multinomials. The solution of the VD method can only be a guidance to find a feasible solution of the spline approach. Furthermore, it cannot even be guaranteed that the corresponding optimal solution of VD will be a global optimum in the space of multinomials. A path containing a sharp turn can be the shortest in VD, but the path that matches with the global minimum of the cost function in equation (3.4) can be situated in a completely different part of environment. To increase the probability that the global optimal solution is reached, we take the advantage of the evolutionary algorithms using a population of several particles that had to be initialized.[1] In the proposed method, the initial population is splitted to several sub-groups that are initialized along several promising paths found in VD.

Such an approach should deal with the problem of a multiple local extreme of the cost function that was introduced in Section 3.3.3 as well as it should reduce the time of the optimization process. Using this method the particles are initialized close to the desired solution and so the exploration part of the optimization process can be shortened. This enables to extend the exploitation part, when the area close to the desired solution has already been found and the final result is improving.

The VD algorithm cannot be utilized in the standard form with edges evaluated by their length for our purpose. In the presented Voronoi Strains (VS) method, the edges are supplemented with a number correlating with the fitness function in equation (3.4). This approach decreases the difference between the minimum of the fitness function and the cheapest path in VD. The proposed evaluation of the edges of VD, collected in a set E, is

$$f(e) = l(e) + \left(\frac{\alpha}{\min_{o \in \mathcal{O}_{obs}} d(e,o)}\right)^2, e \in E \tag{4.1}$$

where α has a similar meaning as in equation (3.4), $l(e)$ is the length of edge e and $d(e,o)$ is the shortest distance between edge e and obstacle o. The shortest path in such evaluated graph connecting the actual and desired positions of the formation is still needlessly long and also not smooth, but it is a good estimation of the region where the optimal trajectory could be situated.

Instead of a randomly generated initial population uniformly covering the whole search space, which is usually used in evolutionary algorithms, we would like to generate the population close to such a region. As a result, the algorithm converges faster but the population diversity is lost which causes problems when the estimation of the solution

[1]The PSO technique is employed in this section as the appropriate evolutionary method chosen in Section 3.3.3, but any population based optimization algorithm can be utilized with the same concept.

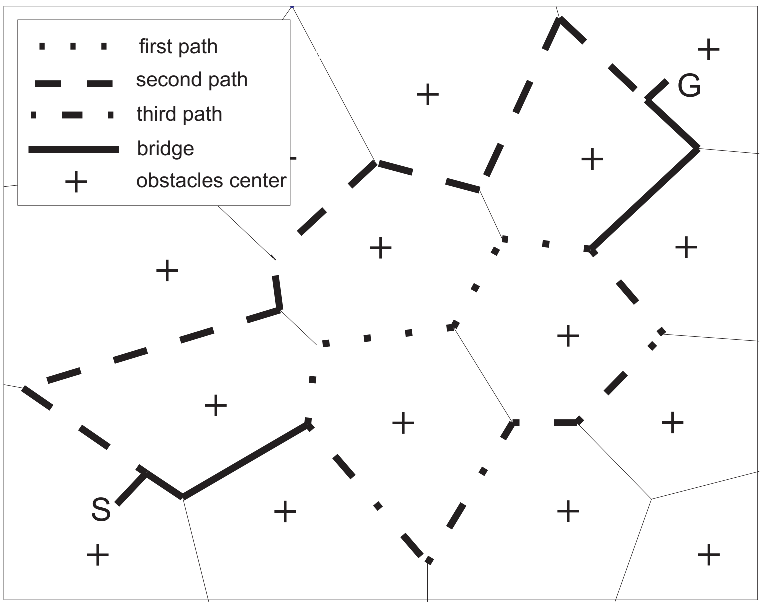


Figure 4.1: A Voronoi diagram and the cheapest paths used for the initialization of the strains.

is not correct. As mentioned above, we propose to create new strains in the population that are identified in areas of alternative solutions (the second cheapest path, the third cheapest path, etc) of the evaluated VD to solve this problem. Concretely, the price of the edges of the cheapest path is always risen two times and the next strain is located along the new cheapest path in such a changed graph. The obtained different solutions can contain the same edges, but these "bridges" are usually short. A simple example of the cheapest paths in a Voronoi graph is shown in Fig. 4.1.

Then the population of the evolutionary algorithm is initialized so that the number of particles in each strain is

$$s(i) = s_{total}\frac{2^{\tilde{m}-i}}{2^{\tilde{m}}-1}, i \in \{1, \cdots, \tilde{m}\} \tag{4.2}$$

where s_{total} is the total number of particles in the population and $\tilde{m}$ is the number of strains. Using this approach, the first obtained Voronoi paths, where the probability of the global solution is the highest, have the biggest strains.

In the end, we should shortly describe how the particles are placed along the paths found in the diagram. In the first step of the initialization, the paths are divided into $\tilde{n} - 1$ parts of identical length ($\tilde{n}$ is the number of splines in the string). In the second step, The control points of the splines P_k, where $k \in \{1, \ldots, \tilde{n} - 1\}$, are randomly generated on each part in consideration of their order in a geometric representation.[1]

[1]In the same way, the positions $\bar{p}_L(k)$, $k = \{1 \ldots N + M\}$ of transition points gathered in vectors $\Psi_{L,N}$ and $\Psi_{L,M}$ can be distributed during initialization of the method presented in Chapter 2.

Table 4.1: The mean values of the fitness function of the best particle after 50 iterations for different values of V_{max} and w_{start}.

	V_{max}				
w_{start}	10	70	150	220	300
0.1	14.51	11.10	11.21	11.65	11.86
0.2	14.34	11.03	11.18	11.52	11.57
0.4	13.43	10.98	11.24	11.43	11.65
0.6	12.28	11.00	11.43	11.80	12.00
0.8	11.82	11.23	11.82	12.06	12.39

4.1.2 Results and parameters tuning

The Voronoi Strains algorithm has been verified in two experiments. Both experiments are based on a statistically processed set of runs, because each run of evolutionary approaches is unique and thus the final trajectories are different. Each value presented in this subsection has been obtained from 400 runs of the algorithm.

The first experiment has been performed to study the influence of different parameters of the PSO algorithm on the quality of the resulting trajectories obtained by VS. In the second experiment, we have compared the VS algorithm with a simple PSO method using different initializations.

All these results have been obtained by the PSO algorithm using parameters $w_{end} = 0.05$, $\Upsilon_1 = 2$ and $\Upsilon_2 = 2$. The resulting trajectories are composed from 10 splines ($dim(\overrightarrow{x}) = 18$) and the population consists of 28 particles (16 particles in the 1th strain, 8 in the 2nd strain and 4 in the 3rd strain). Each optimization process has been interrupted after 50 iterations and 150 obstacles have been distributed randomly in the workspace.

The mean costs of the solutions of the VS algorithm with different settings of w_{start} and V_{max} in equations (A.2) and (A.4) are presented in Table 4.1. The best results have been obtained with $w_{start} = 0.4$ and $V_{max} = 70$. It is interesting to compare these results with the simple PSO method using random initialization [162], where the optimal values are $w_{start} = 0.6$ and $V_{max} = 250$. The most significant is the difference of V_{max}. Particles in the VS algorithm are initialized close to the optimal solution and they are kept in this position by the low value for the maximal velocity. If V_{max} is too big, the particles start to explore the whole space and the advantage of the initialization is lost. Contrariwise the simple PSO algorithm with a too low value of V_{max} can converge only slowly and the optimal trajectory cannot be found during 50 iterations. The lower value of w_{start} is advantageous to the VS approach, because thus the exploratory process is skipped (due to better initialization) and the optimization starts directly in an exploitative mode, where the space is searched already close to the most promising solution.

The progression of the mean cost of the best particle is presented in Fig. 4.2 for different numbers of splines used to evolute the trajectory. The lowest mean value of the best particle was achieved with $dim(\overrightarrow{x}) = 18$ (10 splines in the string) after 50

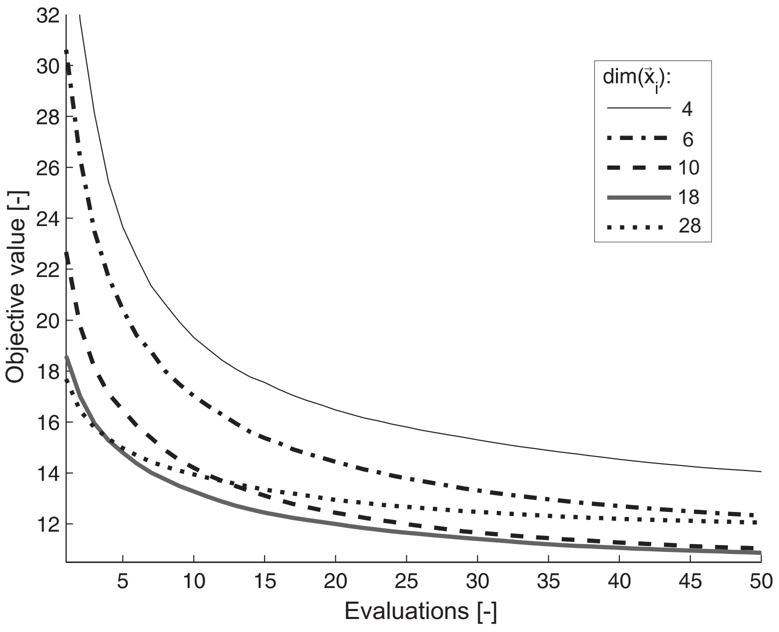


Figure 4.2: Progression of the best particle with different length of particle.

iterations. A higher dimension results in a better initial population (a higher mean value of the best particle), but also slows down the optimization process. Contrariwise the PSO is sped up if the value of $dim(\overrightarrow{x})$ is low, but the space is reduced and the optimal solution can be of poor quality (e.q. not free of collisions).

Three different algorithms were compared in two dissimilar scenarios in the experiments described in this section. The first algorithm (PSO-simple) only uses a basic initialization. Particles in the initial population are generated randomly and so the control points of the splines cover the complete workspace and the trajectories possibly contain loops. In the second approach (PSO-line) it is supposed that the optimal trajectory is short and without loops or backwards motion. Therefore, the line connecting the points $\bar{p}_S$ and $\bar{p}_G$ is divided into $\tilde{n}$ same parts and the control points P_i are randomly generated close to the dividing points.

The first scenario is identical to the situation with 150 obstacles used in the previous experiment and demonstrates the ability of the VS algorithm to work in complicated environments. The mean value and covariance of the fitness function of the best particle after 50 iterations is presented in Table 4.2. The mean value of solutions achieved by the VS approach is much better than the mean value of solutions found by the other methods, but the most significant fact is found by comparing the covariances. The low covariance in the VS approach shows its high robustness in comparison to the PSO-simple and the PSO-line methods.

The second scenario consists of only 10 obstacles in a workspace of identical size to the one in the first experiment. The results presented in the second column of Table 4.2 illustrate a low effect of the initialization on the optimization process in such a simple situation. The biggest advantage of the VS approach (missing exploratory mode) is lost, because in this case the exploratory mode in the simple PSO approach is very

Table 4.2: The mean fitness values and covariances (in brackets) of the fitness function of the best particle after 50 iterations.

method	150 obstacles	10 obstacles
PSO-simple	56.56 (444.53)	1.68 (0.0047)
PSO-line	21.31 (44.57)	1.67 (0.0039)
Voronoi Strains	10.98 (0.89)	1.67 (0.0035)

short. Additional experiments and a more detailed decription of the VS method can be found in [158].

4.2 Hierarchical approach

One of the crucial problems of the approaches that are solving the *formation to target zone problem* is to decide how big the dimension of space of solution is optimal before the optimization. For the approach in Chapter 2 it means to find the optimal value of M, while for the approach in Chapter 3 it is the optimal number of multinomials in the string $\tilde{n}$. A too big value of the parameters causes the growth of the computational time and contrariwise if a too small value is used a feasible solution may not exist (e.g. see study in Section 2.4.1). The optimal setting of these variables is impossible in most applications and it can change as new obstacles are detected during the task executing.

A solution proposed in this section is a hierarchical decomposition of the planning process to smaller tasks. The hierarchical plunge is iteratively realized only in the part of solution where it is required by the complexity of the environment. At the beginning the formation can follow a simple plan that represents a coarse structure of the map. In the part of the map where such result is insufficient (this can be easily detected using the value of cost function), the appropriate piece of solution is replaced by a more complicated path. The formation therefore does not have to wait for the complete plan before the beginning of the movement and in addition the detailed plan is obtained when the robots approach to the concerned area. It has no sense to compute a particular solution when the exact position of dynamic obstacles is not known. The provided description of the method is adapted to the approach in Chapter 3, but the utilization of the method in Chapter 2 is similar as you can see in a short note at the end of this section.

We should mention that only an epitome of the method is presented here, while the complete version can be found in [156] which is supplemented by a formation driving simulation in [157].

4.2.1 Method description

In the hierarchical approach proposed in this section the optimization task is decomposed to multiple subtasks. The basic idea of the algorithm can be seen in Fig. 4.3. The actual state of the robot and the desired state generated by a higher *Task planning* module are the inputs for a global optimization method analogous to the basic approach presented in Chapter 3. The obtained path is checked in the *Collision detection* module and a collision free solution is sent to the *Control module*, where the path is executed. If a collision is detected, the string of splines is divided and the control points of each unfeasible spline are used as a new input for the optimization. These sets of control points are put into the memory of the *Global optimization* module. For the following optimization processes, the piece of path that has minimum distance to the virtual leader is always chosen as the first. The remaining parts will be optimized during the robot's movement. Therefore the time needed for the initial planning before the start of the mission is reduced several times.

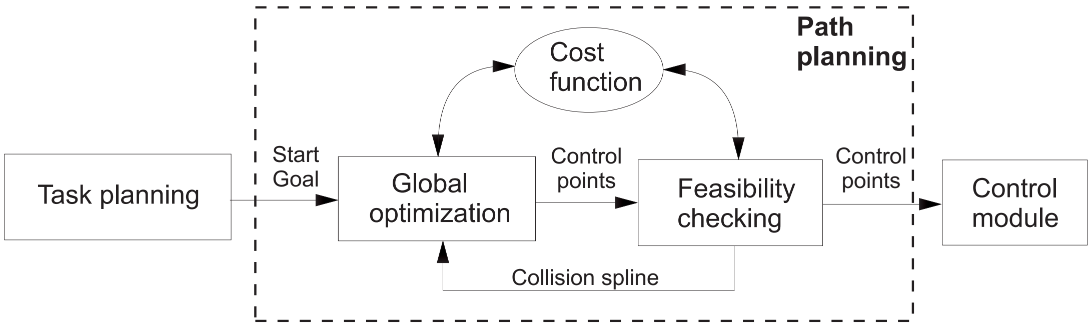


Figure 4.3: Schema of the presented hierarchical method.

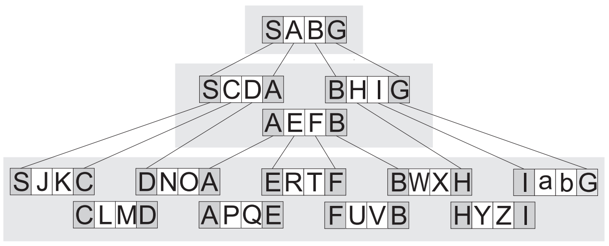


Figure 4.4: Example of the path decomposition.

An example of the path decomposition is shown in Fig. 4.4. Each optimization vector used in the optimization processes contains only two positions $\bar{p}_L(k)$, where $k \in \{1, 2\}$ which correspond to three splines in the string, $\tilde{n} = 3$. In this simplified example the hierarchical process is always stopped in the third level, independently from the number of collisions. The input states for each optimization are printed with a dark background (e.g. S and G in the first level) whereas the optimized positions are represented by letters in white rectangles (e.g. A and B in the first level). As we have explained in Chapter 3 the spline optimization method provides only a sequence of positions $\bar{p}_L(\cdot)$, but the complete states ψ_S and ψ_G are needful for its input. Fortunately the missing values of θ_S and θ_G are uniquely defined by the function $\varphi(\cdot, \cdot)$ that can be easily derived from the obtained solution. The final path in the lowest level consists of 27 splines and it is characterized by actual and desired states and by positions of the virtual leader in between of them. Such a path can be found as a minimum of the nonlinear cost function in a 52-dimensional space (26 optimized points multiplied by 2 numbers describing each position). This procedure is difficult (or impossible) in real applications with a limited computational time. The scanned space is reduced 13 times and the separate optimization subtask can be done in tens of iterations in the hierarchical approach. Furthermore only three optimization processes (in the Fig. 4.4 marked by $SABG$, $SCDA$, $SJKC$) are necessary for the beginning of the mission.

4.2.2 Objective functions

Two different cost functions are used in the presented hierarchical approach. The solutions found in the last hierarchical level (the lowest big rectangle in Fig. 4.4) are evaluated by the cost function, which is equivalent to the cost function applied in the simple method introduced in Chapter 3,

$$f_1(\overrightarrow{x}) = f_{length}(\overrightarrow{x}) + \alpha f_{distance}(\overrightarrow{x}) + \beta f_{radius}(\overrightarrow{x}). \tag{4.3}$$

The paths in the other levels are evaluated by the extended cost function

$$f_2(\overrightarrow{x}) = f_1(\overrightarrow{x}) + \gamma f_{extension}(\overrightarrow{x}), \tag{4.4}$$

where the component $f_{extension}(\overrightarrow{x})$ pushes the control points of the splines $P_k \equiv \bar{p}_L(k)$, where $k \in \{1, \ldots, \tilde{n} - 1\}$, to the obstacle free space. These points are fixed for a lower level planning and therefore collisions close to these points cannot be repaired afterwards. The function $f_{extension}(\overrightarrow{x})$ is defined as

$$f_{extension}(\overrightarrow{x}) = \begin{cases} \delta(\varphi(\cdot,\cdot), \mathcal{O}_{obs})^{-2}, & \text{if} \quad r_{a,L} < \delta(\varphi(\cdot,\cdot), \mathcal{O}_{obs}) \\ \delta(\varphi(\cdot,\cdot), \mathcal{O}_{obs})^{-2} + p_{in}, & \text{else} \end{cases} \tag{4.5}$$

where the constant p_{in} strongly penalizes solutions with a state whose position $p_L(\cdot)$ is situated inside a region created by the dilatation of obstacles with radius $r_{a,L}$.[1] The function $\delta(\varphi(\cdot,\cdot), \mathcal{O}_{obs})$ returns the minimum distance of any point $p_L(k)$, for $k \in \{1, \ldots, \tilde{n} - 1\}$, to the border of any obstacle $o \in \mathcal{O}_{obs}$. Finally, the value of constant γ, which adjusts the influence of the term $f_{extension}(\overrightarrow{x})$ on the optimization, should satisfy the inequality $\gamma > \alpha$. In the experiments, we utilized values of constants α and β in accordance with the simple method introduced in Section 3.3.1 and $\gamma = 1$.

4.2.3 Results and comparison with simple optimization method

The described method has been tested in a scenario with 3000 randomly generated obstacles in a quadratic field. The scenario is depicted and described in detail in [156], where a similar method was applied to a single robot. To solve the global optimization problem we have used the PSO technique again. In Table 4.3, the hierarchical approach has been compared with the simple method introduced in Section 3.3.1. Each resulting

[1]The value of penalization was set in the experiments as $p_{in} = 100$.

Table 4.3: The number of paths unfeasible for an unchanged formation. (The number of paths that can be executed by formation changing.)

maximum level	I	II	III	IV	V
iterations	30	117	272	568	1103
hierarchical	954(382)	521(210)	189(76)	121(51)	97(43)
simple ($n = 2$)	843(340)	622(252)	511(205)	489(196)	442(181)
simple ($n = 3$)	954(382)	667(268)	538(218)	475(193)	438(178)
simple ($n = 4$)	998(413)	852(342)	677(272)	633(255)	560(226)

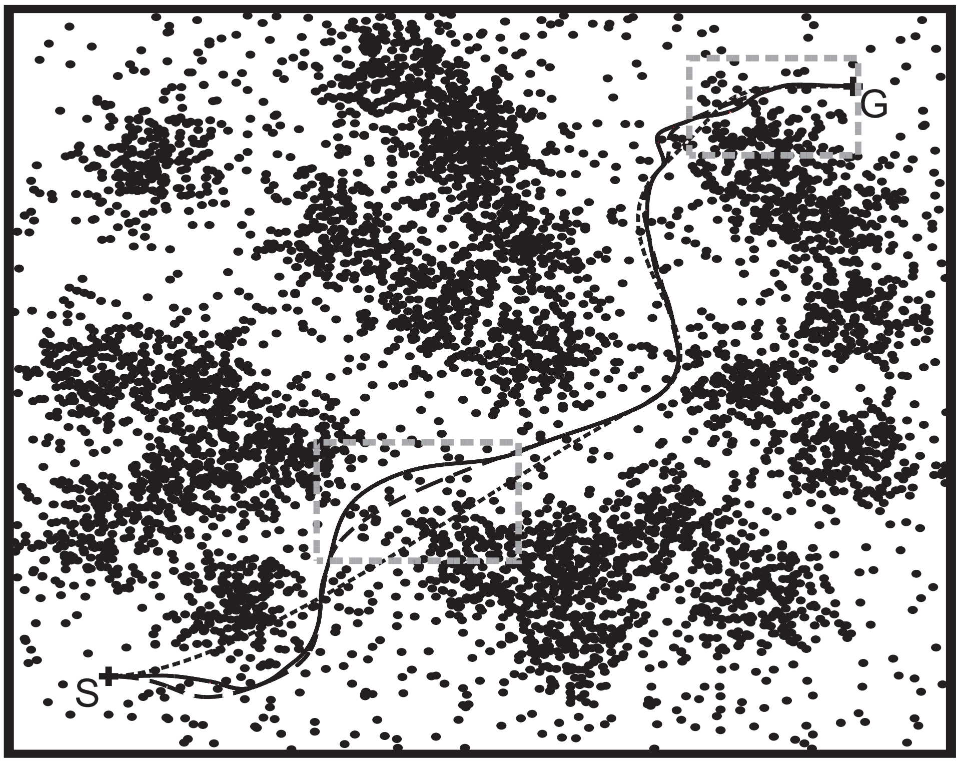
value was obtained by 1000 runs of the algorithms. Such a statistic is necessary due to the indeterministic PSO method. The second row of the table contains the amount of iterations executed by the PSO modules in the hierarchical algorithm with different settings of the maximum level. These values acted as inputs for the relevant runs of the simple PSO algorithm that was executed with a different number of splines in the string. The amount of solutions including a collision with an obstacle is listed in the last four rows. The number of solutions with collisions that remain even after the changing of the formation is depicted in brackets. The obtained results were several times better than by utilization of all modifications of the simple PSO algorithm already in the third hierarchical level. This accrues from the reduction of the search space. Additionally, the number of situations in which the modification of the shape of the formation is required is reduced by the hierarchical process.

An example of a randomly generated situation with a solution found by the hierarchical algorithm is shown in Fig. 4.5. The decreasing amount of collisions in the path at different hierarchical levels can be observed in the zoomed parts of the overall scenario in Fig. 4.5(b) and Fig. 4.5(c). The solution found in the first level (dotted line) contains 21 collisions along the whole path, but the control points are situated far away from the obstacles and the big clusters are avoided. Only 4 collisions are contained in the corrected path at the second level (dashed line) and the final solution (solid line), which was obtained after three hierarchical optimizations, is collision free.

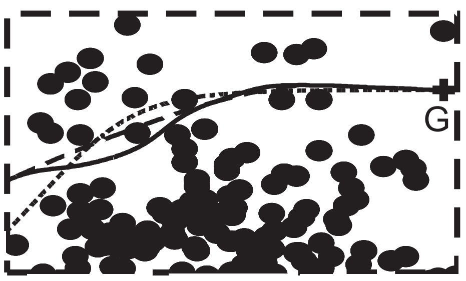
4.2.4 Hierarchical concept for RHC approach

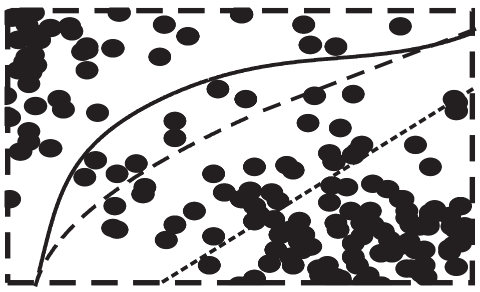
The hierarchical approach can be utilized for the method based on RHC presented in Chapter 2 in the same way as we have shown for the algorithm with a solution in the space of multinominals. This paragraph will be focused on slight differences of the application of the hierarchical concept for the algorithm from Chapter 2 and a general description will be omitted to prevent unnecessary repetitions.

The hierarchical decomposition of the optimization problem from Section 2.3.2 is possible only within the second time interval T_M, because the distance between the sampling points on interval T_N is constant and it is determined by control demands. Based on this, collisions with the environment in the interval T_N are considered as unavoidable using the current dynamics of the system. Contrariwise, collisions in the



(a) A complete trajectory of the leading robot with denoted zoomed areas depicted in Fig. 4.5(b) and Fig. 4.5(c).





(b) A zoomed part close to the goal position.

(c) A zoomed part of Fig. 4.5(a).

Figure 4.5: A situation with 3000 randomly generated obstacles. Dotted line - the path found in the first level, dashed line - the path found in the second level, solid line - the final solution found in the third level. The radius of the obstacles is dilated by the size of the formation.

interval T_M can be caused by an insufficiently small dimension of the space of solution as we have investigated in Section 2.4.1. Similarly as the number of splines $\tilde{n}$ also the value of parameter M can be increased by iteratively adding additional states $\psi_L(\cdot)$ into the part of the optimal trajectory $\Psi_L^\circ(t)$ where a collision with the obstacles is detected.

Let us suppose that the avoidance inequality constraint $g_{r_{a,L}}(\Omega_L^\circ, \mathcal{O}_{obs})$, which is defined by equation (2.27), is violated between the states $\psi_L^\circ(i-1)$ and $\psi_L^\circ(i)$, for $i \in \mathbb{Z}^+$, where $N < i \leq N + M$, and the constraint is satisfied in the rest of the trajectory. Then the states $\psi_L^\circ(i-1)$ and $\psi_L^\circ(i)$ can be used as inputs of a new optimization process $\mathcal{P}^\star(\cdot)$ starting from $\psi_L^\circ(i-1)$. Here, the optimization vector will be defined as $\Omega_L^\star = [\Psi_{L,M^\star}^\star, \mathcal{U}_{L,M^\star}^\star, \mathcal{T}_{L,M^\star}^{\Delta,\star}] \in \mathbb{R}^{6M}$, where $M^\star$ is constant that characterizes the hierarchical process.

The trajectory planning and the obstacle avoidance problem can again be transformed to the minimization of cost function $J_L(\cdot)$ subject to sets of inequality constraints $g_{T_M}(\cdot)$ (equation (2.26)), $g_{r_{a,L}}(\cdot)$ (equation (2.27)) and equality constraints $h_{T_M}(\cdot)$ (equation (2.24)), constraint $h_{con}(\cdot)$, that is

$$\min J_L(\Omega_L^\star) \tag{4.6}$$

$$\begin{aligned} \text{s.t. } h_{T_M}(k) &= 0, \forall k \in \{N, \ldots, N + M - 1\} \\ h_{con}(\psi_L^\circ(i), \psi_L^\star(M^\star)) &= 0 \\ g_{T_M}(k) &\leq 0, \forall k \in \{N + 1, \ldots, N + M\} \\ g_{r_{a,L}}(\Omega_L^\star, \mathcal{O}_{obs}) &\leq 0. \end{aligned} \tag{4.7}$$

The cost function $J_L(\cdot)$ can be used in the form presented in equation (2.22) with constants $N := 0$ and $M := M^\star$. The constraint $h_{con}(\cdot)$, which replaced the stabilization constraint $g_{S_F}(\cdot)$ from equation (2.28), keeps the new part of the trajectory $\Omega_L^\star$ consistent with the previous solution Ω_L°. It is defined as

$$h_{con}(\psi_L^\circ(i), \psi_L^\star(M^\star)) := \begin{bmatrix} x_L^\circ(i) - x_L^\star(M^\star) \\ y_L^\circ(i) - y_L^\star(M^\star) \\ \theta_L^\circ(i) - \theta_L^\star(M^\star) \end{bmatrix}. \tag{4.8}$$

The new optimal trajectory for the virtual leader can be obtained by integration $\Omega_L^{\star,\circ}$ into Ω_L° as $\Psi_{L,M}^\circ = [\psi_L^\circ(N+1), \ldots, \psi_L^\circ(i-1), \Psi_{L,M}^{\star,\circ}, \psi_L^\circ(i+1), \ldots, \psi_L^\circ(N+M)]$, $\mathcal{U}_{L,M}^\circ = [\bar{u}_L^\circ(N+1), \ldots, \bar{u}_L^\circ(i-1), \mathcal{U}_{L,M}^{\star,\circ}, \bar{u}_L^\circ(i+1), \ldots, \bar{u}_L^\circ(N+M)]$, $\mathcal{T}_{L,M}^{\Delta,\circ} = [\Delta t^\circ(N+1), \ldots, \Delta t^\circ(i-1), \mathcal{T}_{L,M}^{\Delta,\star,\circ}$ and $\Delta t^\circ(i+1), \ldots, \Delta t^\circ(N+M)]$. This process can be iteratively repeated for the remaining collisions in the new Ω_L° using the concept presented in Fig. 4.4.

Finally, we should discuss an appropriate initialization of the optimization problem $\mathcal{P}^\star(\cdot)$. The best available information about the desired solution is included in Ω_L°, concretely it is the trajectory defined by the states $\psi_L^\circ(i-1)$ and $\psi_L^\circ(i)$, control inputs $\bar{u}_L^\circ(i)$ and time $\Delta t_L^\circ(i)$. The optimal initialization could then be to equally distribute the states along such a trajectory. We propose the complete initialization as $K_{L,init}^\star(k) := K_L^\circ(i)$, where $k \in \{1, \ldots, M^\star\}$, $v_{L,init}^\star(k) := v_L^\circ(i)$, where $k \in \{1, \ldots, M^\star\}$, $\Delta t_{L,init}^\star(k) := \Delta t^\circ(i)/M^\star$, $k \in \{1, \ldots, M^\star\}$, $\psi_{L,init}^\star(1) := KinMod(\psi_L^\circ(i-1), \bar{u}_{L,init}^\star(1),$

$\Delta t^{\star}_{L,init}(1))$ and $\psi^{\star}_{L,init}(k) := KinMod(\psi^{\star}_{L,init}(k-1), \bar{u}^{\star}_{L,init}(k), \Delta t^{\star}_{L,init}(k))$, where $k \in \{2, \ldots, M^{\star}\}$. The function $KinMod(\cdot)$ represents the discrete kinematic model described by equations (2.2).

5

Application - airport snow shovelling

5.1 Introduction and motivation

A complete multi-robot system for efficiently removing snow from an airport will be presented in this chapter. This application was chosen as an interesting utilization of the theoretical approach presented in Chapter 2. Such an application is essential to show the procedure of appropriate tuning of crucial parameters which could be complicated in the general case. Furthermore, we are convinced of the usefulness and feasibility of such a project which has been confirmed by positive feedback given from the scientific community at several international conferences [77; 159; 160].

The main task of the ground staff at the airport is to maintain airports' operations safe and uninterrupted. One of the critical periods for the safeness of air traffic is snowy weather. This is not because of the reduced visibility during landing, but mainly due to runway conditions. It is a complicated task to remove the snow from the huge surface of the main runways and the big amount of auxiliary roads that are necessary for the planes as well as for the ground vehicles.

Today the tracks of airports are freed from snow by utilizing a fleet of human driven snowploughs (see Fig. 5.1). An increase of efficiency, but also the saving of expenses (now the crew of ploughs must be on alert day and night all winter) could be achieved using an autonomous system of mobile robots. Due to the fact that the big airports are already equipped by the essential sensors, i.e. a global positioning system and automatic detection of runway conditions, such a system can be set up relatively easily. Also the periodicity of the task and low presence of the obstacles during the cleaning process (e.g. in comparison to highways) predestinate the use of autonomous robots.

A description of an appropriate autonomous system is provided in this thesis. The basic idea is motivated by current approaches commonly used for shoveling of runways by human driven snowploughs. Since partly cleaned paths could be dangerous especially in emergency situations, it is required that the main runways as well as the auxiliary roads have to be cleaned up at once. Thus we avoid forced landing planes as well as rescue and fire fighting vehicles facing roads with an irregular snow surface. The main

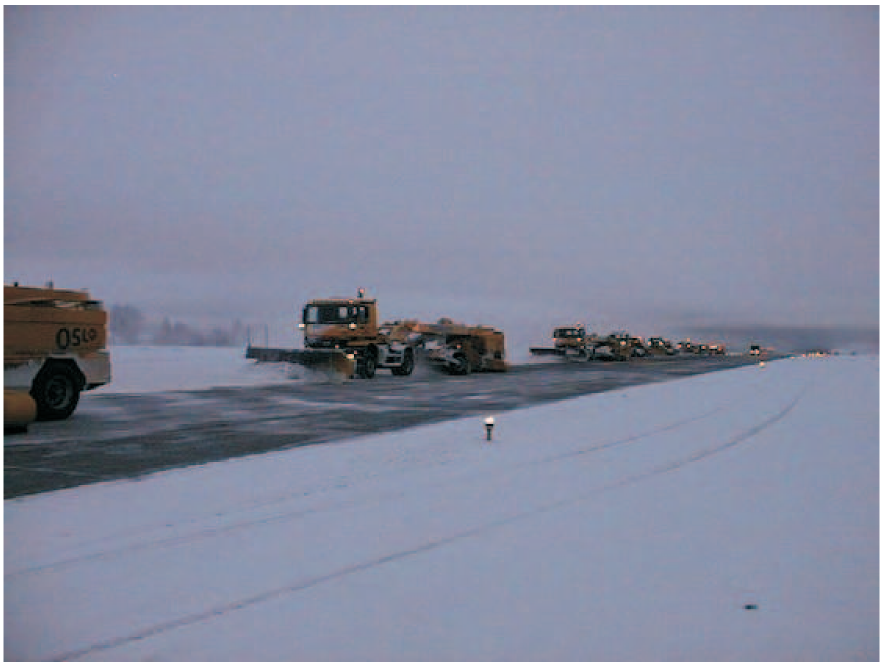


Figure 5.1: Commercial snowploughs from the Norwegian company Øveraasen. (Source: [137])

runway should be therefore cleaned up by a big group with sufficient numbers of vehicles. This approach enables to clean such a large surface quickly and a possibility that some snow remains in the landing area is decreased. When the cleaning of the runway is accomplished the big group is splitted into smaller teams with sizes appropriate for the shoveling of smaller roads. In our case the advantage of such an approach could be an opportunity to use the formation driving method that can easily arrange the ploughs into positions optimal for shoveling, but also simplify the robotic system and increase its robustness.

In this chapter we provide an overview of several achievements we have obtained during the investigation of the airport sweeping problem. Note that parts of it have been published before in an international journal [79] and in proceedings of international conferences [77; 159; 160]. In these publications one can find additional information about the methods described in this chapter and also relevant approaches developed by other members of our team.

For the purpose of airport snow shoveling, the conventional point-to-point path planning presented in Chapter 3 needs to be extended to so called coverage path planning that determines a path for the robots visiting all points (in our application all roads) in the workspace. The coverage by mobile robots is a quite large domain including e.g. robotic demining, mobile sensing networks, lawn mowing, car-body painting, field harvesting or floor cleaning. An inexhaustible amount of coverage approaches have been developed for these scenarios. We would like to mention only few of the most cited. In the first of them, the differential drive robots are used for application of mobile sensing networks without any guarantee of collision avoidance [34]. The authors introduce control and coordination algorithms for the optimal coverage using gradient descent algorithms. Another sensor networks application provides an effective coverage control for bidirectional partially connected mobile structures [83]. A cooperative based strategy using neural networks for complete coverage path planning of multiple cleaning robots is presented in [117; 118]. This work has been followed by an approach proposed in [189] using the neural network for complete coverage path planning of cleaning robots with obstacle avoidance in nonstationary environments. For a better overview of coverage path planning methods we recommend well-arranged surveys published in [30] and in the literature review of [34]. A comprehensive survey of existing projects in this field is published in [26] while 30 different prototypes designed for this specific application are listed in [146].

Multi-robot coverage techniques can be generally divided into two branches: off-line and on-line coverage. In the off-line coverage, the robots have a-priori knowledge of the workspace and the path planning is done beforehand [76] while in the on-line coverage the map of the environment is obtained during the task accomplishing and the plan is adaptively built [183]. In this thesis, both approaches are discussed. In the static route scheduling method (Section 5.3.2) a known map is assumed and the path is obtained off-line completely before the mission, while the dynamic task allocation method (presented in Section 5.3.3) enables an on-line response to detected changes in the airport environment.

From the applications perspective, the snow shoveling task addressed in this chapter is related to the field of autonomous sweeping. The task sweeping has been defined as a motion with aim to cover a 2-dimensional area by robots' effecters (in our case the snowploughs' shovels). An approach of path planning for a single robot followed by a decomposition of the obtained path for the multiple robots has been applied in the initial work of cooperative sweeping by multiple mobile robots (see [98; 99]). The application of this method in a real environment without any global positioning system has been enabled by work in [100] improving the autonomous floor cleaning system by a novel cooperative positioning system. From the work of other teams, we will refer to a decentralized method based on partitioning a cleaning area into polygons dynamically allocated to the robots in [85]. Another example is an approach inspired by cooperative behavior of ants introduced in [182]. In this decentralized cleaning algorithm only using local interactions between the robots without a central supervisor, the task completion with an upper bound on the time complexity is guaranteed. A behavior based approach for multi-robot cooperative cleaning is also proposed in [87]. Here, the authors use a distributed action selection mechanism that unifies path planning, cooperative interactions and reactive behavior. In [170], ethological-inspired behaviors are applied on a swarm of agents for the cleaning tasks. In the method, the robots are assigned to individual territories that can be dynamically resized. Another decentralized approach of the on-line cooperative sweeping using a market-like structure and a negotiation mechanism to resolve the task sharing is published in [127]. On-line cooperative cleaning is also discussed in [73] where a method using an internal spiral coverage is published. In [5], the problem of sweeping is extended to a more dynamic variant considering the spreading of contamination that needs to be cleaned. This work is completed by a theoretical study [188] discussing the effect of geometric features of the dirty region on the cleaning time of the robot swarm.

The most relevant approach to our application is presented in [3] where authors introduced a task of continuous area sweeping. In the method, a group of robots must repeatedly visit all points in fixed areas, while the rate of visiting separate parts of the cleaning sequence is non-uniform and even non-stationary. This perfectly corresponds with our application where the airport during the snow-fall must be cleaned repeatedly, the rate of cleaning of particular runways depends on their importance (the auxiliary roads for ground staff can be freed from snow less frequently than the landing areas) and the cleaning frequency rests with intensity of snowing.

Projecting the problem of cooperative sweeping onto the airfield, the snow shoveling task generates some problems that have not yet been solved. Our approach considers the nonholonomic kinematics of usual snowploughs as well as the position and orientation of the ploughs' shovels. Furthermore, the working space is more structured due to the airfield environment.

In the airport sweeping scenario, the first problem to solve is to assign tasks for each snowplough so that a short total execution time is achieved. Two different task allocation approaches with different features and areas of application are proposed in this chapter. The first, static method, is based on a completely explored tree of solutions of the task allocation problem. The obtained result is globally optimal regarding the total time needed for shoveling. The second, dynamic multi-agent based approach, effectively manages the partitioning of the vehicles into formations resulting in a locally optimal execution time. The introduced method applies simple heuristics and an algorithm for the reduction of the tree of possible solutions. The task list is computed online and can be adjusted immediately e.g. in order to react on unforeseen events.

Another subproblem is how to translate the instructions from the task list to the commands for the low level control of the vehicles. As a solution of this problem we introduce a method that constructs desired paths for the virtual leaders of the formations as a string of line segments connecting centers of crossroads. To track these paths with the snowplough formations we apply a leader-follower approach that maintains the arrangement of the group in curvilinear coordinates. Low level control inputs for the virtual leader as well as for the followers are then obtained using the RHC method described in Chapter 3 (the virtual leader) and in Chapter 2 (the followers).

The remainder of this chapter is organized as follows. Section 5.2 gives an overview of the snow shoveling system and the necessary communication channels. In Section 5.3, both task allocation approaches are described and compared. After this, we introduce suitable motion coordination techniques for the vehicle formations based on RHC in Section 5.4. Finally, results from the hardware experiments are presented in Section 5.5.

5.2 System overview

In this section we introduce the system structure of our approach. We decided to rely on a central supervision for the high level coordination of the system. The main reason for this is safeness, since a central command center (usually placed on an airport control tower) has a complete overview of the whole system and it is the appropriate place for

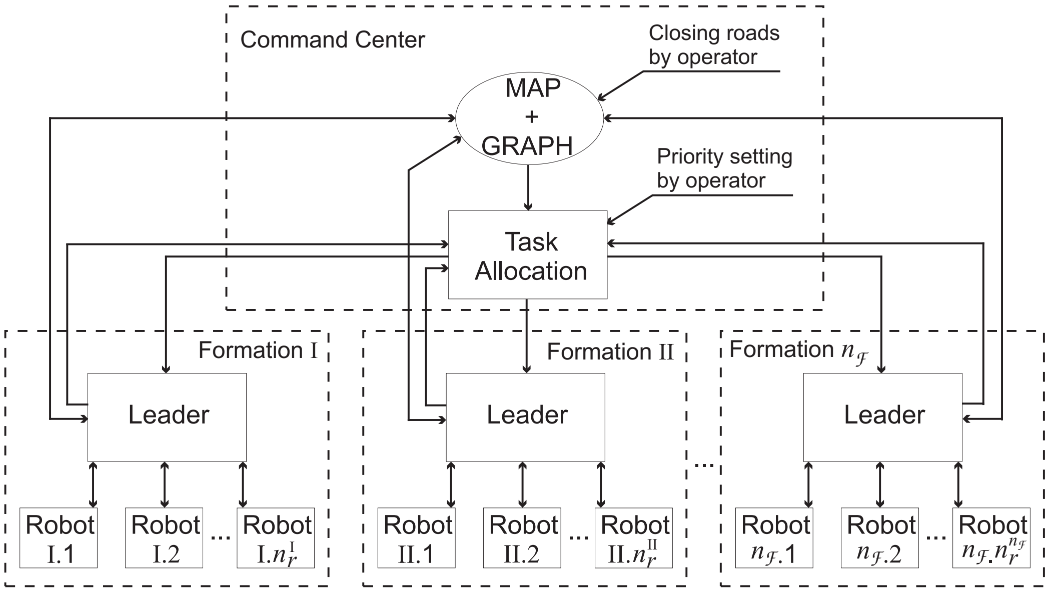

Command Center
Closing roads by operator
MAP + GRAPH
Priority setting by operator
Task Allocation
Formation I
Formation II
Formation $n_{\mathcal{F}}$
Leader
Robot I.1
Robot I.2
Robot I.n_r^{I}
Robot II.1
Robot II.2
Robot II.n_r^{II}
Robot $n_{\mathcal{F}}$.1
Robot $n_{\mathcal{F}}$.2
Robot $n_{\mathcal{F}}.n_r^{n_{\mathcal{F}}}$

Figure 5.2: Scheme of the complete snow shoveling system. The arrows denote communication links between the different modules.

operator intervention in case of trouble. The single point of failure problem, which rises from a single command center can be dealt with one or two redundant command center units that take over in case of a failure. Another reason for the centralized approach is that the workspace of the robots is well known in size and structure. Therefore, the scalability provided by a decentralized approach with agents exchanging parts of the map etc. is not necessary.

The highest level of the proposed scheme (see Fig. 5.2) is divided into two types of units. The first one, which we call the *Command Center*, is responsible for the central tasks. The second kind of units (blocks denoted as *Formation I - Formation* $n_{\mathcal{F}}$) represents the current constellation of vehicles where each unit corresponds to one formation. These units, which are independent from the *Command Center* most of the time, are primarily responsible for putting the assigned task into the appropriate formation motion.

The core of the *Command Center* is the *Task Allocation* module that utilizes two data structures: *MAP* and *GRAPH*. The *GRAPH* data structure is prepared off-line from the *MAP* by assigning nodes for every crossroad and edges for every road in between. Furthermore, we assign two numbers to the edges of the graph: the first one is an integer value equal to the number of ploughs needed for the sweeping of the corresponding road and the second number is equal to the road's length, which is assumed to be proportional to the needed cleaning time. These values are used by the *Task Allocation* module in order to generate a reasonable output. Additionally, a flag marking the roads that are already cleaned is assigned to each edge. In contrast to the static *MAP*, the *GRAPH* structure can be adjusted by a human operator closing and reopening roads according to the current airport traffic. Also ploughs that detect an obstacle on the road are meant to update the *GRAPH* structure.

The current *GRAPH* structure is used as an input for the *Task Allocation* module that will be described in detail in Section 5.3. An execution step of the *Task Allocation* module is triggered by snowploughs that have just accomplished (or failed) their current task, so they are waiting for new instructions. Besides the *GRAPH*, the module maintains actual plans of all coalitions as well as a priority setting for each road, which depends on the airport traffic as well as on the snowing intensity. Note that the priority setting is not investigated in detail in this thesis, which is focused on the formation driving and path planning but further information on this topic can be found in [160].

In the *Formation I - Formation* $n_{\mathfrak{F}}$ units the *Leader* module is responsible for generating a reference trajectory at the beginning of each task. This is done with the information received from the *Task Allocation* module and with the appropriate coordinates from the *MAP* structure. The leader is just one designated robot in the formation, usually the one in front of the unit. Besides the snow shoveling the leader acts as a connection between the robots and the *Command Center*. It informs the *Command Center* about detected obstacles, finished or aborted tasks as well as the need of additional robots to compensate failures. Note that the leader is not meant to be the reference point in terms of the formation movement, in which it acts like the other robots of the group. The individual control inputs are calculated separately by each follower in order to follow the reference trajectory while maintaining the formation. Necessary signals for synchronization are provided by the leader. As a result the *Formation I - Formation* $n_{\mathfrak{F}}$ units are independent of the *Command Center* most of the time. Further details about the motion coordination of the formation will be presented in Section 5.4.

5.3 Task allocation

The highest reasoning level in the system is an algorithm for designing an optimal sequence of cleaning tasks for the autonomous snowploughs. This is also the level where human operators are allowed to influence the shoveling process. Concretely the operator can adjust the sweeping priorities of the routes, which depend on current airport operations and weather conditions.

This section differs from the rest of the thesis that is focused on the formation control, but it is necessary to get the complete overview of the snow shovelling project. Furthermore it can explain relations and communication links between the ploughs which is the key factor of multi-robot systems. Finally, using the results of the task allocation we can identify all specific situations and maneuvers that can occur during the complete snow shovelling. These cooperative sub-tasks (e.g. formations' splitting, merging, turning, shrinking or obstacle avoidance within the formations) will be utilized in the experimental results demonstrating abilities of the formation driving method.

Two different task allocation algorithms have been designed for the specific application of autonomous multi-vehicle snow shoveling. The first approach provides results off-line, but the global optimum of the solution with respect to the total cleaning time needed for all ploughs can be easily proven. The core of the algorithm is a graph describing all roads that need to be cleaned up and a tree of solutions used for designing the optimal cleaning sequences. The second method is based on forming temporary coalitions of ploughs for each specific task and has been developed for the biggest airports with highly dynamic environment. The decision which route will be cleaned by which robots is based on a heuristic approach combined with a partly explored tree of solutions. There is no guarantee that this method will find the global optimal solution, but the fact that the decisions are made immediately allows the system to respond to sudden changes in the environment.

According to the commonly used taxonomy of the task allocation methods in multi-robot systems published in [68], the problem mentioned above is ST-MR-TA: Single-Task robots, Multi-Robot tasks, Time-extended Assignment. ST means that each robot is capable of performing one task at a time, MR means that tasks require multiple robots and TA means that information for future planning is available, like e.g. the set of all tasks needed for completing a mission. This class of tasks includes coalition formation and scheduling and it belongs to the class of $\mathcal{NP}$-hard problems (cf. [67]). An example for cooperative coalition formation can be found in [1] where an underlying organization is used to guide the formation process. Moreover a reinforcement learning technique is applied to increase the quality of local decisions as agents gain more experience. A local learning strategy for improving the aggregate performance of sensor network agents has also been used in [24].

Another approach designed for reconnaissance scenarios, where a team of scout robots observe several areas of interest, solves the task allocation problem using a market-based strategy [196]. In the investigated scenario, each area can be seen from a set of observation points, thus enabling a task decomposition. In contrast to our algorithm each task does not necessarily need to be accomplished at once, but different observation points of the same area could be visited subsequently. The problem of multiple tasks with multiple robots investigated in [181] is more related to our work. The objective in [181] was to assign teams of robots to certain tasks in the way that the system's overall efficiency is maximized. The authors used heuristics to find approximate solutions and adjusted them for an application in box pushing, room cleaning or sentry duty applications.

The remainder of this section is structured as follows: a complexity study will be introduced in Section 5.3.1. This part should clarify the needfulness of the assumptions applied in the static graph approach as well as the necessity of applying the sub-optimal agent method for large airports. The basic ideas of the proposed static algorithm will be presented in the following Section 5.3.2, while a more detailed description can be found in [77; 159]. The next Section 5.3.3 gives an overview of the second approach, which is not influenced by the complexity of the airfield structure and it allows online changes to the road priorities by operator intervention. Detailed descriptions and additional results of the second approach have been published in [79; 160]. Finally, both algorithms will be compared in the last Section 5.4.2. The comparison is supplemented by results that verify robustness and applicability of the presented system.

5.3.1 Complexity study

A natural description of the complicated airport structure could be a graph. All runways as well as smaller roads can be described by multiple edges where the multiplicity is equal to the number of ploughs needed for their shoveling. Similarly crossings of the roads or places where the runway changes to a narrower path can be replaced by nodes in the graph.

In such a well defined structure an optimal coverage by multiple robots can be found using a graph method. The optimal coverage means that each simple edge is visited by a robot at least once and the total time needed for the complete cleaning is minimal. Let us analyze the utilization of a simple ”breath first” algorithm where the ability to find the global optimal solution can be easily proven.

Fig. 5.3 shows a scenario with two runways a, e connected by three smaller roads b, c, d. We postulated for the simplification of the description that the field is composed of two identical squares and the time needed for shoveling each side is identical and does not depend on the cleaning track. An extension caused by different radii of the curves as well as time lost by turning on the spot are vanished in this case. It means that e.g. the way from node 3 to 4 along the big runway is always two times longer (in both tracks) than the shortest way from 1 to 3.

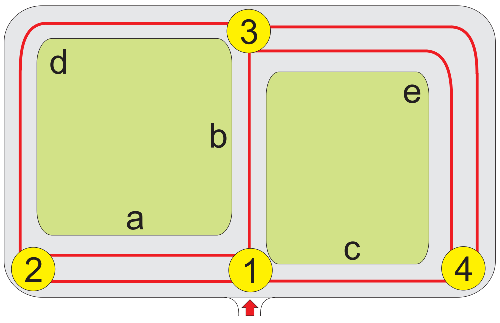


Figure 5.3: Shoveling scenario containing 4 nodes, 2 bigger roads and 3 auxiliary roads.

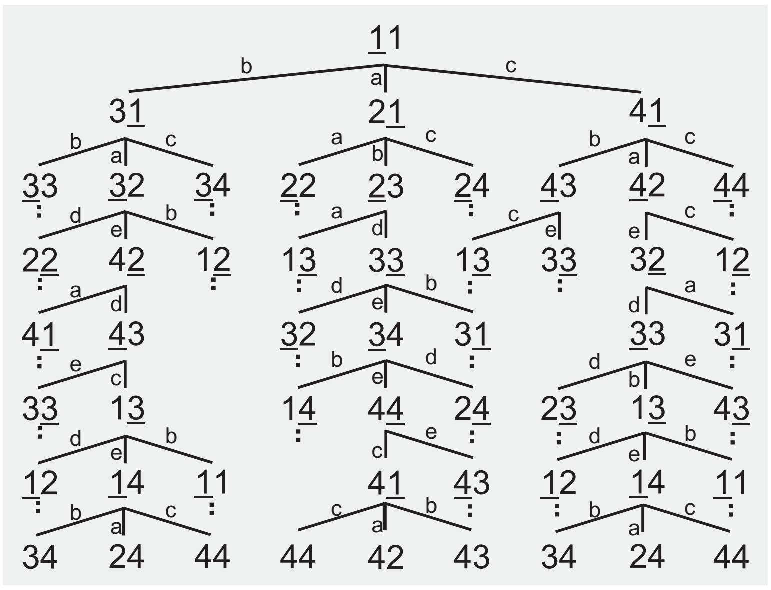


Figure 5.4: Reduced tree with an optimal schedule in the middle column.

Let us analyze the situation where two robots are coming to node 1 facing the arrow. The vehicles are approaching to the node one after another in a short interval. This state is labelled by vector $\underline{1}1$ in the root of the tree depicted in Fig. 5.4. The first underlined digit represents the robot that is already prepared in the crossroad 1 while the second component of the vector without underline means that the second robot is still coming to the node. On the assumption that both robots can shovel the airport independently, the first robot can immediately continue to one of the following nodes 2, 3, or 4 using roads a, b, or c. The new possible states created by the expansion of the root are depicted in the second level of the tree. Similarly, the level 3 is achieved by using the roads a, b, or c by the second plough. In our simple case the complete tree of depth equal to 8 must be explored to find the optimal solution. On the picture of the tree most of the branches are replaced by dots due to size of figure, because the complete tree contains about a 1000 leaves. The path from the root to a leaf, which matches the complete cleaning sequence with minimal total time, will be the desired solution. In Fig. 5.4 an optimal solution can be found in the middle column of the tree, but also in the middle columns of the left branch or the right branch.

Now we would like to find an estimation of the number of leaves that have to be explored in general. Let us suppose that the total time of the best solution of the cleaning problem is equal to the sum of the times needed for cleaning all tracks separately. It is obvious that a tree, whose total cleaning time of corresponding solution is less than or equal to the time of the best solution, has to be explored completely to prove the optimality of the found schedule. The minimal depth of the tree will then be equal to the number of tracks in our simplified example. The total number of tracks that need to be cleaned is

$$W = \sum_{e=1}^{n_E} W_e, \tag{5.1}$$

where n_E is number of all edges in the graph and W_e is the number of robots that are necessary for cleaning the edge e. We are now able to define a simple lower bound of the number of leaves in the tree as

$$n_l = b_{min}^W, \tag{5.2}$$

where b_{min} is the minimal branching factor of the tree. In our simple example

$$n_l = 2^7 = 128, \tag{5.3}$$

which is much lower than the actual number of leaves in Fig. 5.4.[1] Nevertheless for our purpose such a lower bound is sufficient. The estimated number of schedules that must be explored for finding an optimal complete coverage of the Frankfurt international airport (for the map of airport see Fig. 5.5), which was chosen as an example for the verification of the results, is

$$n_l = 3^{761} \approx 10^{363}. \tag{5.4}$$

The number of tracks that should be cleaned by the robots was obtained taking width of all roads from the airport map and width of the runway sweeper RS 200 (for technical

[1]Number of adjacent branches is higher than the minimal branching factor of the tree in most of the nodes which increases the amount of the leaves.

date refer to [137]) usually used for the cleaning of airports. It is obvious that a complete tree with the number of leaves equal to 3^{761} cannot be created.

An idea how to reduce the problem mentioned above is to consider several tracks covering each road as one edge. This means that shovelling of each road will be described as one task that must be accomplished by a sufficient number of robots at one time.[1] Such a simplification makes sense, because each runway should be shovelled at once. The minimal number of ploughs used for the scenario of the Frankfurt airport is then equal to 17, because the width of the biggest runway is about 60 meters and the working width of the sweepers is 3.6 meters.

Using this restriction the exponent in equation (5.2) will be equal to the number of tasks and therefore significantly lower. Nevertheless the estimated number of successors of each state (the branching factor of the tree) has to be changed. In the previous case this number was equal to the number of roads adjacent to the crossroads, because only one robot could be on the crossroad at the same time. Now up to 17 robots can be on the same crossroad (e.g. after cleaning a main runway which needs all ploughs) and the successors must cover all possible options of how the robots can continue. The total number of successors can be computed by

$$S = \sum_{i_1=1}^{n_r+1} \sum_{i_2=1}^{i_1} \cdots \sum_{i_{b-2}=1}^{i_{b-3}} i_{b-2}, \tag{5.5}$$

where b is the number of adjacent roads and n_r is the number of ploughs. For example, 696 successors should be expanded if a group of 17 robots comes to a crossing point with three adjacent roads. This amount together with the still unsatisfactory length of the plan (147 roads need to be cleaned at the Frankfurt airport) make the problem still unsolvable in real time. Therefore, we have to accommodate some restrictions of the task which will be described in the next section.

5.3.2 Static route scheduling approach

The static route scheduling approach for snow shoveling of the airport by autonomous mobile robots is introduced in this section. The core of the algorithm is the graph describing all roads that need to be cleaned up by edges and whose nodes represent the centers of all crossings. The final schedules, which are an output of the algorithm, should cover all edges at least once. Moreover the result should be optimal with respect to the total time needed and it should fulfill the constraint that the robots do not take the edge from where they came from.

As shown in the previous subsection 5.3.1, the complete tree of solutions cannot be explored due to the high complexity of the task allocation problem. Fortunately not all of the states of the tree are feasible with respect to safety regulations applied on the airport. Usually each airport has several parallel runways with appropriate service roads. In order to prevent collisions between snowploughs and the airport traffic, such parts should be shoveled separately. During the cleaning phase, the airport traffic is not allowed to enter the area and the other parts of the airport are forbidden for the

[1]Each task is defined as a movement of a group of robots from node to node.

ploughs at this time. The Frankfurt airport usually utilizes two parallel runways, called A and B. We divide all roads surrounding the runways to the three non-overlapping set for the task allocation algorithms. These sets will be denoted A, B and AB as can be seen in Fig. 5.5. The edges of set A (resp. B) need to be cleaned up completely while runway A (B) is in shoveling mode. The corresponding roads are used by the airport traffic, when the appropriate runway is in active mode. Edges of the set AB will be cleaned up with set A or B, or with both of them. These auxiliary roads are only used by the ground crew that is able to avoid the snowploughs. Therefore they do not need to be closed for the robots.

Another indispensable simplification of the searched space is enabled by the safety regulations, which require shoveling each road at once and by the specific character of the airport: the service roads are almost two times narrower than the main runways. Due to this we can expect that the optimal solution will be to clean the runway by a big formation firstly and to divide it into two equal parts afterwards in order to clean up the remaining roads. Hence, in the final solution there is no time wasted for robots waiting for others to form up.

The reduced number of states needed for scheduling both parts is unfortunately still unsatisfactory. To improve the algorithm the selection rule that determines the state that will be expanded in the next step was changed. In the "breadth first" algorithm the first found state that is still unexpanded is usually chosen. But if instead the unexpanded state with the lowest waste time[1] would be expanded, the most promising solutions are favored. Furthermore, once a complete solution is found, we can prove its optimality because all states with lower waste time than in the first found solution are already evaluated. This approach has been motivated by work in [27], where the branch and bound algorithm has been utilized for scheduling thermal generation units in power system economic operations.

A pseudo code of such an algorithm proposed for airport snow shoveling can be found in Algorithm 1. The cleaning loop for the complete Frankfurt airport can then be summarized in the following 6 steps:

1. All robots clean the runway of *Area A* in a big formation. The schedule is a sequence of the edges from node 2 to node 31.

2. Two smaller formations clean the remaining edges of *Area A*. The edges of *Area AB* can also be used. The schedule of this step depicted in Fig. 5.6 was found by Algorithm 1.

3. Both formations move to node 19. The shortest path in the graph is calculated by the Dijkstra algorithm [44].

4. The reunited formation cleans the runway of *Area B* from node 19 to node 79.

5. The newly divided formations clean all edges of *Area B* and *Area A+B* that were not cleaned up in steps 2.-4. The edges of *Area B* and *Area A+B* that are already cleaned can be used during this step too.

[1]Wasted time is defined as the time lost by ploughs moving without sweeping or by waiting for additional vehicles.

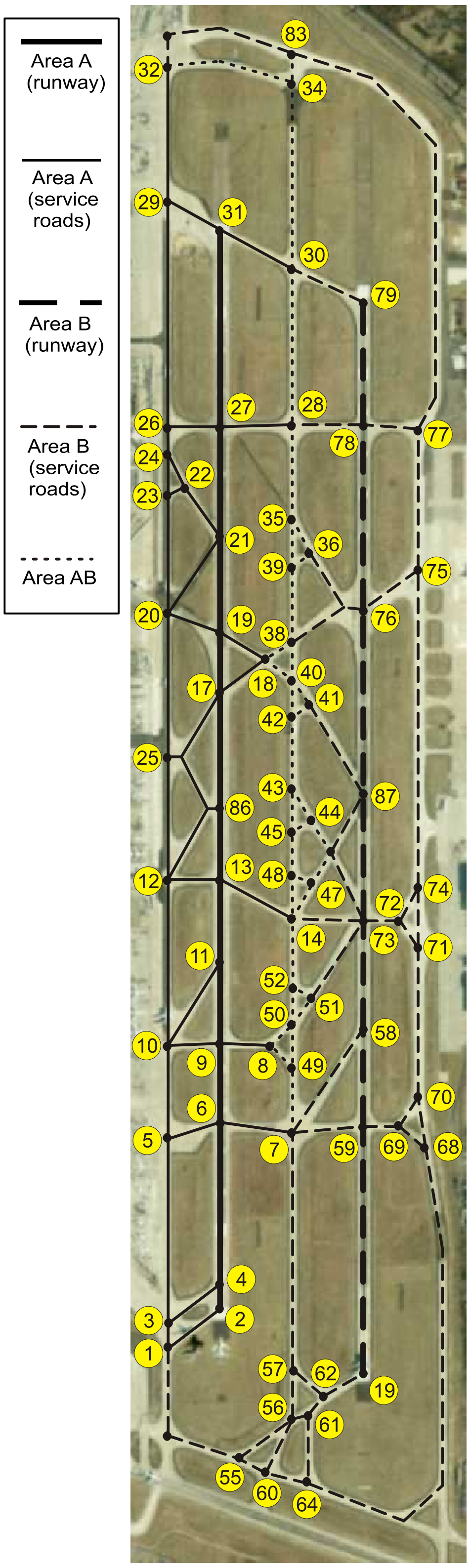


Figure 5.5: Map of Frankfurt Airport with roads and runways partitioned into 3 sets.

6. Both formations move to node 2 and the sequence is restarted with step 1.

Algorithm 1: Pseudo code of the scheduling algorithm.

```
SOLUTIONS ⟵ $\emptyset$
UNEXPANDED ⟵ initialState
expandingState ⟵ ShortestTime(UNEXPANDED)
while UNEXPANDED $\neq \emptyset$ && UncleanedRoads(expandingState) $\neq \emptyset$ do
    UNEXPANDED ⟵ UNEXPANDED − expandingState
    node ⟵ StaticNode(expandingState)
    NEWNODES ⟵ Successors(node)
    foreach newNode of NEWNODES do
        newState ⟵ AdvancedState (newNode,expandingState)
        UNEXPANDED ⟵ UNEXPANDED + newState
        if UncleanedRoads(newState) = 0 then
            SOLUTIONS ⟵ SOLUTIONS + newState
    expandingState ⟵ ShortestTime(UNEXPANDED)
if UNEXPANDED = $\emptyset$ then
    return NoSolution
else
    return ShortestTime(SOLUTIONS)
```

`ShortestTime` returns the state with the lowest wasted time.

`UncleanedRoads` creates a list of uncleaned roads in the current state.

`StaticNode` returns the node (of the graph) where a formation recently finished its task.

`Successors` returns all feasible nodes that can be reached from the actual node in one step.

`AdvancedState` returns the new state which results from moving the formation from `StaticNode` towards the specified node.

The total number of states that were evaluated during the scheduling of part A is equal to 61520. The optimal solution, which is depicted in Fig. 5.6, was found during 32 minutes using a Pentium 4 CPU 3.2GHz, 512MB of RAM. Such a run time is sufficient in the case of a static map and a constant number of ploughs, where the plan can be prepared off-line before the mission.[1] The problem occurs if the schedule has to be changed. In reality the map of an airport can be changed due to obstacles blocking a road and also the number of ploughs can be reduced by failures. The scheduling approach should be able to respond to the new situation promptly which should reduce time when the ploughs are waiting for a new plan.

[1]The computational time of the scheduling for *Area B* and the time of the path planning necessary for moving between the areas are significantly shorter than time needed for the *Area A*.

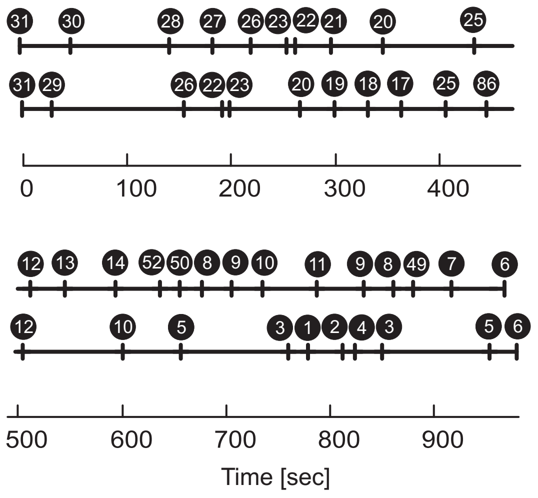


Figure 5.6: Time schedule for cleaning the service roads in *Area A*. The sequences for two formations show order and time in which each node should be visited.

5.3.3 Dynamic task allocation method

In this method, which is designed to provide the dynamic response to sudden changes in the environment, a multi-agent approach is used to describe the behavior of the ploughs. In the *Task Allocation* module, introduced in Fig. 5.2, each *Formation* unit is considered to be one object (agent) with an assigned task. When a cleaning task is finished, the same agent can only be used for another road, if the number of ploughs is still sufficient. If a bigger group is needed for the new task, the agent has to ask the remaining robots for help. Contrariwise, if the new road is narrower, so much that the group could be reduced, the surplus ploughs will be offered to the other agents as available robots. The crucial part of this approach is to assign which road should be swept by which agent.

5.3.3.1 Heuristics

The idea of the presented method can be described in two steps. In the first step a sorted list L_t is generated. This list consists of all uncleaned roads (tasks) ordered by the distance from the beginning of sweeping of the area to the intersection where the cleaning of the corresponding road would be started. This guarantees that no road will be omitted during the cleaning process and that the ploughs will not have to move too far before they continue to sweep. In the second step free robots are allocated to the most prioritized roads. For this we choose the first $n_{\mathcal{F}}$ roads from L_t, where $n_{\mathcal{F}}$ is the biggest integer satisfying

$$\sum_{i=1}^{n_{\mathcal{F}}} W_i \leq n_{fr}. \tag{5.6}$$

Thereby W_i denotes the number of ploughs needed for accomplishing the i-th task in L_t and n_{fr} denotes the number of robots that are currently free[1]. After this the free

[1]Free robots are ploughs that just finished the task or redundant robots offered by other agents.

robots are assigned to groups $\mathcal{F}_i$, where $i \in \{1, \dots, n_{\mathcal{F}}\}$, in a way that the term

$$\max_{i \in \{1,\dots,n_{\mathcal{F}}\}} \left(\max_{j \in \{1,\dots,n_r^i\}} (\mathrm{time}_{i,j}) \right) \tag{5.7}$$

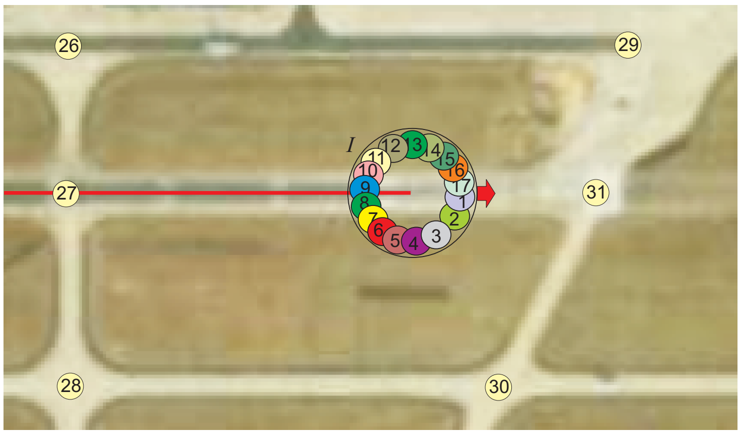
is minimal. Constant $\mathrm{time}_{i,j}$ denotes the time needed for plough j to move from its actual position to the beginning of road i and n_r^i is the number of ploughs in $\mathcal{F}_i$.

The application of this method provides one solution of the task allocation problem for the momentary free robots. This solution is optimal with respect to the used informations, but from the global point of view it may not be. Unfortunately, the heuristics mentioned above cannot solve all possible situations in a satisfactory manner. An example of an obviously wrong decision is shown in Fig. 5.7. In the first snapshot of the simulation the big formation of robots I is coming to the end of the main runway and should be splitted for shoveling smaller auxiliary roads. A situation after applying the heuristics is depicted in the second snapshot (see Fig. 5.7(b)). While the sub-formation II is shoveling the road between crossroads 31 and 30 and the sub-formation I is shoveling the road between 31 and 29, the sub-formation III is going to clean up road 29-26. In the following snapshot (see Fig. 5.7(c)) the formation II is cleaning the road between 30 and 28 and the formation I is going back to the crossroad 31, which is the shortest way to the road 27-28 that is the next in the sorted list of tasks. It is obvious that a more optimal solution could be to continue with the formation I to clean the road 29-26 and to use freed III for shoveling another uncleaned way.

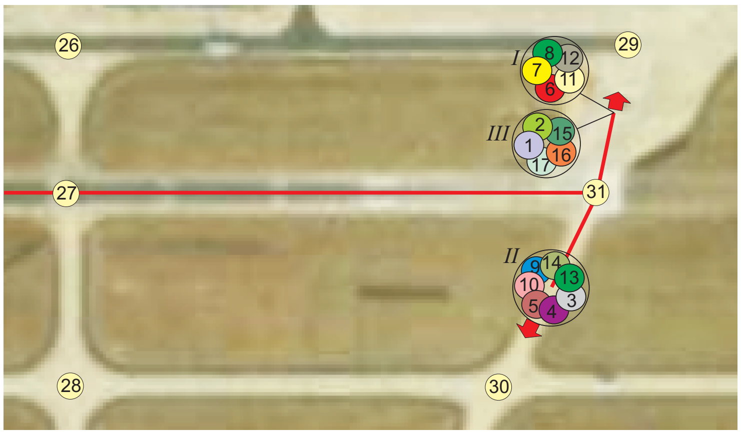
Such a smarter result can be obtained using more "sophisticated" heuristics or using an approach, which enables to expand the local information about the actual state of ploughs. The biggest disadvantage of the heuristics' extension is the impossibility to cover all situations that can happen. The heuristics become too complicated and a debugging of the scheduling process is troublesome. On the contrary the second approach could solve the problem more generally and transparently. An improvement of the method will therefore be achieved by selecting the most promising solution within a certain set.

5.3.3.2 Exploring the state of solution

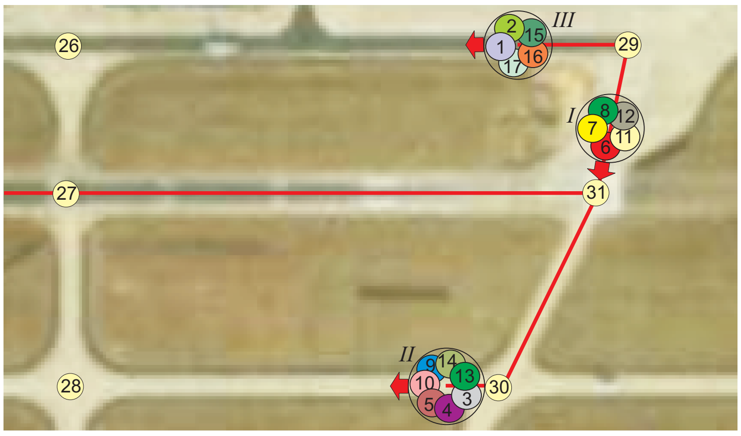
First we will generate all possible sets of tasks that satisfy the inequality (5.6). After this the sets are put in a list L_s ordered by the sum of the tasks' indeces in the sorted list L_t. The lowest index in the set is decisive for elements with an identical sum of indeces. Note that for the planning only the first b_t sets from L_s will be used, where the parameter b_t is an input of the algorithm that depends on the available computation capability. If the number of elements of L_s is less than b_t all sets from L_s will be used. To illustrate this approach with an example we assume that each task from a list $L_t = (A_1, B_2, C_3, D_4)$ requires exactly one plough (the subscripts correspond to the task's index within the sorted list L_t). We assume that there are just two ploughs available for sweeping at the moment. As a result we receive $L_s = (AB_{1+2=3}, AC_{1+3=4}, AD_{1+4=5}, BC_{2+3=5}, BD_{2+4=6}, CD_{3+4=7})$. It is obvious that in general the whole tree cannot be explored, but due to the heuristics described in Section 5.3.3.1 it is not necessary, since the most promising solutions will be preferred.



(a) One formation of 17 snowploughs sweeps the main runway.

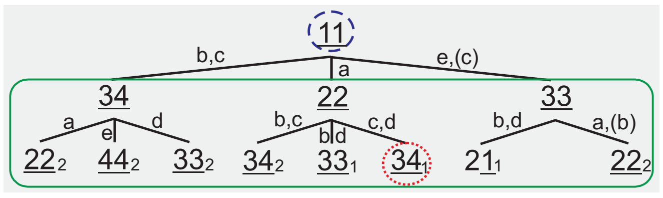


(b) Formations *I* and *II* are shoveling auxiliary roads. Formation *III* is moving to clean road 29-26.

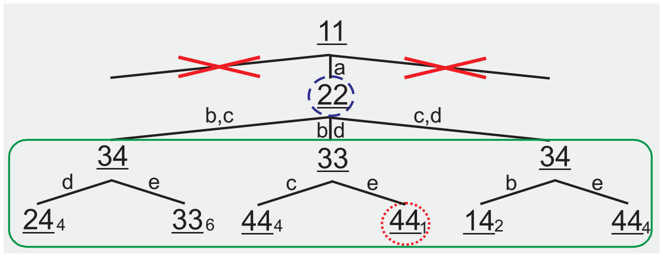


(c) Formations *I* and *III* are shoveling uncleaned roads 30-28 and 29-26. Formation *I* is coming to shovel road 27-28.

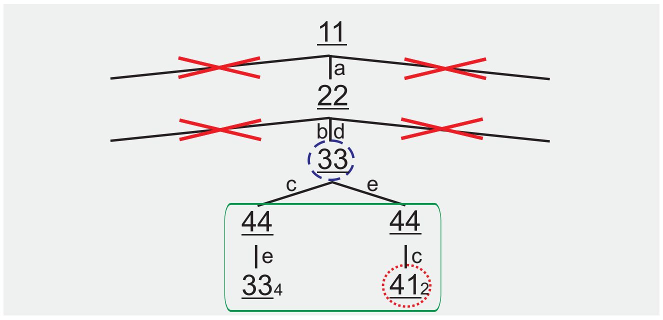
Figure 5.7: Suboptimal solution of the route scheduling task. Each small bubble denotes one robot. Roads that are already cleaned are depicted by solid line. The positions of robots in the circles denoting the formations are only illustrative. The real shape of shoveling formations will be shown in Section 5.4.



(a) step 1



(b) step 2



(c) step 3

Figure 5.8: Stepwise partial exploration of the tree of solutions. The actual state of the ploughs is marked by a dashed circle and the evaluated states are encircled with a solid line. The state with the shortest wasted time in the lowest level is marked by a dotted circle. The subscript numbers denote the summed amount of wasted time.

Notice that without the extension presented in this section only the first element AB_3 would be chosen automatically. In contrast, the extended method also examines other promising solutions, which could be closer to the global optimum for the complete sweeping task.

In the second step of this extended task allocation one set of tasks has to be chosen from the first b_t elements of L_s. Here, we designed a decision-making process that is based on the partial exploration of the tree of possible solutions. To explain the idea of the method, we look at the scenario depicted in Fig. 5.3 and at the steps of exploring the tree of solutions that are shown in Fig. 5.8. We assume that two robots are employed in this example. The depth of explored sub-trees is set to $d_t = 2$ and the parameter $b_t = 3$. Furthermore, the edges a and e are meant to require two robots for the sweeping.

The tree depicted in Fig. 5.8(a) reflects the beginning of the decision-making process. Both ploughs are located at the initial node 1 and the lists for the decision process are $L_t = (a_1, b_2, c_3, d_4, e_5)$ and $L_s = (a_1, bc_5, e_5, bd_6, cd_7)$. The initial state is expanded using the first b_t elements of L_s. This procedure is repeated until the obtained subtree has depth equal to $d_t + 1$. Then a state with the shortest wasted time ($\underline{34}$) is chosen in the lowest level of the sub-tree. If there are multiple states with equally wasted time, one of them is chosen randomly. Note that these states are optimal with respect to the limited information obtained by the partial exploration of the tree. Now, the snowploughs can continue to the next state, towards the one with the lowest wasted time. Once the next state ($\underline{22}$) is reached, the planning process is restarted from this state as depicted in Fig. 5.8(b). The local optimal state $\underline{44}$ is selected in the third level of the new sub-tree (fourth level of the tree). In the third decision step (cf. Fig. 5.8(c)) the sub-tree's root is at node $\underline{33}$ and the selected node with the shortest wasted time is $\underline{41}$, which completes the sweeping of the whole field. Note that the final schedules for the robots $R_1 = (1 - 2 - 1 - 3 - 4)$ and $R_2 = (1 - 2 - 3 - 4 - 1)$ have been obtained by the evaluation of only 15 states. For comparison, more than 1000 states need to be evaluated to explore the whole tree in this small scenario as is shown in the complexity study in Section 5.3.1.

Finally, we should make an estimation of complexity of the dynamic task allocation approach presented in this section, which is necessary to setup the parameters b_t and d_t appropriately and with respect to the available computational power. Since, in each step of the algorithm there is at least one road that will be cleaned, an upper bound for the states that have to be evaluated is given by

$$n = n_{roads} b_t^{d_t}, \tag{5.8}$$

where n_{roads} denotes the number of roads that need to be shoveled.

5.3.4 Simulations and parameters tuning

In this section we present the results of the simulation of the developed task allocation approaches with their comparison at the end. Thereby we use the area of the Frankfurt international airport as the sweeping scenario with all roads divided into the three non-overlapping sets A, B and AB as was introduced in Section 5.3.2. Such an assumption is necessary for the utilization of the static algorithm and is also required by the safety rules.[1]

The number of vehicles needed for each road has been derived from the roads' width and the shovel's width of the runway sweeper RS 200 which was shown in Section 5.3.1. Based on this the following results were obtained by utilizing a group of 17 ploughs, because minimal number of ploughs needed for the sweeping of runways at the Frankfurt airport is equal to 17.

First, let us analyze the dynamic algorithm, which is more sensitive to the appropriate setting of variables. The important problem is to find values for the algorithm's branching factor b_t and its sub-tree size d_t. The results of the simulations with different settings can be found in Table 5.1. In the table the quality of the cleaning process

[1]In a general case the partitioning of the airport is not necessary for the dynamic approach, which is independent of the complexity of the map.

Table 5.1: Total time in seconds needed for cleaning *Area A* using schedules found by the algorithm with different parameters d_t and b_t.

time [s]	$d_t = 2$	$d_t = 3$	$d_t = 4$	$d_t = 7$
$b_t = 2$	1385	1289	1071	1071
$b_t = 3$	1385	1138	898	898
$b_t = 4$	1161	1132	898	898
$b_t = 7$	1161	898	898	898

refers to the time needed for the complete sweeping of *Area A*. It is easy to see that better solutions are found with the increasing depth of the explored sub-tree d_t, which matches the planning horizon. The same relation holds for the branching factor b_t. Unfortunately both parameters significantly influence the computational time of the algorithm as you can see in the equation (5.8). The best solution has been found by setting the parameters to $b_t \geq 3$ and $d_t \geq 4$. For $b_t = 3$ and $d_t = 4$ only 81 states have been evaluated every time a new agent was created or new tasks were distributed among existing agents. Hence a decision was found within fractions of milliseconds. This shows that the decision-making process can quickly respond to sudden changes in the workspace.

Snapshots from the simulation of the best solution are depicted in Fig. 5.9 and Fig. 5.10. An animation of the complete sweeping process can be found on the website [55]. Fig. 5.9(a) shows formation *I* with all 17 robots that shovel the last part of the runway in *Area A*. Note that the figures represent the current state of the *Task Allocation* module, while the vehicles actually move along the trajectories designed in the level of the formation unit as it was described in Section 3.4.[1] As depicted in Fig. 5.9(b) the shoveling process continues by dividing the big formation at the end of the runway into three groups denoted by *I*, *II* and *III* with corresponding tasks 31-29, 27-26 and 31-30. In contrast to the Fig. 5.7 the robots are distributed optimally and the redundant movement of two formations along the road 31-29 is corrected. The formations are independent from the *Command Center* after the tasks are assigned. They will report to the *Task Allocation* module once the task is accomplished or in case of a failure (e.g. blocked road, defect plough). Note that formation *II* consists of 7 ploughs, even though the cleaning of the road 27-26 requires only 5 vehicles. This is a result of the task allocation algorithm that also considers the progress of the cleaning process in the near future. Two additional robots in formation *II* increase the solution's value, since the task allocation plans to assign the formation to road 26-24 afterwards, which requires 7 ploughs. The sequence of snapshots depicted in Fig. 5.10 shows the appropriately sized formations, which continue with the sweeping of the auxiliary roads. One can see that the coalitions have changed compared to the preceding situation in Fig. 5.9(c).

[1]The agents are depicted as circles with identification numbers of ploughs for an illustration of the scheduling process. The real shape of the shoveling formation will be shown in Section 5.4.

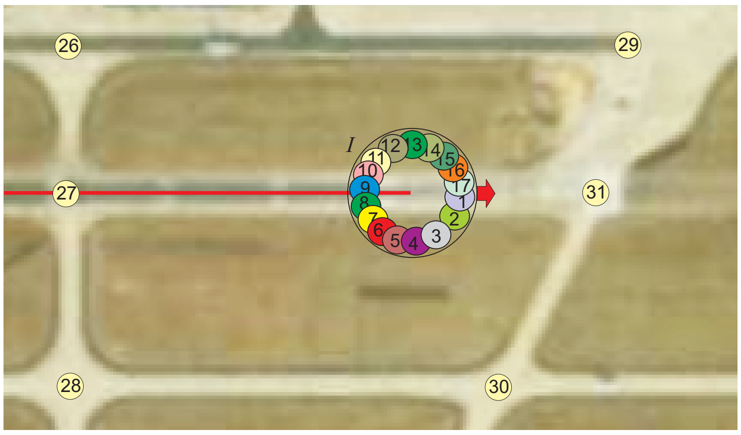
Table 5.2: Comparison of the static and dynamic task allocation methods. − denotes insufficient attribute, + sufficient attribute and ++ very good quality

	Static method	Dynamic method
Computational complexity	−	++
Response to changes	−	++
Total time of scheduling	+	+
Proof of optimality	++	−
Accommodated restrictions	−	++
Environment complexity	+	++

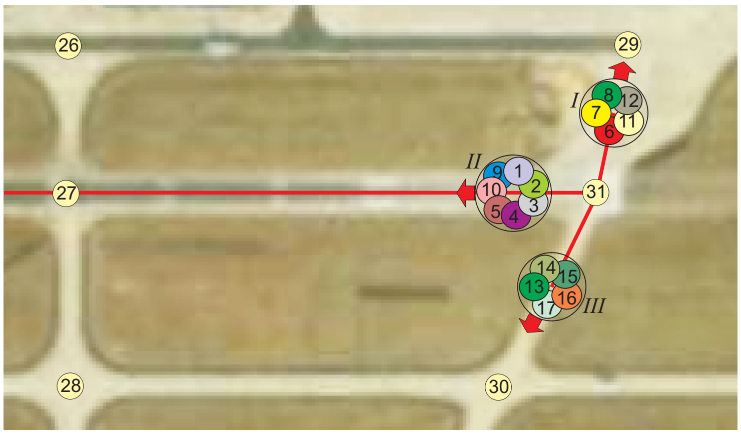
Fig. 5.11 shows snapshots from the same time interval as in Fig. 5.10 with the difference being the road 21-20 that is blocked by an obstacle. When formation III detects the obstacle, the group's leader reports its coordinates to the *Command Center*. After the *GRAPH* structure has been updated the *Task Allocation* module computes and assigns a new task to the robots of formation III. As one can see in Fig. 5.11(b), formation III heads back to node 21 in order to continue towards node 19 where formation II is waiting for additional vehicles. Formation I has been extended with two ploughs from formation II in order to clean the wider road 20-25, which originally should have been cleaned by formation III. Fig. 5.11(c) shows the extended formation II shoveling the road 19-18 and the remainder of formation III is coming to clean the road 17-25.

A direct comparison of both task allocation methods is difficult due to the different applied assumptions. In the static approach the ploughs have to be splitted into two equivalent groups and it is assumed that such formations will be sufficient for cleaning the auxiliary roads. On the contrary in the static approach the ploughs cannot turn on the spot which could have provided better solutions. The only exact comparison of the approaches can be obtained from Table 5.1 and from Fig. 5.6 where the method was applied for the scheduling of the *Area A* of the Frankfurt international airport. The total time needed for the realization of the plan obtained by the static approach is 962 seconds which is slightly worse then the result designed by the dynamic approach with setting $b_t \geq 3$ and $d_t \geq 4$. Nevertheless, such confrontation cannot show the practical advantages of the algorithms. We will try to give a better overview of the features of the methods by a summary in Table 5.2

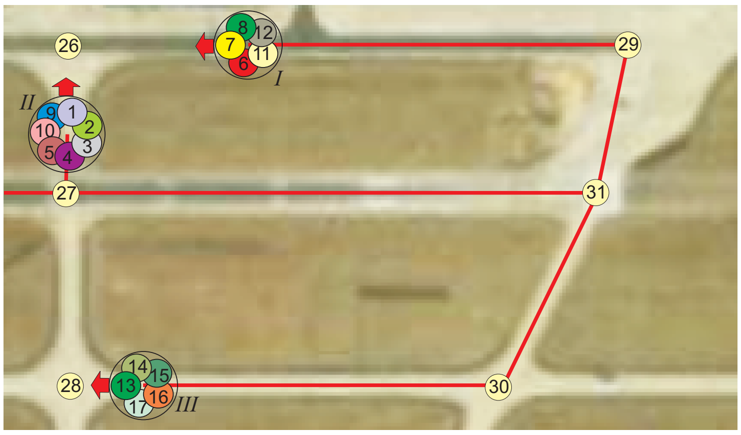
The first row in the table "Computational complexity" is quite obvious. While the required computational time for the static approach is more than 30 minutes, using the dynamic approach the complete schedule for all ploughs can be designed in less than one second. Furthermore, in the static method the ploughs have to wait the whole time even for the first part of the plan, whereas in the dynamic method the plan is designed subsequently and the first result is available in less than 20ms. This fact is crucial for the second row of the table "Response to changes", where the immediate result of planning is essential. Additionally, an immediate reaction to robot failures as



(a) One formation of 17 snowploughs sweeps the main runway.

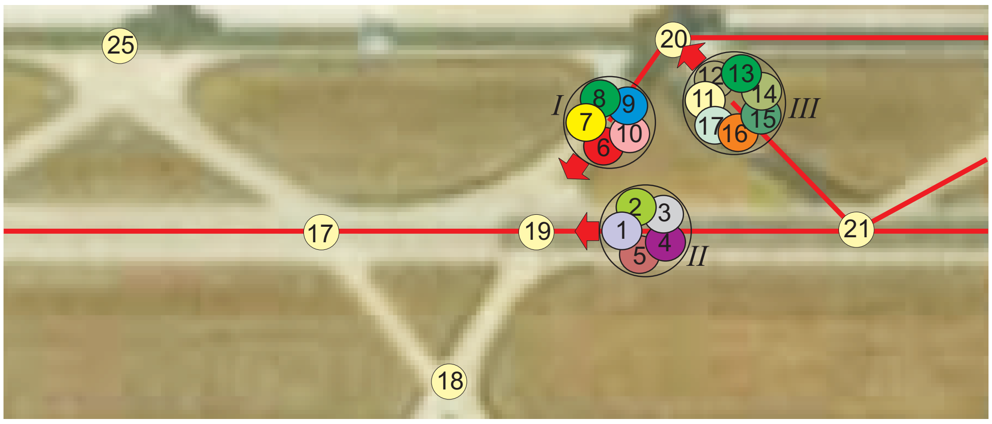


(b) Formation *I* and formation *III* sweep roads 31-29 and 31-30. Formation *II* is preparing to clean road 27-26.

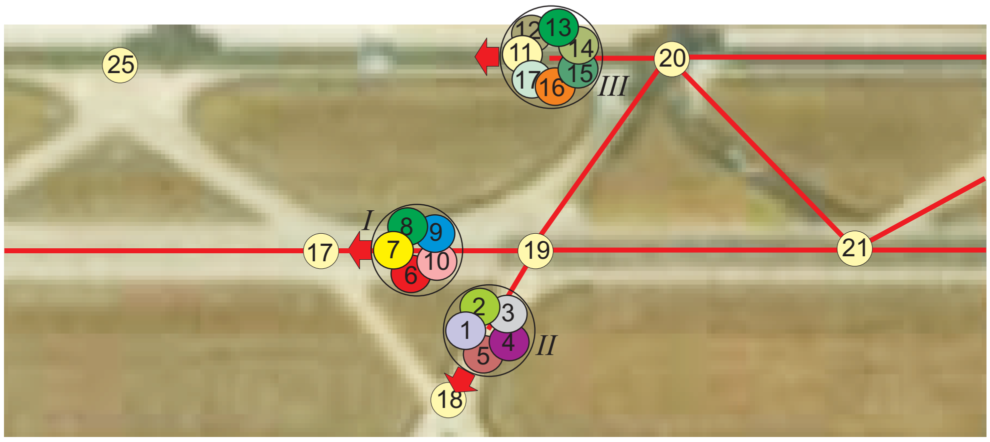


(c) Three formations with a variable number of ploughs are shoveling a part of the international airport in Frankfurt.

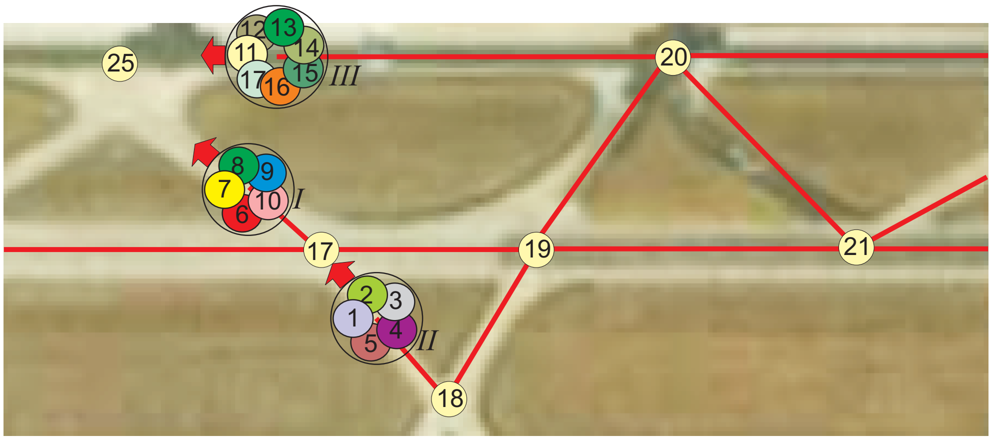
Figure 5.9: The division of a big formation into 3 formations that start cleaning the auxiliary roads at the end of the runway *A*. Numbered circles in the intersection denote the nodes of the *GRAPH*. All other numbered circles correspond to one robot. Roads that are already cleaned are depicted by a solid line.



(a) Formation I sweeps road 20-19, formation III sweeps road 21-20 and formation II is preparing to clean road 19-18.

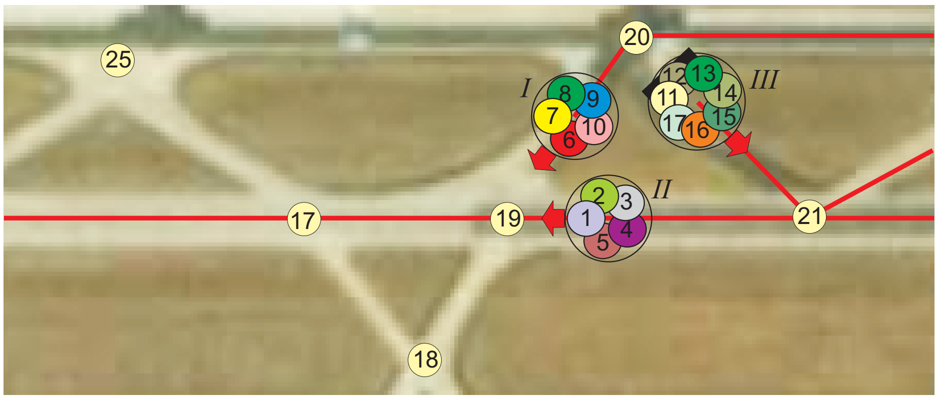


(b) Formation II sweeps road 19-18, formation III sweeps road 20-25 and formation I is preparing to clean road 17-25.

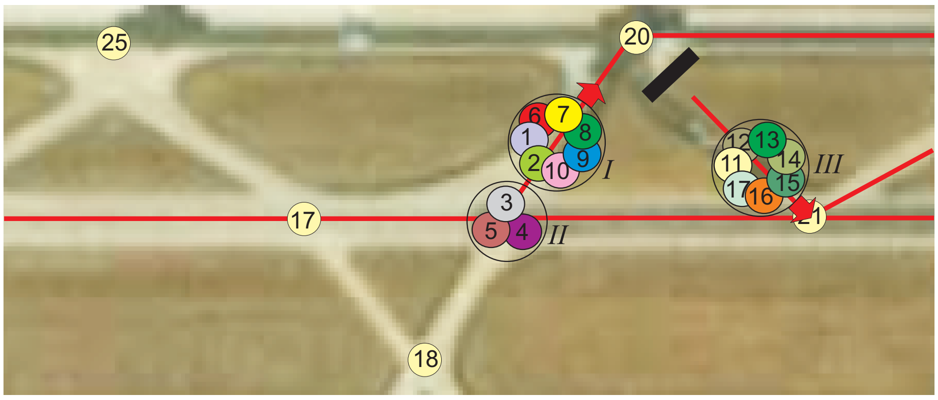


(c) Formations are cleaning roads 18-17, 17-25 and 20-25.

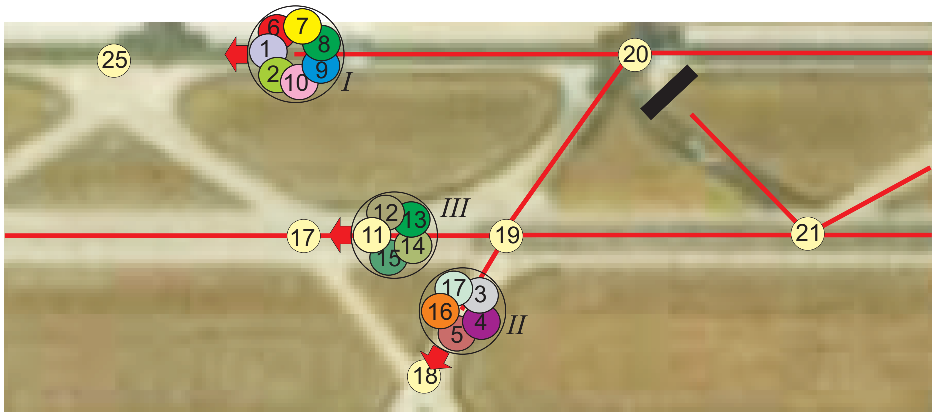
Figure 5.10: Snapshots from the task allocation simulation for *Area A* of Frankfurt airport.



(a) Formation *III* detected a blocked road and is waiting for a new task. Formation *I* is sweeping road 20-19 and formation *II* is moving towards node 19 in order to extend formation *I*.



(b) Formation *I* is preparing to clean road 20-25, while formation *II* is waiting for additional robots and formation *III* is going to support formation *II* in the sweeping of roads 19-18 and 17-25.



(c) Formation *I* is sweeping road 20-25, formation *II* is sweeping road 19-18 and formation *III* is preparing to clean road 17-25.

Figure 5.11: Snapshots from the task allocation simulation for *Area A* of the Frankfurt airport where road 21-20 is blocked by an obstacle.

well as to sudden changes in the environment like e.g. detected obstacles or blocked roads increase the robustness of the algorithm.

The quality of the solution is compared in the rows 3 and 4. As presented above, the "Total time of scheduling" is comparable for both method and is sufficient for the application at the airport. Nevertheless we cannot say anything about the optimality of the result obtained by the dynamic method. The solution, which is sent back by the static approach, is always optimal with respect to applied restrictions. We should mention that the property "Accommodated restrictions" is one of the biggest disadvantages of the static method that derogates the applicability to the existing safety regulations. Contrariwise the dynamic method is very flexible and it is not restricted by the present conditions on the airport. The last quality index "Environment complexity" is satisfactory for both methods, but in the case of the static approach only at the cost of necessary decomposition of the airport to sufficiently small areas. The utilization of the dynamic method is independent of the airport structure at all without any restrictions.

5.4 Formation driving

5.4.1 Method description

The formation driving algorithm for the airport snow shoveling can be easily implemented using the approach introduced in Section 3.4. The only difference can be found in the desired path which is provided by the *Path Planning* module (see Fig. 3.1). In the previous approach the input was a smooth and feasible path composed from a sequence of multinomials. Here, the formation should follow axes of the runways to cover the surface that is mainly used by the airplanes. The desired path is then a sequence of connected axes of roads provided by the *Task Allocation* module, which was introduced in Fig. 5.2. Such a path composed from line segments is not feasible for the formation of car-like robots, but an appropriately adjusted RHC method, that is described in Section 3.4, can overcome any unsmooth connections.

To reutilize the description as well as the proof of convergence from Section 3.4 the k-th line segment will be described by equation

$$\varphi(k, s) \quad = \quad (P_k - P_{k-1})\, s + P_{k-1}, \tag{5.9}$$

where parameter s is within the interval $\langle 0, 1\rangle$, the points P_k, where $k \in \{1, \ldots, \tilde{n} - 1\}$, are crossings of axes of neighboring roads, P_0 is the beginning of the first axis and $P_{\tilde{n}}$ is the end of the last axis. The whole string of the segments is expressed as $\varphi(k, \cdot) = \{\varphi_x(k, \cdot), \varphi_y(k, \cdot)\}$, where $k \in \{1, \ldots, \tilde{n}\}$, in the case of 2D. Parameter $\tilde{n}$ is the number of roads provided by the *Task Allocation* module. The rest of the description in Section 3.4.1 and the proof in Section 3.4.2 can be used without any changes.

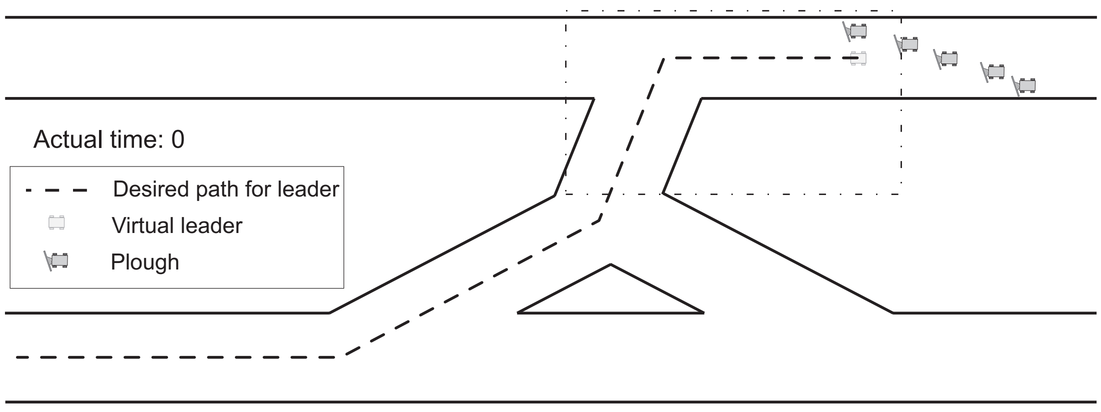


Figure 5.12: Workspace of ploughs with the depicted desired path for the formation and with denoted area which will be closely investigated in the snapshots presented in Fig. 5.14-5.16.

5.4.2 Experimental results and parameters tuning

In this section, results obtained applying the RHC method for the driving of formations of ploughs in an airport environment will be presented. We will investigate an influence of the length of planning horizons on the characteristics of solutions. Furthermore, we will verify an usefulness of the part of the cost function penalizing control inputs that are too aggressive.

An environment of two parallel runways connected with several roads with the same width was chosen as the test scenario (see Fig. 5.12). The task of the ploughs is to follow a preplanned path tracking the axes of the roads that need to be cleaned. The crucial problem of the path following method is to overcome unsmooth connections of the line segments and still to maintain the optimal coverage of the runways. The initial position of five ploughs and the dashed line representing the desired path is depicted in Fig. 5.12. The dot-dash rectangle borders region around one of the path breaks that will be closely investigated in the study of lengths of time horizons. Finally, we should remark that the robots are initialized with their maximum velocity and with their position randomly generated around the desired position within the formation. This initialization should bring the simulation close to real application.

The optimal setting of the length of the planning horizon N as well as the number of states n that will be applied in each optimization loop should satisfy contending claims as mentioned in Section 2. Intuitively the number of states N used for planning significantly increases time needed for obtaining desired control inputs for the ploughs. The maximal time T_{max} that can be used for computing can be easily defined by the parameter n as $T_{max} = n\Delta t$, because the new plan should be prepared during the pursuit of the n control inputs. The time interval Δt is in the proposed approach fixed as $\Delta t = 0.25s$[1]. Experimental measurements of the maximal time required during the

[1]The maximal working speed of the runway sweeper RS 200 usually used for the snow shoveling of airports is 65km/h (see [137]) and so the distance between the two states in the plan with $\Delta t = 0.25s$ is less than 4.5m. This is approximately one third of the total length of the sweeper which is proposed as an optimal setting in literature (see e.g. [148]).

Table 5.3: Maximal computational time needed for leader planning and maximal computational time required for the followers during the mission using different setting of parameter N. The algorithm was implemented in Matlab environment and the results were obtained using a Pentium 4 CPU 3.2GHz, 512MB of RAM.

N [-]	2	3	4	5	6	7	8	9	10
time [s] (leader)	0.02	0.05	0.11	0.27	0.47	0.67	1.02	1.74	2.23
time [s] (followers)	0.02	0.05	0.06	0.09	0.11	0.12	0.16	0.18	0.21

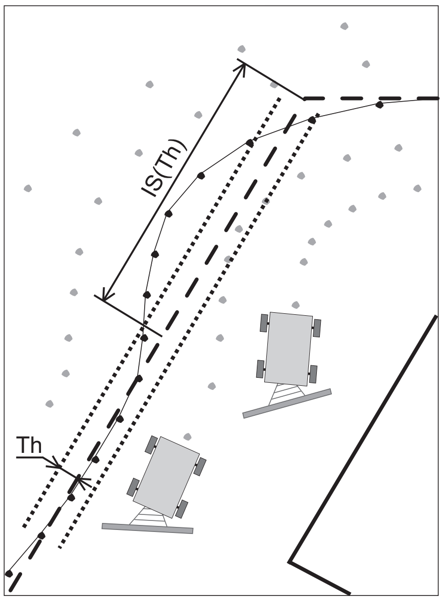


Figure 5.13: Explanation of the interval of stabilization used for the evaluation of results.

formation driving in the airport environment (Fig. 5.12) are presented in Table 5.3. An interesting result can be seen in comparison of the time complexity of the optimization process for the virtual leader and for the followers. While the time necessary for the virtual leader grows more than quadratically, the time used by the followers is only linearly dependent on the parameter N. This fact is caused by the formation driving method used for the initialization of the followers' planning as introduced in Section 2.3.3.2. The optimization is then used only for suppression of the sensors and actuators uncertainty or to avoid possible collisions with obstacles and within the team. For the tuning of the horizons' length, only the time required for the virtual leader must be considered, because the shorter optimization for the followers is done in parallel.

The aim of the path following approach presented here is to overcome the connections of the line segments so that the center of the formation is deviated from the desired position minimally and for the shortest possible interval. We will use an interval of stabilization to evaluate the different combination of values N and n. The interval, which is graphically introduced in Fig. 5.13, is defined as follow.

Table 5.4: Values of $IS(d_r/4)$ of solutions obtained by the algorithm with a different setting of parameters N and n. The sign ∞ denotes simulations in which the position of the virtual leader does not satisfy the conditions in the definition of $IS(Th)$ for $Th = d_r/4$. Constant d_r is the width of the plough's shovel. Unfeasible settings of parameters are marked by the sign x.

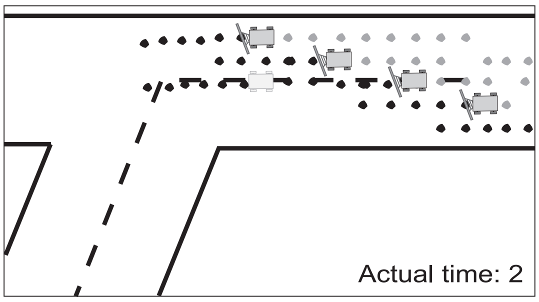
$n \backslash \backslash N$	2	3	4	6	8
1	∞	2.44	2.19	1.50	1.50
2	∞	2.35	1.99	1.48	1.44
3	x	∞	2.44	2.05	1.50
4	x	x	∞	2.16	1.49
5	x	x	x	2.72	2.69

Definition 5.4.1. Interval of stabilization Let us define the interval of stabilization $IS(Th)$ as the length of the desired path between the connection of line segments and the point from which the difference between the virtual leader position and the desired position on the followed path stays less than or equal to a threshold Th as long as the planning interval N does not reach the next connection of line segments.

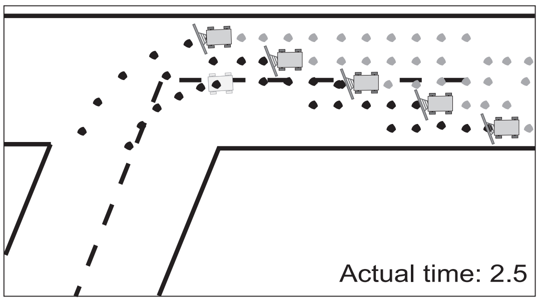
The values of $IS(Th)$ obtained from simulations of formation driving using the method with different settings of parameters N and n are presented in Table 5.4. The obvious result is the correlation of solutions' quality with the value of N. Longer time horizon can response to the sudden change of the formation heading better as well as suppress the consequence oscillations better. For the lowest values of N the process can even be unstable. The sign ∞ denotes solutions of the algorithm where the position of the virtual leader did not satisfy the conditions in the definition of $IS(Th)$ for $Th = d_r/4$ (e.g. see Fig. 5.17). Contrariwise a too big value of the parameter n causes a long period of the robots' driving without any possibility to respond to the obstacles or to the breakage of the desired path. Problematic settings are mainly the values of n similar to values to N. In the case of $n = N$, the system is unstable even for a bigger value of N. This observation shows the usefulness of the RHC approach with two time horizons which is presented in this work.

Combining both, the table with computational time complexity and the table evaluating the quality of results, parameters $n = 2$ and $N = 6$ were chosen as the optimal setting. The higher value of N can improve the properties of the method only slightly, but at the cost of the considerable extension of computational demands[1]. Similarly, the lower value of n does not guarantee better results, but it can significantly reduce the threshold T_{max}.

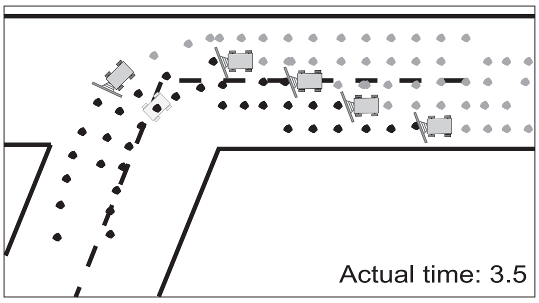
[1]Notice that the code used for obtaining the result presented in this section was not optimized with respect to computational efficiency and also to the environment Matlab, which is well suitable for research purposes, is not time efficient. Nevertheless the setting $n = 2, N = 6$ requires at the most 0.47s which is less than $T_{max} = n\Delta t = 0.5s$, in which the plan has to be provided.



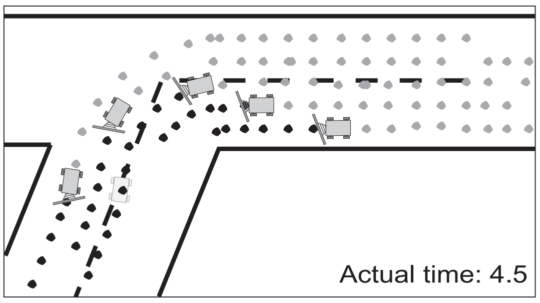
(a) First response to the path break.



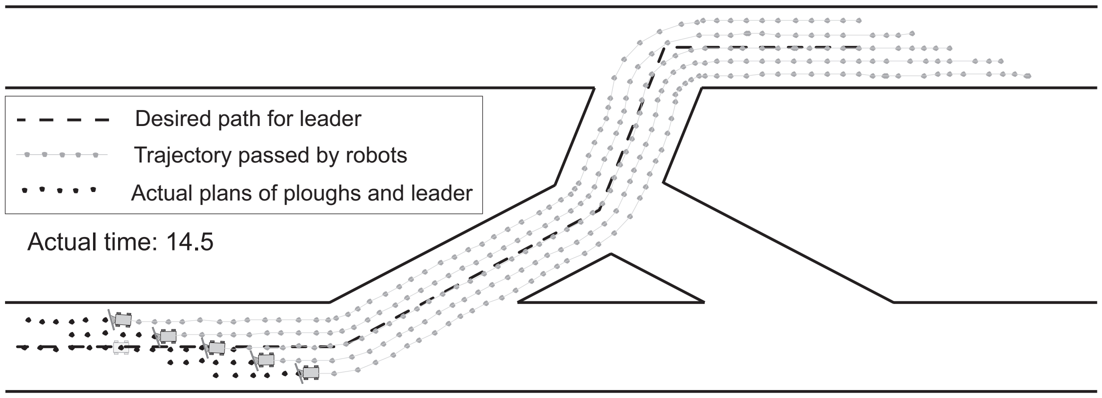
(b) Optimal plan for the passage through the path break.



(c) The virtual leader just returned to the desired path.

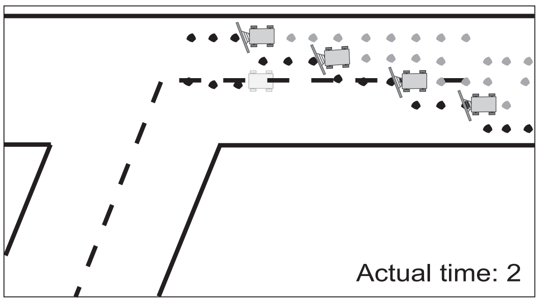


(d) The formation follows the turning of the virtual leader.

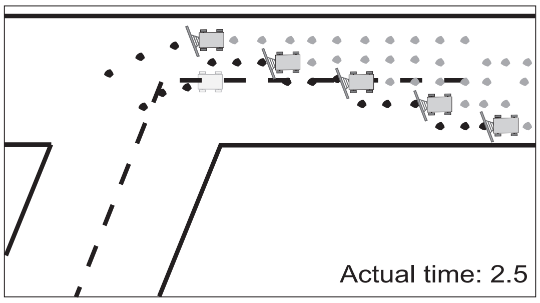

Desired path for leader
Trajectory passed by robots
Actual plans of ploughs and leader
Actual time: 14.5

(e) The robots after the mission accomplishment with depicted passed trajectories.

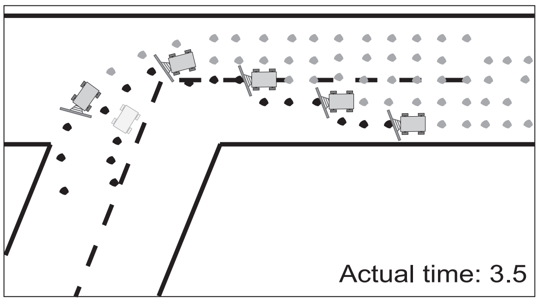
Figure 5.14: Formation driving with the setting of the algorithm: $N = 6$ and $n = 2$.



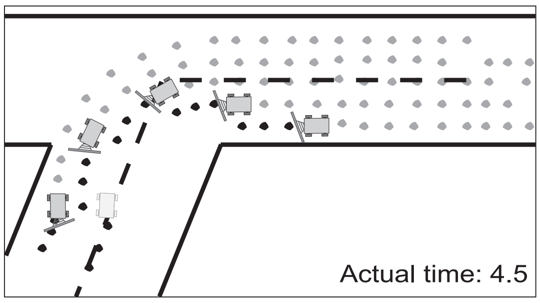
(a) The last optimization result before the path break.



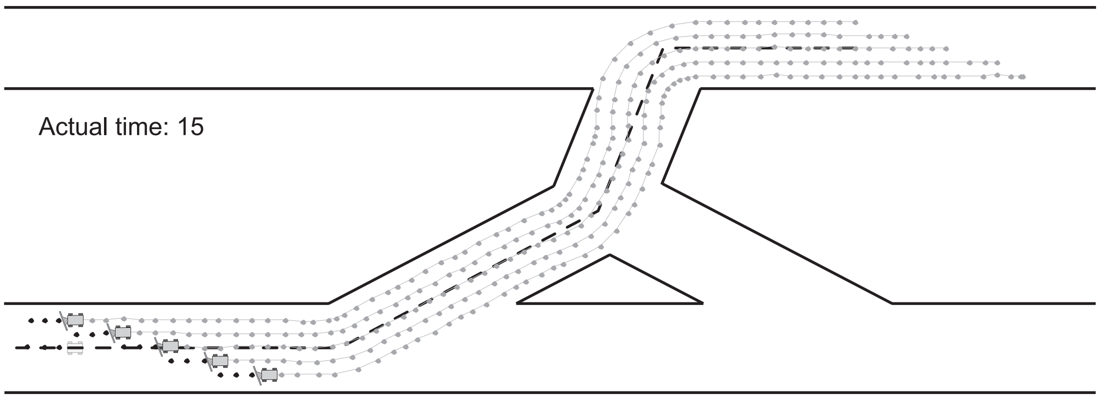
(b) Response to the path break.



(c) The ploughs turn with minimal turning radius.

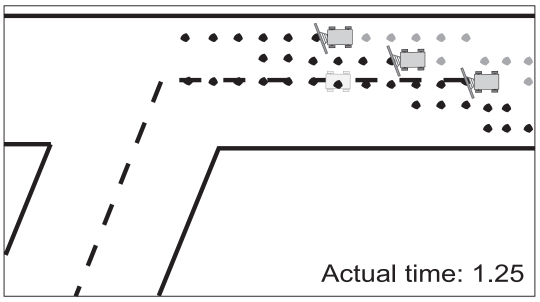


(d) The virtual leader just returned to the desired path.

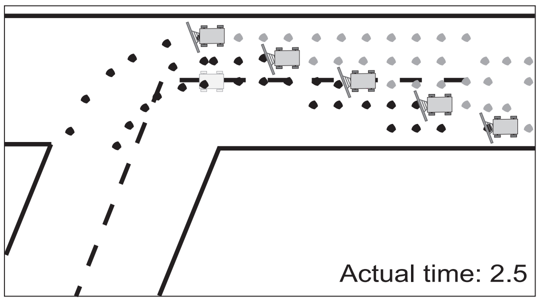


(e) The formation after the mission accomplishment with depicted passed trajectories of the ploughs.

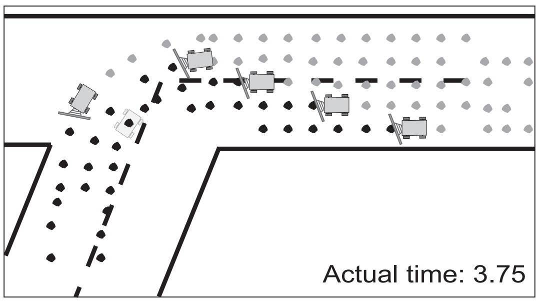
Figure 5.15: Formation driving with the setting of the algorithm: $N = 3$ and $n = 2$.



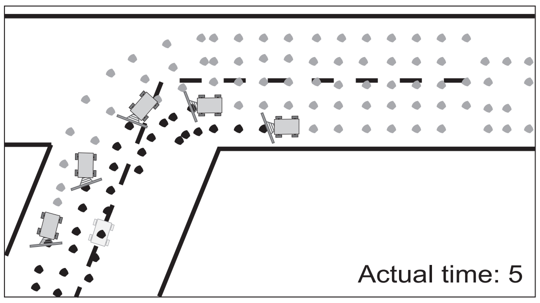
(a) The last optimization result before the path break.



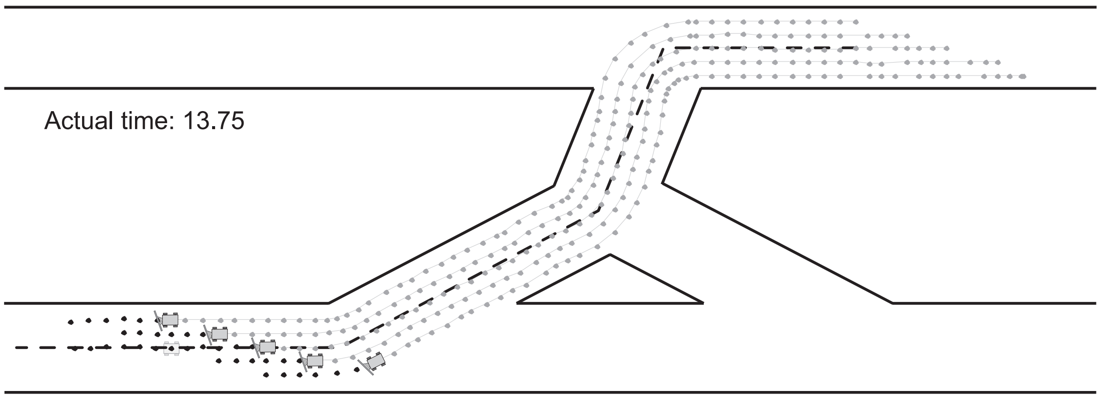
(b) The first response to the path break.



(c) The first plough turns deviated from the optimal way.



(d) The virtual leader just returned to the desired path.



(e) The robots after the mission accomplishment with depicted passed trajectories.

Figure 5.16: Formation driving with the setting of the algorithm: $N = 6$ and $n = 5$.

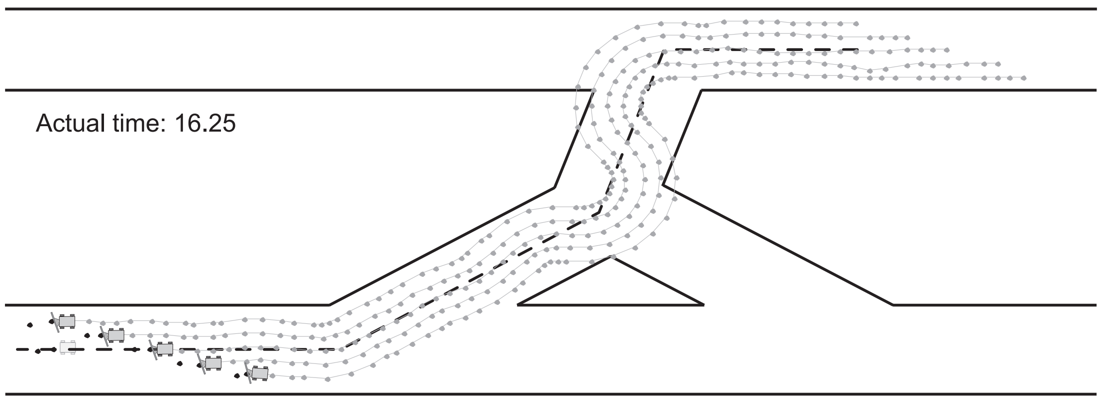
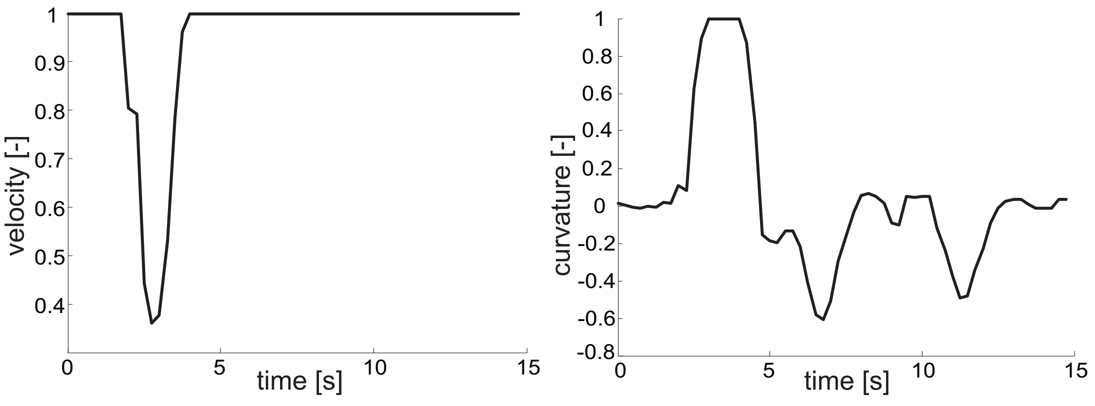
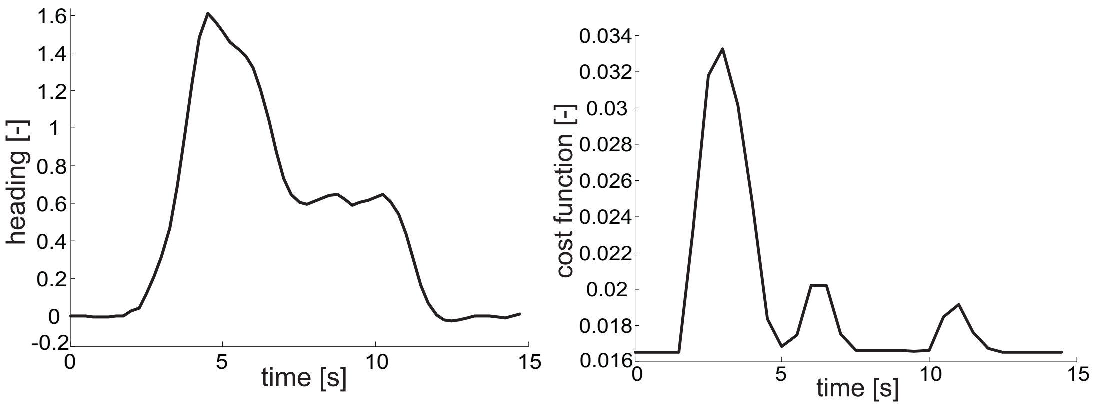


Figure 5.17: Formation driving with the setting of the algorithm: $N = 2$ and $n = 1$.

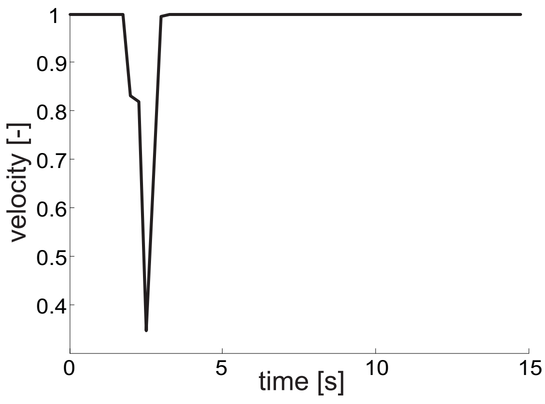


(a) (b)

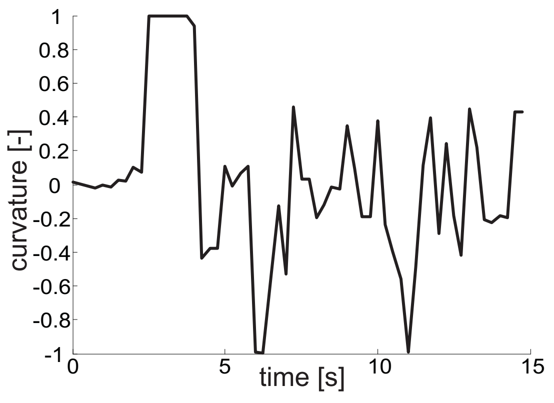


(c) (d)

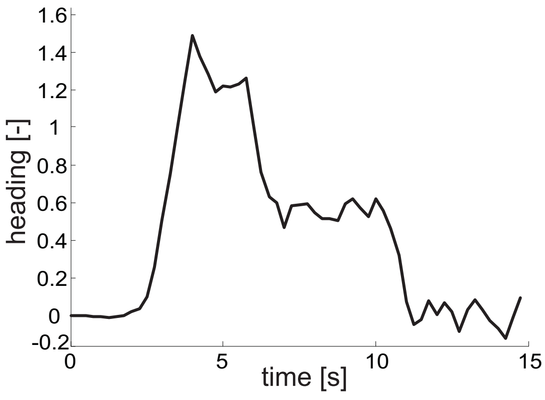
Figure 5.18: Formation movement characteristics of the virtual leader (velocity v_L, curvature K_L, heading $\theta_L(t)$ and values of best solution of cost function designed for the virtual leader) describing the simulation presented in Fig. 5.14 with setting $\alpha = \beta = 1$, $\gamma = 0.4$.



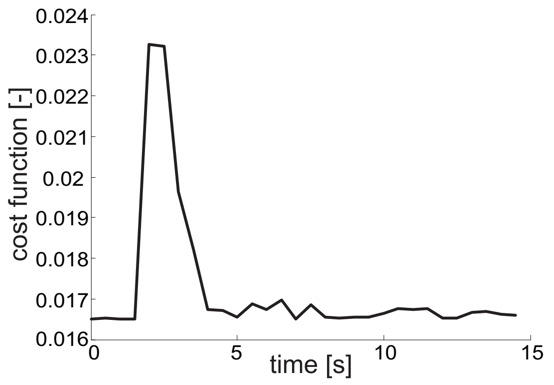
(a)



(b)



(c)



(d)

Figure 5.19: Formation movement characteristics ($v_L(t)$, $K_L(t)$, $\theta_L(t)$ and the values of cost function) of the simulation in the same environment as in Fig. 5.18, but with the setting of the algorithm: $\alpha = \beta = 1$, $\gamma = 0$

Snapshots of the simulation with the setting of parameters $n = 2$ and $N = 6$ are presented in Fig. 5.14. In all of the pictures, the black points denote an actual plan for the ploughs as well as for the virtual leader and the grey points denote states visited by the robots during the previous movement. The dash line represents the desired path that has to be followed by the virtual leader, which is drawn by a contour in front of the formation.

In the first snapshot of the formation driving in Fig. 5.14(a), the plan of the virtual leader is already slightly influenced by the second line segment of the path. Nevertheless, complete information about the expected change of heading is obtained in the next planning step in Fig. 5.14(b) where the formation is guided through a curve with minimal deviation from the desired path. In the next snapshot, the virtual leader reaches the second segment of the path (see Fig. 5.14(c)). The fourth picture shows small oscillations around the path, which has to be followed by the formation. These perturbations will be promptly suppressed and the position of the virtual leader will remain in the zone bordered by the distance $Th = d_r/4$ from the desired path. The trace of the complete movement can be seen in Fig. 5.14(e) where we depict the final snapshot of the simulation.

In Fig. 5.15, the simulation with setting of algorithm $n = 2$ and $N = 3$ is presented to demonstrate the effect of the insufficient long time interval N. The first snapshot in Fig. 5.15(a) was captured at the same time of the simulation as the snapshot from Fig. 5.14(a). Here, only the first line segment is contributing to the leader's cost function and the formation blindly continues to the line break. The first response to the approaching change of heading is enabled in Fig. 5.15(b). Nevertheless, it is too late to come through the curve optimally and so the deviation from the second path segment produces an uncleaned part of the runway (see Fig. 5.15(c)). Beyond this, the outer plough could get to close to the margin of the runway and due to the obstacle avoidance part of the equation (3.13) width of the cleaned track would be narrower.

The second problem, the interval n being too long, is clarified in Fig. 5.16 where the algorithm with $n = 5$ and $N = 6$ was utilized. In the first snapshot in Fig. 5.16(a), the formation is still too far from the line break to predict the approaching curve. In the next planning step in Fig. 5.16(b) the formation is already too close to the line break due to the long period where the robots just blindly executed preplanned control inputs. Therefore, the ploughs overshoot the desired path again (see Fig. 5.16(c)), but using the longer interval N the return to the path is smoother than in the case presented in Fig. 5.15.

The last example of the setting of the parameters, $n = 1$ and $N = 2$, presented in Fig. 5.17, is obviously wrong. Here, the method converges to a local optimization of the control inputs without a possibility to consider any situation in the near future. We should mention that the ability of the followers to avoid the borders of the runways was switched off in this simulation. This setting shows behavior of the virtual leader without any compensation from the followers.

Another important parameter, which can significantly affect the behavior of the algorithm, is the last part of the cost function presented in the equation (3.13) that penalizes aggressive control inputs. The importance of this penalization is hard to see from the results depicted in Fig. 5.14 - Fig. 5.17, but its influence changes the behavior of the robots dramatically. The significant parameters of the virtual leader's driving, curvature $K_L(t)$, heading $\theta_L(t)$, velocity $v_L(t)$ and the value of the cost function of the final solution in each planning step, are depicted in Fig. 5.18 - Fig. 5.20. The first set of curves in Fig. 5.18 matches with the simulation presented in Fig. 5.14. Here, the constants that adjusts the influence of the components in equation (3.13), are $\alpha = \beta = 1$ and $\gamma = 0.4$. We should remark that the algorithm is sensitive mainly to the proportion of the last component to the others. The values of velocity $v_L(t)$ in Fig. 5.18(a) remain on the desired maximum during the complete task except for the region around the first turn where the ploughs move slower. This deceleration could be reduced using a smoother desired path which however makes the approach more complicated mainly during the splitting and decoupling of the formations shown in Section 5.4.3. Since $v_L(t)$ is equal to maximal value most of the time the only active control input is curvature $K_L(t)$ (see Fig. 5.18(b)). It is obvious that the biggest absolute value of $K_L(t)$ is required when the ploughs move through the breaks of the desired path. The height of these peaks correlates with a difference in the heading of neighboring line segments.

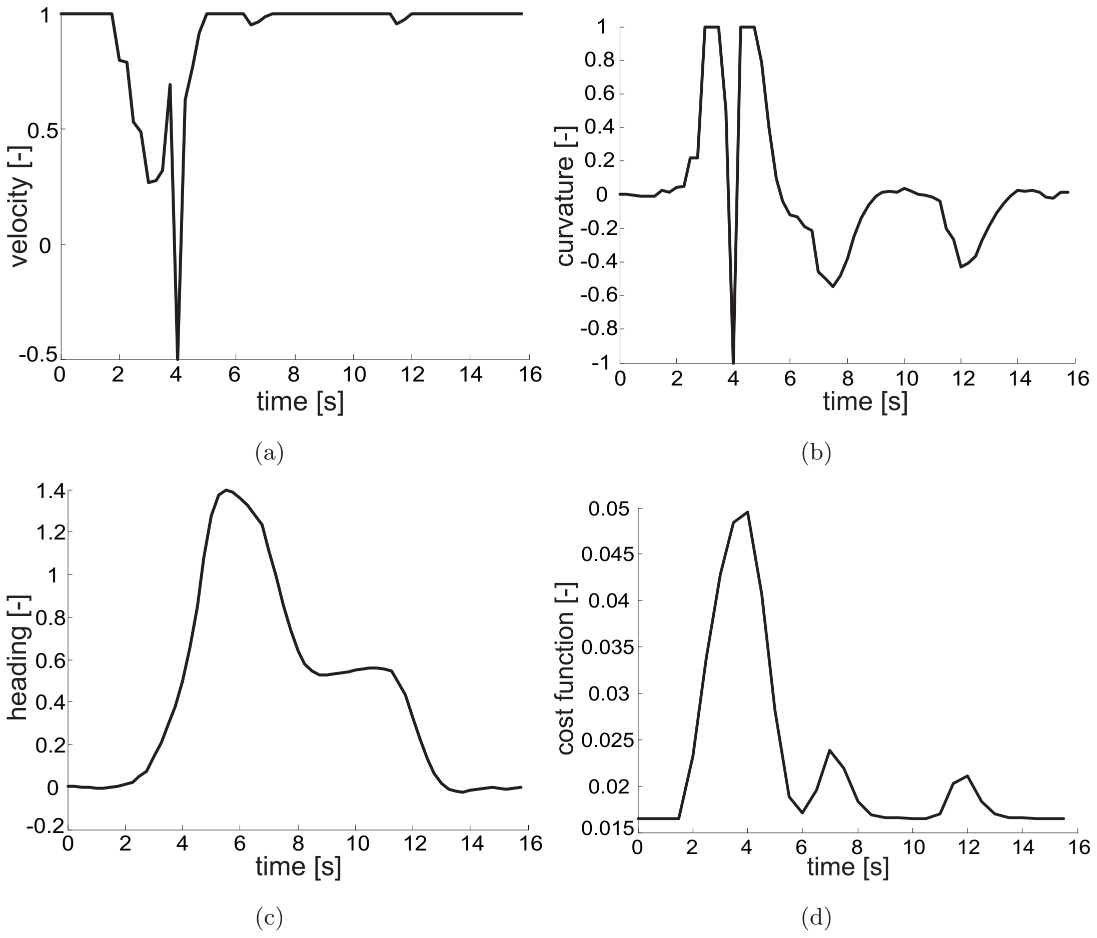


Figure 5.20: Formation movement characteristics ($v_L(t)$, $K_L(t)$, $\theta_L(t)$ and the values of cost function) of the simulation in the same environment as in Fig. 5.18, but with the setting of the algorithm: $\alpha = \beta = 1$, $\gamma = 1$

Beyond the peaks, we can see slight oscillations in the values. These oscillations were already described in Fig. 5.14 and can also be found in the graph of the heading $\theta_L(t)$ (see Fig. 5.18(c)). The last picture in Fig. 5.18(d) presents values of the cost function of the optimal solution. The value grows with the approaching of the virtual leader to the connection of line segments due to the necessary deviation from the desired path as well as due to the increase of the curvature as expected in the study of convergence.

To demonstrate usefulness of the curvature penalization, the same characteristics like in Fig. 5.18 are presented in Fig. 5.19 using the algorithm setting: $\alpha = \beta = 1$ and $\gamma = 0$. Although the algorithm is still stable as you can see from the values of the cost function in Fig. 5.19(d), the oscillations of the curvature and therefore also oscillations of the heading are inadmissible. Such behavior could damage the steering of ploughs and the movement could be dangerous on the slippery frostbitten runways.

The opposite problem of the curvature penalization is presented in Fig. 5.20, where the same task was solved with the algorithm setting $\alpha = \beta = \gamma = 1$. Such a bigger value of constant γ can provide results almost without oscillations and with very smooth

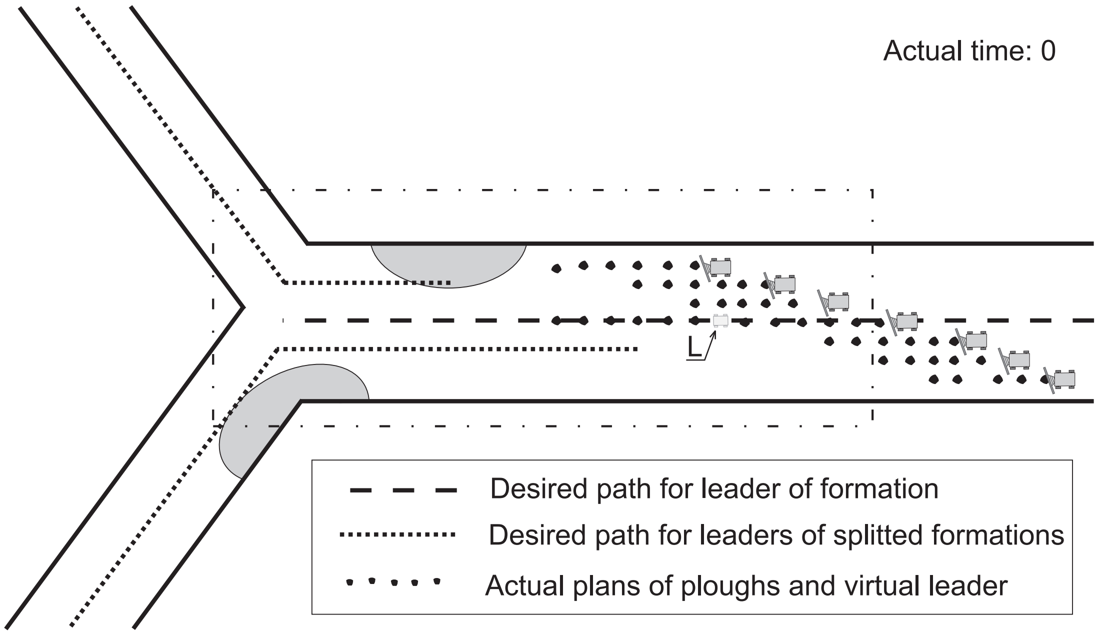

Actual time: 0
L
Desired path for leader of formation
Desired path for leaders of splitted formations
Actual plans of ploughs and virtual leader

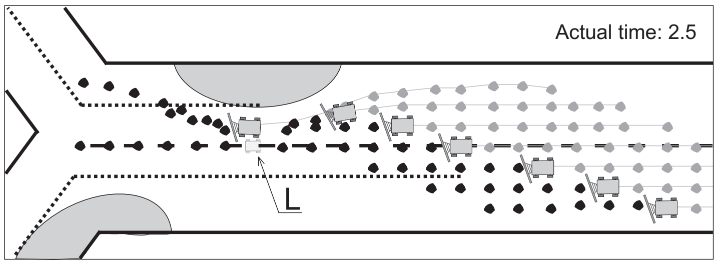
Figure 5.21: Big formation cleaning a runway with denoted paths necessary for splitting to two sub-formations.

changes of the heading (see Fig. 5.20(c)). Unfortunately the algorithm does not work satisfactorily during the sharper curves. In such cases the solution corresponding with the minimum of the cost function may not just be a simple turning, but it could be a plan with complicated maneuvers including stopping and reverse driving (see the negative values of velocity in Fig. 5.20(a)). Such a result cannot be optimal with respect to the total time needed for shoveling. Again the solution could be the utilization of a smoother desired path with problems mentioned above.

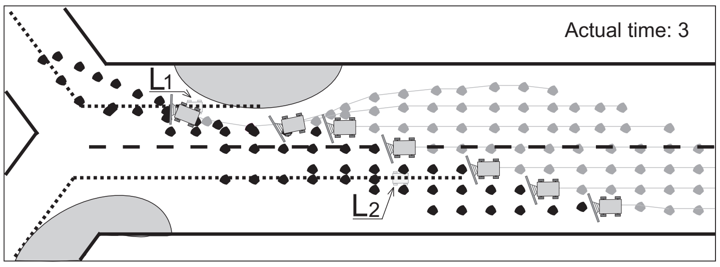
5.4.3 Splitting and merging

The approach for splitting and merging of the formations of snow-ploughs presented in this section is necessary for the completeness of the airport shoveling project. In this part we do not need to build a new theory, because the techniques and convergence studies developed in the previous chapters can be utilized without significant changes. However, important technical details that cannot be left out for the successful application will be presented here. Due to the strong application dependence, both skills, the formation splitting as well as the formation merging, will be explained using specific examples containing all problems that need to be solved during the snow-shoveling of airports.

The results presented in these chapters have been obtained using the formation driving algorithm introduced in Section 5.4.1 with the settings $N = 6$, $n = 2$, $\alpha = \beta = 1$, $\gamma = 0.4$ and $\Delta t = 0.25s$. Therefore, the time difference between two subsequent planning steps in the simulations splitting and merging is 0.5s.

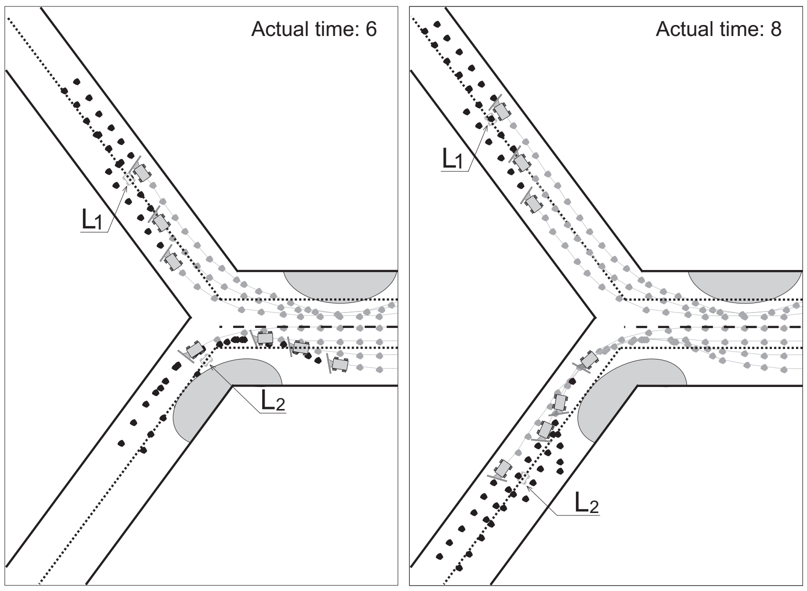


(a) Formation closely before the splitting.



(b) Formations that have just been split.

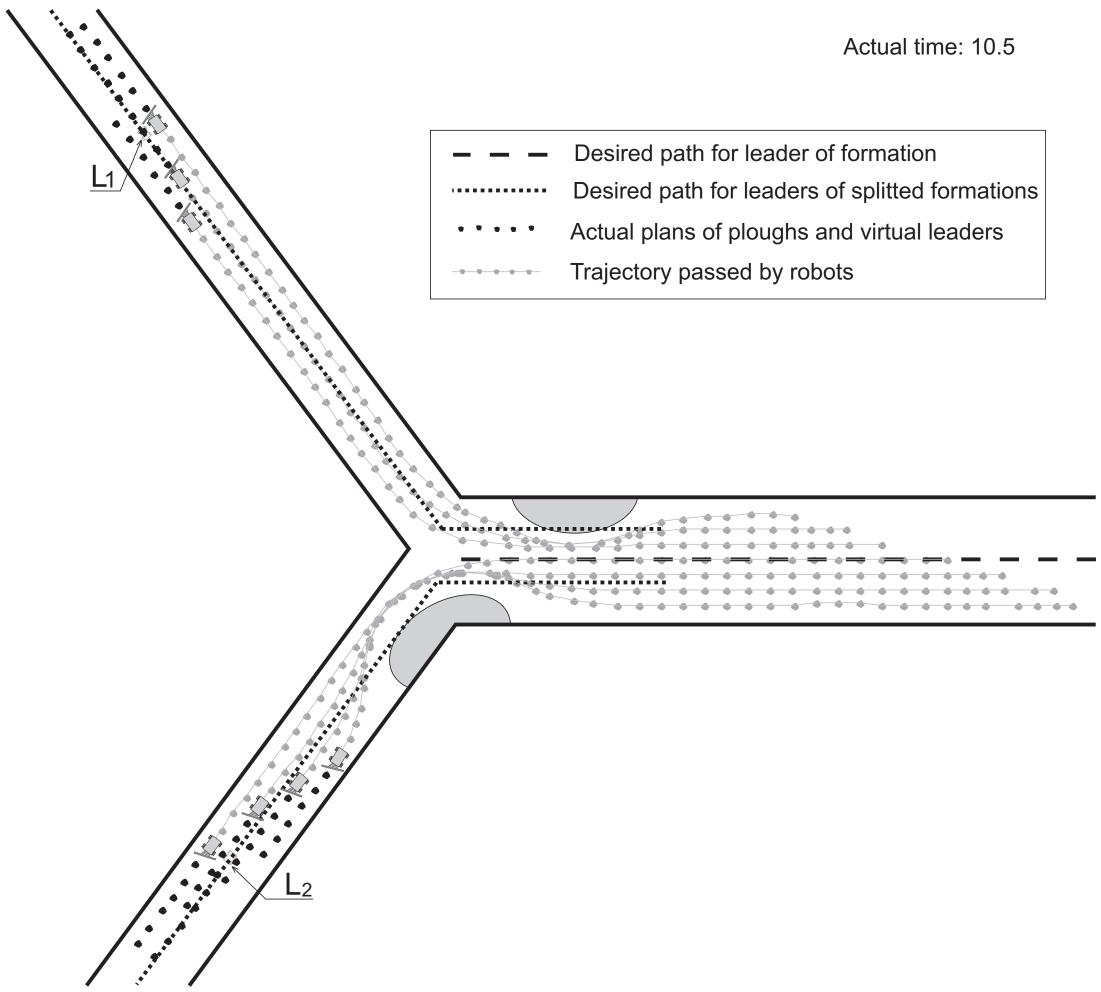
Figure 5.22: Splitting of the formation during an obstacle avoidance.



(a) Formation led by L_2 is avoiding an obstacle partly blocking the cleaned road.

(b) The formation led by L_2 is going back to the desired shape.

Figure 5.23: Obstacle avoidance maneuver immediately after the formation splitting.



Actual time: 10.5
Desired path for leader of formation
Desired path for leaders of splitted formations
Actual plans of ploughs and virtual leaders
Trajectory passed by robots
L1
L2

Figure 5.24: Complete history of the formation movement during the simultaneous splitting and obstacle avoidance.

5.4.3.1 Splitting

The idea of the formation splitting mechanism presented in this chapter is using the concept of formation driving relatively simply. The key problem of the splitting approach is to find an appropriate time when to transfer the leading tasks from a single virtual leader to several virtual leaders that belong to the lately formed formations. The simplest solution could be to consider the initial big formation as several subformations with assigned leaders from the very beginning of the shoveling. Unfortunately, the forming of different coalitions during the complete airport cleaning would degrade the method to a single robot driving. This approach decreases the robustness of the method as well as it increases the complexity of planning and communication. The ploughs connected to a team could avoid the collision better with the rest of the formation and they can avoid obstacles in a more effective way. All these facts lead us to postpone the splitting point to as late as possible.

Let us now analyze the problem from the opposite view. The desired path, which is followed by the virtual leader of the big formation, will differ from those used by the new leaders as you can see from Fig. 5.21. In addition, we can suppose that during the splitting the new formations have to change their heading for cleaning the following road. To take advantage of the planning horizon, the new desired path should be utilized when the new direction of movement can influence the optimization process. For simplification we can say that the formation should be divided under the commands of new virtual leaders in the distance of the former virtual leader from the center of the crossroad greater than or equal to the length of the planning horizon. Due to the prior knowledge of the maximal leader's speed we can consider an upper bound l_{spl} of the length of the planning horizon as

$$l_{spl} = N\Delta t \max_{\tau\in\langle t;t+N\Delta t\rangle}(v_{max,L}(\tau)). \tag{5.10}$$

The formation will be splitted in the distance from the center of the crossroad equal to l_{spl} to satisfy both requirements mentioned above: i) to keep the robots in the big formation as long as possible and ii) to switch a desired path sufficiently far from the crossroad. Perceive that both, the path for the big formation as well as the new desired paths, must be overlapping to employ the RHC concept. Before the switch-point, the previous path contributes to the cost function of the virtual leader. This virtual point is situated on the axis of the formation and in the way that $p_1(\cdot)$ coordinate of the first plough in the sub-formation is zero. Once the formation reaches the switch-point, the new virtual leaders are placed on the axes of established formations again next to the first ploughs. In case the first plough of the new formation is the same as the first plough in the previous formation the overlapping part will be equal to l_{spl}. For the formations created from ploughs moving at the back of the previous formation the overlapping can be even bigger, because the new path must begin at the position of the new virtual leader at the time of splitting.

An example of such desired paths is depicted in Fig. 5.21 where a formation splitting is presented. At the beginning of the simulation, seven ploughs form one formation for shoveling a wider runway. The formation is led by the virtual leader denoted L and has to be divided into two smaller formations at the end of the runway. Two neckings of the runway were added to the map to demonstrate an robustness of the algorithm in the environment with obstacles. In our case the obstacles have been known before the formation driving but similar results could be obtained using obstacles detected by ploughs how it will be shown in the simulation of formations decoupling. The first obstacle is situated close to the splitting point which should show an ability to optimally split the formation even if the ploughs are not in the desired positions. The formation shortly before the splitting is depicted in Fig. 5.22(a). The outer ploughs are deviated from their desired positions to avoid the obstacle while the rest is following the virtual leader L. The following snapshot in Fig. 5.22(b) presents the situation where the robots are already led by two independent leaders L_1 and L_2. The switching of the leadership has been realized just before this planning step. During the nonzero interval needed for the switching of the communication topology, the ploughs can use the old plans obtained by the former virtual leader and their movement is continuous. Once, the ploughs get the sequence of states used for their planning from the new leader, the coordinates $p_i(t)$ and $q_i(t)$, where $i \in \{1, \dots, n_r\}$, must be adapted for the new formations.[1]

The splitting maneuver continues in Fig. 5.23(a) where the ploughs led by the virtual leader L_1 are already maintained in the desired position while the second formation is passing through the narrow corridor between the obstacle and the border of runway. The snapshot of this simulation depicted in Fig. 5.23(b) presents the plans of the robots in the second formation to reach the desired shape appropriate for snow shovelling. The final snapshot with a delineated complete history of the ploughs movements is depicted in Fig. 5.24.

5.4.3.2 Merging

The method for merging several sub-formation to one big formation guided by one virtual leader will be introduced in the following paragraph. The core idea of the approach is to iteratively divide the merging process to the simple jointing of two formations as illustrated in the following simulation of three formation merging. In Fig. 5.25, the formations F_1, F_2 and F_3 guided by the virtual leaders L_1, L_2 and L_3 are approaching to the point of merging in the middle of the crossroad. In addition, there are depicted paths that will be followed by the subsequently formed groups. The formations should pass through the crossroad in an order in which they will be maintained in the big formation. This procedure should minimize snow remaining on the runway as well as decrease the collision risk. Due to the shovels' orientation, the ploughs must be arranged in sequence F_1, F_2, F_3 from the front to the back of the composed shoveling column. Based on this, the formations F_2 and F_3 have to stop and wait for the formation F_1. In the first snapshot of the simulation in Fig. 5.25, the formation F_2 brakes from the initial velocity to reach the appropriate waiting position which should happen before getting to the crossroad.

The snapshots presented in Fig. 5.26 could help to clarify the merging process as well as the rules for creating the desired paths. In the first picture in Fig. 5.26(a), the formation F_3 also breaks to join the group as the last one. Before the suspension in the waiting position, the formation F_2 detected obstacles which caused deviation of the ploughs from the desired shape. In the following snapshot in Fig. 5.26(b), the formation led by L_3 reached the waiting position while the formation led by L_2 has accelerated to continuously merge with the formation F_1. It could be difficult to estimate the optimal beginning of the acceleration in a real experiment. Fortunately, the inaccuracy in the spacing between the sub-formations can be suppressed using the periodical replanning and obstacle avoidance during the movement.

Similarly to the paths that should be followed by the virtual leaders during the formation splitting also in this case the path for the virtual leaders of the small formations and for the virtual leader of the merged formation have to be overlapping. As you can see in Fig. 5.26(c), the position of the first plough in the newly merged formation F_{12} remains the same as in the previous formation F_1. Therefore, the position of the new

[1]The coordinates of the ploughs for all formations are simply computed using the width of the runway, the number of ploughs in the formation, the coverage of their shovels and the safety spacing in p-direction.

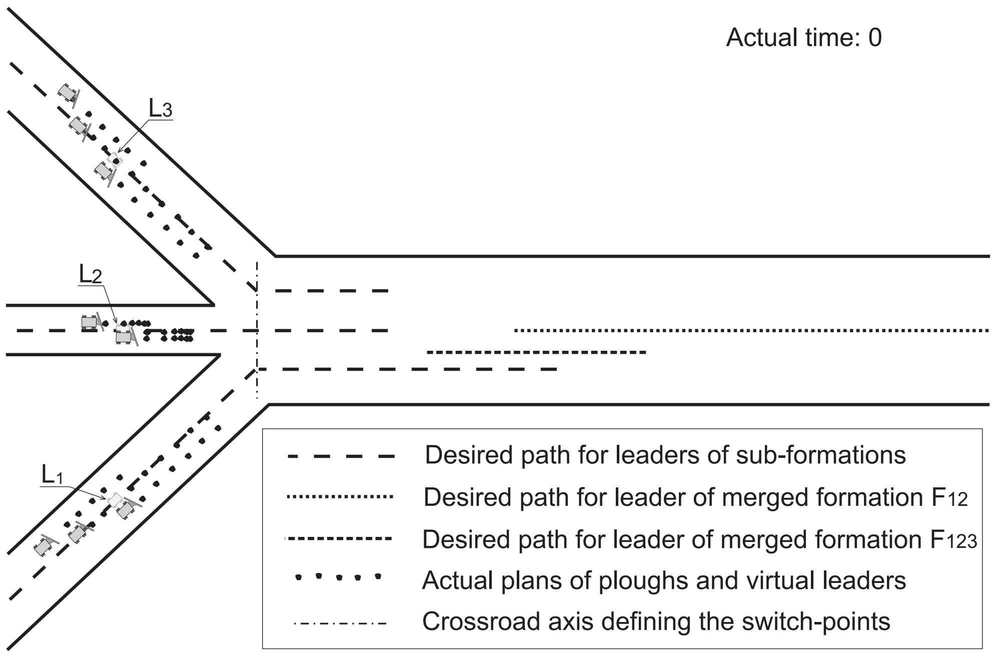
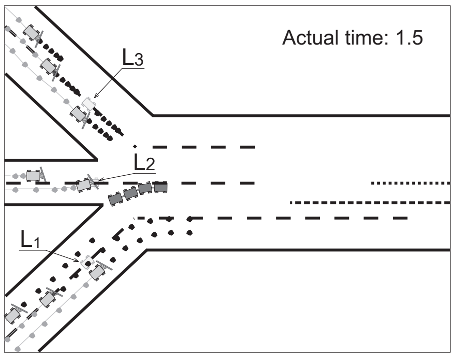

Actual time: 0
L3
L2
L1
Desired path for leaders of sub-formations
Desired path for leader of merged formation F12
Desired path for leader of merged formation F123
Actual plans of ploughs and virtual leaders
Crossroad axis defining the switch-points

Figure 5.25: Three small formations shoveling auxiliary roads with depicted paths needed for merging to a big formation optimal for sweeping the main runway.

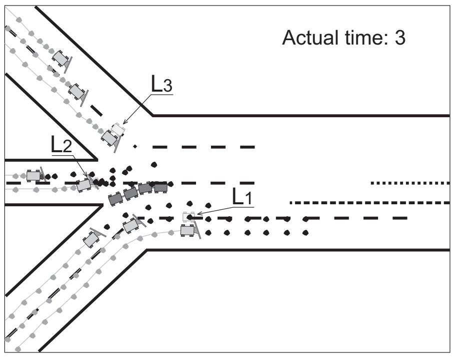
virtual leader L_{12} stays unchanged with respect to the direction of the movement. The new position is only shifted from the former position of the leader L_1 in the perpendicular direction to the desired path. Physically, the planning module of the virtual leader denoted L_{12} can be placed on board of the same plough as before. The formations, similarly as for the splitting, are merged when the position of the second virtual leader L_2 is behind the crossroad.[1] In this area, the virtual leaders of the formations that have to be merged are already behind the break of the desired line and their heading is parallel. All ploughs from that point can follow the same path that will be placed in the axis of the new formation. Beside this, the idea of a time receding horizon must be considered again in definition of the length of the paths utilized by the old formations as well as by the merged one. The path followed by the formation F_2 has to overlap the switch-point with length of l_{spl} (defined in equation (5.10)), because the path is contributing to the cost function used for planning until the leader L_2 reaches the switch-point. To determine the overlapping of the path followed by F_1, we have to find out the position of the virtual leader L_1 at the time of switching. For simplification let us suppose that the merging of the formation will be done exactly when the leader L_2 reaches the switch-point.[2] Then the distance of the leader L_1 from the perpendicular dot-dash line in the crossroad has to be $p_{n_r^{F_1}} + \Delta p$. The number $n_r^{F_1}$ is the index of

[1]In Fig. 5.25 this point matches the crossing of the dash desired path and the perpendicular dot-dash line.

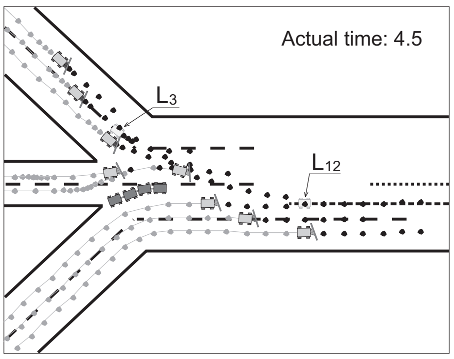
[2]Because of the nonzero interval $n\Delta t$ the physical switch-over will be executed behind that point, but due to the necessary synchronization the optimal spacing between the formation should already be kept.



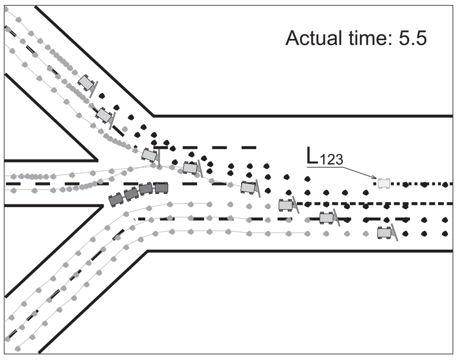
(a) Formations modify their velocities with a goal to continuous connection. The formation led by L_2 is avoiding the detected obstacle.



(b) Formation F_2 accelerates with an aim to join the formation F_1. Formation F_3 waits for the formations going by.



(c) Formations F_1 and F_2 are already merged to formation F_{12} and formation F_3 accelerates to join them.



(d) All formations are just merged to one big formation $F_{1,2,3}$.

Figure 5.26: Zoomed snapshots of the simulation explaining the merging process.

the last robot in formation F_1 and Δp is the recommended spacing between two neighboring ploughs. The desired line is extended again to satisfy the requirements of RHC and so the total length of the path behind the crossroad has to be $p_{n_r^{F_1}} + \Delta p + l_{spl}$. The switch-over to the merged formation arrangement is done if the leader L_1 (but also the new leader L_{12}) is in a distance from the crossroad that is equal to $p_{n_r^{F_1}} + \Delta p$. Therefore, the new desired path should start in this distance.

The first plan of the ploughs after the formations connection can also be found in Fig. 5.26(c). The last two robots of formation L_{12} are still avoiding the obstacle, but the formation will be merged continuously using the periodical replanning even without the slacking up of the first group. You can observe that the formation F_3 is accelerating to join with the new formation F_{12} in the same time in which the formations F_1 and F_2 are in the process of merging into the formation F_{12}. The complete formation consisting of 8 ploughs is firstly employed in the snapshot shown in Fig. 5.26(d). From that time, the ploughs are guided by one shared virtual leader and they are following the same path. One can observe that the final formation was created even before the

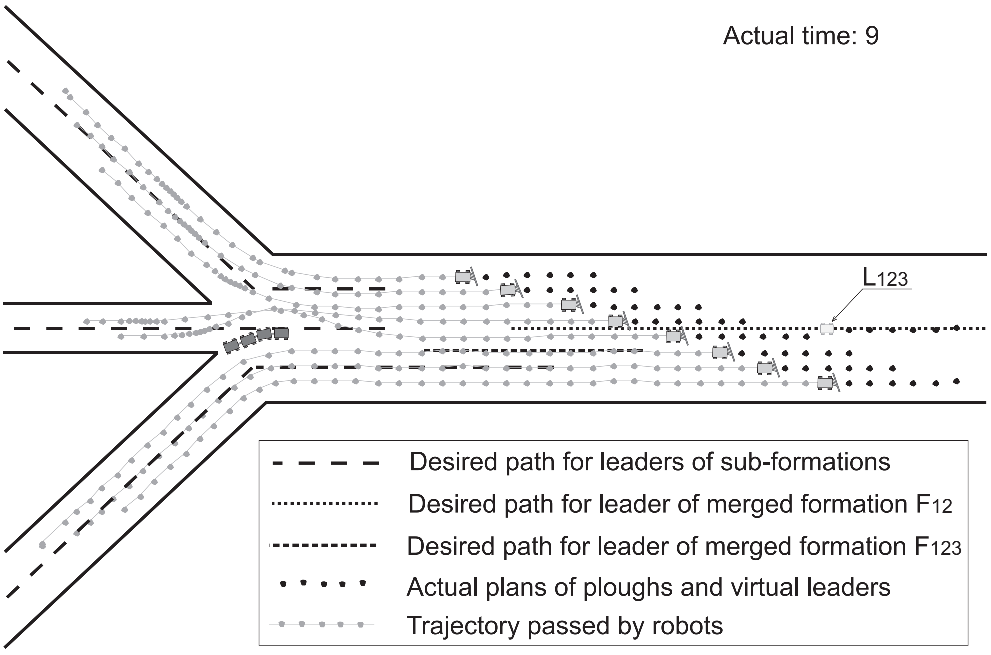

Actual time: 9
L123
Desired path for leaders of sub-formations
Desired path for leader of merged formation F12
Desired path for leader of merged formation F123
Actual plans of ploughs and virtual leaders
Trajectory passed by robots

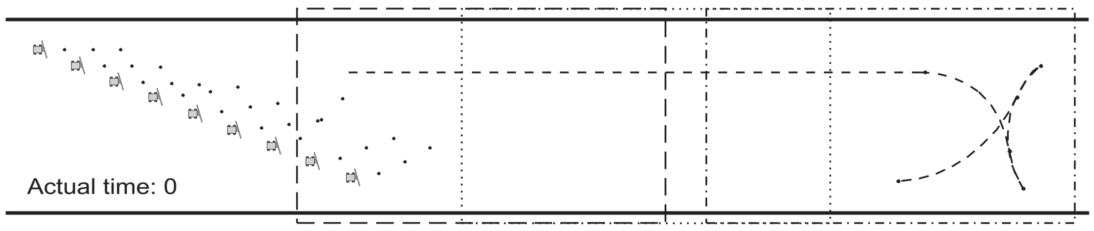
Figure 5.27: The complete history of the formations movement during the merging.

last ploughs turned to the big runway. This is possible using the time delay between the ploughs following the virtual leader's trajectory and the virtual leader itself. The continuous driving of the ploughs, originally from the formation F_1, is then obtained by a sequential following of the plans of the leaders L_1, L_{12} and L_{123}. This seemingly too complicated structure of switching between several plans enables a much smoother connection of the robots than with the application of independent plans. Our method also provides a general approach independent from the number of merging formations. The last snapshot of the simulation presented in Fig. 5.27 shows the merged formation of ploughs that can be utilized for the shoveling of a big runway.

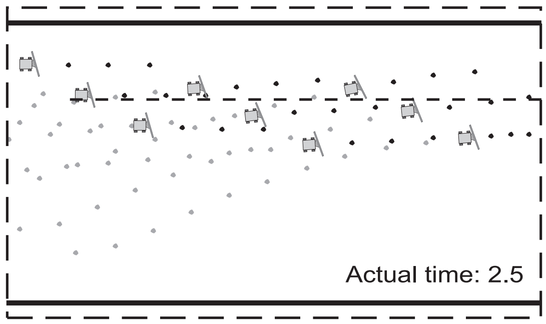
5.4.4 Formation turning

The remaining skill of the formation driving necessary for the fulfilment of an arbitrary desired plan of the ploughs is the formation turning. The robots should be able to turn on the spot in case of a blind runway or a road that is blocked by obstacles detected during the task accomplishing. Furthermore, such a U-turn can be part of the ordinary optimal schedule designed by the task allocation module.

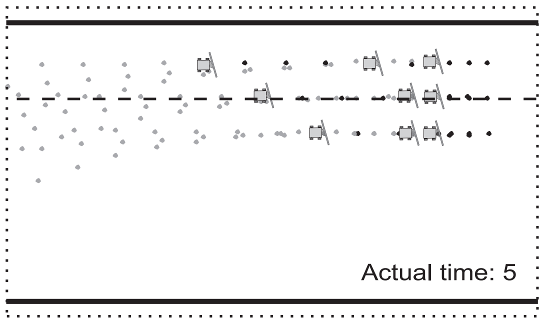
The maneuver suitable for this purpose can be derived from the general concept presented in the Section 2.5 as following: once the U-turn is required by the *Task Allocation* module, the optimization problem $\mathcal{P}_2(\cdot)$, which is solved during the snow shoveling in each sampling time $n\Delta t$, is replaced by the problem $\mathcal{P}(\cdot)$ extended with the concept of two alternating virtual leaders. Unfortunately, the cleaning formation is too widespread in the directions of p as well as q and the U-turn of the formation keeping the shoveling shape unchanged is impossible in the area bordered by the road-sides. An obvious solution is to relocate ploughs from the actual formation to a more



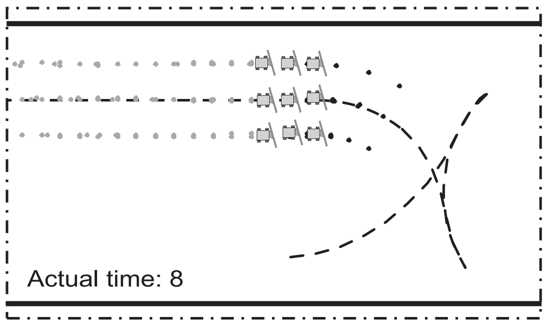
(a) The initial formation optimal for the snow sweeping.



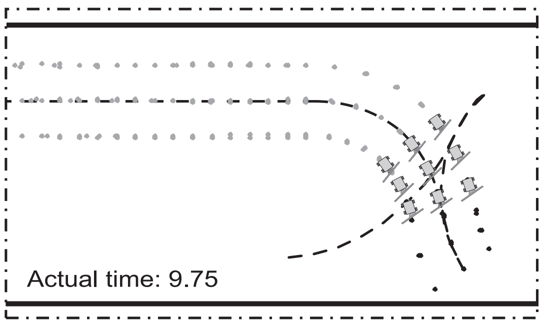
(b) The transformation between the sweeping and the turning formations.



(c) The turning formation building.



(d) The compact shape appropriate for turning.

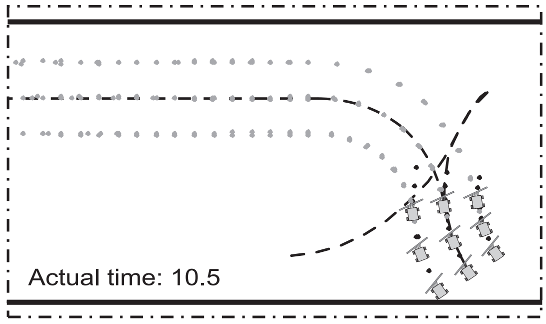


(e) The formation in the turning maneuver.

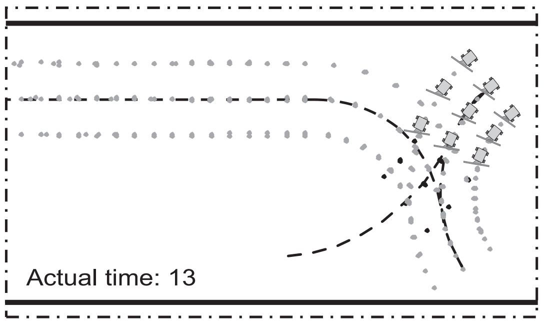
Figure 5.28: The U-turn of snowploughs at the end of runway. Part I.

compact shape for the turning. Once the U-turn is accomplished, the robots return to the previous positions suitable for the shoveling.

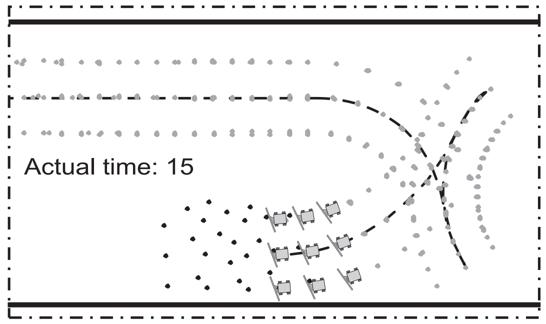
In the first snapshot of the simulation presented in Fig. 5.28(a), we can see the beginning of such a turning maneuver with 9 ploughs shoveling a runway of the airport. The complete plan of the turning designed for the virtual leader in one optimization step consists of the three parts that were introduced above: i) the transformation to the more compact shape, ii) an optimal U-turn, iii) the transformation back to the shoveling formation. In part i), the position of the virtual leader is shifted to the upper part of the runway which is optimal for the turning in part ii) and the variables which describe the position within the formation are changed from the initial values optimal for snow shoveling $p_j^{i)}(\cdot)$, $q_j^{i)}(\cdot)$ to the values appropriate for turning $p_j^{ii)}(\cdot)$, $q_j^{ii)}(\cdot)$, where $j \in \{1, \ldots, n_r\}$. The movement of the followers is planned autonomously,



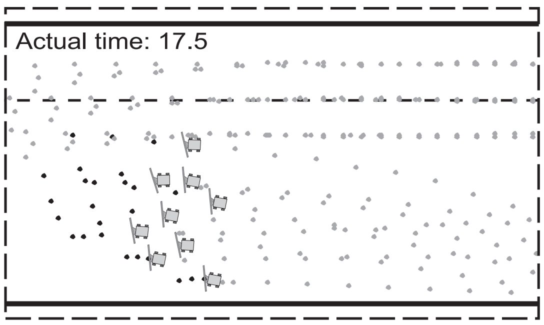
(a) Time of taking command by the second virtual leader.



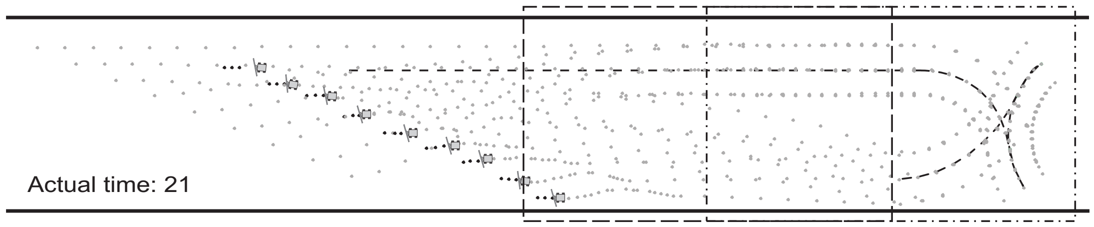
(b) Time of switching the leadership back to the first virtual leader.



(c) The U-turn accomplished. The denoted plans lead the robots back to the sweeping formation.



(d) The transformation between the turning and the sweeping formations.



(e) The ploughs converge back to the previous formation.

Figure 5.29: The U-turn of snowploughs at the end of runway. Part II.

only the values $p_j^{ii)}(\cdot)$, $q_j^{ii)}(\cdot)$ are designed as an input of the turning maneuver.[1]

The switching between the two shapes of the formation is shown in the snapshots in Fig. 5.28(b) and Fig. 5.28(c). The desired compact formation, which is more appropriate for the turning, is reached in the snapshot in Fig. 5.28(d). The accomplishing of part ii), the turning, is similar to the simulation presented in Fig. 2.19 and it is depicted by snapshots in Fig. 5.28(e), Fig. 5.29(a) and Fig. 5.29(b). The specific control inputs for the forward and backward movement during the turning are also obtained here using the two virtual leaders approach presented in Section 2.5. The turning maneuver is finished in Fig. 5.29(c) where the formation is oriented backwards and the ploughs can return to the previous positions. A snapshot demonstrating this last part of the

[1]In practical applications, the same predefined values $p_j^{ii)}(\cdot)$, $q_j^{ii)}(\cdot)$ can be used for all U-turn maneuvers and therefore the system can still be fully autonomous.

complete turning simulation is presented in Fig. 5.29(d).

A quite large part of the runway stays uncleaned using such an approach as you can see from the Fig. 5.29(e) where the complete history of the robots movement is denoted. Furthermore, during the backward movement the ploughs must go over the snowy surface of the runway which could be dangerous. A simple solution of this problem is to continue in the cleaning formation to the end of the runway and then to move back to the sufficient distance needed for the transformation of the formation. In addition, the ploughs can already change their positions to a compact formation during the backward movement which reduces the total turning time.

5.5 Hardware experiment

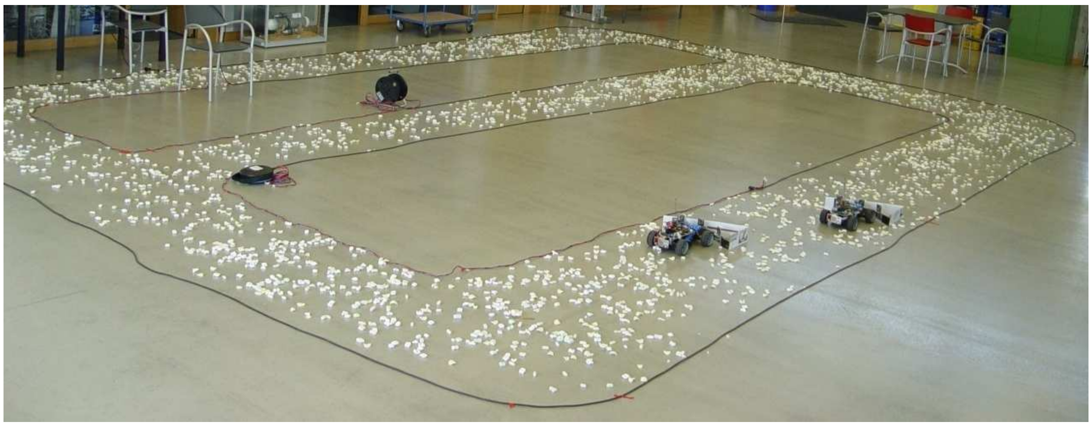
The hardware experiment presented in this section was performed to verify the functionality of the complete airport snow shoveling system that was developed by the subgroup of the PhD program "Identification, Optimization and Control with Applications in Modern Technologies" allocated at University of Wuerzburg. The experiment was set up for the utilization of indoor MERLIN-Testbed [165] that was developed within the university. The MERLIN car-like robots are equipped with sensors for obstacle detection (ultrasonic range finders) and also sensors for position determination (wheel encoders, gyroscope), which are crucial in the presented task. Furthermore, a wireless communication enables robots to receive commands as well as to inform the other vehicles about their actual state which is important mainly for the formation stabilization and the groups coordination. The possible communication between the robots and an internal PC can simulate the commanding of vehicles from an airport control tower via the *Task Allocation* module.

The chassis of the MERLINs has been slightly modified by adding a simple shovel in front of the "ploughs" for the experiment. Due to technical limitation the orientation of the shovels has been fixed and cannot be turned to sweep the snow into the other direction which must be considered in the planned schedule. Such equipped rovers have a shoveling coverage of approximately $0.55m$ and the length $0.95m$ that results in formations with distances between the centers of the robots $0.4m$ in the q coordinate and $1.5m$ in the p coordinate.

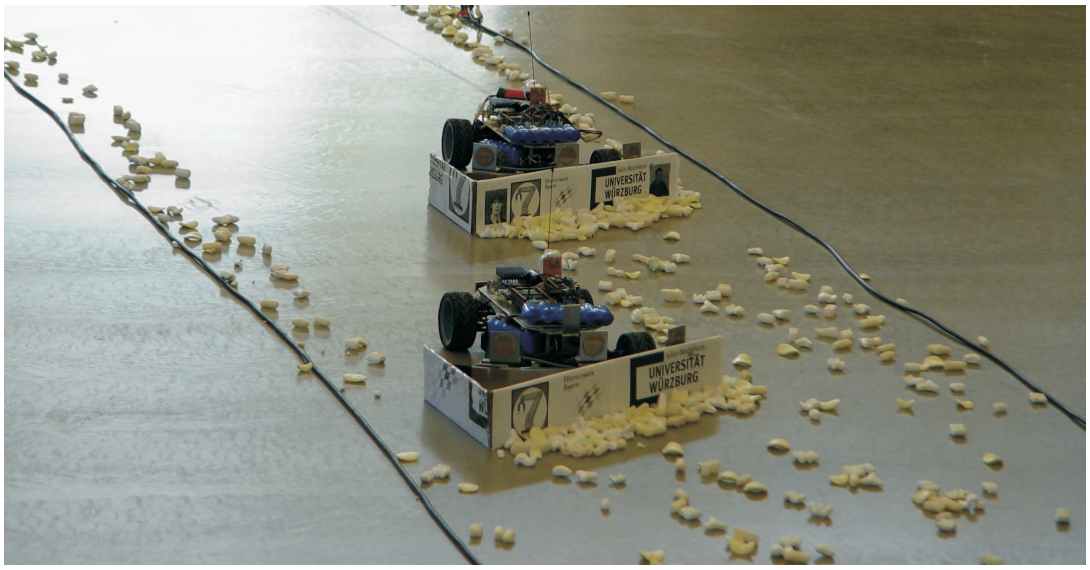
The experiment scenario has been adapted for indoor application replacing the real snow by small pieces of polystyrene. The size of runways have been decreased to fit to the robotic hall of University Wuerzburg. The complete scenario consists of one big runway that has to be cleaned by a formation of two robots and two auxiliary roads that require shoveling by single ploughs (the "indoor" airport can be seen in Fig. 5.30(a)).

An optimal schedule for such a simple scenario can be designed by the *Task Allocation* module in one step of the tree exploration, but due to the splitting and decoupling parts included in the obvious solution the entire snow shoveling system needs to be employed. A utilization of a more complex environment has been impossible due to the space limitation of the hall but also because of employed position determination of the ploughs which enables only short experiments. In our experiment the pose of the robots during movements has been obtained by the fusion of data from wheel encoders and a gyroscope. In real outdoor applications, an external global position system is necessary, because of the error accumulation in the applied ”dead reckoning”. An example of such an approach solving the problem of wheel skidding and slipping by GPS-based tracking is presented in [116].

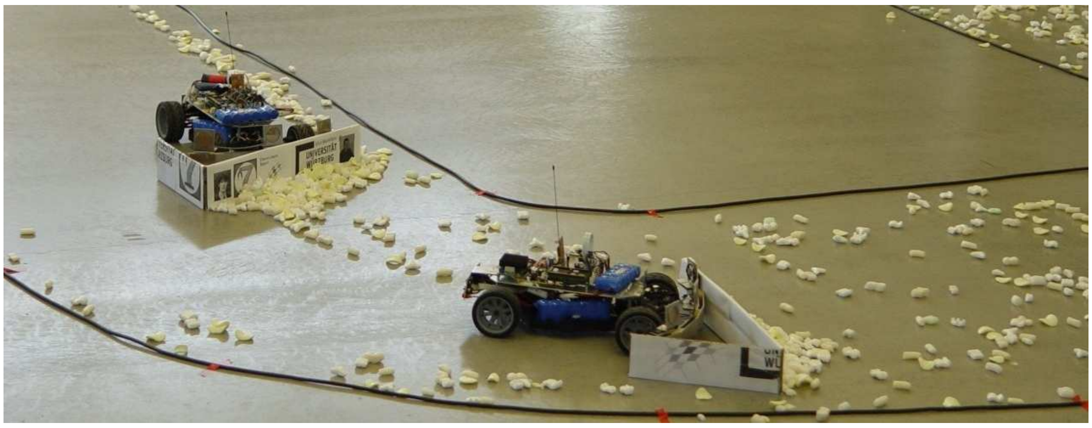
On the first picture of the hardware experiment in Fig. 5.30(a), two ploughs are waiting in the initial position for commands from the task allocation level. The next picture in Fig. 5.30(b) shows the robots in the formation pushing the ”snow” into the side following shovels' orientation. The beginning of the splitting maneuver is pictured in Fig. 5.30(c) where the wide runway verges into a narrower road. From this point the vehicles are considered as independent unites and they clean the auxiliary roads separately. The formation has to be built-up again to finished the main runway cleaning and to be prepare for the next run of shoveling. The details of the merging process have been explained in Section 5.4.3.2. In our experiment, the merging is shown in Fig. 5.31(a), where the second rover of the big formation already finished its task and is waiting for the first robot, which is cleaning the longer road. In the next snapshot in Fig. 5.31(b), the first robot is passing by the waiting robot, which is accelerating to complete the formation. Finally, the robots at the end of the first cleaning cycle are shown in Fig. 5.31(c).



(a) Initial position of the ploughs.

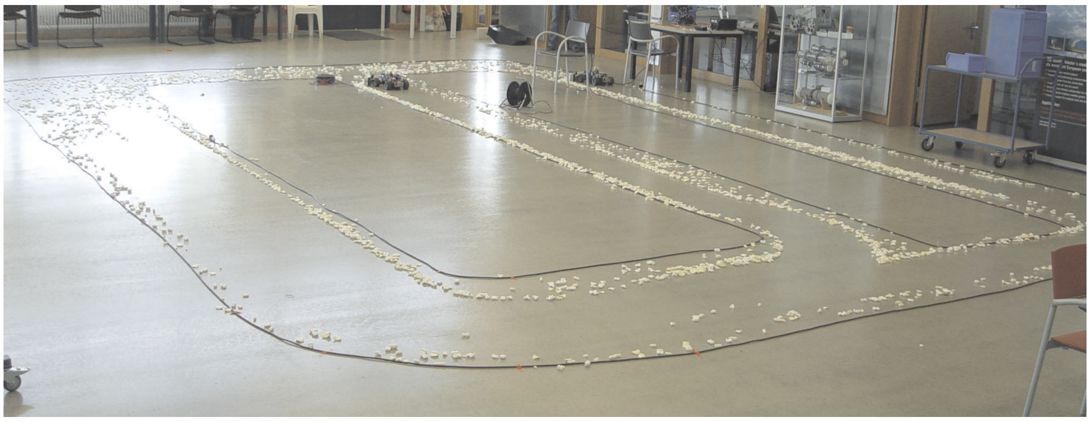


(b) The robots cooperatively shoveling the main runway.

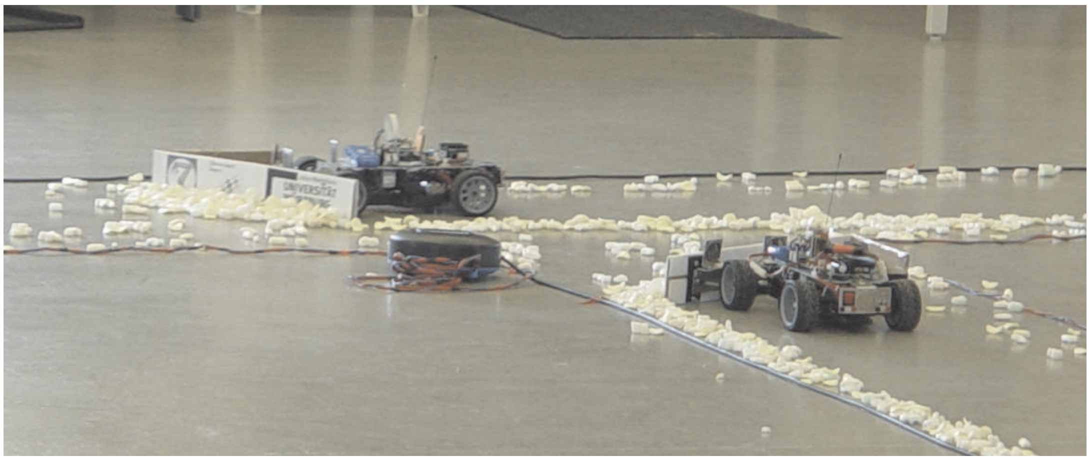


(c) The formation splitting to two independent units.

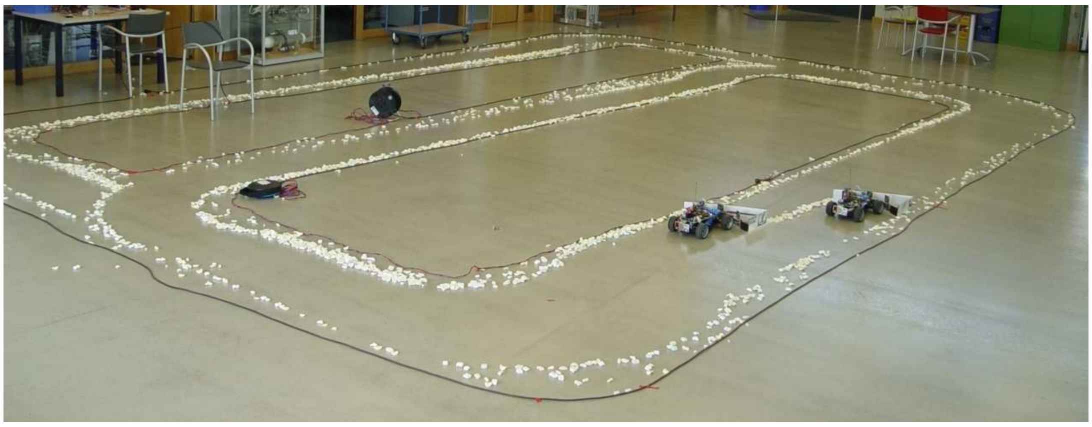
Figure 5.30: Snapshots from the snow shoveling hardware experiment. Part I.



(a) The robot shoveling the middle road waits for the other one to compose the formation. The picture was taken from the opposite side of the room unlike the picture in Fig. 5.30(a).



(b) The robot shoveling the middle road accelerates to join the first plough.



(c) The ploughs have completed one cleaning cycle.

Figure 5.31: Snapshots from the snow shoveling hardware experiment. Part II.

6

Conclusion and future work

6.1 Conclusion

This thesis described the stabilization and trajectory planning for formations of the nonholonomic car-like robots based on the receding horizon control. The presented methods offered general and robust solutions considering static as well as dynamic obstacles, the dynamic allocation of the vehicles to variable groups and robots' failures tolerance. We proposed a novel general concept of the integration of the global information describing the known structure of the environment and the local image of the vehicles' neighborhood, which is frequently updated by on-board sensors. We showed that this optimization based method, which provides control inputs for all members of the formation, converges to a given target region. As an extension of this approach, we developed an algorithm employing two virtual leaders in the task of complicated manoeuvring with a view to optimally manage a U-turn of the whole compact formation in arbitrary workspaces. Beside this, we proposed several methods based on path planning for the virtual leader of the formation in the space of multinomials to reduce the complexity of the optimization problem. The algorithms derive benefits from the principle of the chosen optimization techniques and adjust them for the purpose of robotics. Furthermore, they enable a decomposition of the optimization process which is useful for the initial determination of the length of time horizons utilized in RHC.

In the part of the thesis focussed on applications, we proposed a compete system designed for the task of airport snow shoveling using formations of autonomous ploughs. Two different approaches were proposed in the highest reasoning level of the system, which solves the task allocation problem with a view to optimally cover all runways and auxiliary roads. The first method can be employed only for a static map of the airport, but the optimality of the result is ensured. The second method is providing only sub-optimal solutions, but with the possibility of an on-line respond to detected changes in the environment. The output of the task allocation methods are performed by the general formation driving approach that was adapted for the purpose of the optimal coverage. Additionally, we extended the approach with abilities to form temporary formations, to split formations to several smaller teams, to merge the sub-formations and to turn wide-spread formations, which are required by the task allocation strategies.

All separate methods and their extensions as well as the complete systems were verified by various simulations and hardware experiments. With the tests, we simulated the real workspace of robots using existing maps of buildings and a map of the Frankfurt airport. We studied the behavior of the proposed system in situations with static and dynamic obstacles. We verified the robustness of the method in case of a vehicle failure or in a situation where no feasible solution for the complete formation is possible.

6.2 Future work

The general formation driving method described in this thesis was adapted only for one application of car-like robots' formations, but as shown in the motivation presented in Section 1.1, the other scenarios introduce additional research topics. The future work in *3D cooperative mapping* could be an integration of the optimization of the stereo system baseline length to the formation stabilization. Additionally, the information obtained by the merging of images from different cameras could be utilized as a feedback for the position determination of the robots within the formation.

In the second application, which is called the *cloud of toxic gas determination*, the measurements of the concentration on each robot should influence not only a future movement of the group, but also the size of the formation. This again leads to an adapting of the cost function for the virtual leader (movement of the group) as well as for the followers (resizing of the formation). Another extension could be motivated by the gas cloud itself. The mathematic models describing the behavior of the clouds usually require three-dimensional information [4; 57; 195] and therefore it is useful to utilize heterogenous teams of mobile robots and helicopters. The high dynamic of the cloud can also be better captured using unmanned aerial vehicles. This requires the combining of different kinematic models and mainly an extension of the formation driving methods to 3D. Additional applications of formations of autonomous underwater vehicles, airplanes or even satellites are a logical continuation of such an effort.

Also the third scenario, *airport snow shoveling*, needs to be further investigated and improved. The optimal coverage of the surface on the crossroads of runways should be considered in the task allocation. Now the remaining snow is neglected on the crossroads and only the roads are completely cleaned as you can see from the simulations. Finally, the complete system and mainly the task allocation strategies should be verified in a long-term hardware experiment in a more realistic environment with moving obstacles and with a structure comparable to the map of a real airport. This supposes the utilization of robust outdoor robots (which can be e.g. outdoor MERLIN [54; 167] developed within the University of Wuerzburg) equipped with an external positioning system.

Appendix A

Utilized optimization techniques

A.1 Global optimization using Sparse Grids

The algorithm described in this appendix was originally presented in [59] and for utilization in robotics and was extended in terms of collaboration within the group of PhD students in program "Identification, Optimization and Control with Applications in Modern Technologies" supported by the Elite Network of Bavaria [155].

The key feature of the method is an adaptively working iterative technique that only uses function evaluations at specific points inside a given box in order to find a point that is in some way sufficiently close to a global solution. This part of the method is an enhancement of the HCP-Algorithm presented in [133] by Novak and Ritter. A clusteranalysis of the generated grid points is following this global part in order to find suitable starting points for the subsequent local part of the algorithm. This approach uses a Penalty-Barrier-Multiplier-method (PBM) [17] in order to find a more accurate solution. Each optimization in this local part is done with the Nelder-Mead algorithm.

The optimization problem holds certain challenges, such as the non-smoothness (even discontinuity) of the cost function and the expensiveness of each function evaluation. We also assume the existence of many local solutions and we aim at finding a global minimum. There are a few common methods able to deal with these difficulties. Unfortunately, most of the known solvers require at least a gradient information and provide only local solutions.

The optimization method described here is based on the algorithm presented in [133], which is a deterministic global optimization method using Sparse Grids (SG) with the aim to find a point that is in some sense close enough to the global optimum. The method is also able to consider some nonlinear constraints and employs clusteranalysis [29] and a PBM-algorithm [17] in order to find a solution with the required accuracy.

The core of the algorithm is the global part where a kind of sparse grid (see Fig. A.1 for examples) is built up iteratively.

Definition A.1.1. The one dimensional grid G_l with level l is given by

$$G_l = \left\{ \overrightarrow{x}_i : i2^{-(l+1)}, i = \pm\{0, 1, 2, \dots, 2^l - 1\} \right\}.$$

This is an equidistant grid[1] with step size $2^{-(l+1)}$. The corresponding $d-$dimensional sparse grid of level n is defined as a certain combination of the cartesian product of different one dimensional grids. Since we omit the points on the border this grid is also called *Noboundary-Grid.*

Definition A.1.2. The $d-$dimensional sparse grid $G^d_{l_n}$ with level l_n is given by

$$G^d_{l_n} = \bigcup_{l_1+\ldots+l_d=l_n} G_{l_1} \times \ldots \times G_{l_d}.$$

The utilized method does not create a whole sparse grid but starts at the center of the considered cube and successively adds new gridpoints by evaluating all existing grid points, choosing the best one and adding the neighbors (i.e. the closest grid points in the sparse grid of the next higher level) of this point. This evaluation of all existing points takes place by the regarding of the value of the function β_ζ:

$$\beta_\zeta(\overrightarrow{x}) = (\texttt{level}(\overrightarrow{x}) + \texttt{degree}(\overrightarrow{x}))^\zeta \texttt{rank}(\overrightarrow{x})^{1-\zeta}. \tag{A.1}$$

Where $\zeta \in [0,1]$ controls the adaptiveness of the algorithm. The `level` of a point $\overrightarrow{x}$ is defined by the level of the grid in which the points appears. The `degree` of a grid point counts how often this point was already chosen as "best" point. The `rank` of the point is important in order to allow the adaptiveness because it marks the number of points with a smaller (or equal) objective value. So if ζ is close to zero we get a local method, whereas when $\zeta = 1$ the sparse grid is set up without considering the objective function.

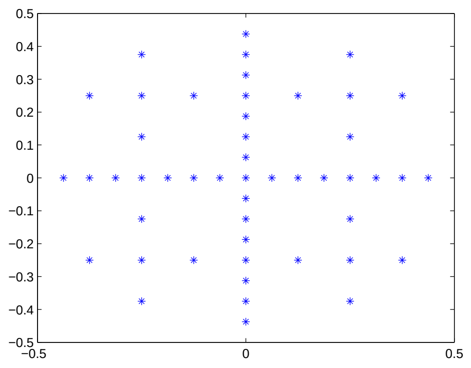
The first part of the algorithm, the global search, stops after a given number of function evaluations. Depending on the parameter ζ one can usually find certain clusters in the set of the considered grid points. This is done by using a subtractive clustering method. The minimum of each cluster is a suitable starting point for a local method. Here a PBM-method is used in order to deal with the constraints. The local optimization is done with the Nelder-Mead method.

A.2 Particle Swarm Optimization

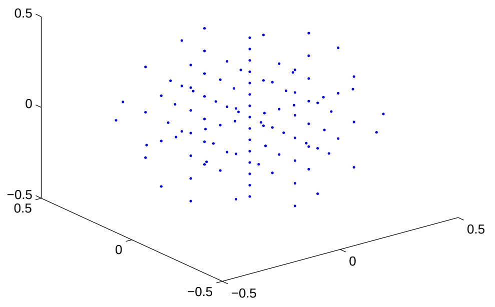
The PSO method was developed for finding a global optimum of a nonlinear function [91; 119]. This optimization approach has been inspired by the social behavior of birds and fish. Each solution consists of a set of parameters and represents a point in a multidimensional space. The solution is called "particle" and the group of particles (population) is called "swarm". The method has been adapted for the application in mobile robotics in [162] and an abbreviated description will be provided here.

In the PSO technique, two kinds of information is available to the particles. The first is their own experience, i.e. their best state and its fitness value so far. The other information is their social knowledge, i.e. the particles know the momentary best solution $\overrightarrow{x}_{g,b}$ of the group found during the evaluation process.

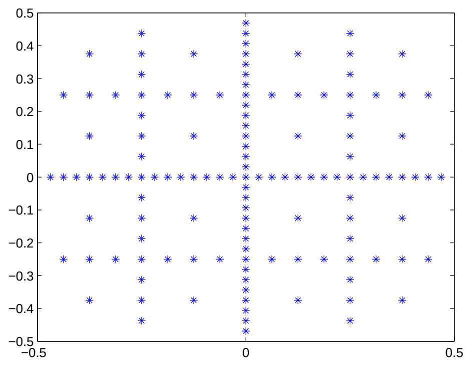
[1]Only the cube $[-0.5, 0.5]^d$ is considered to simplify matters.



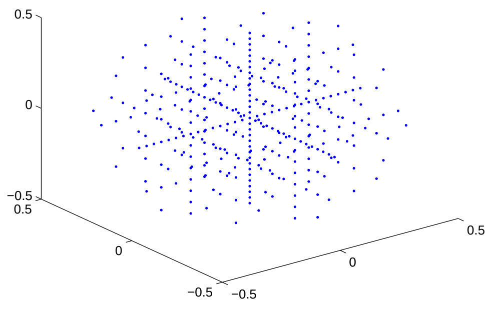
(a) A two dimensional grid of level 3.



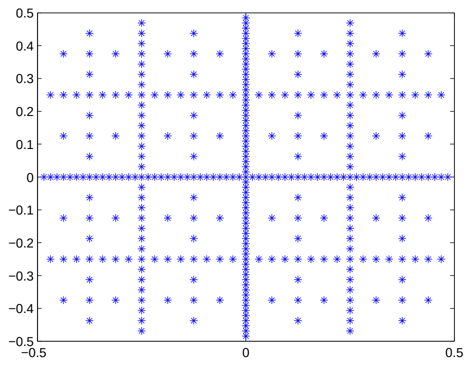
(b) A three dimensional grid of level 3.



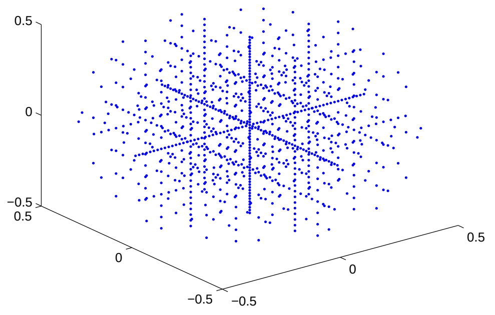
(c) A two dimensional grid of level 4.



(d) A three dimensional grid of level 4.



(e) A two dimensional grid of level 5.



(f) A three dimensional grid of level 5.

Figure A.1: Two and three dimensional Noboundary-Grids of different levels.

Each particle is represented as a D-dimensional position vector $\overrightarrow{x}_i(\iota)$ in discrete time ι and has a corresponding instantaneous velocity vector $\overrightarrow{v}_i(\iota)$. Furthermore, it remembers its individual best value of fitness function and position $\overrightarrow{x}_{i,b}$ which has resulted in that value.

During each iteration ι, the velocity update rule is applied on each particle in the swarm as

$$\overrightarrow{v}_i(\iota) = w\overrightarrow{v}_i(\iota-1) + \Phi_1(\overrightarrow{x}_{i,b} - \overrightarrow{x}_i(\iota-1)) + \Phi_2(\overrightarrow{x}_{g,b} - \overrightarrow{x}_i(\iota-1)), \qquad \text{(A.2)}$$

where parameter w is called inertia weight and it decreases linearly from w_{start} to w_{end} during all iterations. The symbols Φ_1 and Φ_2 are computed according to the equation

$$\Phi_j = \Upsilon_j \begin{pmatrix} r_{j1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & r_{jD} \end{pmatrix}, \qquad \text{(A.3)}$$

where $j \in \{1, 2\}$. The parameters Υ_i are constants that weight the influence of the particles' own experience and of the social knowledge. In our experiments, the parameters were set to $\Upsilon_1 = 2$ and $\Upsilon_2 = 2$. The r_{jk}, where $k \in \{1, \ldots, D\}$, are random numbers drawn from a uniform distribution between 0 and 1.

The position of the particles is then computed according to the update rule

$$\overrightarrow{x}_i(\iota) = \overrightarrow{x}_i(\iota-1) + \overrightarrow{v}_i(\iota), \qquad \text{(A.4)}$$

where the following guideline is applied for the stabilization of the optimization process. If any component of $\overrightarrow{v}_i$ is less than $-V_{max}$ or greater than $+V_{max}$, the corresponding value is replaced by $-V_{max}$ or $+V_{max}$, respectively, where V_{max} is the maximum velocity parameter.

The update formulas (A.2) and (A.4) are applied during each iteration and the values of $\overrightarrow{x}_{i,b}$ and $\overrightarrow{x}_{g,b}$ are updated simultaneously. The algorithm is stopped if the maximum number of iterations is reached or if any other predefined stopping criteria is satisfied.

References

[1] S. Abdallah and V. Lesser. **Organization-based cooperative coalition formation**. In *Proc. of IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2004)*, September 2004.

[2] P. K. Agarwal, T. Biedl, S. Lazard, Robbins S., Suri S., and S. Whitesides. **Curvature-constrained shortest paths in a convex polygon**. *SIAM journal on computing*, **31**(6):1814–1851, 2002.

[3] M. Ahmadi and P. Stone. **Multi-robot Learning for Continuous Area Sweeping**. *Lecture Notes in Computer Science, Learning and Adaption in Multi-Agent Systems*, **3898**:47–70, 2006.

[4] V. N. Alexandrov, W. Owczarz, P. G. Thomson, and Z. Zlatev. **Parallel runs of a large air pollution model on a grid of Sun computers**. *Mathematics and Computers in Simulation*, **65**(6):557–577, 2004.

[5] Y. Altshuler, A.M. Bruckstein, and I.A. Wagner. **Swarm robotics for a dynamic cleaning problem**. In *Proc. of the IEEE Swarm Intelligence Symposium*, pages 209–216, June 2005.

[6] A. Alvarez, A. Caiti, and R. Onken. **Evolutionary path planning for autonomous underwater vehicles in a variable ocean**. *IEEE Journal of Oceanic Engineering*, **29**(2):418–429, 2004.

[7] I. Ashiru, C. Czarnecki, and T. Routen. **Characteristics of a genetic based approach to path planning for mobile robots**. *Journal of network and computer applications*, **19**(2):149–169, 1996.

[8] M. Athans and P. L. Falb. *Optimal Control: An Introduction to the Theory and Its Applications.* Sydney : McGraw-Hill, 1th edition, 1963.

[9] F. Aurenhammer and R. Klein. *Voronoi diagrams. Hand book of Computational Geometry.* Elsevier Science Publishers, Amsterodam, 2000.

[10] T. Balch and R. C. Arkin. **Behaviour-based Formation Control for Multi-robot Teams**. *IEEE Transactions on Robotics and Automation*, **14**(6):926–939, December 1998.

[11] T. Balch and M. Hybinette. **Social potentials for scalable multi-robot formations**. In *Proc. of IEEE Conference on Robotics and Automation*, **1**, pages 73–80, April 2000.

[12] D. J. Balkcom, P. A. Kavathekar, and M. T. Mason. **Time-optimal Trajectories for an Omnidirectional Vehicle**. *International Journal of Robotics Research*, **25**(10):985–999, 2006.

[13] T. D. Barfoot and C. M. Clark. **Motion Planning for Formations of Mobile Robots**. *Robotics and Autonomous Systems*, **46**:65–78, February 2004.

[14] T. D. Barfoot, C. M. Clark, S. M. Rock, and G. M. T. D'Eleuterio. **Kinematic Path-planning for Formations of Mobile Robots with a Nonholonomic Constraint**. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2002.

[15] R.W. Beard, J. Lawton, and F.Y. Hadaegh. **A coordination architecture for spacecraft formation control**. *IEEE Transactions on Control Systems Technology*, **9**(6):777 – 790, November 2001.

[16] S. Behzadipour and A. Khajepour. **Time-optimal trajectory planning in cable-based manipulators**. *IEEE Transactions on Robotics*, **22**(3):559–563, 2006.

[17] A. Ben-Tal and M. Zibulevsky. **Penalty/Barrier Multiplier Methods for convex Programming Problems**. *SIAM Journal on Optimization*, **7**(2):347 – 366, 1997.

[18] J.G. Bender. **An overview of systems studies of automated highway systems**. *IEEE Transactions on Vehicular Technology*, **40**(1):82– 99, 1991.

[19] R. M. Bhatt, C. P. Tang, and V. N. Krovi. **Formation optimization for a fleet of wheeled mobile robots - A geometric approach**. *Robotics and Autonomous Systems*, **57**(1):102–120, 2009.

[20] L.T. Biegler. **Advances in nonlinear programming concepts for process control**. *Journal of Process Control*, **8**(5):301–311, 1998.

[21] J. Borenstein and Y. Koren. **The Vector Field Histogram: Fast Obstacle Avoidance for Mobile Robots.** *IEEE Journal of Robotics and Automation*, pages 278–288, 1991.

[22] T. Braunl and N. Tay. **Combining configuration space and occupancy grid for robot navigation.** *Industrial Robot*, **28(3)**:233–41, 2001.

[23] L. Breger, J. How, and A. Richards. **Model predictive control of spacecraft formations with sensing noise**. In *Proc. of American Control Conference, 2005.*, **4**, pages 2385–2390, 2005.

[24] B. Bulka, M. Gaston, and M. Desjardins. **Local strategy learning in networked multi-agent team formation**. *Autonomous Agents and Multi-Agent Systems*, **15**(1):29 – 45, 2007.

[25] Vasudevan C. and Ganesan K. **Case-Based Path Planning for Autonomous Underwater Vehicles**. *Autonomous Robots*, **3**(2):79–89, 1996.

[26] C. G. Cassandras and W. Li. **Sensor networks and cooperative control**. *European Journal of Control*, **11**(4–5):436–463, 2005.

[27] C. Chen and S. Wang. **Branch-and-bound scheduling for thermal generating units**. *IEEE transactions on energy conversion*, **8**:184–189, 1993.

[28] D.Z. Chen, R.J. Szczerba, and J.J. Uhran. **A framed-quadtree approach for determining Euclidean shortest pathsin a 2-D environment**. *IEEE Transactions on Robotics and Automation*, **13**(5):668–681, 1997.

[29] S.L. Chiu. **Fuzzy Model Identification Based on Cluster Estimation**. *Journal of intelligent and fuzzy systems*, **2**:267–278, 1994.

[30] H. Choset. **Coverage for robotics A survey of recent results**. *Annals of Mathematics and Artificial Intelligence*, **31**(1–4):113 – 126, 2001.

[31] L. Consolini, F. Morbidi, D. Prattichizzo, and M. Tosques. **Leader-follower formation control of nonholonomic mobile robots with input constraints**. *Automatica*, **44**(5):1343–1349, 2008.

[32] M. Corless and G. Leitmann. **Adaptive Controllers for Avoidance or Evasion in an Uncertain Environment: Some Examples**. *Computers & Mathematics with Applications*, **18**:161–170, 1989.

[33] M. Corless, G. Leitmann, and J. Skowronski. **Adaptive Control for Avoidance or Evasion in an Uncertain Environment**. *Computers & Mathematics with Applications*, **13**:1–11, 1987.

[34] J. Cortés, S. Martínez, T. Karatus, and F. Bullo. **Coverage control for mobile sensing networks**. *IEEE Transactions on Robotics and Automation*, **20**(2):243–255, 2004.

[35] X. Cui, T. Hardin, R.K. Ragade, and A.S. Elmaghraby. **A swarm-based fuzzy logic control mobile sensor network for hazardous contaminants localization**. In *Proc. of IEEE International Conference on Mobile Adhoc and Sensor Systems*, 2004.

[36] T.B. Curtin, J.G. Bellingham, J. Catipovic, and D. Webb. **Autonomous oceanographic sampling networks**. *Oceanography*, **6**(3):86–94, 1993.

[37] M. J. Daigle, X. D. Koutsoukos, and G. Biswas. **Distributed Diagnosis in Formations of Mobile Robots**. *IEEE Transactions on Robotics*, **23**(2):353 – 369, April 2007.

[38] A.K. Das, R. Fierro, V. Kumar, J.P. Ostrowski, J. Spletzer, and C.J. Taylor. **A Vision-Based Formation Control Framework**. *IEEE Transactions on Robotics and Automation*, **18**(5):813–825, October 2003.

[39] B. Dasgupta, A. Gupta, and E. Singla. **A variational approach to path planning for hyper-redundant manipulators**. *Robotics and Autonomous Systems*, **57**(2):194–201, 2009.

[40] G. De Nicolao, L. Magni, and R. Scattolini. **Stabilizing receding-horizon control of nonlinear time-varying systems**. *IEEE Transactions on Automatic Control*, **43**(7):1030–1036, 1998.

[41] J.C. Derenick and J.R. Spletzer. **Convex Optimization Strategies for Coordinating Large-Scale Robot Formations**. *IEEE Transactions on Robotics*, **23**(6):1252–1259, December 2007.

[42] J.P. Desai, J.P. Ostrowski, and V. Kumar. **Modeling and control of formations of nonholonomic mobile robots**. *IEEE Transactions on Robotics and Automation*, **17**(6):905–908, December 2001.

[43] A. Dhariwal, G.S. Sukhatme, and A.A.G. Requicha. **Bacterium-inspired robots for environmental monitoring**. In *Proc. of IEEE International Conference on Robotics and Automation*, 2004.

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

[45] D. V. Dimarogonas, S. G. Loizou, K. J. Kyriakopoulos, and M. M. Zavlanos. **A Feedback Stabilization and Collision Avoidance Scheme for Multiple Independent Non-point Agents**. *Automatica*, **42**(2):229–243, 2006.

[46] R. Dittmar and B. M. Pfeiffer. *Modellbasierte praediktive Regelung*. Oldenbourg Verlage GmbH, Muenchen, 2004.

[47] K. D. Do. **Bounded Controllers for Formation Stabilization of Mobile Agents With Limited Sensing Ranges**. *IEEE Transactions on Automatic Control*, **52**(3):569–576, 2007.

[48] K. D. Do. **Formation Tracking Control of Unicycle-Type Mobile Robots With Limited Sensing Ranges**. *IEEE Transactions on Control Systems Technology*, **16**(3):527–538, 2008.

[49] K. D. Do. **Output-feedback formation tracking control of unicycle-type mobile robots with limited sensing ranges**. *Robotics and Autonomous Systems*, **57**(1):34–47, 2009.

[50] K. D. Do, Z. Jiang, and J. Pan. **A global output-feedback controller for simultaneous tracking and stabilization of unicycle-type mobile robots**. *IEEE Transactions on Robotics and Automation*, **20**(3):589–594, 2004.

[51] F. Driewer, H. Baier, and K. Schilling. **Robot/Human Rescue Teams: A User Requirement Analysis**. *Advanced Robotics*, **19**(8):819–838, 2005.

[52] W.B. Dunbar and R.M. Murray. **Receding horizon control of multi-vehicle formations: A distributed implementation**. In *Proc. of 43rd IEEE Conference on Decision and Control.*, 2004.

[53] W.B. Dunbar and R.M. Murray. **Distributed receding horizon control for multi-vehicle formation stabilization**. *Automatica*, **42**(4):549–558, April 2006.

[54] D. Eck, M. Stahl, and K. Schilling. **The Small Outdoor Rover MERLIN and its Assistance System for Tele-Operations**. In *Proc of the 6th International Conference on Field and Service Robotics*, 2007.

[55] ENB. **International Doctorate Program: Identification, Optimization and Control with Applications in Modern Technologies. Movies**. *http://www2.am.uni-erlangen.de/elitenetzwerk-optimierung/index_en.php?page=movies&lang=en*, February 2009.

[56] F. Fahimi. **Sliding-Mode Formation Control for Underactuated Surface Vessels**. *IEEE Transactions on Robotics*, **23**(3):617 – 622, June 2007.

[57] J. A. Farrell, J. Murlis, X. Long, W. Li, and R. T. Carde. **Filament-Based Atmospheric Dispersion Model to Achieve Short Time-Scale Structure of Odor Plumes**. *Environmental Fluid Mechanics*, **2**(1-2):143–169, 2002.

[58] J. A. Fax and R. M. Murray. **Information flow and cooperative control of vehicle formations**. *IEEE Transactions on Automatic Control*, **49**(9):1465–1476, 2004.

[59] I. Ferenczi. *Globale Optimierung unter Nebenbedingungen mit dunnen Gittern*. Diploma thesis, Technische Universitat Munchen, 2005.

[60] R. Fierro, A.K. Das, V. Kumar, and J.P. Ostrowski. **Hybrid Control of Formations of Robots**. In *Proc. of IEEE Conference on Robotics and Automation*, **1**, May 2001.

[61] R. Findeisen and F. Allgower. **An Introduction to Nonlinear Model Predictive Control**. In *Proc. of the 21st Benelux Meeting on Systems and Control*, Veldhoven, 2002.

[62] FMINCON. **Optimization Toolbox, Matlab**. *http://www.mathworks.com/access/helpdesk/help/toolbox/optim/ug/fmincon.html*, April 2009.

[63] E. Franco, L. Magni, T. Parisini, M.M. Polycarpou, and D.M. Raimondo. **Cooperative Constrained Control of Distributed Agents With Nonlinear Dynamics and Delayed Information Exchange: A Stabilizing Receding-Horizon Approach**. *IEEE Transactions on Automatic Control*, **53**(1):324–338, February 2008.

[64] J. Fredslund and M.J. Mataric. **A General Algorithm for Robot Formations Using Local Sensing and Minimal Communication**. *IEEE Transactions on Robotics and Automation, special issue on Advances in Multi-Robot Systems*, **18**(5):837–846, October 2002.

[65] C.H. Fua, S.S. Ge, K. Duc Do, and K.-W. Lim. **Multirobot Formations Based on the Queue-Formation Scheme With Limited Communication**. *IEEE Transactions on Robotics*, **23**(6):1160–1169, 2007.

[66] K. Fujimura. **Path planning with multiple objectives**. *IEEE Robotics and Automation Magazine*, **3**(1):33–38, 1996.

[67] B.P. Gerkey and M.J. Mataric. **A framework for studying multirobot task allocation**. In *Proc. of the 2nd International Naval Research Laboratory Workshop on Multi-Robot Systems*, March 2003.

[68] B.P. Gerkey and M.J. Mataric. **A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems**. *The International Journal of Robotics Research*, **23**(9):939–954, 2004.

[69] F. Giulietti, M. Innocenti, M. Napolitano, and L. Pollini. **Dynamic and control issues of formation flight**. *Aerospace Science and Technology*, **9**(1):65–71, 2005.

[70] D. Gu and H. Hu. **Receding horizon tracking control of wheeled mobile robots**. *IEEE Transactions on Control Systems Technology*, **14**(4):743–749, 2006.

[71] F. Hadaegh, W. M. Lu, and P. Wang. **Adaptive Control of Formation Flying Spacecraft for Interferometry**. In *Proc. of Large Scale Systems: Theory and Applications*, 1998.

[72] Y. Hao, B. Laxton, E. R. Benson, and S. K. Agrawal. **Differential flatness-based formation following of a simulated autonomous small grain harvesting system.** *Transactions of the ASAE*, **47**(3):933–941, 2004.

[73] Z. B. Hao, N. Sang, and H. Lei. **Cooperative Coverage by Multiple Robots with Contact Sensors**. In *Proc. of IEEE Conference on Robotics, Automation and Mechatronics*, 2008.

[74] C.T. Hardin, X. Cui, R.K. Ragade, J.H. Graham, and A.S. Elmaghraby. **A modified particle swarm algorithm for robotic mapping of hazardous environments**. In *Proc. of World Automation Congress*, 2004.

[75] T. Hatanaka, N. Kitudmrat, and M. Fujita. **Formation Control via Receding Horizon Control : A Set Theoretic Approach**. In *Proc. of international conference on Instrumentation, Control and Information Technology (SICE 2008)*, August 2008.

[76] N. Hazon and G. A. Kaminka. **Redundancy, Efficiency and Robustness in Multi-Robot Coverage**. *The International Journal of Robotics Research*, pages 735–741, 2005.

[77] M. Hess, M. Saska, and K. Schilling. **Autonomous multi-vehicle formations for cooperative airfield snow shoveling**. In *Proc. of the 3rd European Conference on Mobile Robots (ECMR07)*, Freiburg, Germany, 2007.

[78] M. Hess, M. Saska, and K. Schilling. **Enhanced Motion Planning for Dynamic formations of Nonholonomic Mobile Robots**. In *Proc. of the 6th IFAC Symposium on Intelligent Autonomous Vehicles (IAV2007)*, September 2007.

[79] M. Hess, M. Saska, and K. Schilling. **Application of Coordinated Multi Vehicle Formations for Snow Shoveling on Airports**. *Inteligent Service Robotics*, **2**(4):205 – 217, October 2009.

[80] A. Howard, L. E. Parker, and G. S. Sukhatme. **Experiments with a Large Heterogeneous Mobile Robot Team: Exploration, Mapping, Deployment and Detection**. *International Journal of Robotics Research*, **25**(5–6):431–447, 2006.

[81] H. Hu and M. Brady. **Dynamic global path planning with uncertainty for mobile robots inmanufacturing**. *IEEE Transactions on Robotics and Automation*, **13**(5):760–767, 1997.

[82] J. Huang, S.M. Farritor, A. Qadi, and S. Goddard. **Localization and follow-the-leader control of a heterogeneous group of mobile robots**. *IEEE/ASME Transactions on Mechatronics*, **11**(2):205 – 215, April 2006.

[83] I. Hussein and D. M. Stipanović. **Effective Coverage Control for Mobile Sensor Networks.** In *Proc. of the IEEE Conference on Decision and Control*, pages 2747–2752, San Diego, CA, 2006.

[84] A. Jadbabaie, Jie Lin, and A.S. Morse. **Coordination of groups of mobile autonomous agents using nearest neighbor rules**. *IEEE Transactions on Automatic Control*, **48**(6):988 – 1001, June 2003.

[85] M. Jager and B. Nebel. **Dynamic decentralized area partitioning for cooperating cleaning robots**. In *Proc. of the IEEE International Conference on Robotics and Automation*, **4**, pages 3577–3582, 2002.

[86] G. Jan, K. Yin Chang, and I. Parberry. **Optimal Path Planning for Mobile Robot Navigation**. *IEEE/ASME Transactions on Mechatronics*, **13**(4):451 – 460, August 2008.

[87] D. Jung, G. Cheng, and E. Zelinsky. **Robot Cleaning: An Application of Distributed Planning and Real-time Vision**. In *Proc. of International conference on Field and Service Robotics*, 1998.

[88] G.A. Kaminka, R. Schechter-Glick, and V. Sadov. **Using Sensor Morphology for Multirobot Formations**. *IEEE Transactions on Robotics*, **24**(2):271 – 282, April 2008.

[89] W. Kang, N. Xi, and A. Sparks. **Formation control of autonomous agents in 3D workspace**. In *Proc. of IEEE International Conference on Robotics and Automation.*, 2000.

[90] S. S. Keerthi and E. G. Gilbert. **Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: stability and moving-horizon approximations**. *J. Optim. Theory Appl.*, **57**(2):265–293, 1988.

[91] J. Kennedy and R.C. Eberhart. **Particle Swarm Optimization**. In *Proc. International Conference on Neural Networks IEEE*, **4**, pages 1942–1948, 1995.

[92] T. Keviczky, F. Borrelli, K. Fregene, D. Godbole, and G.J. Balas. **Decentralized Receding Horizon Control and Coordination of Autonomous Vehicle Formations**. *IEEE Transactions on Control Systems Technology*, **16**(1):19–33, Jan. 2008.

[93] H.K. Khalil. *Nonlinear Systems*. Prentice Hall, Michigan State University, MI, 3th edition, 2001.

[94] O. Khatib. **Real-Time Obstacle Avoidance for Manipulators and Mobile Robots**. *The International Journal of Robotics Research*, **5**:90–98, 1986.

[95] Y. Kim, D.-W. Gu, and I. Postlethwaite. **Real-time path planning with limited information for autonomous unmanned air vehicles**. *Automatica*, **44**(3):696–712, 2008.

[96] M. Kloetzer and C. Belta. **Temporal Logic Planning and Control of Robotic Swarms by Hierarchical Abstractions**. *IEEE Transactions on Robotics*, **23**(2):320 – 330, April 2007.

[97] W.H. Know and S. Han. *Receding Horizon Control: model predictive control for state models (Advanced Textbooks in Control and Signal Processing)*. Springer, London, 1th edition, 2005.

[98] D. Kurabayashi, J. Ota, T. Arai, S. Ichikawa, S. Koga, H. Asama, and I. Endo. **Cooperative Sweeping by Multiple Mobile Robots with Relocating Portable Obstacles**. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1472 – 1477, Osaka, Japan, 1996.

[99] D. Kurabayashi, J. Ota, T. Arai, and E. Yoshida. **Cooperative Sweeping by Multiple Mobile Robots**. In *Proc. of the IEEE International Conference on Robotics and Automation*, Minneapolis, Minnesota, 1996.

[100] R. Kurazume and S. Hirose. **Development of a Cleaning Robot System with Cooperative Positioning System**. *Autonomous Robots*, **9**(3):237 – 246, 2000.

[101] Y. Kuwata and J.P. How. **Stable trajectory design for highly constrained environments using receding horizon control**. In *Proc of American Control Conference*, **1**, pages 902– 907, 2004.

[102] E. Lalish, K. A. Morgansen, and T. Tsukamaki. **Formation Tracking Control using Virtual Structures and Deconfliction**. In *Proc. 42th IEEE Conference on Decision and Control*, 2006.

[103] D. Langer, J.K. Rosenblatt, and M. Hebert. **A behavior-based system for off-road navigation**. *IEEE Transactions on Robotics and Automation*, **10**(6):776–783, December 1994.

[104] J.C. Latombe. *Robot Motion Planning*. MA: Kluwer, Norwell, 1991.

[105] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[106] J.R.T. Lawton, R.W. Beard, and B.J. Young. **A Decentralized Approach to Formation Maneuvers.** *IEEE Transactions on Robotics and Automation*, **19**(6):933–941, December 2003.

[107] D. V. Lebedev, J. J. Steil, and H. J. Ritter. **The dynamic wave expansion neural network model for robot motion planning in time-varying environments**. *Neural networks*, **18**(3):267–285, 2005.

[108] D. Lee and P. Y. Li. **Passive decomposition approach to formation and maneuver control of multiple rigid bodies**. *ASME Journal of Dynamic Systems, Measurement and Control*, **129**(5):662–677, 2007.

[109] G. Leitmann. **Guaranteed Avoidance Strategies**. *Journal of Optimization Theory and Applications*, **32**:569–576, 1980.

[110] G. Leitmann and J. Skowronski. **Avoidance Control**. *Journal of Optimization Theory and Applications*, **23**:581–591, 1977.

[111] N.E. Leonard and E. Fiorelli. **Virtual leaders, artificial potentials and coordinated control of groups**. In *Proc. of the 40th IEEE Conference on Decision and Control.*, pages 2968– 2973, 2001.

[112] M.A. Lewis and K. Tan. **High Precision Formation Control of Mobile Robots Using Virtual Structures**. *Autonomous Robots.*, **4**(4):387–403, 1997.

[113] Y. Li and X. Chen. **Stability on multi-robot formation with dynamic interaction topologies**. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems on Systems and Control*, 2005.

[114] A. J. Lilienthal and T. Duckett. **Building gas concentration gridmaps with a mobile robot**. *Robotics and Autonomous Systems*, **48**(1):3–16, 2004.

[115] H. T. Liu, S. Jinjun, and S. Dong. **Adaptive synchronization control of multiple spacecraft formation flying**. *Journal of dynamic systems, measurement, and control*, **129**(3):337–342, 2007.

[116] C. B. Low and D. Wang. **GPS-Based Tracking Control for a Car-Like Wheeled Mobile Robot With Skidding and Slipping**. *IEEE/ASME Transactions on Mechatronics*, **13**(4):480 – 484, August 2008.

[117] C. Luo and X. Yang. **A Real-time Cooperative Sweeping Strategy for Multiple Cleaning Robots**. In *Proc. of the IEEE International Symposium on Intelligent Control*, Vancouver, Canada, 2002.

[118] C. Luo, X. Yang, and D. Stacey. **Real-time Path planning with Deadlock Avoidance of Multiple Cleaning Robots**. In *Proc. of the IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, 2003.

[119] M. Macas, D. Novak, and L. Lhotska. **Particle swarm optimization for hidden markov models with application to intracranial pressure analysis.** In *Proc. of Biosignal*, 2006.

[120] M. Macas, M. Saska, L. Lhotska, L. Preucil, and K. Schilling. *Particle Swarm Optimization*, chapter Path Planning for Formations of Mobile Robots using PSO Technique, pages 329–347. InTech Education and Publishing, 2009.

[121] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. **Constrained model predictive control: Stability and optimality**. *Automatica*, **36**(6):789–814, 2000.

[122] D.Q. Mayne and H. Michalska. **Receding horizon control of nonlinear systems**. *IEEE Transactions on Automatic Control*, **35**(7):814–824, 1990.

[123] J. Mejia and D. M. Stipanović. **Asymptotic Stabilization Using a Constructive Approach to Constrained Nonlinear Model Predictive Control**. In *Proc. of the 47th IEEE Conference on Decision and Control*, Cancun, Mexico, 2008.

[124] J.S. Mejia and D.M. Stipanovic. **Safe trajectory tracking for the two-aircraft system**. In *Proc. of IEEE International Conference on Electro/Information Technology*, 2007.

[125] H. Michalska and D.Q. Mayne. **Robust receding horizon control of constrained nonlinear systems**. *IEEE Transactions on Automatic Control*, **38**(11):1623–1633, 1993.

[126] D. Milutinovi and P. Lima. **Modeling and Optimal Centralized Control of a Large-Size Robotic Population**. *IEEE Transactions on Robotics*, **22**(6):1280 – 1285, December 2006.

[127] T.W. Min and H.K. Yin. **A decentralized approach for cooperative sweeping by multiple mobile robots**. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Victoria, Canada, 1998.

[128] P. Moore and J. Crimaldi. **Odor landscapes and animal behavior: tracking odor plumes in different physical worlds**. *Journal of marine systems*, **49**(1–4):55–64, 2004.

[129] A.I. Mourikis and S.I. Roumeliotis. **Optimal sensor scheduling for resource-constrained localization of mobile robot formations**. *IEEE Transactions on Robotics*, **22**(5):917 – 931, october 2006.

[130] A.I. Mourikis and S.I. Roumeliotis. **Performance analysis of multirobot Cooperative localization**. *IEEE Transactions on Robotics*, **22**(4):666 – 681, August 2006.

[131] I. K. Nikolos and N. Tsourveloudis. **Evolutionary path planning for unmanned aerial vehicles cooperation**. In *Proc of International Conference on Informatics in Control Automation and Robotics*, 2007.

[132] N. Noguchi and H. Terao. **Path planning of an agricultural mobile robot by neural network and genetic algorithm**. *Computers and electronics in agriculture*, pages 187–204, 1997.

[133] E. Novak and K. Ritter. **Global Optimization Using Hyperbolic Cross Points**. In *State of the Art in global Optimization.*, pages 19–33. Kluver Academic Publishers, Dordrecht, 1996.

[134] P. Ogren, E. Fiorelli, and N. E. Leonard. **Formations with a Mission: Stable Coordination of Vehicle Group Maneuvers**. In *Proc. 15th International Symposium on Mathematical Theory of Networks and Systems*, 2002.

[135] P. Ogren, E. Fiorelli, and N. E. Leonard. **Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment**. *IEEE Transactions on Automatic Control*, **49**(8):1292–1302, 2004.

[136] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

[137] Overaasen. **Runway Sweepers from Overaasen Snow Removal Systems, Gjovik, Norway.** *http://www.overaasen.no/runway-sweepers*, February 2009.

[138] L. Pallottino, V.G. Scordio, A. Bicchi, and E. Frazzoli. **Decentralized Cooperative Policy for Conflict Resolution in Multivehicle Systems**. *IEEE Transactions on Robotics*, **23**(6):1170 – 1183, December 2007.

[139] P. Papalambros and D. J. Wilde. *Principles of optimal design: Modeling and Computation, 2nd ed.* Cambridge University Press, 2000.

[140] L. Parker. **The effect of heterogeneity in teams of 100+ mobile robots**. *Multi-Robot Systems: From Swarms to Intelligent Automata*, **2**:205–215, 2003.

[141] L. E. Parker. **Current state of the art in distributed autonomous mobile robotics**. In *International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2000.

[142] L.E. Parker. **ALLIANCE: an architecture for fault tolerant multirobot cooperation**. *IEEE Transactions on Robotics and Automation*, **14**(2):220–240, April 1998.

[143] L.E. Parker and F. Tang. **Building Multirobot Coalitions Through Automated Task Solution Synthesis**. *Proc. of the IEEE*, **94**(7):1289–1305, July 2006.

[144] M. Peasgood, C.M. Clark, and J. McPhee. **A Complete and Scalable Strategy for Coordinating Multiple Robots Within Roadmaps**. *IEEE Transactions on Robotics*, **24**(2):283 – 292, April 2008.

[145] C. Petres, Y. Pailhas, P. Patron, Y. Petillot, J. Evans, and D. Lane. **Path Planning for Autonomous Underwater Vehicles**. *IEEE Transactions on Robotics*, **23**(2):331–341, 2007.

[146] E. Prassler, A. Ritter, C. Schaeffer, and P. Fiorini. **A Short History of Cleaning Robots**. *Autonomous Robots*, **9**(3):211–226, 2000.

[147] A.I. Propoi. **Use of linear programming methods for synthesizing sampled-data automatic systems**. *Automatic Remote Control*, **24**(7):837–844, 1963.

[148] S. Joe Qin and Thomas A. Badgwell. **A survey of industrial model predictive control technology**. *Control Engineering Practice*, **11**(7):733–764, 2003.

[149] C. V. Rao, S. J. Wright, and J. B. Rawlings. **Application of Interior-Point Methods to Model Predictive Control**. *Journal of Optimization Theory and Applications*, **99**(3):723–757, 1998.

[150] W. Ren and R.W. Beard. **Virtual structure based spacecraft formation control with formation feedback**. In *Proc. of AIAA Guidance, Navigation, and Control Conference*, 2002.

[151] W. Ren and R.W. Beard. **A decentralized scheme for spacecraft formation flying via the virtual structure approach**. In *Proc. of American Control Conference*, **2**, June 2003.

[152] J. Richalet, A. Rault, J. L. Testud, and J. Papon. **Model Predictive Heuristic Control: Applications to Industrial Processes**. *Automatica*, **14**:413–428, 1978.

[153] J. Rodrigues, D. Figueira, C. Neves, and M.I. Ribeiro. **Leader-Following Graph-Based Distributed Formation Control**. In *Proc. of Robotica 2008 - 8th Conference on Autonomous Robot Systems and Competitions*, 2008.

[154] R. A. Russell. **Tracking chemical plumes in constrained environments**. *Robotica*, **19**(4):451–458, 2001.

[155] M. Saska, I. Ferenczi, M. Hess, and K. Schilling. **Path Planning for Formations Using Global Optimization with Sparse Grids**. In *Proc. of The 13th IASTED International Conference on Robotics and Applications (RA 2007).*, Wuerzburg, Germany, 2007.

[156] M. Saska, M. Hess, and K. Schilling. **Hierarchical Spline Path Planning Method for Complex Environments**. In *Proc. of the 4th International Conference on Informatics in Control, Automation and Robotics.*, Angers, France, 2007.

[157] M. Saska, M. Hess, and K. Schilling. **Path Planning and Motion Coordination for Compact Vehicle-Formations**. In *Proc. of 13th Portuguese Conference on Artificial Intelligence (EPIA'07)*, Guimaraes, Portugal, December 2007.

[158] M. Saska, M. Hess, and K. Schilling. **Voronoi Strains - A Spline Path Planning Algorithm for Complex Environments**. In *Proc. of the IASTED conference on Artificial Intelligence and Applications*, Innsbruck, Austria, January 2007.

[159] M. Saska, M. Hess, and K. Schilling. **Efficient Airport Snow Shoveling by Applying Autonomous Multi-Vehicle Formations**. In *Proc. of IEEE International Conference on Robotics and Automation*, Pasadena, USA, May 2008.

[160] M. Saska, M. Hess, and K. Schilling. **Route Scheduling Approach for Airport Snow Shoveling using Formations of Autonomous Ploughs**. In *Proc. of 10th International Conference on Control, Automation, Robotics and Vision (ICARCV 2008)*, Hanoi, Vietnam, December 2008.

[161] M. Saska, M. Kulich, G. Klančar, and J. Faigl. **Transformed net - collision avoidance algorithm for robotic soccer**. In *Proc. of the 5th Vienna Symposium on Mathematical Modelling.* Vienna: ARGESIM, 2006.

[162] M. Saska, M. Macas, L. Preucil, and L. Lhotska. **Robot Path Planning using Partical Swarm Optimization of Ferguson Splines**. In *Proc. of the 11th IEEE International Conference on Emerging Technologies and Factory Automation. ETFA 2006.*, 2006.

[163] M. Saska, J. S. Mejia, D. M. Stipanovic, and K. Schilling. **Control and Navigation of Formations of Car-Like Robots on a Receding Horizon**. In *Proc of 3rd IEEE Multi-conference on Systems and Control*, 2009.

[164] M. Saska, L. Preucil, and M. Kulich. **Elliptic Net A Path Planning Algorithm for Dynamic Environments**. In *Proc. of the 3th International Conference on Informatics in Control, Automation and Robotics*, 2006.

[165] K. Schilling. **Navigation by cooperating mobile robots**. In *Proc of Intelligent robots and computer vision*, pages 354–359, 2004.

[166] K. Schilling and F. Driewer. **Remote Control of Mobile Robots for Emergencies**. In *Proc of 16th IFAC World Congress*, Prague, Czech Republic, 2005.

[167] K. Schilling and Q. Meng. **The MERLIN vehicles for outdoor applications**. In G. R. Gerhart, C. M. Shoemaker, and D. W. Gage, editors, *Unmaned Ground Vehicle Technology IV, Proc. of SPIE*, **4715**, pages 43–49, 2002.

[168] K. Schilling and L. Preucil. **Tele-Presence Methods Supporting Cooperation of Robots and Humans**. In *Proc of IEEE International Workshop on Safety, Security and Rescue Robotics*, Kobe, Japan, 2005.

[169] M. Schmidt, K. Ravandoor, O. Kurz, S. Busch, and K. Schilling. **Attitude Determination for the Pico-Satellite UWE-2**. In *Proc. of the 17th World Congress*, 2008.

[170] M. Schneider-Fontan and M.J. Mataric. **Territorial multi-robot task division**. *IEEE Transactions on Robotics and Automation*, **14**(5):815–822, 1998.

[171] N. Shahidi, H. Esmaeilzadeh, M. Abdollahi, and C. Lucas. **Memetic Algorithm Based Path Planning for a Mobile Robot**. *International journal of information technology*, **1**(4):174–177, 2004.

[172] J. Shao, G. Xie, and L. Wang. **Leader-following formation control of multiple mobile vehicles**. *Control Theory and Applications, IET*, **1**(2):545–552, March 2007.

[173] D. M. Stipanović, P. F. Hokayem, M. W. Spong, and D. D. Šiljak. **Cooperative avoidance control for multi-agent systems**. *Journal of Dynamic Systems, Measurement, and Control*, **129**:699–707, 2007.

[174] D.M. Stipanović, R. Teo G. Inalhan, and C.J. Tomlin. **Decentralized overlapping control of a formation of unmanned aerial vehicles**. *Automatica*, **40**(8):1285–1296, 2004.

[175] C. Sultan, S. Seereram, and R. K. Mehra. **Deep Space Formation Flying Spacecraft Path Planning**. *International Journal of Robotics Research*, **26**(4):405–430, 2007.

[176] D. Swaroop and J. K. Hedrick. **String stability of interconnected systems**. *IEEE Transactions on Automatic Control*, **41**(3):349–357, 1996.

[177] D. Swaroop and J. K. Hedrick. **Constant spacing strategies for platooning in automated highway systems**. *ASME Journal of Dynamic Systems, Measurement and Control*, **121**(3):462–470, 1999.

[178] H.G. Tanner, S.G. Loizou, and K.J. Kyriakopoulos. **Nonholonomic navigation and control of cooperating mobile manipulators**. *IEEE Transactions on Robotics and Automation.*, **19**(1):53–64, 2003.

[179] H.G. Tanner, G.J. Pappas, and V. Kumar. **Leader-to-Formation Stability**. *IEEE Transactions on Robotics and Automation*, **20**(3):443–455, June 2004.

[180] P.G. Tzionas, A. Thanailakis, and P.G. Tsalides. **Collision-free path planning for a diamond-shaped robot usingtwo-dimensional cellular automata**. *IEEE Transactions on Robotics and Automation*, **13**(2):237–250, 1997.

[181] L. Vig and J.A. Adams. **Multi-Robot Coalition Formation**. *IEEE Transactions on Robotics*, **22**(4):637–649, August 2006.

[182] I. A. Wagner, Y. Altshuler, V. Yanovski, and A. M. Bruckstein. **Cooperative Cleaners: A Study in Ant Robotics**. *The International Journal of Robotics Research*, **27**(1):127–151, 2008.

[183] I.A. Wagner, M. Lindenbaum, and A.M. Bruckstein. **Distributed covering by ant-robots using evaporating traces**. *IEEE Transactions on Robotics and Automation*, **15**(5):918–933, 1999.

[184] P. K. C. Wang, F. Y. Hadaegh, and K. Lau. **Synchronized Formation Rotation and Attitude Control of Multiple Free-Flying Spacecraft**. *Journal of Guidance, Control, and Dynamics*, **22**(1):28–35, 1999.

[185] A.R. Willms and S.X. Yang. **An efficient dynamic system for real-time robot-path planning**. *IEEE Transactions on Systems, Man, and Cybernetics*, **36**(4):755–766, 2006.

[186] A.R. Willms and S.X. Yang. **Real-Time Robot Path Planning via a Distance-Propagating Dynamic System with Obstacle Clearance**. *IEEE Transactions on Systems, Man, and Cybernetics*, **38**(3):884–893, 2008.

[187] W. Wu, H. Chen, and P. Y. Woo. **Time optimal path planning for a wheeled mobile robot**. *Journal of Robotic Systems*, **17**(11):585–591, 2000.

[188] A.M. Bruckstein Y. Altshuler and I.A. Wagner. **Shape Factor‘s Effect on a Dynamic Cleaners Swarm**. In *Proc. of International Conference on Informatics in Control, Automation and Robotics*, 2006.

[189] S.X. Yang and C. Luo. **A neural network approach to complete coverage path planning**. *IEEE Transactions on Systems, Man, and Cybernetics*, **34**(1):718–724, 2004.

[190] T. T. Yang, Z. Y. Liu, H. Chen, and R. Pei. **Formation Control and Obstacle Avoidance for Multiple Mobile Robots**. *Automatica*, **34**(5):588–593, 2008.

[191] J. Ye and R. Qu. **Fairing of parametric cubic splines**. In *Mathematical and Computer Modelling*, **30**, pages 121–31. Elseviers, 1999.

[192] B.J. Young, R.W. Beard, and J.M. Kelsey. **A control scheme for improving multi-vehicle formation maneuvers**. In *Proc. of American Control Conference*, 2002.

[193] Xiaobu Yuan and S.X. Yang. **Multirobot-Based Nanoassembly Planning with Automated Path Generation**. *IEEE/ASME Transactions on Mechatronics*, **12**(3):352 – 356, June 2007.

[194] S. Yue, D. Henrich, W. L. Xu, and S. K. Tso. **Point-to-Point trajectory planning of flexible redundant robot manipulators using genetic algorithms**. *Robotica*, **20**(3):269–280, 2002.

[195] Z. Zlatev. *Computer Treatment of Large Air Pollution Models*. Springer Netherland, 1995.

[196] R. Zlot and A. Stentz. **Complex Task Allocation For Multiple Robots**. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA 2005)*, April 2005.

[197] Y. Zou, P. Pagilla, and R. Ratliff. **Distributed Formation Flight Control Using Constraint Forces**. *Journal of Guidance, Control, and Dynamics*, **32**(1):112–120, 2009.