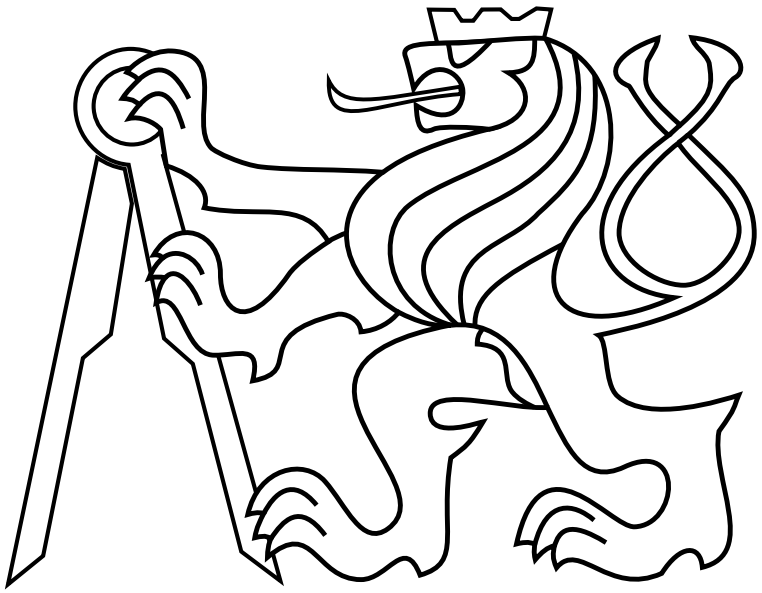
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

DIPLOMA THESIS



Bc. Vojtěch Spurný

Complex Maneuvers of Heterogeneous Formations of Ground and Aerial Robots

Department of Cybernetics

Thesis supervisor: **Ing. Martin Saska, Dr. rer. nat.**

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne .....

.....
Podpis autora práce

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

DIPLOMA THESIS ASSIGNMENT

Student: Bc. Vojtěch S p u r n ý

Study programme: Cybernetics and Robotics

Specialisation: Robotics

Title of Diploma Thesis: Complex Maneuvers of Heterogeneous Formations of Ground and Aerial Robots

Guidelines:

The main purpose of this thesis is to extend system for control and stabilization of heterogeneous teams of ground robots and helicopters with the possibility of making complex maneuvers in environments with obstacles.
Student designs, implements and verifies Model Predictive Control (MPC) based method that enables trajectory planning into a desired location. Obtained solution must satisfy constraints given by the system of relative localization carried onboard of unmanned aerial vehicles for formation stabilization. Furthermore, student implements an initialization of the MPC algorithm based on Rapidly-Exploring Random Tree (RRT). Particular algorithms and the entire system will be verified in various numerical experiments in environments of different complexity. The behavior of the system will be statistically analyzed in dynamic and partly unknown environment. Influence of different setting of the algorithms on the quality of obtained trajectories and time complexity will be evaluated.

Bibliography/Sources:

[1] Saska, M. - Vonásek, V. - Krajník, T. - Přeučil, L. Coordination and navigation of heterogeneous MAV–UGV formations localized by a 'hawk-eye'-like approach under a model predictive control scheme. International Journal of Robotics Research, Volume 33, Issue 10, pp 1393-1412, July 2014.
[2] Saska, M. - Mejía, J. S. - Stipanović, D. M. - Vonásek, V. - Schilling, K. - Přeučil, L.: Control and navigation in manoeuvres of formations of unmanned mobile vehicles, European Journal of Control, Volume 19, Issue 2, Pages 157-171, March 2013.
[3] LaValle, S. M.. Planning Algorithms. Cambridge University Press, Cambridge, U.K., 2006.
[4] Stipanovic, D.M. – Hokayem, P.F. – Spong, M.W., et al. Cooperative avoidance control for multi-agent systems. Journal of Dynamic Systems, Measurement, and Control,129:699–707, 2007.

Diploma Thesis Supervisor: Ing. Martin Saska, Dr. rer. nat.

Valid until: the end of the summer semester of academic year 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 23, 2015

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Bc. Vojtěch S p u r n ý

Studijní program: Kybernetika a robotika (magisterský)

Obor: Robotika

Název tématu: Komplexní manévry heterogenních formací pozemních a vzdušných robotů

Pokyny pro vypracování:

Hlavním cílem práce je rozšířit systém pro řízení a stabilizaci heterogenních týmů pozemních robotů a helikoptér o možnost provádět autonomně manévry ve složitém prostředí s překážkami.
Student navrhne, implementuje a experimentálně verifikuje metodu prediktivního řízení (MPC) umožňující plánovat komplexní trajektorie (opakovanou změnu směru pohybu formace) do požadované pozice formace. Získané řešení bude splňovat omezení dané systémem vizuální relativní lokalizace instalované na palubě bezpilotních helikoptér.
Student dále naimplementuje metodu pro inicializaci MPC algoritmu založenou na rychle rostoucích náhodných stromech. Jednotlivé algoritmy i systém jako celek budou verifikovány sadou numerických experimentů v prostředích různé složitosti. Bude statisticky analyzováno chování systému v dynamicky se měnícím a částečně neznámém prostředí a bude porovnán vliv různých nastavení algoritmů na kvalitu získané trajektorie a výpočetní náročnost.

Seznam odborné literatury:

[1] Saska, M. - Vonásek, V. - Krajník, T. - Přeučil, L. Coordination and navigation of heterogeneous MAV–UGV formations localized by a 'hawk-eye'-like approach under a model predictive control scheme. International Journal of Robotics Research, Volume 33, Issue 10, pp 1393-1412, July 2014.
[2] Saska, M. - Mejía, J. S. - Stipanović, D. M. - Vonásek, V. - Schilling, K. - Přeučil, L.: Control and navigation in manoeuvres of formations of unmanned mobile vehicles, European Journal of Control, Volume 19, Issue 2, Pages 157-171, March 2013.
[3] LaValle, S. M.. Planning Algorithms. Cambridge University Press, Cambridge, U.K., 2006.
[4] Stipanovic, D.M. – Hokayem, P.F. – Spong, M.W., et al. Cooperative avoidance control for multi-agent systems. Journal of Dynamic Systems, Measurement, and Control,129:699–707, 2007.

Vedoucí diplomové práce: Ing. Martin Saska, Dr. rer. nat.

Platnost zadání: do konce letního semestru 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 23. 1. 2015

Abstract

This diploma thesis deals with control and trajectory planning of heterogeneous teams of ground robots and helicopters, in which a system of relative localization carried onboard of unmanned aerial vehicles is used for formation stabilization. The proposed planning method is based on the model predictive control technique. The main goal of this diploma thesis is design, implementation and experimental verification of two extension of the formation driving methodology being developed within Department of Cybernetics, FEE, CTU. The first extension of the system enables usage of rapidly-exploring random tree method for initialization of the model predictive control algorithm. The second extension of the system enables possibility of making complex maneuvers in environments with obstacles. The purpose of both extensions is discussed, and particular algorithms and the entire system are verified in various numerical experiments in environments of different complexity. Furthermore, the behaviour of the system is statistically analyzed in dynamic and partially unknown environment. Influence of different settings of the algorithm on the quality of obtained trajectories and time complexity is evaluated as well.

keywords:
[multi-robot formation, trajectory planning, model predictive control, complex maneuvers, dynamic environment, rapidly-exploring random tree]

Abstrakt

Tato diplomová práce je zaměřena na vývoj metod pro řízení a plánování trajektorie heterogenních týmů pozemních robotů a helikoptér, ve kterých je používán systém vizuální relativní lokalizace instalovaný na palubě bezpilotních helikoptér. Navrhovaná metodologie je zaměřena na použití prediktivního řízení (MPC). Hlavním cílem této práce je navrhnout, implementovat a experimentálně ověřit dvě rozšíření systému pro řízení formací robotů vyvíjeného na katedře kybernetiky, ČVUT v Praze. První rozšíření systému umožní inicializaci MPC algoritmu založenou na rychle rostoucích náhodných stromech. Druhé rozšíření systému umožní provádět komplexní manévry (opakovanou změnu směru pohybu formace) ve složitém prostředí s překážkami. Důvod obou rozšíření je diskutován a jednotlivé algoritmy i systém jako celek verifikovány sadou numerických experimentů v prostředích různé složitosti. Dále je staticky analyzováno chování systému v dynamicky se měnícím a částečně neznámém prostředí a je porovnán vliv různých nastavení algoritmu na kvalitu získané trajektorie a výpočetní náročnost.

klíčová slova:
[multi-robotické formace, plánování trajektorie, metoda prediktivního řízení, komplexní manévry, dynamické prostředí, rychle rostoucí náhodné stromy]

Acknowledgements

I would like to thank to Ing. Martin Saska, Dr. rer. nat. for his supervision, relevant discussions, and for his patience with text correcting. My thanks also belong to my family for their support.

Contents

List of Figures

1 Introduction

With advancing technological progress, mobile robots become more spread than ever and also multi-robot systems become frequently used. One of the currently investigated problems dealing with multi-robot systems is coordination of formations of robots and their motion planning into a target zone. In this task, a group of robots has to find and follow a collision free trajectory while it maintains a desired shape of the formation. For solving this problem positions of the members of the formation need to be known. Unfortunately, the GPS module that is well known as a way how to get the global position cannot be always used (e.g. in indoor application) and its precision is insufficient for control of compact formations. One of the possible ways how to locate the positions of the members of the formation is to use relative visual localization. This diploma thesis deals with a system for control and trajectory planning of heterogeneous teams of ground robots and helicopters where a system of relative localization carried onboard of unmanned aerial vehicles is used for formation stabilization.

The system of control and trajectory planning for the formation presented in this thesis is based on the model predictive control method that is described in [23] and [19]. This system tries to find a solution that satisfies constraints of movement of the formation and also constraints of direct visibility between the members of the formation that is required for relative visual localization. The main purpose of this diploma thesis is design, implementation and experimental verification of two extensions of the mentioned system.

The first extension of the system will enable usage of rapidly-exploring random tree method for initialization of the model predictive control algorithm. The second extension of the system will enable performing complex maneuvers in environments with obstacles. The purpose of both extensions will be discussed and particular algorithms and the entire system will be verified in various numerical experiments in environments of different complexity. Furthermore, the behaviour of the system will be statistically analyzed in dynamic and partially unknown environment. Influence of different settings of the algorithm on the quality of obtained trajectories and time complexity will be evaluated.

2 State of the art

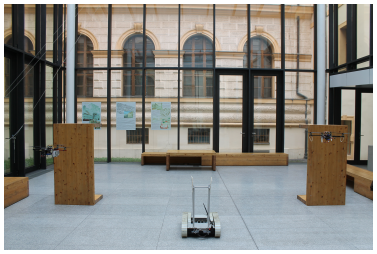
Several methodologies and formulations of controlling multi-robot systems to reach the target region have been proposed. According the literature, these methods are categorised as follow:

- the behaviour-based approach
- the leader-follower approach
- the virtual structure approach.

In the behaviour-based approach, a local behaviour is assigned to each individual robot [1]. The main advantage of this approach is in decentralisation of the problem of the controlling multi-robot system. This approach was inspired by behaviour of animals in nature (e.g. school of fish, group of birds). Based on their perception animals are trying to minimize the chance of being detected by predators or to gather food more efficiently. Craig Reynolds developed a simple egocentric[1] behavioural model for group of birds[11]. Each member of the group follows this model and its behaviour consists of several separate components including: collision avoidance (avoidance of collision with others robots), velocity matching and flock centring.

(a) The behaviour-based approach - swarm of robots (Source: wikipedia.org)

(b) The virtual structure approach - spacecrafts (Source: http://www.supernexus.com/riftspace/LIBERTY.htm)



(c) The leader-follower approach - formation of one ground and 2 aerial vehicles

Figure 1: Examples of using approaches to control a multi-robot system.

In the leader-follower approach, one or more of the robots in the group are typically assigned as leader, and the rest of members become followers[23]. Behaviour of the leader specifies behaviour of the whole group. The advantage is that the global trajectory is computed only for the leader and not for all robots in the group. The control inputs for followers are computed from the leader trajectory with the aim to follow this trajectory.

[1]Egocentrism is a behaviour in which each individual predominantly focuses on himself rather than on others.

The disadvantage of this approach lies in absence of feedback from behaviour of followers to the leader. So if follower fails or temporarily slows down, the leader will not react to it.

In the virtual structure approach, the entire robot formation is considered as a single structure [9]. The formation is not created from leaders and followers (there is no hierarchy in the formation). In this method the desired trajectory for the entire structure is computed and this trajectory is used for controlling the individual robots in the formation.

3 Preliminaries

An introduction of methods used in this thesis is given in this chapter.

3.1 Leader-follower approach

This thesis is built on achievements presented in [15] and [19]. These works are based on the leader-follower approach. It means that the whole trajectory to the desired area is computed only for the leader and trajectory of robots in a formation (followers) is defined relatively to the leader trajectory. In the leader-follower concept, virtual leader, which is located in the front and simultaneously in the axis of the formation, is used.

This approach enables us to meet the requirements of different types of robots as members of the formation. It was also chosen because it satisfies requirements of the direct visibility between team members, which is used for localization of the robots. During the movement it cannot happen that this visibility is lost because this would lead to destruction of coherence of the formation. An example of an area which can not be blocked by an external object is shown in Figure 2.

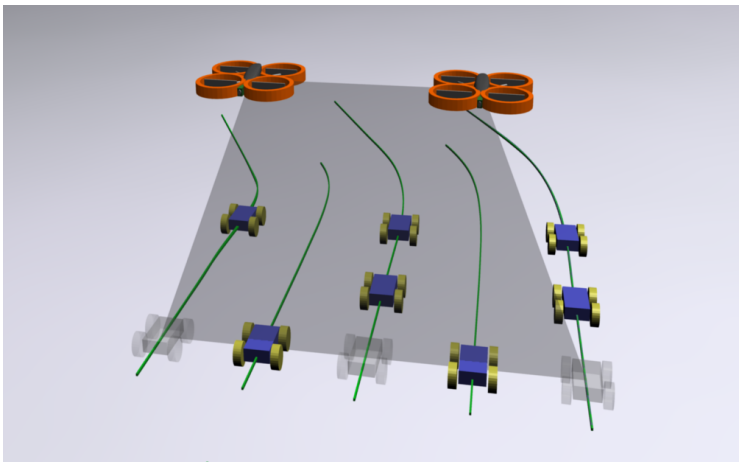


Figure 2: The example of projection of the zone in which the direct visibility has to be satisfied into a plane of the virtual leader.

3.2 Configuration space of robots

Let us define fundamental terms required for the method description. The configuration space of the robots denoted as $\mathbb{C}-space$, represents set of all possible configurations of the

robots in the environment[2] $\mathbb{W}$. A search in this $\mathbb{C}-space$ must be conducted for finding a solution of motion planning problem[7].

We suppose that the environment includes obstacles $\mathbb{O}_{obs}$ that divide the $\mathbb{C}-space$ into two subsets. The first subset of $\mathbb{C}-space$ is obstacle configuration space $\mathbb{C}_{obst} \subseteq \mathbb{C}$. It is a set of all configuration $\vec{\psi_j}$ of the j-th robot, in which the robot intersects with obstacle and can be defined as

$$\mathbb{C}_{obst} = \{\vec{\psi_j} \in \mathbb{C} | \mathbb{A}(\vec{\psi_j}) \cap \mathbb{O}_{obs} \neq \emptyset\}, \tag{1}$$

where $\mathbb{A}(\vec{\psi_j})$ represents body of the robot in configuration $\vec{\psi_j}$. The second subset of $\mathbb{C}-space$ is free configuration space $\mathbb{C}_{free}$, that represents space, where the robot is without collision with an obstacle. It can be defined as

$$\mathbb{C}_{free} = \mathbb{C} \setminus \mathbb{C}_{obs}. \tag{2}$$

Configuration of the virtual leader L and n_r numbers of the followers in this work is described by vector $\vec{\psi_j} = (x_j, y_j, z_j, phi_j) \in \mathbb{C}$, where $j \in \{L, 1, \ldots, n_r\}$. So configuration of the j-th robot is set by its position in Cartesian coordinates $\bar{p}_j = (x_j, y_j, z_j)$ and by its heading phi_j.

3.3 Kinematic model

This approach requires that the kinematic model must be suitable for each robot in the formation. It must be suitable for both ground and aerial vehicles. The solution is to use the extended model for car-like robot[19].

The standard kinematic model for car-like robot was created for ground vehicles and uses two parameters for describing robot movement, velocity v and curvature K. The curvature K is defined as

$$K(t) = \frac{\tan(\theta(t))}{d}, \tag{3}$$

where d represents a distance between the front and rear wheels, and θ is angle of the front pair of wheels (see Figure 3). This model extended by another parameter w, which represents ascent velocity, can be used as the kinematic model for both the mentioned types of robots. So the kinematic model of j-th robot is described by following equations

$$\begin{aligned} \dot{x}_j(t) &= v_j(t)\cos(\varphi_j(t)), \\ \dot{y}_j(t) &= v_j(t)\sin(\varphi_j(t)), \\ \dot{z}_j(t) &= w_j(t), \\ \dot{\varphi}_j(t) &= K_j(t)v_j(t), \end{aligned} \tag{4}$$

[2]In this work a 3-dimensional space is considered.

and ascent velocity w is limited to zero for the ground vehicles. Three parameters of the kinematic model represent control inputs for the robot and can be represented by a vector $\vec{u_j}(t) = (v_j(t), w_j(t), K_j(t))$.

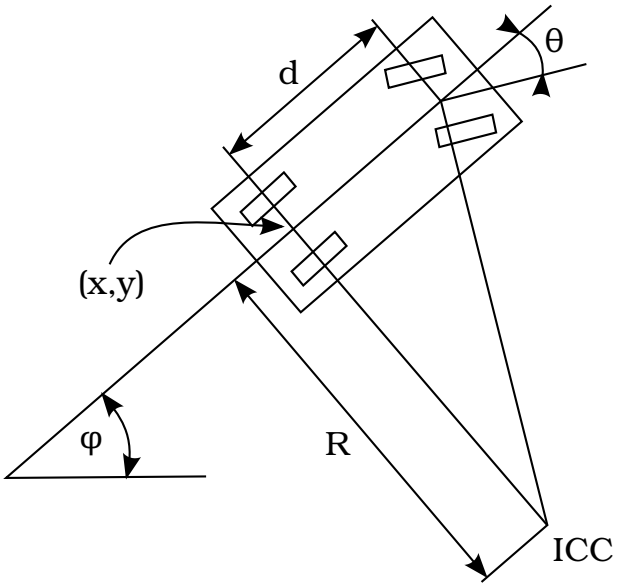


Figure 3: Car-like model

Computed trajectory from initial state to target state is described by a sequence of vectors $\vec{u_1}, \ldots, \vec{u_{end-1}}$, which contains control inputs, and by a sequence of the durations of control inputs $\Delta t_1, ..., \Delta t_{end-1}$. For the time interval $\Delta t_k = t_{k+1} - t_k$, where $k \in \{1, \ldots, end-1\}$, the control inputs are constant (from here index k is used instead of t_k). The model for transition points, where the controls inputs change can be deduced by integration of the kinematic model (eq. (4)) over interval $[t_k, t_{k+1}]$:

$$
\begin{aligned}
x_j(k+1) &= \begin{cases} x_j(k) + \frac{1}{K_j(k+1)} [\sin(\varphi_j(k) + \\ K_j(k+1) v_j(k+1) \Delta t(k+1)) - \\ \sin(\varphi_j(k))], \text{if } K_j(k+1) \neq 0; \\ x_j(k) + v_j(k+1) \cos(\varphi_j(k)) \Delta t(k+1), \\ \text{if } K_j(k+1) = 0, \end{cases} \\
y_j(k+1) &= \begin{cases} y_j(k) - \frac{1}{K_j(k+1)} [\cos(\varphi_j(k) + \\ K_j(k+1) v_j(k+1) \Delta t(k+1)) - \\ \cos(\varphi_j(k))], \text{if } K_j(k+1) \neq 0; \\ y_j(k) + v_j(k+1) \sin(\varphi_j(k)) \Delta t(k+1), \\ \text{if } K_j(k+1) = 0, \end{cases} \\
z_j(k+1) &= z_j(k) + w_j(k+1) \Delta t(k+1) \\
\varphi_j(k+1) &= \varphi_j(k) + K_j(k+1) v_j(k+1) \Delta t(k+1),
\end{aligned}
\tag{5}
$$

where $x_j(k), y_j(k), z_j(k)$ and $phi_j(k)$ are configuration of the j-th robot in transition point with index k. $v_j(k+1), w_j(k+1)$ and $K_j(k+1)$ are control inputs from vector $\vec{u_j}(k+1) =$

$\vec{u_j}(t_k, t_{k+1} - t_k)$ which are used at t_k and for $\Delta t_{k+1} = t_{k+1} - t_k$.

3.4 Robot's constraints

Every robot has limitations of its movement given by the vehicle's mechanical capabilities. This is presented by limitation of control inputs as follows:

$$\begin{aligned} v_{min,j} \leq v_j(k) \leq v_{max,j}, \\ |K_j(k)| \leq K_{max,j}, \end{aligned} \tag{6}$$

and moreover for aerial vehicles

$$w_{min,j} \leq w_j(k) \leq w_{max,j}.$$

As was mentioned before ground vehicles have limited ascent velocity w_j to zero.

Furthermore, we must define two boundary r_a and r_d. Robots should respond only to obstacles that are closer than avoidance boundary r_d to its positions. It is not allowed for robots to come to obstacles closer than avoidance boundary r_a during the movement. It must be satisfied $r_a < r_d$.

3.5 Formation driving

Curvilinear coordinates p, q, h are used for description of the relative states of the followers to the virtual leader states. Conversion from these curvilinear coordinates to Cartesian coordinates for j-th member of the formation can be described by the following equations

$$\begin{aligned} x_j(t) &= x_L(t_{p_j}) - q_j \sin(\varphi_L(t_{p_j})), \\ y_j(t) &= y_L(t_{p_j}) + q_j \cos(\varphi_L(t_{p_j})), \\ z_j(t) &= z_L(t_{p_j}) + h_j, \\ \varphi_j(t) &= \varphi_L(t_{p_j}), \end{aligned} \tag{7}$$

where $\psi_L(t_{p_j}) = \big(x_L(t_{p_j}), y_L(t_{p_j}), z_L(t_{p_j}), phi_L(t_{p_j})\big)$ is state of the virtual leader in time t_{p_j}. So the state of j-th robot is determined from the virtual leader state $\psi_L(t_{p_j})$, which represents state of the virtual leader in the past when it was in distance p_j from its actual state ψ_L. Follower's state is then set by distance q_j in state $\psi_L(t_{p_j(t)})$ which is perpendicular to the virtual leader trajectory and by distance h_j above state $\psi_L(t_{p_j(t)})$. An example of shape of the formation described in curvilinear coordinates is shown in Figure 4.

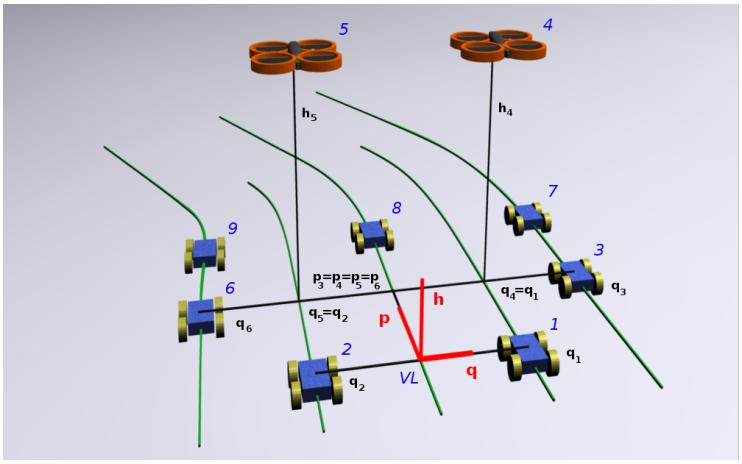


Figure 4: An example of shape of the formation described in curvilinear coordinates.

3.6 Constraints of virtual leader

As was mentioned each robot in the formation has its own limitations of control inputs, which depends on the robot design. These limitations of the members of the formation and with their positions in the formation have to be considered in trajectory planning for the virtual leader. If the formation is turning, each robot has to move with different value of curvature and velocity. All these limitations are included into constraints of the virtual leader movement by following equations

$$\begin{aligned} K_{max,L} &= \min_{i=1,\dots,n_r} \left(\frac{K_{max,i}}{1 + q_i K_{max,i}} \right), \\ K_{min,L} &= \max_{i=1,\dots,n_r} \left(\frac{-K_{max,i}}{1 - q_i K_{max,i}} \right), \\ v_{max,L}(t) &= \min_{i=1,\dots,n_r} \left(\frac{v_{max,i}}{1 + q_i K_L(t)} \right), \\ v_{min,L}(t) &= \max_{i=1,\dots,n_r} \left(\frac{v_{min,i}}{1 + q_i K_L(t)} \right), \\ w_{max,L} &= \min_{i=1,\dots,n_r} \left(w_{max,i} \right), \\ w_{min,L} &= \max_{i=1,\dots,n_r} \left(w_{min,i} \right). \end{aligned} \tag{8}$$

The virtual leader trajectory must be collision free for the virtual leader but also for the members of the formation. So the shape of the formation must be included into planning.

That is done by following equation

$$r_{d,L}(t) = r_s + \max_{i=1,\dots,n_r} |q_i(t)|\,,$$
$$r_{a,L}(t) = r_a + \max_{i=1,\dots,n_r} |q_i(t)|\,, \tag{9}$$

where $r_{d,L}(t)$ is detection zone and $r_{a,L}(t)$ is avoidance zone of the virtual leader.

3.7 Model predictive control

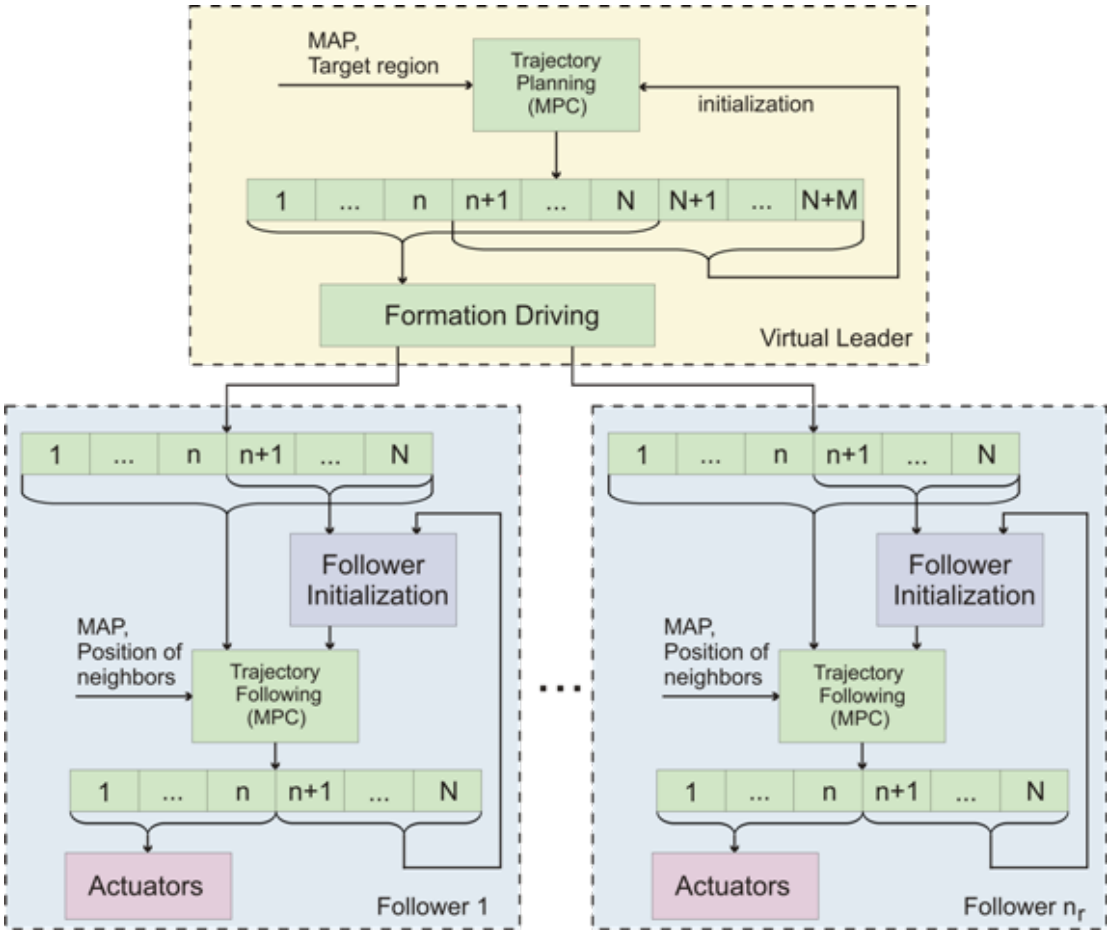


Figure 5: Illustration of the model predictive control approach.[19]

The Model Predictive Control (MPC) is an optimization method for stabilization of nonlinear systems over a finite time horizon. This method is often used in industry but can be also used in mobile robotics. In this thesis, the MPC is used for trajectory planning to

desired area and simultaneously for computing control inputs feasible for the members of the formation.

The standard model predictive control solves a finite horizon optimization control problem starting from current state with constant sampling time Δt between N transition points. This approach uses actual perception of the world for solving. Only first n of the control steps from the result are used and the optimization problem is solved again from the newly achieved state. The perception of the world may change during the movement of the formation and this method allows the robots to react to these changes.

Illustration of the proposed method is shown in Figure 5. This method is divided into 2 parts. In the *Virtual leader* part, the *Trajectory Planning block* provides control inputs for the virtual leader and a complete trajectory to the target zone feasible for the entire formation. For this task, the standard model of the predictive control was extended by a horizon in which the sampling time is variable between M transition points. The entire horizon is therefore formed from two horizons. The first horizon $\langle t_0, t_0 + N\Delta t\rangle$ is denoted as the *control horizon* and the second $\langle t_0 + N\Delta t, t_0 + (N + M)\Delta t\rangle$ as the *planning horizon*. The *control horizon* with the constant sampling time is used to obtain immediate control and the *planning horizon* where lengths of time intervals between transition points are also variables taking part in the planning problem. The resulting trajectory is used as an input for the second main block, which transforms the plan to the desired trajectory for the followers (using eq. (7)), and for re-initialization of the optimization in the next planning step.

In the *Follower* block, the *Trajectory Following* module is responsible for computing trajectory (input controls), which is feasible (avoid collisions with the obstacles and the other members in the formation), and which is as close as possible to the desired trajectory provided by the virtual leader. Only the first n of the computed control inputs are used according to the model predictive control.

The solution of the optimization control problem for the model predictive control is formed by minimizing the cost function. The sequential quadratic programming is a powerful process for solving this problem with nonlinear constraints. Unfortunately, its disadvantage is missing ability to overcome local extrema in the cost function. This problem will be described in section 5, where the initialization of optimization will be solved.

4 Implementation details

In this section the proposed approach for solving optimization problem of trajectory planning will be described. Detailed description of planning will be presented in subsection 4.1 for the virtual leader and in subsection 4.2 for the followers.

4.1 Virtual leader's trajectory planning

As was mentioned in section 3.7, trajectory planning presented in this thesis is defined as optimization problem over two time horizons (*control horizon* and *planning horizon*). The trajectory is coded into the optimization vector Ω_L for the purpose of the MPC method, with N+M elements, where N is length of *control horizon* and M is length of *planning horizon*. Each element contains control inputs and time that represents the duration of every control input. The sampling time is constant for the first N elements and for the rest of elements the sampling time is variable.

The trajectory planning and obstacle avoidance problem can be then transformed into the minimization of cost function $\lambda_L(\cdot)$ subject to sets of inequality constraints $g_N(\cdot), g_M(\cdot), g_{r_{a,L}}(\cdot)$, and $g_{S_F}(\cdot)$, that is

$$\begin{aligned} \min \lambda_L(\Omega_L), \text{s.t. } & g_N(k) \leq 0, \forall k \in \{1, \ldots, N\}, \\ & g_M(k) \leq 0, \forall k \in \{N+1, \ldots, N+M\}, \\ & g_{r_{a,L}}(\Omega_L, \mathcal{O}_{obs}) \leq 0, \\ & g_{S_F}(\psi_L(N+M)) \leq 0. \end{aligned} \tag{10}$$

4.1.1 Objective function

The cost function is given by

$$\begin{aligned} \lambda_L(\Omega_L) = & \alpha \left(\sum_{k=N+1}^{N+M} \Delta t(k) \right) + \beta \min \left\{ 0, \frac{dist(\Omega_L, \mathcal{O}_{obs}) - r_{d,L}}{dist(\Omega_L, \mathcal{O}_{obs}) - r_{a,L}} \right\}^2 \\ & + \gamma \sum_{k=1}^{N+M} |v(k) - \overline{v}| + \eta \sum_{k=1}^{N+M} |w(k) - \overline{w}| + \mu \sum_{k=1}^{N+M} \left| K(k) - \overline{K} \right| \\ & + \xi \quad \|(\psi_L(N+M), C_{S_F})\| \end{aligned} \tag{11}$$

as a weighted sum of several parts. The first part symbolizes the time required to reach the target area. The duration time of each control input is constant for *control horizon* but for *planning horizon* variable. So it does not make sense to include the time in *control horizon* into formula. The second part of $\lambda_L(\cdot)$ represents influence of obstacles close to

the planned trajectory. Function $dist(\Omega_L, \mathcal{O}_{obs})$ provides minimal Euclidean distance from all obstacles $\mathcal{O}_{obs}$ to the leader planned trajectory $\psi_L(.)$. The cost of this part is zero if distance $dist(\Omega_L, \mathcal{O}_{obs})$ is bigger than $r_{d,L}$. The obstacles that are farther than $r_{d,L}$ from $\psi_L(.)$ do not have influence on the cost function. The next three parts of $\lambda L(\cdot)$ are added because control inputs of trajectory without big changes are preferred. Last part of the formula is included for improvement of convergence to desired solution. The desired region is represented as a circle with radius r_{S_F} and center C_{S_F}.

By setting constants $\alpha, \beta, \gamma, \eta, \mu$ a ξ user can set which trajectory is preferred. For example increasing parameter β results in longer trajectories with larger distances from obstacles.

4.1.2 Constraint function

Inequality constraints $g_N(\cdot)$ and $g_M(\cdot)$ of the cost function represent limitations of the virtual leader movement (eq. (8)). Inequality constraint $g_{r_{a,L}}(\cdot)$ that characterizes safety regions around the trajectory is defined as

$$g_{r_{a,L}}(\Omega_L) := r_{a,L} - dist(\Omega_L, \mathcal{O}_{obs}). \tag{12}$$

If inequality constraint $g_{r_{a,L}}(\cdot)$ is satisfied, the trajectory cannot lead to collision with an actually known external objects.

Last term $g_{S_F}(\psi_L(N+M))$ is stability constraint ensuring that the virtual leader trajectory ψ_L will lead to the desired region. The stability constrain is given by

$$g_{S_F}(\psi_L(N+M)) := \|(\psi_L(N+M), C_{S_F})\| - r_{S_F}. \tag{13}$$

The trajectory that satisfies all these constraint is considered as a feasible collision free trajectory from actual state of a robot to a desired region.

4.2 Trajectory following for followers

The trajectory of the virtual leader, which is result of the previous section, will be used according to the leader-follower concept as an input of trajectory following for followers. The virtual leader trajectory must be transformed for every follower of the formation using equation (7). Unfortunately, this plan can be used only for followers with $p = 0$ and for followers, with $p \neq 0$, another approach must be used. The idea of this approach is to use history of the virtual leader movement together with the actual computed trajectory. The states of the virtual leader $\psi_L(t_{p_j(t)})$ where the leader used to be in the past and that are behind its actual computed states in distance p_j need to be determined for each

follower. The states $\psi_L(t_{p_j(t)})$ are then used for computing the desired states of followers using equation (7).

The proposed approach has a problem to compute the desired position for j-th follower when state $\psi_L(t_{p_j(t)})$ do not exist. It happens when the virtual leader does not travel the distance p_j. In this case, state $\psi_L(t_{p_j(t)})$ is computed as remaining distance $l_r(t)$ behind the start state of the virtual leader. The remaining distance is computed as

$$l_r(t) = l_t(t) - p_j, \tag{14}$$

where $l_t(t)$ is leader travelled distance.

In a similar way to the leader planning in section 4.1, the trajectory is coded into optimization vector Ω_j for the purpose of the MPC method. Vector Ω_j is created from N elements, where N is length of *control horizon*. Every element contains control inputs and constant time.

The trajectory tracking for j-th follower, where $j \in (1, \ldots, n_r)$, can be transformed to the minimization of cost function $\lambda_j(\cdot)$ subject to sets of inequality constraints $g_N(\cdot), g_{r_a}(\cdot)$, and $g_{r_{a,j}}(\cdot)$, that is

$$\begin{aligned} \min \lambda_j(\Omega_j), \text{s.t. } g(k) \leq 0, \forall k \in \{1, \ldots, N\}, \\ g_{r_a}(\Omega_j, \mathcal{O}_{obs}) \leq 0, \\ g_{r_{a,j}}(\Omega_j, \Omega_{n_n}) \leq 0. \end{aligned} \tag{15}$$

4.2.1 Objective function

Similarly as for the virtual leader, the cost function of j-th follower is created as a weighted sum of several parts as

$$\begin{aligned} \lambda_j(\Omega_j) = \alpha \sum_{k=1}^{N} \|(\bar{p}_{D,j}(k) - \bar{p}_j(k))\|^2 + \beta \min \left\{ 0, \frac{dist(\Omega_j, \mathcal{O}_{obs}) - r_d}{dist(\Omega_j, \mathcal{O}_{obs}) - r_a} \right\}^2 \\ + \tau \sum_{k=1}^{N+M} |v(k) - \overline{v}| + \eta \sum_{k=1}^{N+M} |w(k) - \overline{w}| + \mu \sum_{k=1}^{N+M} \left| K(k) - \overline{K} \right| \\ + \gamma \min \left\{ 0, \frac{dist(\Omega_j, \Omega_{n_n}) - r_d}{dist(\Omega_j, \Omega_{n_n}) - r_a} \right\}^2. \end{aligned} \tag{16}$$

The first part symbolizes deviation of computed positions $\bar{p}_j$ from desired positions $\bar{p}_{D,j}(k)$, where $k \in (1, \ldots, N)$. The proposed approach, presented in section 4.1, ensures that the virtual leader trajectory is collision free, but virtual leader is located in the front of the formation and an external object can be detected behind its position. So, it is not sure that precomputed trajectory for followers will be collision free. This is reason why the

cost function $\lambda_j(\cdot)$ has a second part that represents influence of obstacles close to the planned trajectory. Meaning of this part is the same as the second term of eq. (11). It is not preferred if control inputs (forward velocity $v_j(k)$, ascent velocity $w_j(k)$ and curvature $K_j(k)$) of robots are changing often. This is ensured by the next three terms of the cost function $\lambda_j(\cdot)$. We also can not expect that the trajectory will be followed by the robot as is planned. The last part of the cost function has to protect the robot from dangerous behaviour of others members of the formation. Function $dist(\Omega_j, \Omega_{n_n})$ provides minimal Euclidean distance from all planned neighbours positions in the formation Ω_{n_n}, where $n_n = (1, \ldots, j-1, j+1, n_r)$, to the follower planned trajectory $\psi_j(.)$. The idea of the proposed approach is that the desired trajectories are provided to each follower after computing the leader trajectory and the followers start to parallely solve the trajectory tracking. When they complete computing of the trajectories in the actual planning step, they will communicate between themselves by messages about their planned positions. So each follower knows the planned positions of the other members of the formation computed in the previous planning step. We suppose that the planned positions in the actual planning step will be similar to the part of planned positions in the previous planning step. So function $dist(\Omega_j, \Omega_{n_n})$ can be computed as

$$dist(\Omega_j, \Omega_{n_n}) := \min_{i \in n_n} \left(\min_{k \in \{1,\ldots,N-n\}} \|(\bar{p}_j(k) - \bar{p}_i(k+n))\| \right), \tag{17}$$

where $\bar{p}_i(\cdot)$ is planned position of i-th robot in the previous planning step and n represents the number of used control inputs.

4.2.2 Constraint function

Inequality constrains $g(\cdot)$, defined in (15), are identical to inequality constrains $g_N(\cdot)$ in (10), where $k \in (1, \ldots, N)$. Inequality constraints are defined for safety regions around the robots to avoid obstacles as

$$g_{r_a}(\Omega_j) := r_a - dist(\Omega_j, \mathcal{O}_{obs}), \tag{18}$$

and to avoid other members of the formation as

$$g_{r_a}(\Omega_j) := r_a - dist(\Omega_j, \Omega_{n_n}). \tag{19}$$

4.3 Experimental verification

This subsection is focused on experimental verification of the proposed approach. For this purpose, A situation was chosen where formation with 8 members has to move into a target region through an environment with one static obstacle. This obstacle is detected

during the movement. Parameters of the formation are shown in Table 1. Progress of values of the cost function (eq. (11)) used for the virtual leader trajectory planning is displayed in Figure 6. Snapshots from the experiment are shown in Figures 7, 8, and 9.

i	1	2	3	4	5	6	7	8
p_i	0	0	0.55	1.1	1.1	1.65	2.2	2.2
q_i	-0.8	0.8	0	-0.8	0.8	0	-0.8	0.8
h_i	0	0	1	0	0	1	0	0

Table 1: Curvilinear coordinates of the followers in the formation used in the experiment presented in section 4.3.

The shape of the formation is presented in Figure 7. The formation is created from 6 ground vehicles and 2 aerial vehicles. As was mentioned before, the proposed approach is suitable for planning of the movement of the formation to the desired region. Localization of members of the formation is realised using onboard visual system[5]. In this experiment, the ground vehicles are in principle localized by two aerial vehicles. The trajectory obtained in the first planning loop of the presented MPC algorithm is shown in Figure 8, which shows that the found trajectory is collision free (in perspective of knowledge about environment at the beginning). Response of the planning approach to an obstacle detected during the movement at time 36 sec is visualised in Figure 9. The trajectory was correctly changed to avoid the collision with the obstacle.

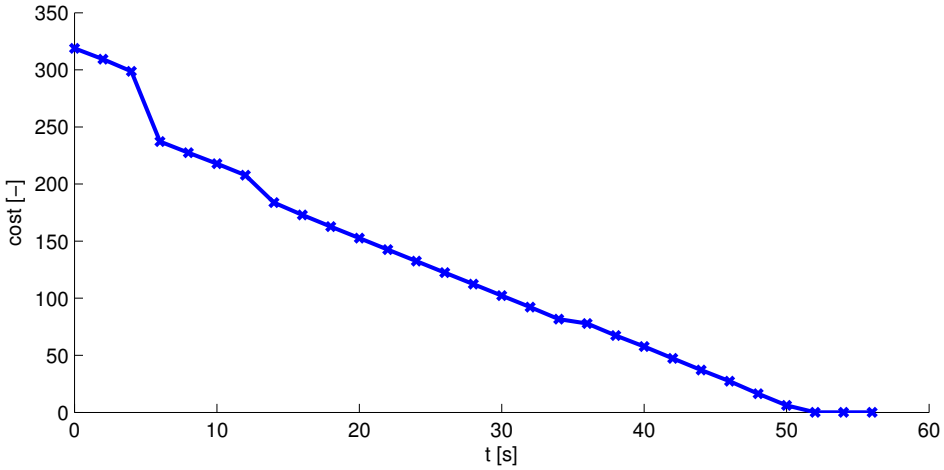


Figure 6: Progress of values of the cost function (eq. (11)) used for the virtual leader trajectory planning for the experiment presented in section 4.3.

Progress of values of the cost function used for the virtual leader trajectory planning is presented in Figure 6. The decrease of values shows convergence of the formation to the desired region. Last three values of the cost function are almost equal to zero, since the *control horizon* already reached the target region and the obstacles do not influence the

cost function. Length of the *control horizon* in this experiment is $N = 5$ and according the MPC concept only a few computed control inputs are used (in this experiment, first 2 control inputs). So if the desired area is reachable in 5 constant time intervals Δt, and in each planning step the first 2 constant time intervals are used, then the desired area will be reached in 3 planning steps.

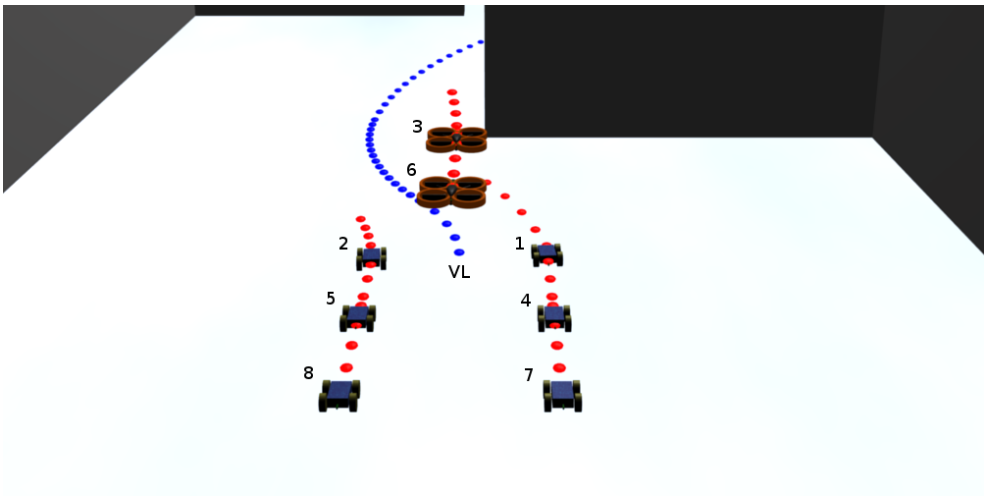


Figure 7: Initial position and shape of the formation for the experiment presented in section 4.3.

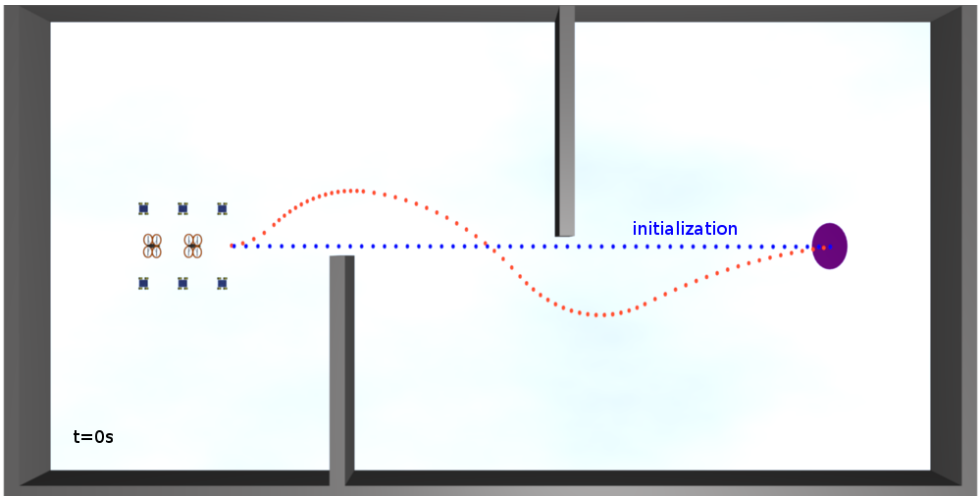
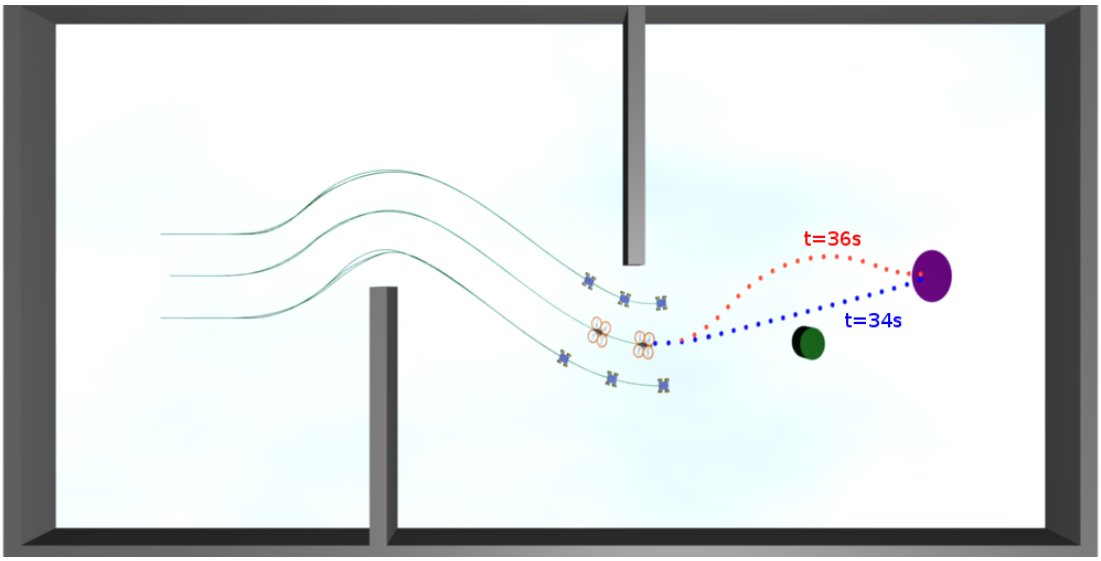
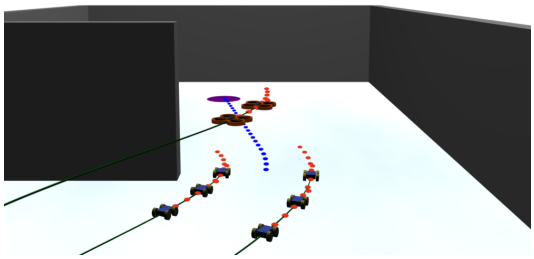


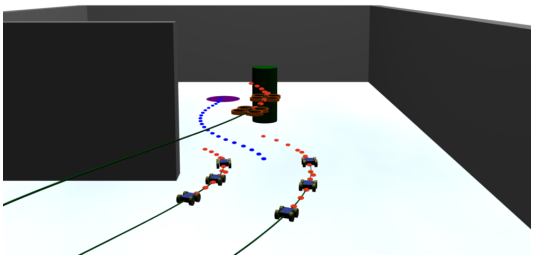
Figure 8: Plotted plan to the target region found by the proposed method in the first planning step.



(a)



(b) t=34s



(c) t=36s

Figure 9: The result of replanning after detecting an obstacle during movement.

5 Initialization of the MPC method

The leader-follower approach and the MPC method for trajectory planning, which were implemented in `C++` within this thesis using literature [19] were introduced in previous sections. From this section, new methodology that is designed, implemented and experimentally verified to extend the existing system will be explained.

5.1 Purpose of the initialization

The scheme of the leader-follower approach for cooperative control of a group of mobile robots is visualized in Figure 5. The MPC method, which was described in section 3.7, is used for solving the optimization control problem of formation movement to the target region. Finding the solution of this problem is achieved by minimizing the cost function with several nonlinear constraints, where satisfying these nonlinear constraints represents feasible and collision free trajectory. The Sequential Quadratic Programming (SQP) is a process used for finding this solution. It is a generalization of the Newton's method and it has disadvantage in missing ability to overcome local extrema in the cost function. Thus the quality of the solution strongly depends on the initialization of the optimization, because the cost function usually in our approach contains local extrema, where the SQP process can easily get stuck. Using a global optimization method for avoiding the local extrema and for finding globally optimal solution would lead to slowing the optimization process, which is not acceptable.

The initialization of the MPC method is necessary for trajectory planning of the virtual leader and also for trajectory tracking of each follower. The initialization for followers is done according to the concept of leader-follower stabilization from the virtual leader trajectory. The problem of the initialization of the MPC method is solved in the first planning step for the virtual leader. In the next planning steps, the initialization is provided from the trajectory obtained in the previous step of MPC method by the reinitialization (see Figure 5).

The manual method, when user sets individually the initial trajectory by hand for each situation, is one way to provide the initialization for the first planning step. The trajectory that was set by such simple method was used as the initialization for the first planning step of the MPC method in the experiment presented in section 4.3. This method of providing the initial trajectory could be appropriate for the simulations, where the correct functionality of the proposed approach is presented, but in the real experiments this approach is insufficient. Obviously, it takes too much time to set the correct initialization from the actual state of the formation (the virtual leader) to the desired region. So, it would be a big benefit, if the initialization could be provided by an automatic method.

5.2 Approaches of trajectory planning for the initialization

Numerous methods of trajectory planning can be found in literature. These methods can be divided into classes given by the type of the approach, namely:

- grid-based approaches [4],
- geometric algorithms [10],
- potential fields [2],
- sampling-based algorithms [7],
- and others.

All of the mentioned approaches have both advantages and disadvantages. The potential fields methods combines attraction to the goal and repulsion from the obstacles for creating the field. The resulting trajectory is a path from the robot's position to the desired position in this field. The advantage of this approach is that the trajectory can be computed quickly. However, the robot can become trapped in a local minima of the potential field, thus failing to find a path. The sampling-based algorithms avoid the problem of local minima. Unfortunately, they are unable to determine that no path exists and the algorithm may run indefinitely[7].

The sampling-based algorithms were chosen in this thesis for finding the initialization of the MPC method for the trajectory planning of the virtual leader, namely the rapidly-exploring random tree.

5.3 Rapidly-exploring Random Tree (RRT)

5.3.1 Description of the algorithm

The RRT was introduced by Steven M. LaValle in [6] and [8]. The objective of this algorithm is to start from an initial configuration and find a path to the goal configuration. This is done by continuously expanding tree using control inputs that drive the system towards randomly-selected points. This tree is expanded in the free configuration space of the robot $\mathbb{C}_{free}$ until the desired state is achieved, or until a maximum number of iteration is reached. The RRT algorithm is probabilistically complete. With enough points the probability that it finds an existing solution converges to one [7]. Unfortunately, as it was mentioned before, the sampling-based algorithms are unable to determine that the desired state cannot be achieved and the algorithm may run indefinitely. This is the reason why the RRT is limited by maximum number of iterations to prevent this situation. The

resulting trajectory will be feasible for the robot when control inputs used for expansion of the tree satisfy the kinematic model of the robot.

Algorithm 1: RRT algorithm

The standard RRT algorithm can possibly grow into the entire feasible space.

input : x_{init}− initial configuration of a robot
MaxIteration − maximum number of iteration

output: Trajectory from x_{init} to the desired region

```
begin
    Tree = x_init ;
    x_new = x_init ;
    i = 0 ;
    while Distance(x_new,goal)> ErrorTolerance do
        x_random = SampleTarget() ;
        x_nearest = NearestVertex(Tree, x_random) ;
        x_new = ExtendTowards(x_nearest, x_random) ;
        if not Tree .contains (x_new) then
            Tree .add (x_new);
        i = i + 1 ;
        if i = MaxIteration then
            break;
    return Trajectory(Tree, x_init)
end
```

The structure of the RRT algorithm is visualised in Algorithm 1. The detection, that the tree is already expanded enough, is necessary for a correct functionality of this algorithm. That is the reason why each newly added vertex into tree is checked, whether it is already located near enough to the desired position. It is also necessary to check the tree if it wasn't already expanded by the vertex x_{new}. Without that the tree could have duplicate vertices. Particular functions of the algorithm are described in the following paragraph.

- **Distance** - This method returns a distance between two points (Euclidean distance is used in this thesis).
- **SampleTarget** - No input parameters are needed for this method. The method returns a random point, that has to be located in the free configuration space of a robot $\mathbb{C}_{free}$.
- **NearestVertex** - The tree must be expanded towards the randomly-selected points x_{random} and therefore the nearest vertex of the tree to the point x_{random} needs to be found. The easiest way how to do this is to compute distances from all vertices of the tree to the point x_{random} and the vertex with the minimal distance represents the

result. However, this method is not effective if the tree has higher number of vertices. A more effective approach will be explained in subsection 5.3.3.

- **ExtendTowards** - In this method, expansion of the tree is done. The method needs two input parameters x_{random} and $x_{nearest}$ from the previously mentioned functions. The steering that represents all possible trajectories from the vertex in the RRT method is used in the expanded node $x_{nearest}$. Only non-collision trajectories from the expanded node $x_{nearest}$ are then used for finding the nearest endpoint of trajectories to a randomly-selected point x_{random}. The trajectory with the nearest endpoint is returned as the result. The structure of this method is described by the pseudocode in Algorithm 2 and an example of its functionality is visualized in Figure 10.

Algorithm 2: RRT algorithm - **ExtendTowards** method

input : $x_{nearest}$− the nearest node of the tree to the point x_{random}
x_{random}− the randomly-selected point
output: x_{new}− extension vertex of the tree

```
begin
    U = Steer() ;
    min_dist = inf ;
    for u ∈ U do
        traj = ComputeTrajectory(x_nearest,u) ;
        if CollisionFree(traj) then
            if Distance(traj .endpoint, x_random)< min_dist then
                best.node =traj .endpoint ;
                best.parent = x_nearest ;
                best.u = u ;
                min_dist =Distance(traj .endpoint, x_random) ;
    return best ;
end
```

- **Trajectory** - This method returns a trajectory from the initial configuration of a robot x_{init} to the vertex of the tree which is closest to the goal configuration. In this part of the algorithm the tree structure is already known and root of the tree represents the initial configuration of a robot x_{init}. Therefore the trajectory can be easily constructed by backtracking through this structure. The advantage of this method is in situations when the tree does not achieve goal configuration in the maximum number of iteration. The trajectory that leads to a configuration that is situated closest to the goal configuration is returned in these situations. The structure of this method is described by the pseudocode in Algorithm 3.

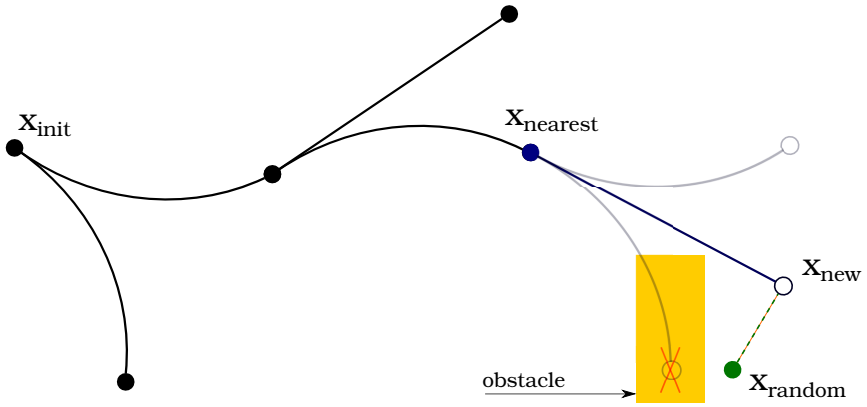


Figure 10: RRT algorithm- extension of the tree towards a randomly-selected point

Algorithm 3: RRT algorithm - **trajectory** method

input : Tree – the tree of RRT algorithm
x_{init}– the initial configuration of a robot
output: traj – the resulting trajectory

begin
  $x =$ **NearestVertex**(Tree, goal) ;
  traj $= \emptyset$;
  while $x.parent \neq \emptyset$ **do**
    traj $= x.u \cup$ traj;
    $x = x.parent$;
  return traj;
end

5.3.2 Modifications of the RRT

The tree is continuously expanded towards a randomly-selected points in the standard RRT algorithm until the desired state is achieved, or until a maximum number of iterations is reached. However, the random character of expansion of the tree means that tree is expanded also into locations which are useless. Numerous modifications of the standard RRT were presented to solve this problem [24].

If the algorithm is informed about the desired position and tries to expand the tree towards it, it is possible that the algorithm will find the solution faster. Unfortunately, it is also possible that it will get stuck because of an obstacle if the algorithm goes only straight forward to the goal. That is the reason why the algorithm also needs to have some random exploring of the area. The structure of the RRT algorithm biased towards the goal is same as the standard RRT described in Algorithm 1, only the function **SampleTarget** is different. Its functionality is described by the pseudocode in Algorithm 4.

Algorithm 4: RRT algorithm biased towards goal - **SampleTarget** method

```
output: x_rand− the selected point
begin
    if Rand() < GoalSamplingProbability then
        return goal;
    else
        return RandomConfiguration() ;
end
```

Another modification of the RRT algorithm expands the tree by selecting a random vertex of the tree and then extending it towards the goal with a probability p. With a probability $1 - p$ it generates a random point then finds the nearest vertex in the tree and expands the tree towards it (same expansion of the tree as the standard RRT algorithm). This modification has similar behaviour as previously mentioned one, since it is also biased towards the goal. The structure of the modification of the RRT algorithm is described by the pseudocode in Algorithm 5.

Algorithm 5: RRT algorithm biased towards goal 2

```
input  : x_init− the initial configuration of a robot
         MaxIteration − maximum number of iteration
output: Trajectory from x_init to the desired region
begin
    Tree = x_init ;
    x_new = x_init ;
    i = 0 ;
    while Distance(x_new,goal)> ErrorTolerance do
        if Rand() < GoalSamplingProbability then
            v_random = RandomVertex(Tree) ;
            x_new = ExtendTowards(v_random, goal) ;
        else
            x_random = SampleTarget() ;
            x_nearest = NearestVertex(Tree, x_random) ;
            x_new = ExtendTowards(x_nearest, x_random) ;
        if not Tree .contains (x_new) then
            Tree .add (x_new);
        i = i + 1 ;
        if i = MaxIteration then
            break;
    return Trajectory(Tree, x_init)
end
```

The previously mentioned modifications of the RRT algorithm try to expand the tree from the initial configuration of a robot in the direction towards the goal. Another modification of algorithm uses two trees for searching in both directions. The first tree is expanded from the initial position of a robot and the second from the goal. This modification is called bidirectional RRT algorithm and in every iteration, it tries to expand one of these two trees, which can be done by any of the previously mentioned methods. In every iteration, the algorithm also tries to check if the trees are already big enough that they are connected to each other. When they are connected, the resulting trajectory from the initial position of a robot to the goal already exists and is returned as the solution.

5.3.3 Kd-tree - Efficient finding of nearest points

The RRT algorithm can solve planning problems quite quickly. The structure of the standard RRT algorithm is visualized in Algorithm 1. All its functions can be implemented easily. However, the speed of the algorithm depends on how it is done especially in the **NearestVertex** method. This method finds the nearest vertex of the tree to the point. Several alternatives how to implement this method can be found in literature [7].

The easiest method is to compute the distances from all vertices of the tree to the point in space. Then the vertex with minimal distance to the point represents the result. The time complexity of this method increases linearly with the number of the vertices in the tree. So, this method is not effective if the tree has higher number of vertices. The solution how to speed up the process of finding the nearest point is to insert the vertices into an efficient data structure.

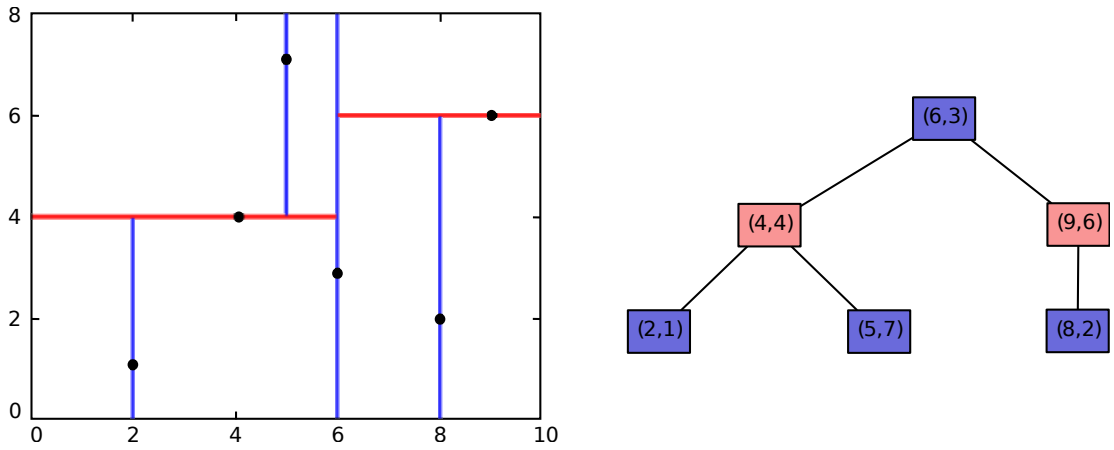


Figure 11: Kd-tree decomposition of 2D space and the resulting kd-tree by a set of points $\{(4,4),(2,1),(5,7),(8,2),(9,6),(6,3)\}$

Widely used and useful data structure is the k-dimensional tree (abbreviated as kd-tree). The kd-tree was developed by Jon Louis Bentley in [3] and can be considered as

a multi-dimensional generalization of a binary search tree. Each node in the kd-tree is associated with one of the k-dimensions. For example, if a node is associated to x-axis, all points with a smaller x value than this node will appear in the left subtree and all points with a larger x value will be in the right subtree. An example of the created kd-tree for some points in 2-dimensional space is shown in Figure 11. The 2-dimensional space is divided by switching between the x and y coordinates. The time complexity of the kd-tree is described in Table 2.

	average	worst case
Search	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$
Insert	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$
Delete	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$

Table 2: The time complexity of the kd-tree.

5.4 Initialization of the MPC method by the RRT algorithm for trajectory planning of the virtual leader

As was mentioned in section 4.1 for the purpose of the MPC method, the trajectory of the virtual leader is gathered into the optimization vector Ω_L , with N+M elements, where N is the length of the *control horizon* and M is the length of the *planning horizon*. Each element contains control input and time that represents the duration of every control input. The sampling time is constant Δt for the first N elements and the sampling time is variable for the rest of the elements. The initial trajectory of the MPC method has to be gathered into the optimization vector Ω_L. The first N elements of this vector have to have the sampling time equal to Δt.

Unfortunately, the trajectory provided by the RRT algorithm gathered into the vector Ω_{RRT} cannot be usually directly used as the initialization of the MPC method, because of the following reasons:

- the sampling time in the first N elements of the vector Ω_{RRT} is not equal to Δt,
- the vector Ω_{RRT} is created from too many parts.

All these reasons depend on the time interval t_s, which has an effect on expanding the tree in the RRT algorithm. The value of the time interval t_s represents the sampling time of the trajectory. The influence of the different settings of values of the time interval t_s will discussed in section 7.2.

Algorithm 6: RRT2MPC algorithm - **ExtendTowards** method

input : $x_{nearest}$− the nearest node of the tree to the point x_{random}
x_{random}− the randomly-selected point
output: x_{new}− extend vertex of the tree

```
begin
    if depth(x_nearest)<N then
        U = Steer(t_s,N) ;
    else
        U = Steer(t_s,M) ;
    min_dist = inf ;
    for u ∈ U do
        traj = ComputeTrajectory(x_nearest,u) ;
        if CollisionFree(traj) then
            if Distance(traj .endpoint, x_random)< min_dist then
                best.node =traj .endpoint ;
                best.parent = x_nearest ;
                best.u = u ;
                min_dist =Distance(traj .endpoint, x_random) ;
    return best ;
end
```

5.4.1 RRT2MPC modification of the RRT algorithm (RRT2MPC algorithm)

The firstly mentioned problem, why the trajectory provided by the RRT algorithm cannot be usually directly used as the initialization of the MPC method, can be solved by setting the sampling time of the RRT algorithm to $t_s = \Delta t$. However, if the sampling time Δt of the *control horizon* is a small number, the trajectory provided by the RRT algorithm will be sampled with high sampling rate. This means that the vector Ω_{RRT} will be created from too many elements. An obvious solution, how to solve it, is to use two sampling times $t_{s,N}$ and $t_{s,M}$ for expanding the tree in the RRT algorithm. The values of these sampling times are $t_{s,N} = \Delta t$ and $t_{s,M} = t_d$, where t_d is value defined by the user. The idea is that the expanding of the tree of the RRT algorithm to the depth N will be done with the sampling time $t_{s,N}$ and for the higher depth of the tree with the sampling time $t_{s,M}$. The consequence of this is that the major part of the provided trajectory will be sampled with the bigger sampling time than Δt and the vector Ω_{RRT} will be create from the less parts than vector Ω_{RRT} provided by the RRT algorithm with only one sampling time $t_s = \Delta t$. The modified **ExtendTowards** method using this idea is described in the pseudocode in Algorithm 6.

By using the previously mentioned solution, the first N parts of the vector Ω_{RRT} is sampled with the sampling time $t_{s,N}$ and rest of the parts with the sampling time $t_{s,M}$.

Algorithm 7: Reduction of numbers of elements that describe the trajectory in the RRT2MPC algorithm

```
input  : traj− the trajectory
output: new− the simplified trajectory
begin
    new = ∅ ;
    actual = traj[N + 1] ;
    for i = N + 2 to length(traj) do
        if actual. hasEqualSteering(traj[i]) then
            actual.t = actual.t + traj[i].t ;
        else
            new = new ∪ actual ;
            actual = traj[i] ;
    new = new ∪ actual ;
    return new ;
end
```

The problem is that the vector Ω_{RRT} still have too many elements with the sampling time $t_{s,M}$ that represent the trajectory in the *planning horizon* of the proposed MPC method. It is eventually possible to use the trajectory in this form as the initial trajectory for the MPC method, but the length of the *planning horizon* M has to be set to the number of parts with the sampling time $t_{s,M}$. However, the time complexity of finding the solution by the SQP process is too high and its usage for the real-time planning is impossible.

A method for minimizing the number of the elements in the vector Ω_{RRT} in the *planning horizon* is necessary for using the provided trajectory as the initialization of the MPC method. The trajectory can have neighboring parts in the *planning horizon* with the same control inputs, because of random expansion of the tree of the RRT algorithm. These parts can be joined together into one part of the trajectory. The proposed method solving this problem is described by the pseudocode in Algorithm 7, and can be applied on every provided trajectory. The consequence of this method is that the same trajectory is described by the vector Ω_{RRT} with a smaller number of elements than the original one, if it is possible. An example of the proposed reduction of number of elements that describe the trajectory is shown in Figure 12.

The RRT algorithm has to be modified for applying the solutions mentioned above (the two sampling times $t_{s,N}$ and $t_{s,M}$ and the method for reduction of the number of elements that describe the trajectory). This modification is called RRT2MPC modification of the RRT algorithm in this thesis (further this modified RRT algorithm will be called as the RRT2MPC algorithm). The RRT2MPC algorithm is described by the pseudocode in Algorithm 8. The RRT2MPC algorithm can be also modified by any of the mentioned modifications in section 5.3.2. In this thesis, the information about the desired position is

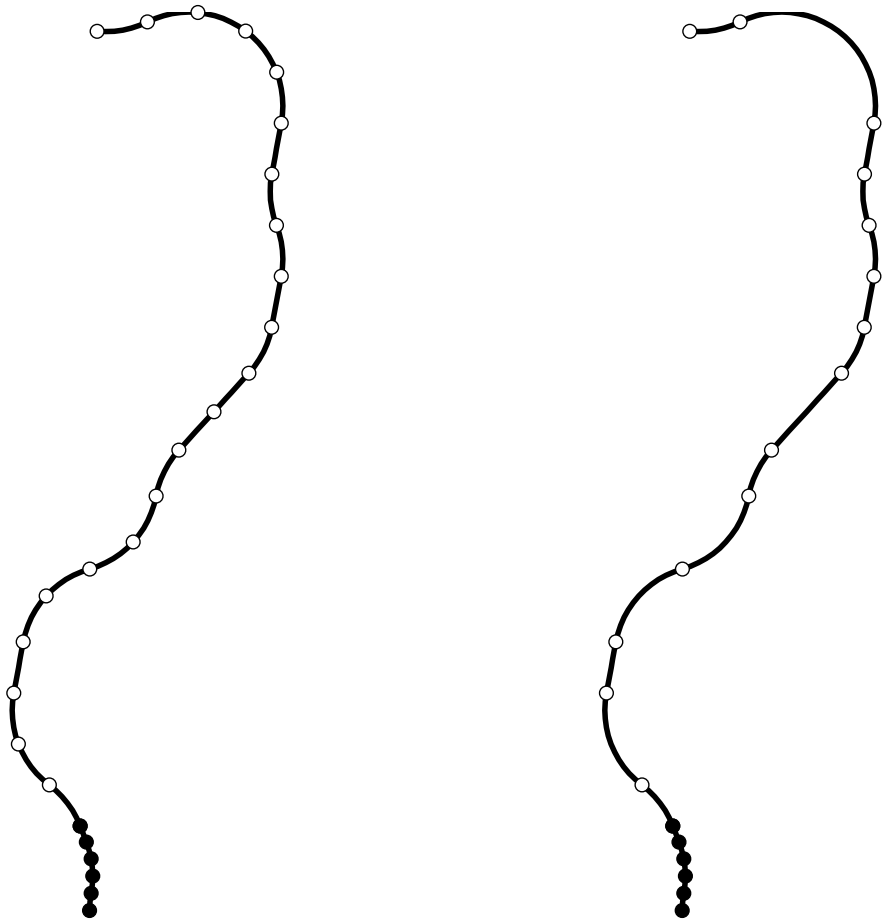


Figure 12: The reduction of the number of elements that describe the trajectory provided by the modified RRT algorithm. The first N parts of the trajectory are sampled with the sampling time $t_{s,N}$ and the rest of the parts with the sampling time $t_{s,M}$. The points in the figures represent transition points of the trajectory. The black points represents transition points in the *control horizon*, and the white points in the *planning horizon*. The original trajectory is on the left side of the figure and the simplified one on the right side. The original trajectory is created from 26 segments and it is simplified by the proposed method to 19 segments.

used for changing the behaviour of the expanding tree. This is done by the modification of the **SampleTarget** method in the same way as is explained in Algorithm 4.

Unfortunately, the proposed method for the reduction is not effective enough (about

Algorithm 8: RRT2MPC algorithm

input : x_{init}− the initial configuration of a robot
MaxIteration − maximal number of iteration
output: trajectory from x_{init} to the desired region

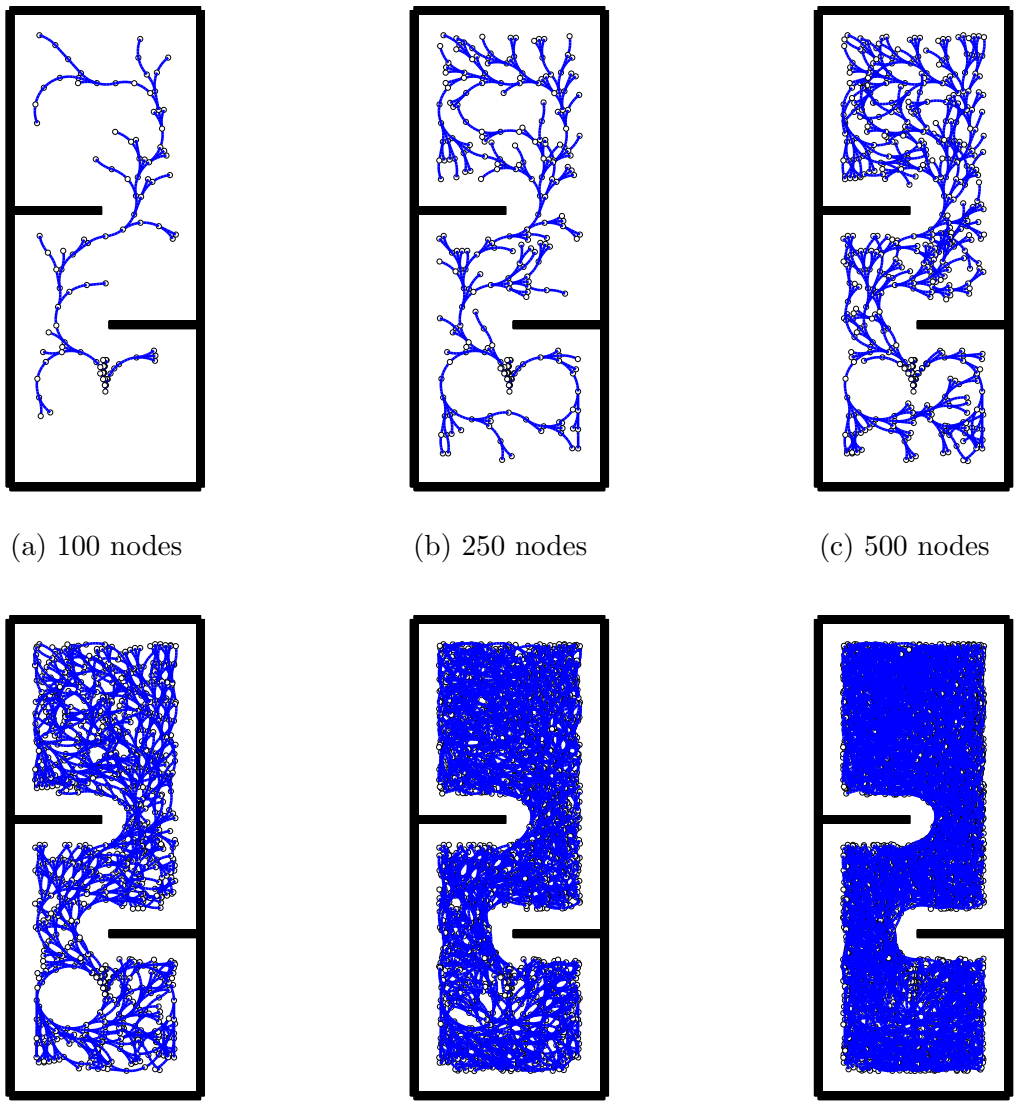
```
begin
    Tree = x_init ;
    x_new = x_init ;
    i = 0 ;
    while Distance(x_new,goal)> ErrorTolerance do
        x_random = SampleTarget() ;
        x_nearest = NearestVertex(Tree, x_random) ;
        x_new = ExtendTowards(x_nearest, x_random) ;
        if not Tree .contains (x_new) then
            Tree .add (x_new);
        i = i + 1 ;
        if i = MaxIteration then
            break;
    traj = Trajectory(Tree, x_init) ;
    return ReductTrajectory(traj)
end
```

30%) because the vector Ω_{RRT} still consists of too many elements for most of the trajectories. So, the resulting trajectory cannot be directly used as the initial trajectory for the MPC method because of the previously mentioned reasons. However the trajectory provided by the RRT2MPC algorithm can be used for the modified MPC method, which will be explain in section 5.4.2.

The growth of the tree generated by the RRT2MPC algorithm in an environment is shown in Figure 13. The free configuration space of the virtual leader $\mathbb{C}_{free}$ in this environment can be seen in Figure 13(f), where the tree covers this space almost completely.

5.4.2 Modificatition of proposed MPC method

As was described before, the initial trajectory for the proposed MPC method, which is applied in the first planning step for the virtual leader, has to be provided. In the next planning steps, the initialization is done from the trajectory obtained in the last MPC step by the reinitialization which is described in [23]. The RRT2MPC algorithm provides the trajectory gathered into the vector Ω_{RRT} that still consists of too many elements for most of the trajectories. Therefore this trajectory cannot be directly used as the initial trajectory for the MPC method. The provided trajectory can be used for initialization of



(a) 100 nodes (b) 250 nodes (c) 500 nodes

(d) 1000 nodes (e) 2500 nodes (f) 5000 nodes

Figure 13: The growth of the tree generated by the RRT2MPC algorithm.

the MPC method if the approach presented in Figure 5 will be changed as follows.

The idea of this modification is to use only the first $N+M$ elements of the trajectory provided by the RRT2MPC algorithm as the initial trajectory for the MPC method (N and M represent lengths of the *control* and *planning horizon*, respectively). Then optimize this part of the trajectory and use the first n control inputs, according to the MPC concept. After that not to use reinitialization for computing the initial trajectory for the next planning steps of the MPC method, but instead of that, use the same approach as for the first planning step. This idea affects the Virtual leader block, which was introduced in

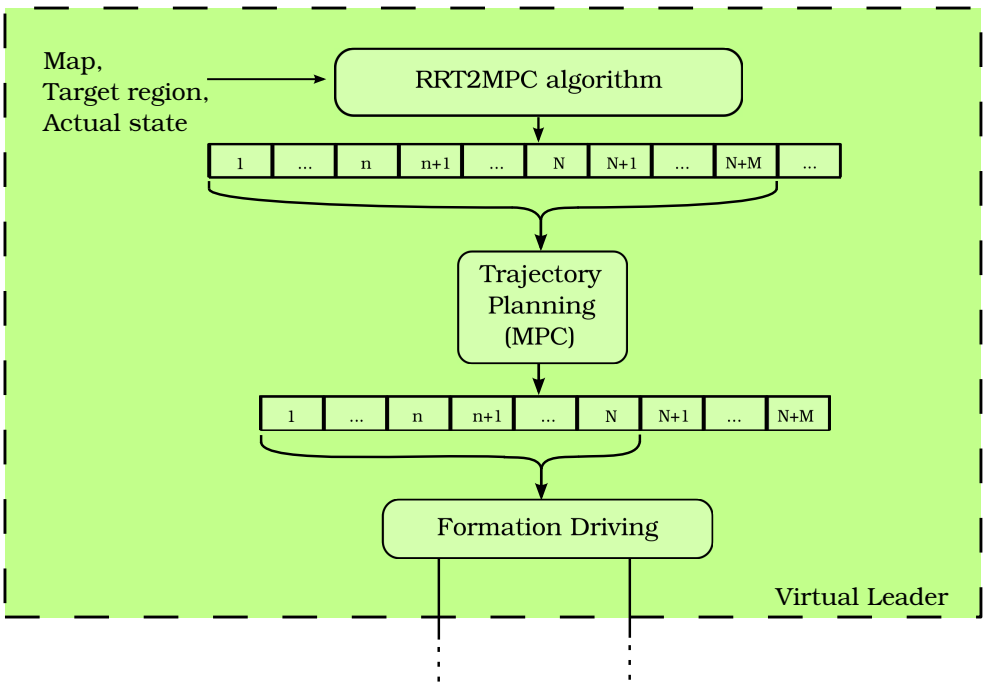


Figure 14: Modified MPC method - Virtual Leader block

Figure 5. The new scheme of this block is shown in Figure 14.

The MPC method is used in this thesis for solving the optimization control problem of the formation movement into the desired region. Unfortunately, the first $N+M$ elements of the trajectory provided by the RRT2MPC algorithm usually do not lead to the desired region and therefore this initial trajectory is considered as an incorrect initialization for the proposed MPC method. This is caused by the stability constraint $g_{S_F}(\psi_L(N+M))$ that ensures that the virtual leader trajectory will lead to the target region. The endpoint of this initial trajectory has to be signed as the goal for the MPC method. Then the MPC method can be used for optimization of the trajectory that will not lead to the target region.

5.5 Experimental verification

This subsection is focused on experimental verification of the proposed approach. For this purpose, the same situation was chosen as for the experiment presented in section 4.3. The same formation with 8 members has to move into a target region through an environment with one static obstacle. This obstacle is detected during its movement. Parameters of the formation are shown in Table 1. Progress of the values of the cost function (eq. (11)) used for the virtual leader trajectory planning is displayed in Figure 15. Snapshots from the experiment are shown in Figures 16, 17 and 18.

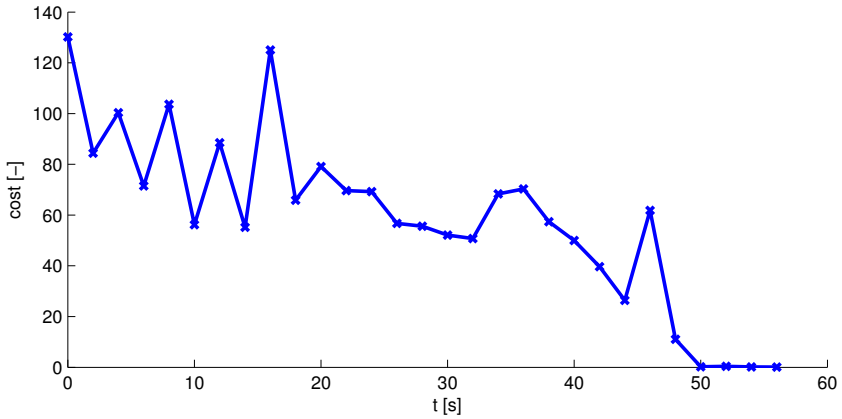
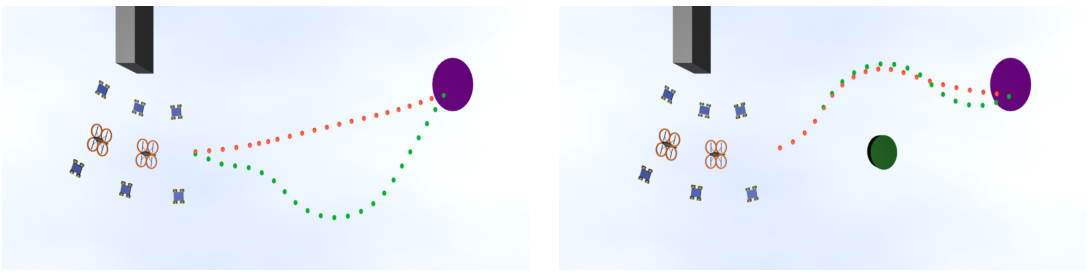


Figure 15: Progress of value of the cost function (eq. (11)) used for the virtual leader trajectory planning.

Progress of values of the cost function used for the virtual leader trajectory planning is presented in Figure 15. It is not clear from the progress of values at the first 20 second of the graph if the formation converges to the desired region in contrast to the graph in Figure 6. It is because the initial trajectories (first $N + M$ parts of the trajectory proposed by the RRT2MPC algorithm) do not lead to the desired region and their lengths differ. In the next part of the graph, similar trajectories are found by the RRT2MPC algorithm besides the optimization at 46s. At this time, the RRT2MPC algorithm found a longer trajectory. The trajectory depends on the expansion of the tree generated by the RRT2MPC algorithm which has a random character. Therefore different trajectories can be found. The finding a longer trajectory than in the previous cases is visualised by the increase of the value of the cost function.

The result of replanning after detecting an obstacle during movement is shown in Figure 16. Plotted plan to the target region found by the RRT2MPC algorithm and the result of optimization of the trajectory created from the first $N+M$ elements of the plan is presented in Figure 17. The paths passed by the members of the formation are shown in Figure 18.



(a) t=34s (b) t=36s

Figure 16: The result of replanning after detecting an obstacle during movement. The trajectory (red points) is the result of local optimization of the initial trajectory (green points).

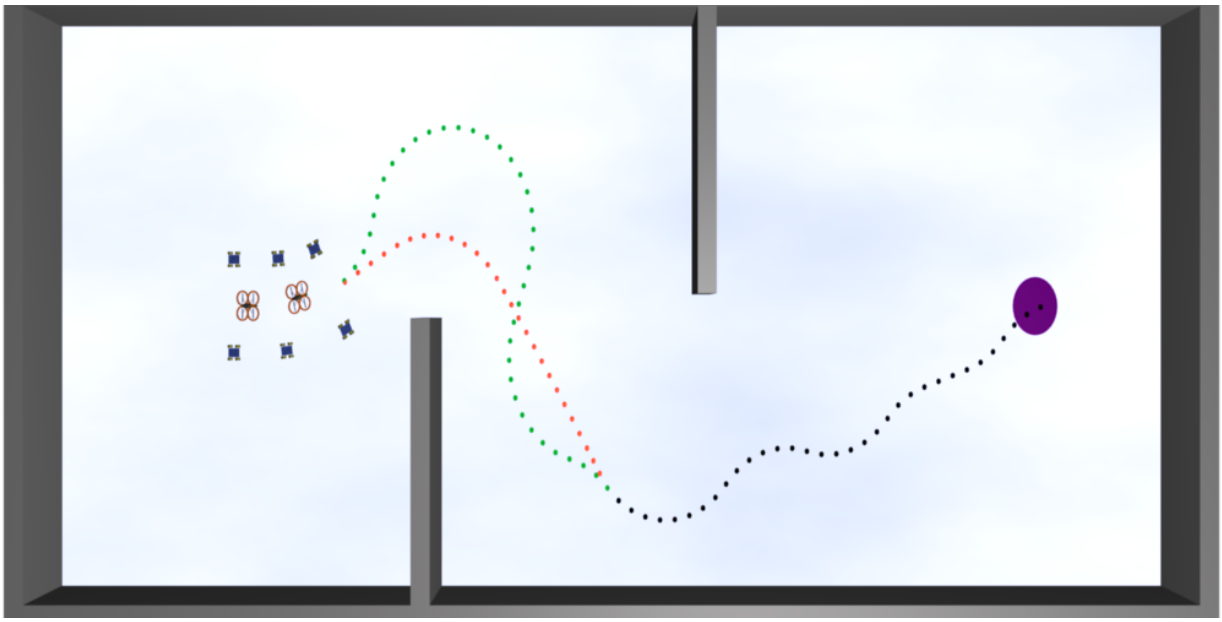


Figure 17: Plotted plan to the target region found by the RRT2MPC algorithm and the optimization of the trajectory created from the first $N+M$ elements of the plan. The plan is visualised in the figure by green points (the first $N + M$ elements) and by black points (the rest elements). The trajectory (red points) is the result of local optimization of the trajectory (green points).

5.6 Summary

The purpose of the initialization for the proposed MPC method was described in this section. The problem lies in the unknown initial trajectory for the virtual leader trajectory planning in the first planning step. For finding this initial trajectory, the RRT algorithm was selected and described. The advantages and disadvantages of this algorithm were mentioned. For example, the advantage is that the algorithm quickly finds the solution, if

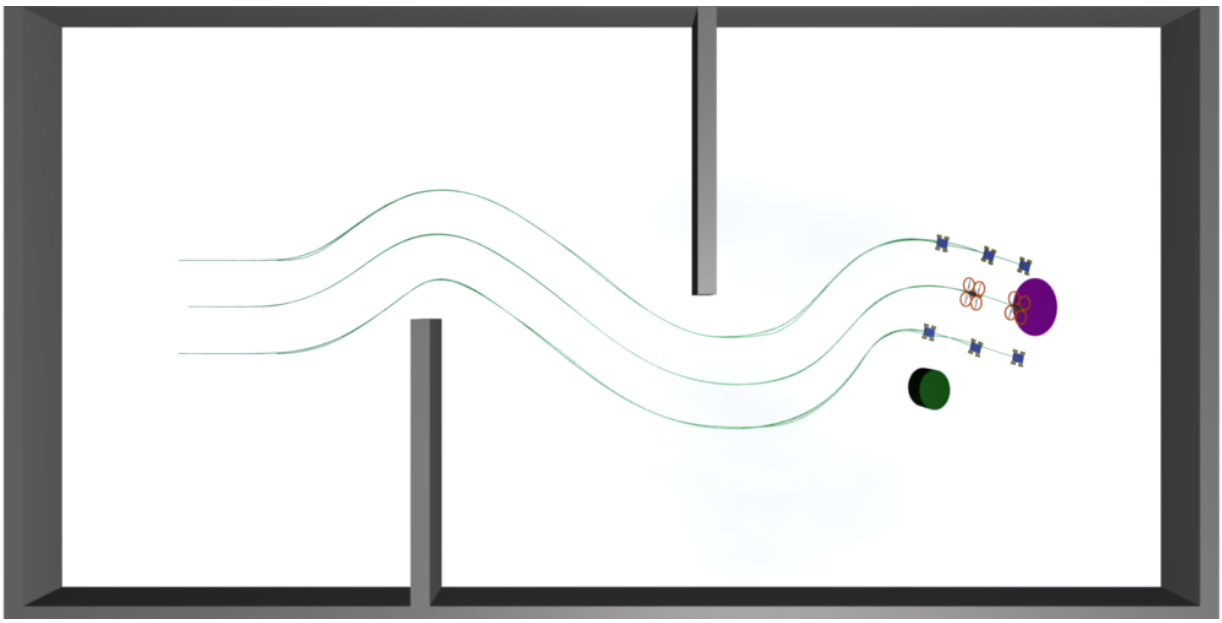


Figure 18: The paths passed by the members of the formation.

exists. Furthermore, the proposed trajectory is feasible for the robot, if the algorithm is run with robot kinematic model. However, the provided trajectory is too complex and can not be directly used as the initial trajectory. For this reason, the modification of the RRT algorithm (so-called the RR2MPC modification in this thesis) was introduced.

This method provides an extension of the tree generated by the RRT algorithm using two sampling times $t_{s,N}$ and $t_{s,M}$. The idea is that the expanding the tree of the RRT algorithm to the depth N (length of *control horizon*) will be done with the sampling time $t_{s,N}$ and for the higher depth of the tree with the sampling time $t_{s,M}$. This part of the modification ensures that provided trajectory can be used for the proposed MPC method. However, the trajectory often consists of too many segments and finding the optimization of this trajectory takes too much time. So the next part of the modification solves reduction of the trajectory. Unfortunately, this reduction is not effective enough to minimize the length to a usable size. Other approach needs to be used in order to use this trajectory as the initial trajectory for the proposed MPC method.

The idea is that only the first $N + M$ segments of the provided trajectory are used as the initial trajectory for the MPC method, where N a M are the numbers of the transition points in the *control* and *planning horizon*. This initial trajectory is locally optimized and only the first n control inputs are applied according to the MPC concept. The same idea is used in next planning steps. This solution works as was illustrated by the experiment in section 5.5. However, finding the new trajectory and optimizing it in every planning step means waste of time.

For future work, other idea how to reduce the length of the trajectory is to use the local optimization for parts of the trajectory. For example, to optimize five parts of trajectory, then to optimize another five parts of trajectory and so on. This approach could minimize the difference of control inputs between the neighboring parts of the trajectory. After that, the reduction of the trajectory described in this thesis can be applied.

6 Complex maneuvers

Only the forward movement of the formation was consider for reaching to the desired area in the experiments in the previous sections. However, a backward movement is necessary for general solutions of the movement of the formation, because in many situations solution of the formation motion problem without any backward movement does not exist. These situations where it is necessary to reverse of the movement of the formation are considered in this section. This part of thesis is based on [23] where these situations are studied.

The proposed MPC method for solving the optimization control problem of the movement to the target region is already general enough to provide a complete plan for the virtual leader including changes of polarity of forward velocity $v_L(\cdot)$ without any modification. The motion constraints of the virtual leader (eq. (8)) allow negative velocity and its maximal value is computed from the allowed maximum negative velocities of all members of the formation. So, the plan for the virtual leader created according to the formation driving concept described in section 3.5 is in principle feasible for each follower in the formation. Unfortunately, the proposed concept for the formation driving works properly only in situations where the forward movement is applied. Applying the reverse of the movement of the virtual leader to the followers exactly following this concept leads to the collision with other members of the formation or to the loss of the desired shape of the formation. This problem arises because the followers behind the leader (with $p > 0$) continue with the direction of the movement until the state where the virtual leader changed the polarity of its velocity has been reached. All members of the formation should change the polarity of their velocity in the same moment for a correct change of the direction of the movement and for keeping the desired shape of the formation. However, this is not possible by the formation driving concept described in section 3.5, because each follower with different value of p_j will achieve the state, when the leader changes the polarity of its velocity in a different time. This problem is demonstrated in Figure 19. In this figure, the proposed formation driving concept leads to collision with other members of the formation and also to the loss of the desired shape of the formation during the reverse of the movement of the formation.

The planned trajectory has to be modified to avoid this problem. An approach solving this problem is shown in Figure 20, where the planned trajectory is extended by a path that has the length equal to maximal value of p_j of the followers in the formation. The reverse of the movement of the formation is applied after the virtual leader travelled this path. The consequence of this extension is that the formation can smoothly continue with a movement while keeping the desired shape.

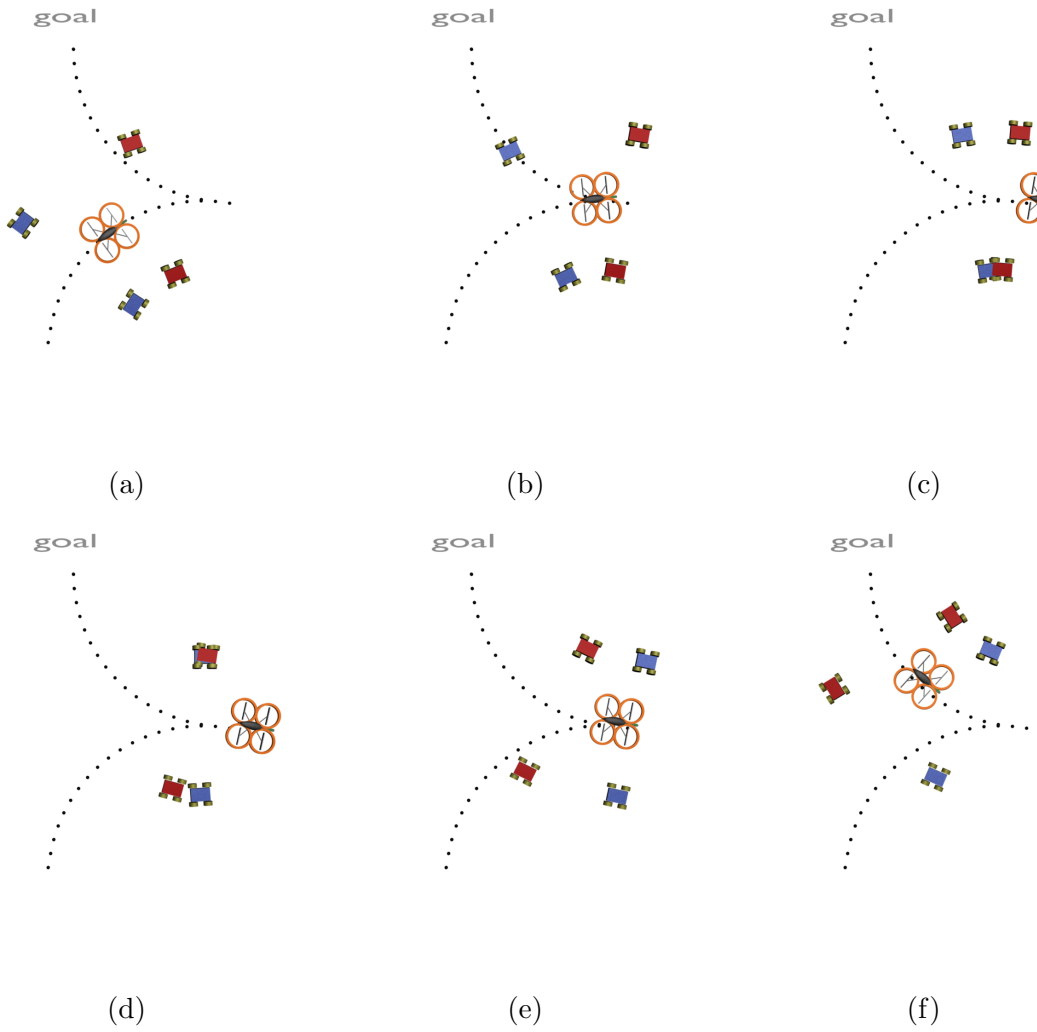


Figure 19: Problem of the proposed formation driving concept during the reverse of the movement of the formation.

6.1 Concept of two alternating virtual leaders

Unfortunately, such simple extension of the planned trajectory may not be feasible because the extended plan could lead to a collision with an obstacle. So the proposed method for trajectory planning has to be improved for finding a plan to the desired area that considers this requirements for applying the reverse of the movement of the formation. In [23], the concept of two alternating virtual leaders was presented. It is an approach that modifies the basic method for the virtual leader in such a way that it plans with two virtual leaders, one for the forward movement and one for the backward movement. Their leading role is always switched when the polarity of the leader velocity changes. So in each moment, one virtual leader leads the formation and other virtual leader controlled as a

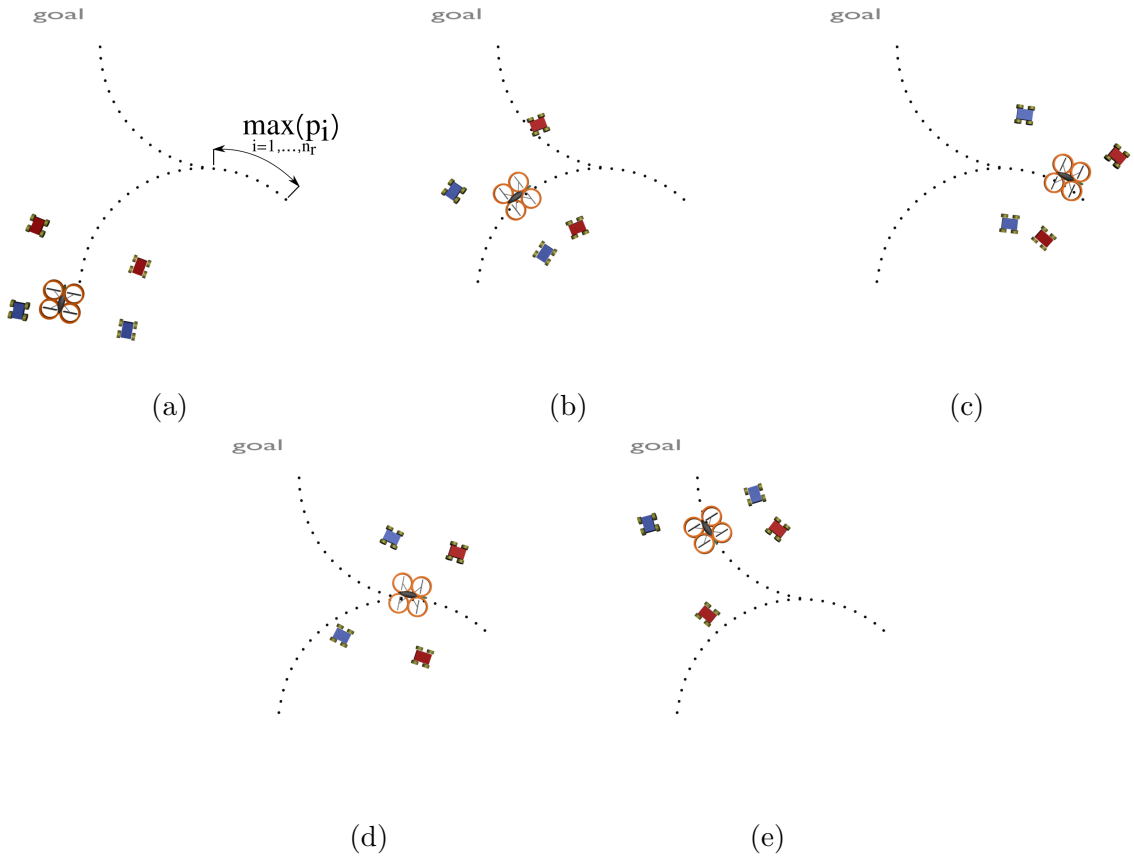


Figure 20: Proposed solution of the problem with the formation driving concept during the reverse of the movement of the formation.

virtual follower. The first virtual leader assigned for the forward movement is located in front of the formation and the second virtual leader in back of the formation. Further, the both virtual leaders are located on axis of the formation.

The plan for both virtual leaders can be solved in a similar way as the plan for the one virtual leader described in section 4.1 where the trajectory is represented as the one optimization vector Ω_L. The vector $\Omega_L = [\vec{u}_{L,1}, \ldots, \vec{u}_{L,N}, \vec{u}_{L,N+1}, \ldots, \vec{u}_{L,N+M}]$ is created from $N + M$ elements where N is the length of the *control horizon* and M is the length of the *planning horizon*. Each of these element contains control inputs and time that represents duration of every control input. The vector Ω_L provides the desired trajectory for both leaders. The parts of the trajectory which have the positive polarity of the value of v_L are assigned to one virtual leader and the rest of the trajectory with the negative polarity to other virtual leader. Furthermore, for the complete plan for both virtual leaders, these trajectories have to be extended by additional paths which are used for shifting the position of one virtual leader to the other one. This is necessary when the polarity of the value of v_L is changed (the leadership is shifted from one virtual leader to the other one) for

a smooth continuation of the movement of the formation while keeping the desired shape.

The maximum number of possible changes of the leadership in the formation during the proposed plan is $N + M - 1$ therefore the plan can be possibly extended by the same number of additional paths. The additional path p_k, where $k \in \{1, \ldots, N + M - 1\}$, which is used for shifting the position of the new virtual leader to the old one can be described by control inputs v_k, w_k, K_k and by a time duration Δt_k of the control inputs which are applied in the state of the old virtual leader. These parameters can be represented as a vector $\vec{q}_k = [v_k, w_k, K_k, \Delta t_k]$. The value of v_k can be determined from the virtual leader constraints as the maximum or minimum allowed value of v_L as the minimization of the driving time is preferred. The decision which one of these values is assigned to v_k depends on the polarity of velocity of the virtual leader in k-th element of Ω_L due to the fact that these polarities have to be the same. The value of Δt_k can be computed as

$$\Delta t_k = \frac{l_a}{v_k}, \tag{20}$$

where l_a is the minimal length of additional paths which is defined as

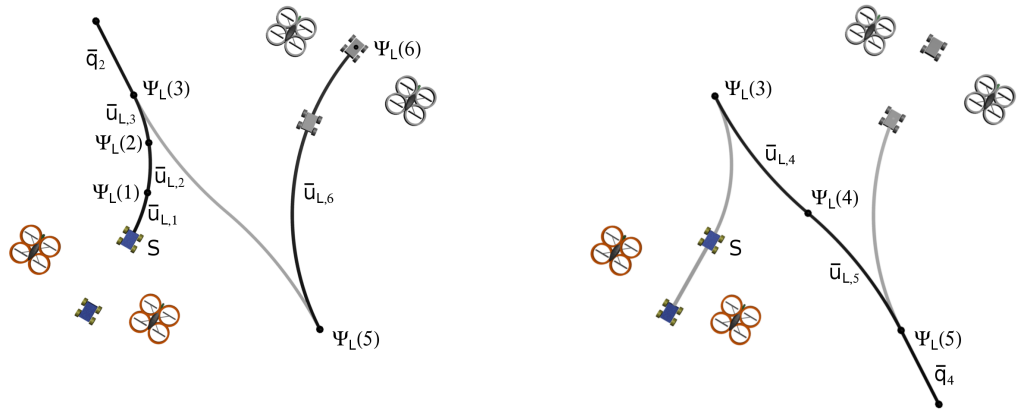
$$l_a = \max_{i=1,\ldots,n_r} (p_i). \tag{21}$$

Therefore, only the values w_k and K_k are not uniquely determined and have to be computed for each additional path. An example of trajectories and appropriate control inputs of the two virtual leaders is shown in Figure 21, where the parameters of the optimization are: $N = 3, M = 3$.

The complete plan including the additional paths for both virtual leaders from the actual position to the target region can be represented as an extended optimization vector $\Omega_L = [\vec{u}_{L,1}, \ldots, \vec{u}_{L,N}, \vec{u}_{L,N+1}, \ldots, \vec{u}_{L,N+M}, w_1, K_1, \ldots, w_{N+M-1}, K_{N+M-1}]$. Finding this plan can be solved in a similar way as finding the plan for the one virtual leader, where this problem is transformed to the minimization of the cost function $\lambda_L(\cdot)$ (eq. (10)) with nonlinear constraints. Satisfying these constraints represents finding a feasible and collision free trajectory.

The relative positions of the followers described by curvilinear coordinates p, q, h to the virtual leader states have to be changed if the leadership is shifted from the one virtual leader to the other one. The positions of the virtual leaders in the formation is not same and therefore the relative positions to the old virtual leader are not the same as the relative positions to the new one. The computation of new values of the coordinates for the follower j, where $j \in \{1, \ldots, n_r\}$, can be done by following equation

$$\begin{aligned} p_{j,new} &= \left| p_{j,old} - \max_{i=1,\ldots,n_r} (p_{i,old}) \right|, \\ q_{j,new} &= q_{j,old}, \\ h_{j,new} &= h_{j,old}, \end{aligned} \tag{22}$$



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

Figure 21: 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 gray curves denote the trajectories where the virtual leader is the virtual follower. The initial positions of the robots are visualised by coloured bodies of the robots. The target positions after applying appropriate control inputs are shown by gray bodies.

where $p_{j,old}, p_{j,old}, p_{j,old}$ are curvilinear coordinates describing vehicle's relative position to the old virtual leader.

6.2 Experiments

This subsection is focused on experimental verification of the proposed approach.

6.2.1 Turning 180 degrees

Turning 180 degrees on narrow roads is one from the expected situations where the proposed approach can be used. The scenario, in which a formation of the nine robots has to find and realize this maneuver, is presented in Figure 24. Parameters of the formation are shown in Table 3.

In this scenario one static obstacle is located in the environment. The collision free plan that was found for turning the formation 180 degrees is shown in Figure 24(b). This plan contains two switches of the virtual leader. The movement of the followers tracking the obtained plan is presented in snapshots of the simulation in Figure 24(c) and in Figure 24(d). The average values of the heading of the followers in rows during the simulation is shown in Figure 22. The complete trajectories travelled by the followers during the maneuver are

i	1	2	3	4	5	6	7	8	9
p_i	0	0	1	1	1	1	2	2	2
q_i	-0.6	0.6	-1.2	-0.6	0.6	1.2	-1.2	0	1.2
h_i	0	0	0	1.1	1.1	0	0	0	0

Table 3: Curvilinear coordinates of the followers in the formation.

denoted in Figure 24(e).

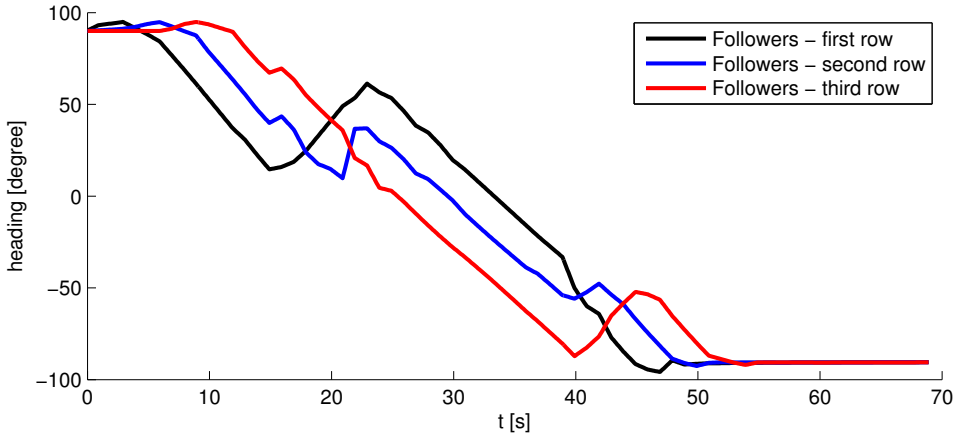


Figure 22: The average values of the heading of the followers in rows during the simulation presented in Figure 24.

6.2.2 Complicated environment with static and dynamic obstacles

The second scenario presented in this section is complicated environment with both static and dynamic obstacles. A formation composed of four members has to move to the target region through narrow paths. The polarity of velocity has to be changed during the movement of formation in order to accomplish the task. Parameters of the formation are shown in Table 4. Snapshots from the simulation of this scenario are shown in Figures 26, 27, and 28. The average values of the heading of the followers in rows during the simulation shown in Figure 23 and applied control inputs of the third follower are presented in Figure 25.

The complete plan for both virtual leaders found in the first planning step is shown on Figure 26(a). This plan includes two changes of leadership in the formation. Snapshot from changing the leadership from the first virtual leader to the second one is shown in Figure 26(b) and from the second virtual leader back to the first one in Figure 26(c). After this the formation detected a static obstacle during the movement thus the plan was changed to avoid the collision. The result of replanning is shown in Figure 26(d).

i	1	2	3	4	5
p_i	0	0	1.1	1.1	0.55
q_i	-0.8	0.8	-0.8	0.8	0
h_i	0	0	0	0	1

Table 4: Curvilinear coordinates of the followers in the formation.

The proposed method in this form is also suitable for planning of the formation movement in an environment with dynamic obstacles. The approach uses the expected positions of obstacles to find a feasible collision free plan. Trajectories after detection of a dynamic obstacle in this scenario are shown in Figure 28. More information about the formation trajectory planning problem using the MPC method in a dynamic environment can be found in our previous work in [16].

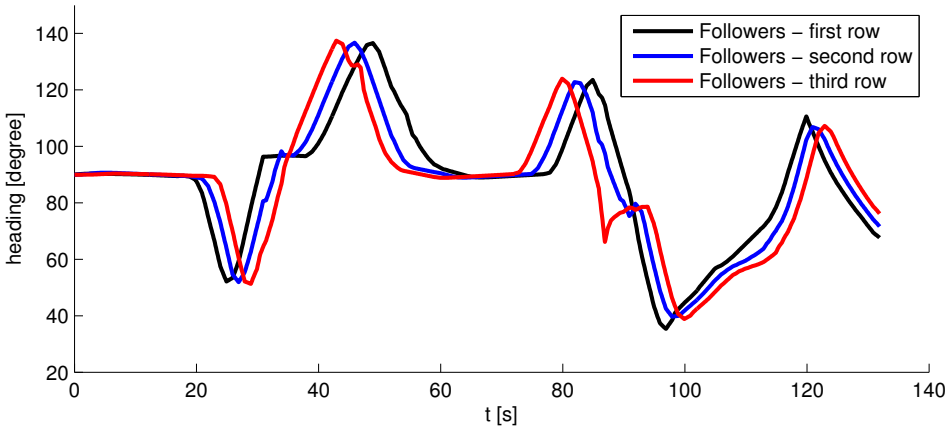
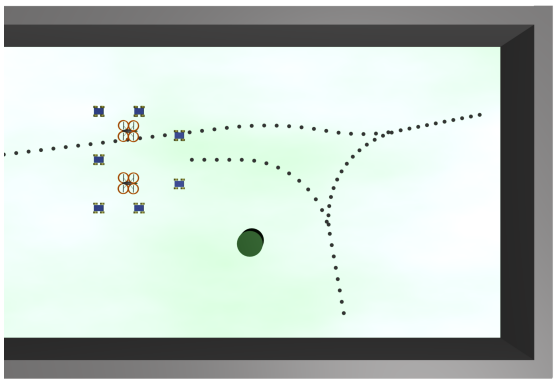


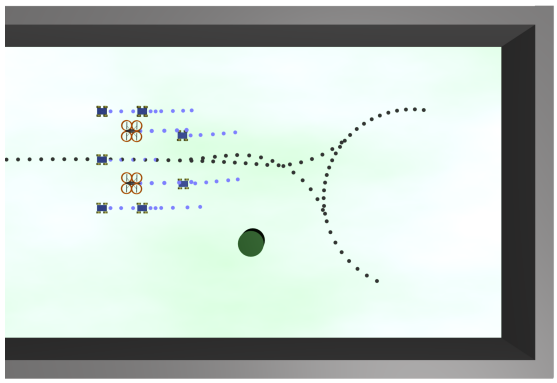
Figure 23: The average values of the heading of followers in rows during the simulation presented in section 6.2.2.

6.3 Summary

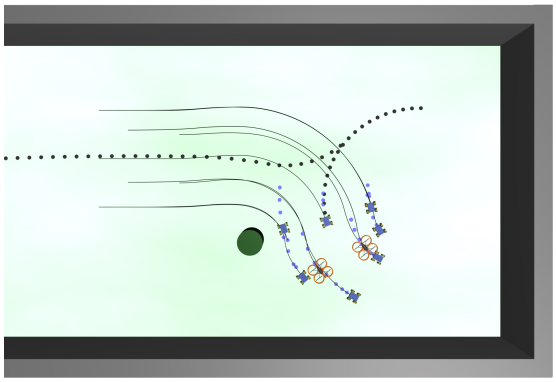
The problem of the proposed formation driving method in situations where the change of the polarity of v_L has to be applied was introduced in this section. The concept of two alternating virtual leaders was mentioned as a solution of this problem. It is an approach that modifies the basic method with the virtual leader in such a way that plans with two virtual leaders, one for the forward movement and one for the backward movement. Their leading role is always switched when the polarity of the leader velocity is changed. So in one moment, one virtual leader leads and other virtual leader is in a role of a virtual follower. Further, the modified MPC method for finding the feasible collision free trajectory for both virtual leaders was introduced and experimentally verified.



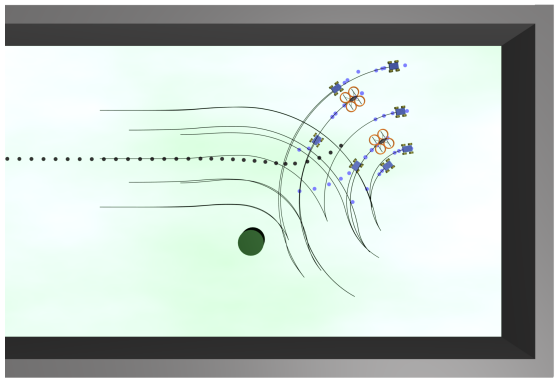
(a) The initial position of the formation with an initial plan for both virtual leaders.



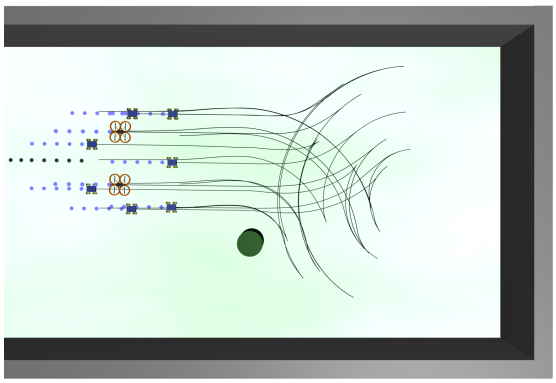
(b) Complete plan for both virtual leaders found in the first planning step of the MPC method.



(c) The second virtual leader overtakes the leadership.



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



(e) The paths passed by the members of the formation in order to accomplish the task.

Figure 24: Snapshots of turning 180 degrees on a blind narrow road.

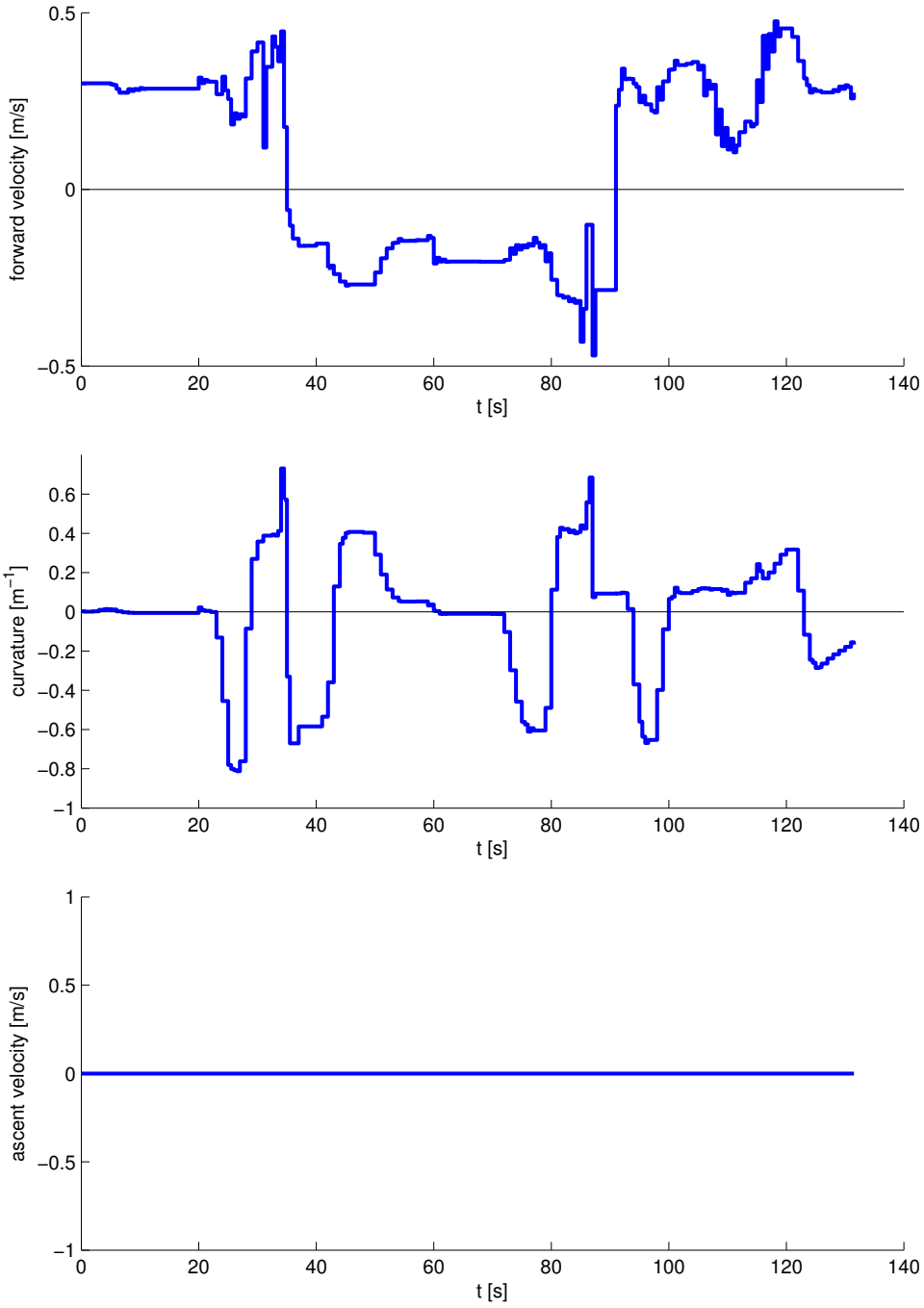
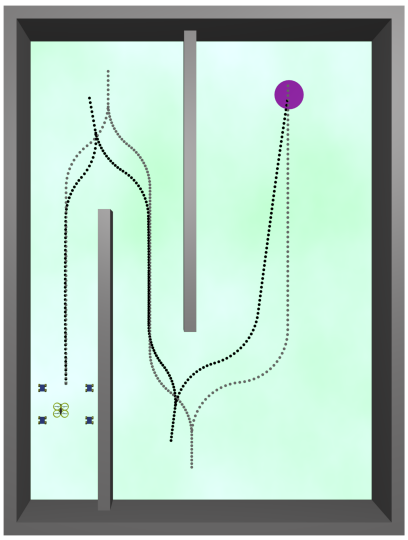
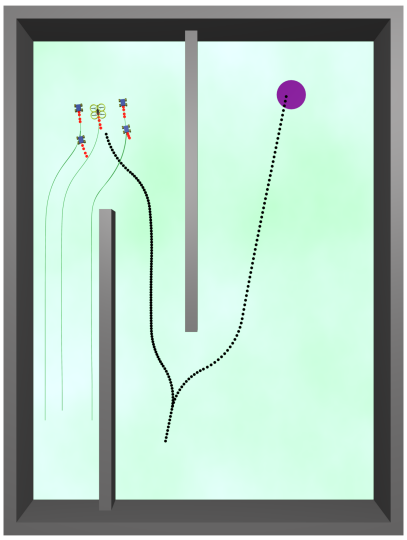


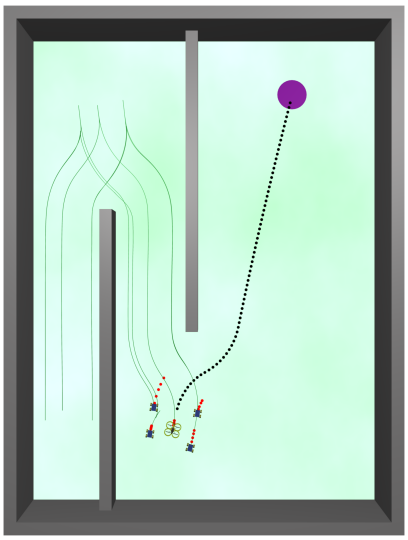
Figure 25: Applied control inputs of the third follower in the experiment presented in section 6.2.2.



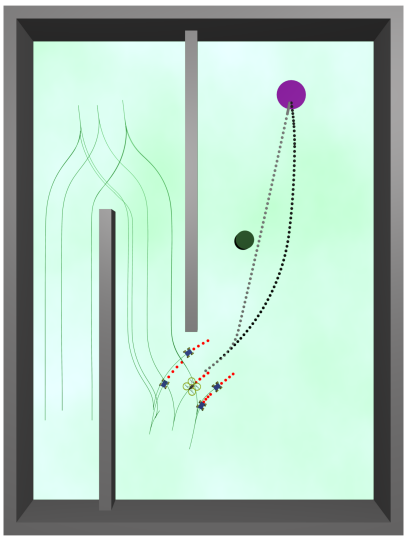
(a) The initial position of the formation with a complete plan for both virtual leaders found in the first planning step. The initial plan is denoted by gray points.



(b) The second virtual leader overtakes the leadership.

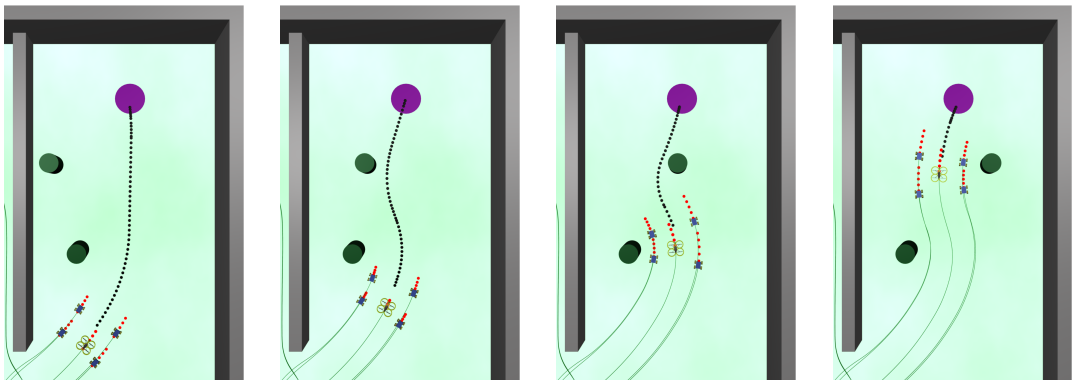


(c) The leadership is returned to the first virtual leader.



(d) The result of replanning after detection of a static obstacle during movement. The plan found in previous planning step is denoted by gray points.

Figure 26: Snapshots of the experiment presented in section 6.2.2.



(a) (b) (c) (d)

Figure 27: Applied trajectories after detecting a dynamic obstacle during the movement in the experiment presented in section 6.2.2. The motion model of the obstacle is known. The proposed system uses this information for prediction of the positions of the obstacle in the future. Therefore, the system found collision free plan for the formation movement.

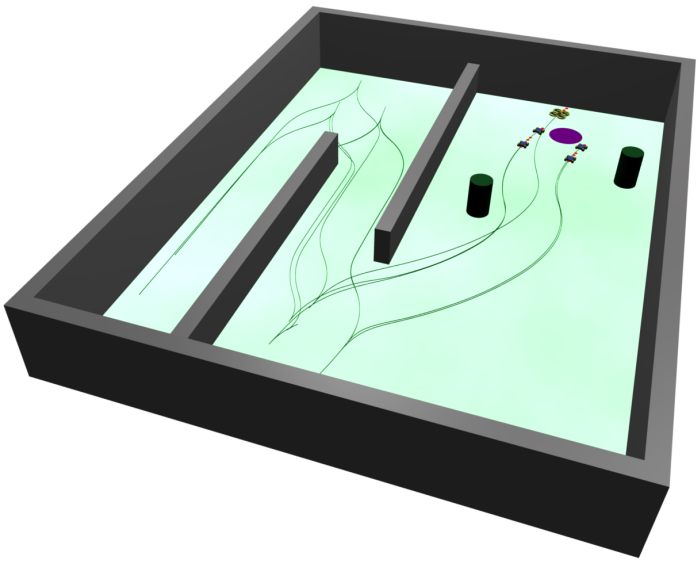


Figure 28: Trajectories passed by followers in the experiment presented in section 6.2.2.

7 Statistical analyses of system performance

In the first part of this section, the behaviour of the proposed system will be statistically analysed in dynamic and partially unknown environment. Influence of different settings of the algorithm on the quality of obtained trajectories and time complexity will be evaluated in the next part.

7.1 Analysis of the movement of the formation in dynamic and partly unknown environment

The proposed way how to use trajectory provided by the RRT2MPC algorithm as the initial trajectory for the MPC method used in this thesis for trajectory planning to the desired area was described in section 5.4.2. The trajectories provided by the RRT2MPC algorithm under the same initial conditions are not typically equal. The reason of this is that the resulting trajectory of the RRT2MPC algorithm is created from the tree that is done by the random expansion.

In this subsection, the capability of this approach, that uses the RRT2MPC algorithm as the initial trajectory for the MPC method to find collision free plan in dynamic and partially unknown environment (an obstacle is detected during the movement of the formation) is presented. For this purpose, scenario where the formation is composed of five robots has to reach the desired area throw an environment with one undetected dynamic obstacle was chosen. Parameters of the formation are described in Table 5. The initial position of the formation and the environment where the formation has to perform the movement is presented in Figure 29. This scenario was solved 50 times using the mentioned method. Results of each simulation are the total time to reach the target region, and minimum distances from the virtual leader and from the followers to the walls and to the obstacle during accomplishing the task.

i	1	2	3	4	5
p_i	0	0	0.4	0.4	1.1
q_i	0	-0.6	0.6	-0.8	0.8
h_i	0	1	1	0	0

Table 5: Curvilinear coordinates of the followers in the formation.

The analysis of the total times to reach the target region is shown in Table 6 and is visualised in Figure 30. The analysis of the resulting minimum distances from the virtual leader and from the followers to the walls are shown in Table 7 and to the obstacle in Table 8. Travelled trajectories for accomplishing the task were collision free in all simulations because the minimum distances are positive values only.

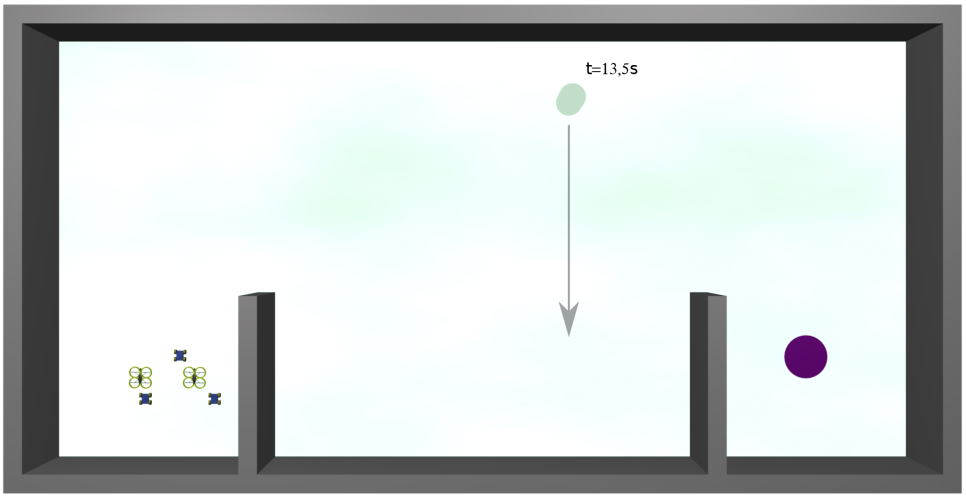


Figure 29: The initial position of the formation, the target area, position of an obstacle (its time of detection and the direction of its movement) in environment used in the experiment presented in section 7.1.

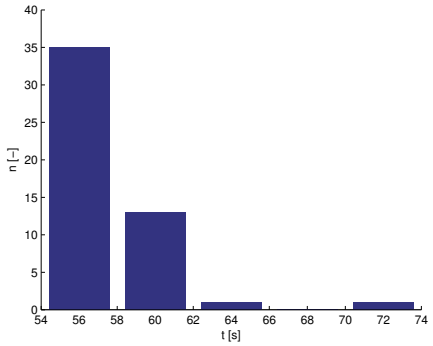


Figure 30: Graph of total times of simulations obtained from 50 runs.

	min	max	mean
total time [s]	56	72	57.36

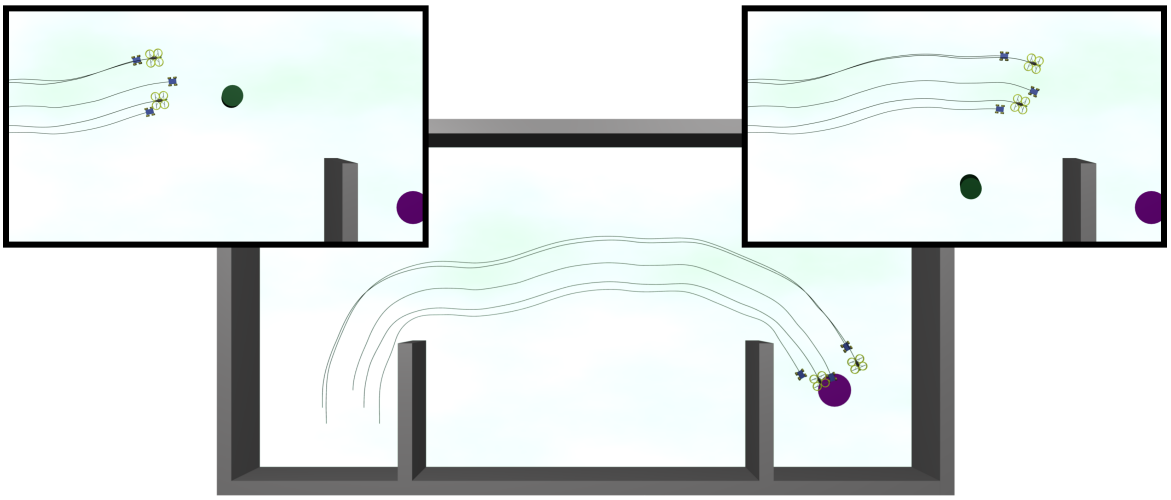
Table 6: Total times to reach the target region obtained from 50 runs.

	leader	follower 1	follower 2	follower 3	follower 4	follower 5
min	0.0052 m	0.8104 m	0.8243 m	1.6733 m	0.1519 m	1.006 m
max	0.1288 m	0.9284 m	0.8807 m	1.7836 m	0.2489 m	1.006 m
mean	0.1103 m	0.9100 m	0.8713 m	1.7669 m	0.1816 m	1.0006 m

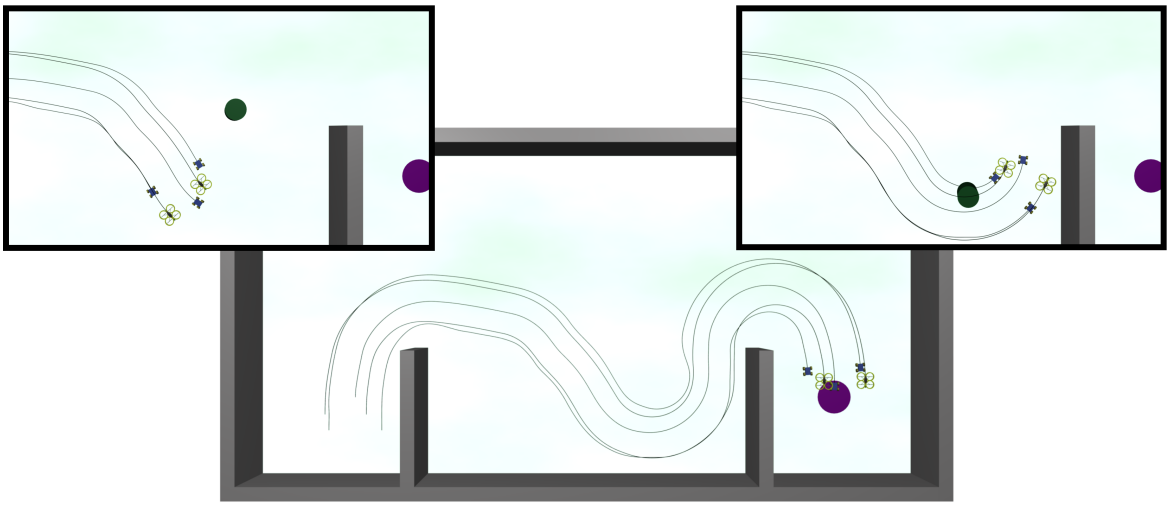
Table 7: Minimum distances from the virtual leader and from the followers to the walls during accomplishing the task obtained from 50 runs.

	leader	follower 1	follower 2	follower 3	follower 4	follower 5
min	0.0246 m	0.8199 m	0.8149 m	0.6505 m	0.3412 m	0.1455 m
max	0.2406 m	1.0453 m	1.5295 m	1.9906 m	1.2515 m	2.1778 m
mean	0.0749 m	0.8779 m	0.9706 m	1.7897 m	0.7430 m	2.0392 m

Table 8: Minimum distances from the virtual leader and from the followers to the obstacle during accomplishing the task obstacles obtained from 50 runs.



(a) Total time to reach the target region = 56s



(b) Total time to reach the target region = 72s

Figure 31: An example of trajectories passed during the simulation.

7.2 Influence of different settings of parameter $t_{s,M}$

In this subsection, the influence of different settings of time interval $t_{s,M}$ of the RRT2MPC algorithm on the provided trajectory will be presented. The time interval $t_{s,M}$ has effect on the expansion of the tree generated by the proposed RRT2MPC algorithm. 100 trajectories were generated for each of the seven different settings of the value of $t_{s,M}$ in the environment presented in Figure 32 using AMD A4-3300M CPU 1.9GHz. Results of a statistical test of performance of the algorithm are shown in Table 9.

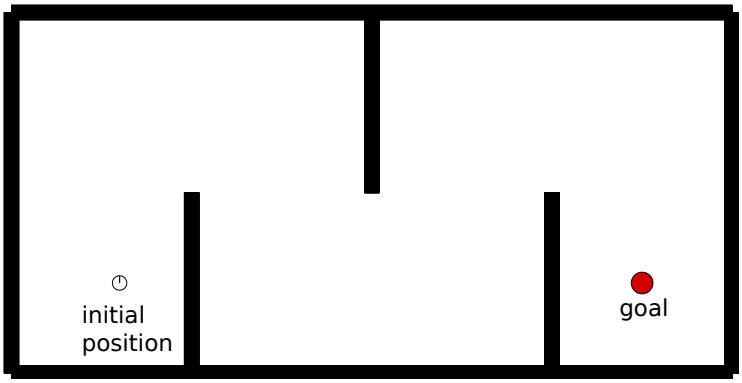
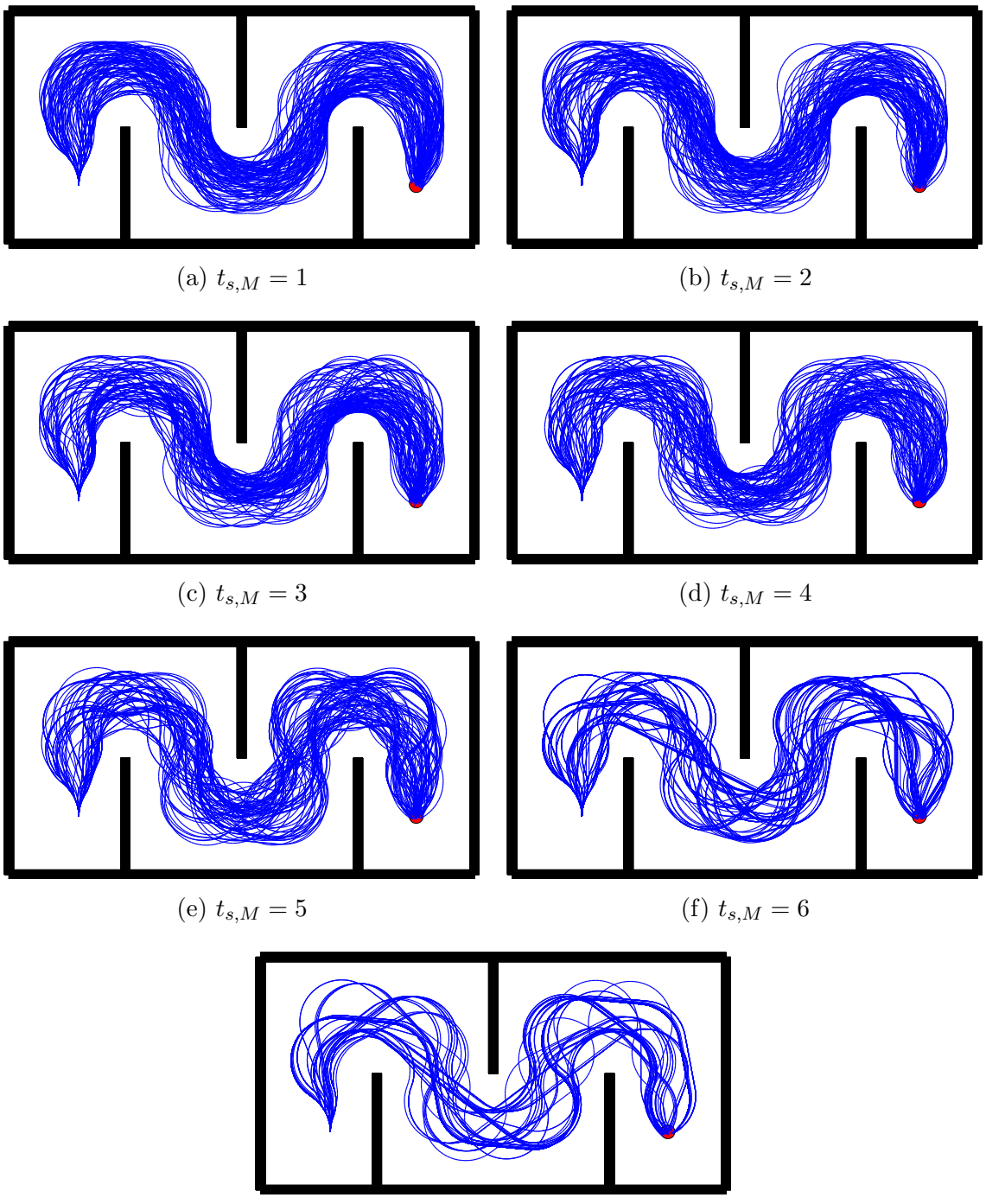


Figure 32: The initial position of the virtual leader and the target region in an environment.

$t_{s,M}$ [s]	1	2	3	4	5	6	7
comput. time [ms]	278.15	159.42	195.79	159.7	175.5	415	919
length [m]	29.5954	29.1634	28.5120	29.2830	31.4197	31.2237	32.8154
length of Ω_L [-]	38.3	24.29	19.4	17.68	16.36	15.19	15.99

Table 9: Mean computation time, average length of the trajectory and average length of vector Ω_L that describes the trajectory for different setting of $t_{s,M}$ obtained from 100 runs.

The results presented in Table 9 show that different settings of parameter $t_{s,M}$ have influence on the length of the trajectory, the computation time and the length of vector Ω_L that describes the trajectory. The setting of $t_{s,M}$ between 1s and 5s should be the best choice when the minimum computing time is preferred as is in real-time systems. However, small value of $t_{s,M}$ has effect on the length of vector Ω_L therefore the RRT2MPC algorithm provides a trajectory with many changes of the values of control inputs (which is not convenient).



(a) $t_{s,M} = 1$ (b) $t_{s,M} = 2$

(c) $t_{s,M} = 3$ (d) $t_{s,M} = 4$

(e) $t_{s,M} = 5$ (f) $t_{s,M} = 6$

(g) $t_{s,M} = 7$

Figure 33: Trajectories found by the RRT2MPC method with different settings of the value of $t_{s,M}$.

8 Conclusion

This diploma thesis deals with control and trajectory planning for heterogeneous formations of ground and aerials vehicles based on the approach described in [15] and [19]. The core of our method is based on leader-follower technique and on the model predictive control approach that offers solutions considering known structure of the environment for finding a feasible collision free trajectory. The method is suitable for any group of robots where the visual relative localization system is used.

The first task of this thesis is to implement a modification of the approach using an initial trajectory provided by the method based on the Rapidly-exploring Random Trees (RRT). This task is solved in section 5 in which a proposed modification is described and experimentally verified.

The second goal of this thesis is to extend system for control and stabilization of heterogeneous teams of ground robots and helicopters with the possibility of making complex maneuvers. In section 6, an extension with two alternating virtual leaders was introduced. It is an approach that modifies the basic method with the single virtual leader in such a way that it plans with two virtual leaders, one for the forward movement and one for the backward movement. The extended approach is then suitable for finding the complete plan that considers complicated maneuvring (repeated reverse of the movement) for accomplishing the task while compact shape of the formation is preserved.

The statistical analysis of the behaviour of the system and influence of different settings of the system is presented in section 7. The analyse shows the capability of our approach, that uses the modified RRT algorithm as the initial trajectory for the MPC method, to find collision free plan in dynamic and partially unknown environment.

Correct functionality of particular algorithms and the entire system were experimentally verified in various scenarios in which the formation succeeded to move from the initial position to its destination without loss of the shape and direct visibility between the members of the formation. Therefore, all goals of the thesis were completely fulfilled.

The presented work exceeds thesis assignment by contributing in relevant research areas of Department of Cybernetics at CTU in Prague. Author of this thesis significantly contributed as a co-author in the following publications. A trajectory planning method for formations of ground vehicles using a similar MPC mechanism as proposed in this thesis is presented in [16]. A study of failure tolerance principles in MPC formation control of heterogeneous MAVs-UGVs groups is proposed in [13]. Finally, an extension of the system in [16] that enables formation driving along predefined spline paths is presented in [17].

Furthermore, relevant results in the field of formation control, which were achieved by other members of Multi-robot Systems group, can be found in [19, 12, 14, 22, 18, 21, 20].

8.1 Future work

The extension of the proposed approach with two alternating virtual leaders enables to find a solution of the movement of the formation, if the kinematic model of car-like robots is used. However, if the formation is composed only from aerial vehicles the movement constraints of the formation will be different than for the heterogeneous formation with at least one ground robot. The future work could find and integrate a method that enables complex maneuvers of groups composed only from unnamed aerial vehicles.

Bibliography

[1] Tucker Balch and Ronald C Arkin. Behavior-based formation control for multirobot teams. *Robotics and Automation, IEEE Transactions on*, 14(6):926–939, 1998.

[2] J. Barraquand, B. Langlois, and J.-C. Latombe. Numerical potential field techniques for robot path planning. *Systems, Man and Cybernetics, IEEE Transactions on*, 22(2):224–241, Mar 1992.

[3] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[4] František Duchoň, Andrej Babinec, Martin Kajan, Peter Beňo, Martin Florek, Tomáš Fico, and Ladislav Jurišica. Path planning with modified a star algorithm for a mobile robot. *Procedia Engineering*, 96(0):59 – 69, 2014. Modelling of Mechanical and Mechatronic Systems.

[5] J. Faigl, T. Krajnik, J. Chudoba, L. Preucil, and M. Saska. Low-Cost Embedded System for Relative Localization in Robotic Swarms. In *ICRA2013: Proceedings of 2013 IEEE International Conference on Robotics and Automation*, pages 985–990, Piscataway, 2013. IEEE.

[6] Steven M LaValle. Rapidly-exploring random trees a new tool for path planning. *TR 98-11, Computer Science Dept., Iowa State University*, 1998.

[7] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[8] Steven M LaValle and James J Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[9] M.Anthony Lewis and Kar-Han Tan. High precision formation control of mobile robots using virtual structures. *Autonomous Robots*, 4(4):387–403, 1997.

[10] Tomás Lozano-Pérez and Michael A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM*, 22(10):560–570, October 1979.

[11] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.*, 21(4):25–34, August 1987.

[12] M. Saska, Z. Kasl, and L. Preucil. Motion Planning and Control of Formations of Micro Aerial Vehicles. In *Proceedings of The 19th World Congress of the International Federation of Automatic Control*, pages 1228–1233, Pretoria, 2014. IFAC.

[13] M. Saska, T. Krajnik, V. Vonasek, Z. Kasl, V. Spurny, and L. Preucil. Fault-Tolerant Formation Driving Mechanism Designed for Heterogeneous MAVs-UGVs Groups. *Journal of Intelligent and Robotic Systems*, 73(1-4):603–622, January 2014.

[14] M. Saska, T. Krajnik, V. Vonasek, P. Vanek, and L. Preucil. Navigation, Localization and Stabilization of Formations of Unmanned Aerial and Ground Vehicles. In *Proceedings of 2013 International Conference on Unmanned Aircraft Systems*, pages 831–840, New York, 2013. Springer.

[15] 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 Multiconference on Systems and Control*, 2009.

[16] M. Saska, V. Spurny, and L. Preucil. Trajectory Planning and Stabilization for Formations Acting in Dynamic Environments. In *Progress in Artificial Intelligence*, pages 319–330, Heidelberg, 2013. Springer.

[17] M. Saska, V. Spurny, and V. Vonasek. Predictive control and stabilization of nonholonomic formations with integrated spline-path planning. In *Submitted to Robotics and Autonomous Systems*, 2015.

[18] M. Saska, V. Vonasek, T. Krajnik, and L. Preucil. Coordination and Navigation of Heterogeneous UAVs-UGVs Teams Localized by a Hawk-Eye Approach. In *Proceedings of 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 2166–2171, Piscataway, 2012. IEEE.

[19] M. Saska, V. Vonasek, T. Krajnik, and L. Preucil. Coordination and navigation of heterogeneous MAV-UGV formations localized by a 'hawk-eye'-like approach under a model predictive control scheme. *International Journal of Robotics Research*, 33(10):1393–1412, September 2014.

[20] M. Saska, V. Vonasek, and L. Preucil. Control of ad-hoc formations for autonomous airport snow shoveling. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010*, volume 1, pages 4995–5000, Taipei, 2010. IEEE Industrial Electronics Society.

[21] M. Saska, V. Vonasek, and L. Preucil. Roads Sweeping by Unmanned Multi-vehicle Formations. In *ICRA2011: Proceedings of 2011 IEEE International Conference on Robotics and Automation*, pages 631–636, Madison, 2011. Omnipress.

[22] M. Saska, V. Vonasek, and L. Preucil. Trajectory Planning and Control for Airport Snow Sweeping by Autonomous Formations of Ploughs. *Journal of Intelligent and Robotic Systems*, 72(2):239–261, November 2013.

[23] Martin Saska. Trajectory planning and optimal control for formations of autonomous robots. *Schriftenreihe Würzburger Forschungsberichte in Robotik und Telematik, Band 3*, 2011.

[24] Ritu Tiwari. *Intelligent Planning for Mobile Robotics: Algorithmic Approaches: Algorithmic Approaches*. IGI Global, 2012.

Appendix A CD Content

In Table 10 are listed names of all root directories on CD.

Directory name	**Description**
Spurny_DT.pdf	diploma thesis in pdf format.
DT_sources.zip	LaTeX– source code of the diploma thesis
code.zip	source codes of program
180_degrees.avi	video from the experiment in section 6.2.1
complicate_environment.avi	video from the experiment in section 6.2.2
experiment_1.zip	data from the experiment in section 7.1
experiment_2.zip	data from the experiment in section 7.2

Table 10: CD Content