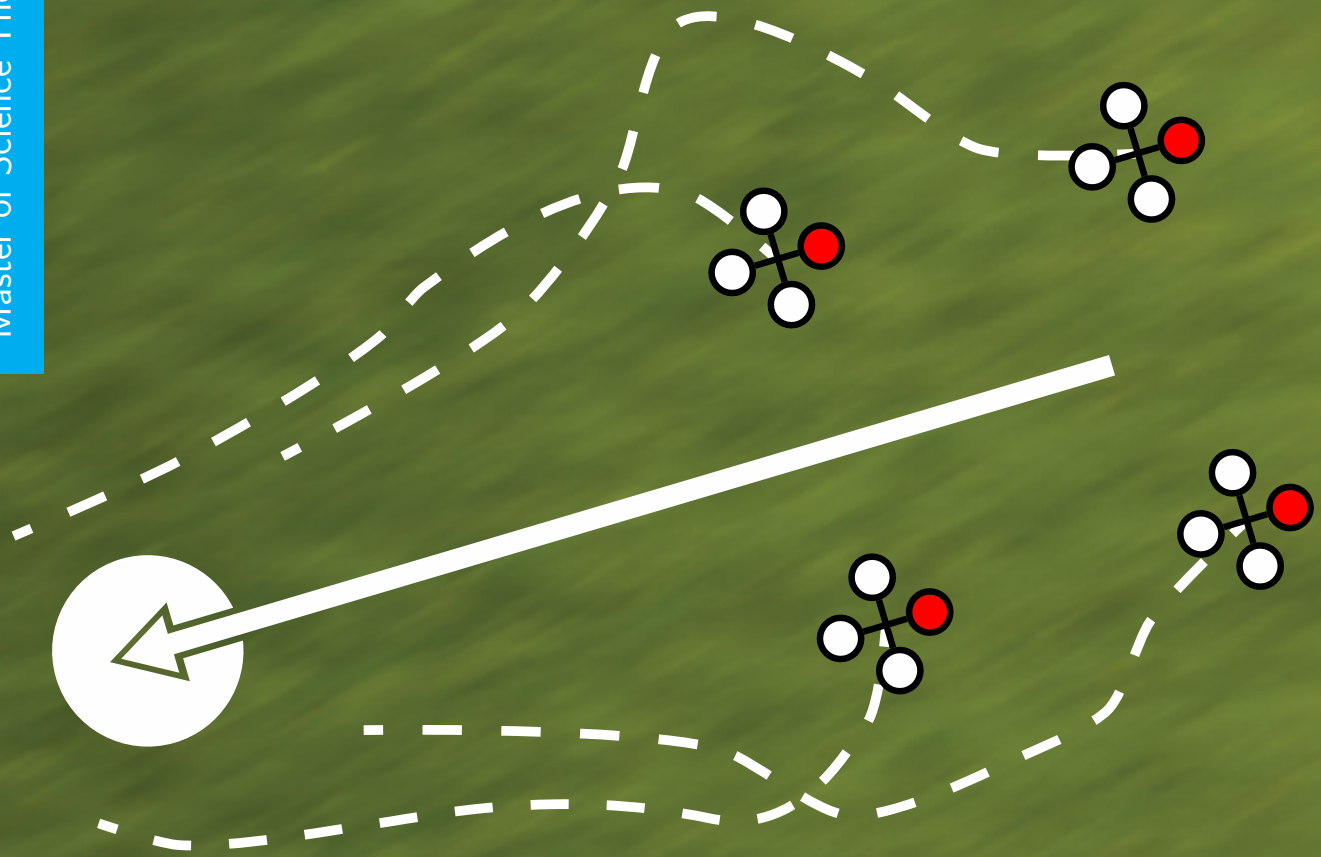
Splitting and Merging of Quadrotor Teams Flying in Formation

Jelle Juhl

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Splitting and Merging of Quadrotor Teams Flying in Formation

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Jelle Juhl

October 31, 2017

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Splitting and Merging of Quadrotor Teams Flying in Formation

by

Jelle Juhl

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: October 31, 2017

Supervisor(s):

dr. Javier Alonso-Mora

Reader(s):

dr.ir. Tamás Keviczky

dr. Laura Ferranti

dr. Riccardo Ferrari

Abstract

Quadrotor UAV's have become extremely popular over the last decade, as they combine great agility with mechanic simplicity. For tasks such as area surveillance or maintaining a communication network, the deployment of multi-agent systems is required. In these scenarios, it often is necessary for the robot teams to navigate in formation while avoiding static and dynamic obstacles. In situations, where team keeping is not crucial it can be beneficial to split and merge robot teams. This could be the case, if multiple targets are supposed to be tracked, or due to obstacle avoidance in order to improve goal progress and help avoiding deadlocks.

Within this thesis a novel approach for splitting and merging of robot teams moving in formation is developed. The method extends an existing geometric method for local formation control, which uses the approximation of typically non-convex workspaces by obstacle free convex regions. The regions are used to constrain a non-linear cost function, which is minimized through Sequential Convex Programming (SCP). This determines the references position for the robots and furthermore allows for the adjustment of the formation shape according to the environment. This thesis presents a goal directed version of the obstacle-free convex regions. These regions are used to determine team splitting and merging based on region intersections. The method takes into account limited sensing and communication range of the agents. Furthermore computations are done in a distributed manner to allow for better scaling with the number of team members.

The presented method is validated through simulations in various scenarios. Furthermore, an experimental framework is recreated and experiments with up to four quadrotor UAVs of the type Parrot Bebop 2 are conducted. The approach shows sufficient real-time capabilities in static and dynamic environments, while maintaining safe navigation of the drones.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor dr. Javier Alonso-Mora for his assistance during the master thesis project.

Delft, University of Technology
October 31, 2017

Jelle Juhl

Chapter 1

Introduction

1-1 Organization

The report of this thesis is organized as follows. The following part of Chapter 1 contains the motivation for the thesis topic, followed by the related works and the main contributions. In Chapter 2, the preliminary definitions are stated, which are used throughout the report. Additionally, the research question is defined and an overview over the presented method is given. The conditions which are supposed to lead to splitting and merging of robot teams are derived in Chapter 3. The detailed explanation of the method and the parts which it consists of, are given in Chapter 4. The method is evaluated through simulations first. Results for teams with the scope of general performance, method robustness and computational complexity are presented and discussed in Chapter 5. Experiments are conducted with up to four quadrotor UAVs. The experimental setup is explained and results in static and dynamic environments are presented and discussed in Chapter 6. Finally, conclusions on the master thesis project and recommended options for future work are given in Chapter 7.

1-2 Motivation

While robots were initially controlled individually, research now also focuses on the control of multiple robots at the same time.

As practical examples, the case of area surveillance, setting up a communication network, or automated factories, soon require the deployment of multi-robot systems. To execute those tasks or to minimize the occupied workspace of multiple robots moving from one location to another, the robot teams may be required to navigate in formation and switch in between several formation shapes according to the environment. While past most approaches for multi-robot formation control assumed obstacle-free workspaces or static obstacles, recent works, such as [2], [3] and [4] among others, take into account static and dynamic obstacles.

However, these algorithms so far do not allow for splitting and merging of the robot teams, or do not account for the limited sensing capabilities of the robots, with the exception of

[3]. In the view of the author, splitting and merging of quadrotor teams can be extremely beneficial. It can be required to split a robot team if multiple references with corresponding number of agents are supposed to be tracked. Target assignment of each robot can in this case be done according to [5] or [6], which execute a marked-based coordination algorithm or solve a distributed linear program. Another trigger for splitting of formations can be obstacle avoidance.

Not allowing obstacles within the team, or intending to avoid the obstacle as an entire team on one side might be necessary if formation keeping is crucial. This may however increase the change for deadlock scenarios to occur, or leads to suboptimal trajectories for some robots. By suboptimal it is referred to the case that robots would individually prefer to avoid the obstacle on a different side than determined through the formation control algorithm. In cases where formation keeping is not crucial, it can therefore be beneficial to reorganize the team into several teams which then avoid the obstacle of different sides while each team moves in formation. After the avoidance manoeuvre, teams re-merge if possible.

As the splitting and merging due to tracking multiple targets can simply be solved with one of the above mentioned approaches, the focus of this master thesis lies on the splitting and merging of robot formations due to obstacle avoidance, as the approach of [3] does not cover this topic.

The developed framework can then serve as a foundation, which can be extended to allow for temporary merging for robot teams with similar paths in proximity to each other.

Over the last decade, quadrotor Unmanned Aerial Vehicle (UAV)s have become extremely popular due to their high agility, relatively low cost and wide operating range. As a consequence quadrotor UAVs, more precisely several drones of the type Parrot Bebop 2, are the robot type used to evaluated the approach through experiments.

1-3 Related Works

Splitting and merging behaviour of robot teams is achieved in different ways so far.

The proposed method of [7] builds on the well known flocking algorithm introduced by [8]. Here, splitting and merging of robot teams is achieved by adding additional repulsive forces in between the agents which are supposed to be split. Different to the here proposed method, the approach assumes an obstacle free workspace and does not allow for operator defined formation shapes. The approach presented by [9] combines artificial potentials with a leader-follower formation control to achieve splitting and merging behaviour due to collision avoidance, however only for static obstacles in a two dimensional environment.

An approach taking into account a static environment of [10] includes a switching control law in between leader- and follower-mode for the agents to achieve splitting and merging. It leads to a hybrid system, where the transition in between the systems corresponds to splitting and merging. A receding horizon dynamic window approach is used for goal progress and collision avoidance. Formation keeping is achieved by minimizing a control Lyapunov function as in [11]. Furthermore, it assumes static two dimensional environments.

A global method for static obstacles in combination with a two dimensional workspace is presented in [12], where robot team splitting and merging actions are added to an augmented

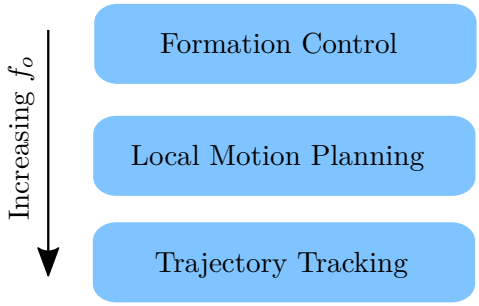


Figure 1-1: Scheme of the three control layers usually involved, with operating frequency f_o.

A* algorithm. It relies on the robust local clearance triangulation of the workspace [13] and represents the robot team through covered area within the workspace. The approach results in team splitting and merging in addition to paths for the centroids of the teams. It does not include formation control however. In addition and at the same time in contrast to the here presented local method, it does not ensure real-time capabilities due to the computational complexity of the approach.

The approach introduced by [3] is based on [14] and is formulated as a distributed receding horizon optimization problem. It accounts for limited communication range of the agents using a communication graph. The team splitting and merging is triggered due to certain changes in the communication graph, while the approach presented here uses the comparison of Goal-Directed Obstacle-Free Convex Regions (GDOFCR) in order to determine team splitting and merging. Furthermore, the approach here allows for automatically choosing from different formation shapes according to the environment.

Multi-robot navigation in formation with obstacle avoidance is often solved, as illustrated in Figure 1-1, in a cascade way combining formation control, motion planning and trajectory tracking in order to assure the real-time capabilities of the algorithms. Furthermore, the formation control can be divided into the categories of controlling unstructured formations or structured formations. The first category evolved from the imitation of animal flocks and ensures that agents stay within proximity to each other while navigating in an environment, at the same time avoiding collision with each other and obstacles. The ones from the latter category aim for achieving the same, with the additional condition of the agents moving as a defined geometric pattern.

The work on multi-robot navigation in formation includes several methods containing reactive behaviours. Unstructured formation control was introduced by [8]. Structured formations are achieved by behaviour based formation control [15] and artificial potentials [16], [17]. A similar approach using directed potentials based on formation graphs is [18]. All the approaches are based on attractive and repulsive forces caused by inter-agent distances and velocities. Consequently, these approaches are computationally cheap. Due to their reactive nature, they struggle in highly dynamic environments.

The concept of representing the formation as a virtual rigid body moving in space is introduced by [19] and later used and tested in experiments by [20] and [21]. In contrast, the algorithm proposed here treats each agent as an individual and allows for reconfiguration, thus being able to adapt to the environment more quickly.

A different approach for controlling rigid formations is presented by [22]. It is based on the combination of gradient-distance based formation control with graph rigidity theory to allow for continuous scaling of rigid formations. However, the approach assumes an obstacle free workspace.

Real-time capable optimization based approaches are presented by [23], which solves an incremental Sequential Convex Program (SCP) for multiple agents but without considering formations and for static obstacles only.

The approach of [24] solves an SCP in the centralized case while accounting for reconfiguration and dynamic obstacles. The approach is later distributed in [2], which serves as the formation control foundation of the here proposed method. Both approaches are able to compute locally optimal formation and rely on the computation of obstacle free convex regions computed according to [25]. Furthermore, the latter approach takes into account limited sensing and communication range.

Further distributed formation control approaches contain the method presented in [4], which solves a receding horizon optimization problem in a distributed fashion using the Alternating Direction Method of Multipliers (ADMM) [26]. A different distributed receding horizon MPC scheme for formation control of UAVs is the above mentioned approach by [3], where each agent solves a Mixed Integer Linear Program (MILP).

Motion planning algorithms for multiple quadrotor UAVS contain global optimization based control methods, such as [27] or [28] among others. The first one solves a MILP, whereas the latter one formulates the problem as a Mixed Integer Quadratic Program (MIQP). Due to their global guarantees, the approaches are computationally expensive and do not have real-time capabilities, which is one aspect of this work.

The approach of [29] is based on the principle of Velocity Obstacle (VO). It computes globally collision free paths through graph search. On-line computation can be achieved by only expanding one node and leads to locally collision free paths. This approach is augmented by [30], to account for multiple decision taking agents leading to reciprocal collision avoidance.

The approach of [31] solves an SCP to cope with non-convex constraints, which are usually present in case of obstacle avoidance. Several MPC control schemes for local motion planning are presented including, among others, [32] and [33].

Additional approaches contain [34], which is based on Probabilistic Roadmaps (PR). The approach of [35] builds one the concept of Rapidly Exploring Random Trees (RRT). Both methods execute a graph search algorithm and therefore struggle with real-time performance in a dynamic three dimensional environment, which would lead to a four dimensional search space. The approach of [36] uses artificial potentials for the motion planning of a quadrotor UAV in a three dimensional dynamic environment.

A great variety of quadrotor trajectory tracking is available. They range from linear model based controllers, such as PID-type [37], Linear Quadratic (LQ) controller [38], robust control schemes [39], MPCs including [40] and [41] to non-linear control schemes. The latter category inludes approaches entirely formulated on $SE(3)$ [42], or build on the concept of differential flatness [43]. The latter approach is used for simulations in this thesis. Feedback linearization can be applied as in [44]. Futhermore, a non-linear MPC scheme is introduced in [45].

1-4 Contribution

The main contribution of this thesis is a novel distributed method for splitting and merging of robot teams moving in formation using the approach of [2], which allows for reconfiguration and choosing from different formation shapes according to the environment. The splitting and merging is induced by obstacle avoidance and the method takes into account limited communication range and a limited field of view.

Verifiable conditions for team splitting and merging are defined.

An adaptation of obstacle-free convex regions [25] is presented. These goal-directed obstacle-free convex regions GDOFCR, which include a collision avoidance action implicitly, are used to determine splitting and merging. Afterwards these are reused to constrain the optimization problem which determines a locally optimal formation state.

This allows ground and aerial robots to move in reconfigurable formations and avoid static and dynamic obstacles. While doing so, the formations split or merge with other formations in order for formation members to have locally optimal trajectories, which are homotopic. The individual formations are not connected to each other, in order to respect the limited communication range of the agents.

Good results of the approach are shown trough simulations with up to 16 quadrotor UAVs. Furthermore, an experimental setup is recreated. Here, the method is shown to have real-time capabilities, which is verified with up to four quadrotor UAVs in static and dynamic environments.

Chapter 2

Preliminaries

This Chapter provides needed definitions for the presentation of the method for splitting and merging of robot teams navigating in formation. The problem is formulated, leading to the research question to be answered in this thesis. Afterwards, an outline of the proposed method is given.

2-1 Definitions

In this Section, the necessary definitions for the proposed method are given.

2-1-1 Robots and Formation

Robots

The robots considered are of the type quadrotor UAV, whose dynamics can be modelled as in [46]. Furthermore, all quadrotors UAVs are assumed to be identical. Consequently, they contain the the same dynamic model and cylindrical non-rotating shape of radius r and height $2h$. For each robot $i \in \mathcal{I}_k = \{1, \ldots, n_k\}$ within a team of robots $k \in \mathcal{K}$, its position at time t is described by $\boldsymbol{p}_i(t) \in \mathbb{R}^3$. The volume occupied by a robot at position $\boldsymbol{p}$ is denoted by $\mathcal{A}(\boldsymbol{p}) \subset \mathbb{R}^3$. The velocity of each robot is denoted by $\boldsymbol{q}_i(t) \in \mathbb{R}^3$.

Each robot i has a limited field of view, which is modelled as a sphere $\mathcal{B} \in R^3$ with radius $r_{\mathcal{B}}$ and origin at $\boldsymbol{p}_i$.

Furthermore, the communication range of the robots is limited to a sphere $\mathcal{C}$ defined by the radius $r_{\mathcal{C}}$ and its origin at $\boldsymbol{p}_i$. The set of robots with which robot i is able to communicate is denoted by $\mathcal{J}_i$.

Teams

Each set of robots which executes a task as a multi-robot system, is called 'a team of robots' from here on.

It is assumed that each robot team $k \in \mathcal{K}$ consists of a set of robots $i \in \mathcal{R}_k$ and shares a common reference $\boldsymbol{p}_{r,k}(t)$, which is known to all robots $1, \ldots, n_k$ within team k.

Furthermore, a preferred velocity $\boldsymbol{q}_{k,pref}(t) \in \mathbb{R}^3$ is assigned to each team of robots towards a common reference $\boldsymbol{p}_{r,k}(t)$. The preferred velocity corresponds to a simple proportional control law written as

$$\boldsymbol{q}_{k,pref} = q_{max} min\left(1, \frac{\|\boldsymbol{p}_r - \boldsymbol{p}_k\|}{d_p}\right) \frac{\boldsymbol{p}_r - \boldsymbol{p}_k}{\|\boldsymbol{p}_r - \boldsymbol{p}_k\|}. \tag{2-1}$$

Here, $\boldsymbol{p}_k$ represents the centroid of the robots position within team k. Furthermore, $q_{max} > 0$ denotes the maximum speed of the robots and $d_p > 0$ the distance from the goal from which $\boldsymbol{q}_{k,pref}$ is reduced linearly. It is assumed $\boldsymbol{q}_{k,pref}$ is known to all robots within team k.

Taking into account the limited communication range of each robot, it requires for modelling the communication structure within a team of robots k as a communication graph $\mathcal{G}_k = (\mathcal{I}, \mathcal{E})$. The robots are represented by the nodes $\mathcal{I}$, while the graph edges $(i, j) \in \mathcal{E}$ represent the possibility of direct communication in between robot i and robot j. The neighbours of each robot i are represented by $\mathcal{N}_i = \{j \in \mathcal{I} | (i, j) \in \mathcal{E}\}$. It is assumed that for each pair of robots i, j there exists a path in $\mathcal{E}$, which links the robots. This requires $\mathcal{G}$ to be a connected graph. The diameter $d_{k,\mathcal{G}}$ of $\mathcal{G}_k$ denotes the longest among all shortest paths in between a pair of robots. Additionally, the set of teams with which team k is able to communicate is denoted by $\mathcal{S}_k$. As the data exchange within the communication network is not in focus of this thesis, infinite bandwidth is assumed.

Formation

For each robot team $k \in \mathcal{K}$, a pre-defined set of template formations $f_t \in \mathcal{I}_f^k$ is considered. Each template formation is defined by a set of vertices $\left\{\boldsymbol{f}_1^{f_t}, \ldots, \boldsymbol{f}_{n_f}^{f_t}\right\}$ and a set of positions $\{\boldsymbol{r}_{f_t,1}, \ldots, \boldsymbol{r}_{f_t,n_k}\}$ for the robots, where n_f corresponds to the number of vertices defining the template formation f_t and n_k the number of robots in team k. Both sets are defined with respect to c_{f_t}, the center of rotation of the template formation f_t. Additionally, a minimum inter-agent distance within f_t is denoted by d_0. Such a vertex representation of each f_t is especially useful for formations including larger number of robots, as it reduces the complexity.

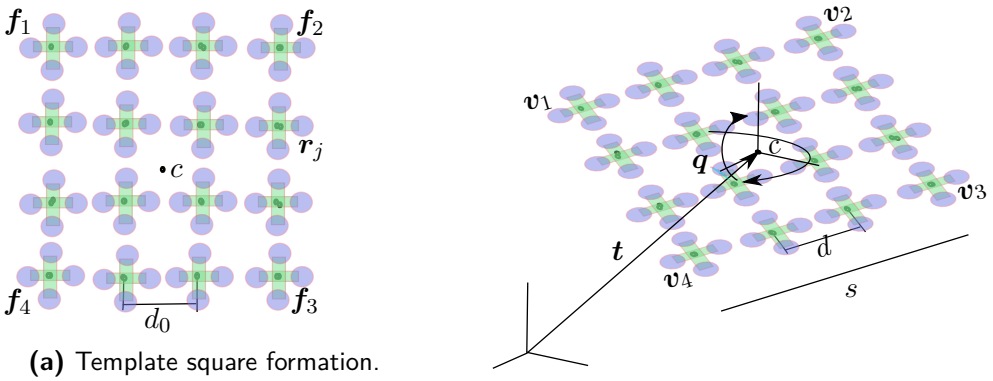
Each formation in space is then defined by an isomorphic transformation of a template formation f_t. This transformation consists of a translation $\boldsymbol{t} \in \mathbb{R}^3$, a rotation $R(\boldsymbol{q}_{rot})$ defined by a unit quaternion $\boldsymbol{q}_{rot} \in SO(3)$ and a scaling factor $s \in \mathbb{R}_+$. The rotation in $SO(3)$ due to the quaternion operation $R(\boldsymbol{q}_{rot})$ can be written as

$$\left[0, R(\boldsymbol{q}) \times f_i^{f_t}\right]^T = \boldsymbol{q}_{rot} \times \left[0, f_i^{f_t}\right] \times \bar{\boldsymbol{q}}_{rot} \tag{2-2}$$

This leads to the transformation of the vertices and positions as

$$\begin{aligned} \boldsymbol{v}_i^{f_t} &= t + sR(\boldsymbol{q}_{rot}) f_i^{f_t}, \\ \boldsymbol{r}_j^{f_t} &= t + sR(\boldsymbol{q}_{rot}) r_{f_t,j}. \end{aligned} \tag{2-3}$$

The transformation state is defined as $z = [\boldsymbol{t}, s, \boldsymbol{q}_{rot}] \in \mathbb{R}^3 \times \mathbb{R}_+ \times SO(3)$, which includes all necessary information to fully describe the transformation of any template formation in space.



(a) Template square formation.

(b) Transformed formation.

Figure 2-1: Scheme of the transformation of a template square formation for 16 quadrotor UAVs [1].

2-1-2 Obstacles and Workspace

Obstacles

The workspace can contain static as well as dynamic obstacles. Static obstacles are define first, followed by the dynamic obstacles.

Static Obstacles

Considering a set of static obstacles $\mathcal{O} \in \mathbb{R}^3$, the set of static obstacles visible for robot i is then defined by the set $\mathcal{O}_i \subset \mathcal{O}$. To include the cylindrical shape of the robots, $\mathcal{O}_i$ is inflated by half of the robot's volume. The set of positions for which the robot is in collision with static obstacles is then expressed as

$$\tilde{\mathcal{O}}_i := \left\{ \boldsymbol{p} \in \mathbb{R}^3 | \mathcal{A}(\boldsymbol{p}) \cap \mathcal{O}_i \neq \emptyset \right\}, \tag{2-4}$$

where $\tilde{\mathcal{O}}_i$ denotes the dilated set $\mathcal{O}_i$.

Dynamic Obstacles

Similarly to the static obstacles, dynamic obstacles are defined with respect to robot i. For each of dynamic obstacles $j \in \mathcal{J}_i = [1, \ldots, n_{i,DO} \subset N]$ observable by robot i, the occupied

volume is expressed by $\mathcal{D}_{i|j}(t) \subset \mathbb{R}^3$. Furthermore, the dilation as for the static obstacles is applied leading to

$$\tilde{\mathcal{D}}_{i|j}(t) := \left\{ \boldsymbol{p} \in \mathbb{R}^3 | \mathcal{A}(\boldsymbol{p}) \cap \mathcal{D}_i(t) \neq \emptyset \right\}. \tag{2-5}$$

The trajectory predictions of the dynamic obstacles are done by assuming that each dynamic obstacle j maintains its current velocity $\boldsymbol{q}_{oj}(t_0)$. Here, t_0 denotes the current time. Finally, the centroid for each obstacle is denoted by $\boldsymbol{p}_{o,j}(t) \in \mathbb{R}^3$.

Workspace

The obstacle free workspace is described by $\mathcal{W}_i(t_0) := \mathbb{R}^3 \times [0, \tau]$. In order to represent the obstacle free workspace $\mathcal{F}_i(t_0)$, the static and dynamic obstacles are represented for $[0, \tau]$ as

$$\hat{\mathcal{O}}_i(t_0) := \tilde{\mathcal{O}}_i \times [0, \tau] \bigcup_{\substack{t \in [0,\tau], \\ j \in \mathcal{J}_i}} \tilde{\mathcal{D}}_{i|j}(t_0 + t) \times t \quad \subset \mathbb{R}^4. \tag{2-6}$$

The time dimension added is here to account for dynamic obstacles within the time horizon τ. This leads to an obstacle representation in $\mathbb{R}^4$ being called position-time space and is similar to the representation of obstacles by [47]. An example of such obstacle representation is illustrated in Figure 2-2 for circular obstacles in a two dimensional workspace with time t being the third dimension.

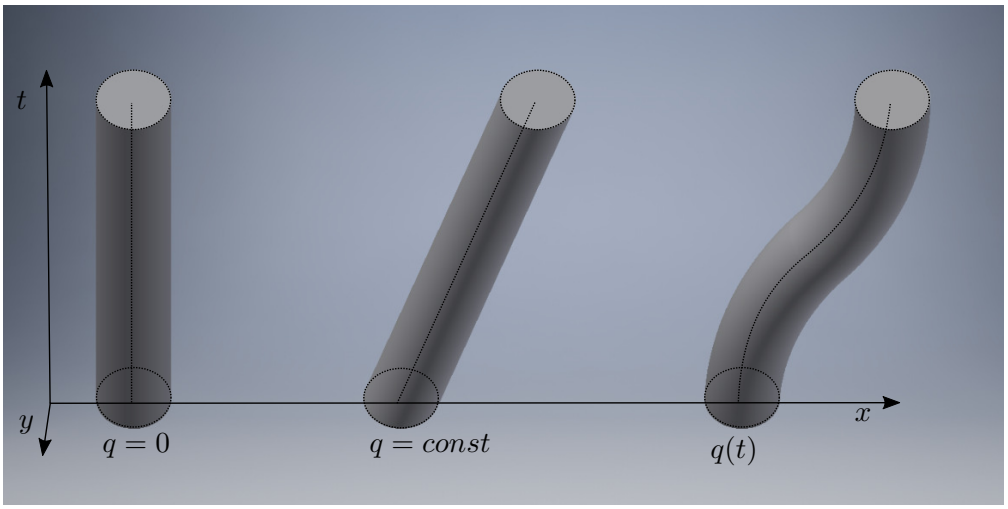


Figure 2-2: Representation of circular obstacles with different velocities in position-time space.

The obstacle-free position-time space for robot i is finally expressed as

$$\mathcal{F}_i(t_0) := \mathcal{W}(t_0) \setminus \hat{\mathcal{O}}_i(t_0) \subset \mathbb{R}^4 \tag{2-7}$$

and represents the set of positions in which robot i is not in collision with any of the static and dynamic obstacles for $t \in [t_0, t_0 + \tau]$.

If requested, further conservativeness can be added to the regions. In this case, analogously to the dilation of the set of static and dynamic obstacles, an extended dilatation can be applied

using the factor $\alpha_d > 1$. This factor is incorporated by virtually increasing the dimensions of the robots $r_d = \alpha_d r$, $h_d = \alpha_d h$, where r_d and h_d denote the dilated radius and height of the robot.

Obstacle Free Convex Space

A crucial part of the presented method is the representation of parts of the obstacle free space $\mathcal{F}$ through obstacle free convex regions, which can be denoted by

$$\mathcal{P} := \left\{ \boldsymbol{x} \in \mathbb{R}^4 | \boldsymbol{A}_c \boldsymbol{x} \leq b_c \right\}. \tag{2-8}$$

Here, the linear constraints $\boldsymbol{A}_c$ and b_c define the obstacle-free convex region. Furthermore, the ellipsoid

$$\mathcal{E}_c(\boldsymbol{C}_c, \boldsymbol{d}) = \left\{ \boldsymbol{x} \in \mathbb{R}^4 | \boldsymbol{x} = \boldsymbol{C}_c \tilde{\boldsymbol{x}} + \boldsymbol{d} \wedge \|\tilde{\boldsymbol{x}}\| = 1 \right\}$$

denotes the largest ellipsoid within $\mathcal{P}$. As an approximation of the size of $\mathcal{P}$, the determinant of $\boldsymbol{C}_c$, $det(\boldsymbol{C}_r)$ is used, as it is proportional to the size of the ellipsoid. An example of the space representation is illustrated in Figure 2-3 for the two dimensional case.

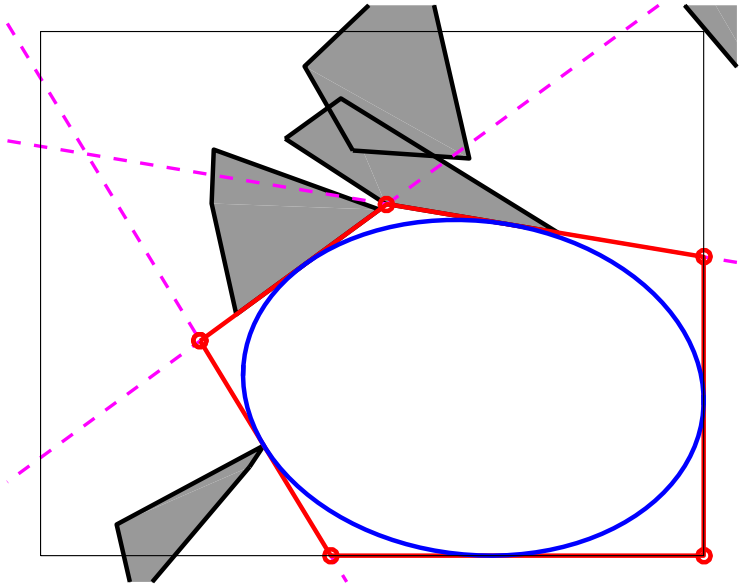


Figure 2-3: Representation of obstacle free convex space in a two dimensional environment. Obstacles (grey), workspace boundaries (black thin lines), linear constraints due to obstacles (pink dashed line), largest ellipsoid (blue), linear constraints representing $\mathcal{P}$ (red)

2-1-3 Motion Planning

The method presented in this thesis is for local navigation. It is assumed that a global reference $\boldsymbol{p}_{r,k}(t)$ for each team k is known to all agents. The global reference could be an

input given by the operator, or the output of a global planning algorithm. The distributed local planner then computes the configuration of the optimal target formation. According to Eq. (2-3), this directly translates to the references for the robots for a given time horizon $\tau > 0$. For safety reasons, the τ must be larger than the time which is needed for the drones to stop from q_{max}.

2-2 Problem Formulation

A team of n quadrotor UAVs is considered, while each robot has a limited communication range and a limited field of view. Furthermore, the quadrotors fly within a formation f which is an isomorphic transformation of one of the m template formations f_t. The transformation is fully described by the transformation state $\boldsymbol{z} = [\boldsymbol{t}, s, \boldsymbol{q}]$. The three dimensional workspace $\mathcal{W}$ contains static and dynamic obstacles, which can be non-convex. In the case of non-convexness, the convex hull of the non-convex obstacle is used.

Each of the quadrotors has knowledge of the positions $\boldsymbol{p}_{oi}$ and velocities $\boldsymbol{v}_{oi}$ of the obstacles. Its trajectories are predicted for a time horizon τ.

Based on the motivation in Section 1-2, the following research question is considered:

How to incorporate efficient splitting and merging behavior to quadrotor formation control?

This leads to the strategy which is followed to investigate the research question. It is believed that a thorough answer of the research question above will be achieved by taking on the three tasks below:

- Conditions under which splitting/merging is beneficial for quadrotor formations will be determined.
- Splitting/Merging behaviour in combination with an existing formation control approach will be implemented. The approach shall account for real-time capabilities, limited communication range and limited field of view.
- The implemented approach will be validated through simulation and experiments on a lab setup.

2-3 Method Overview

The overview of the implementation of the splitting and merging behaviour in combination with formation control, motion planning and trajectory tracking is illustrated in Figure 2-4. It represents a leader-less approach for a team of robots in a three dimensional scenario with limited field of view and communication range. It assumes that the global reference $\boldsymbol{p}_r(t)$ is known to every robot of the the corresponding formation.

The method allows for a distributed computation of splitting and merging of teams of robots, based the comparison of Goal-Directed Obstacle-Free Convex Regions (GDOFCR) which implicitly include the obstacle avoidance actions of the agents and therefore serve as an

approximation of the desired paths of each agent. This is done, as these convex constraints can be re-used for the efficient computation of the optimal formation state inside the formation control algorithm.

Each robot team is fully autonomous in the sense that there is no need for a permanent communication in between the robot teams. After splitting and merging, each robot team follows the approach of [2] to compute a target formation out of several template formations and assigns each robot i of team k a reference position r^k_{ri}, which is reachable on a direct path without obstacle collision in $[0, \tau]$. Afterwards, each robot uses a local motion planner accounting for inter-agent collision avoidance and computing a feasible trajectory. Finally, a trajectory tracking controller computes the inputs for each robot.

1. The robots perform distributed consensus to agree on a set of static and dynamic obstacles $\mathcal{F}(t_0)$.
2. Each robot computes an intermediate goal. While taking into account static and dynamic obstacles, the intermediate goal minimizes the heading deviation from the reference $\boldsymbol{p}_r(t)$ for static obstacles and the deviation from $\boldsymbol{q}_{i,pref}(t)$ for dynamic obstacles.
3. Each robot computes a goal-directed obstacle-free convex region initialized at the robots position and containing an intermediate goal.
4. The robots perform distributed consensus to agree on splitting or merging with an other formation, based on the intersections of the convex regions.
5. Each robot computes the optimal target formation within the intersection of all convex regions of the team.
6. The robots perform distributed consensus to assign each robot to a position within the target formation while minimizing the square of the total distance travelled.

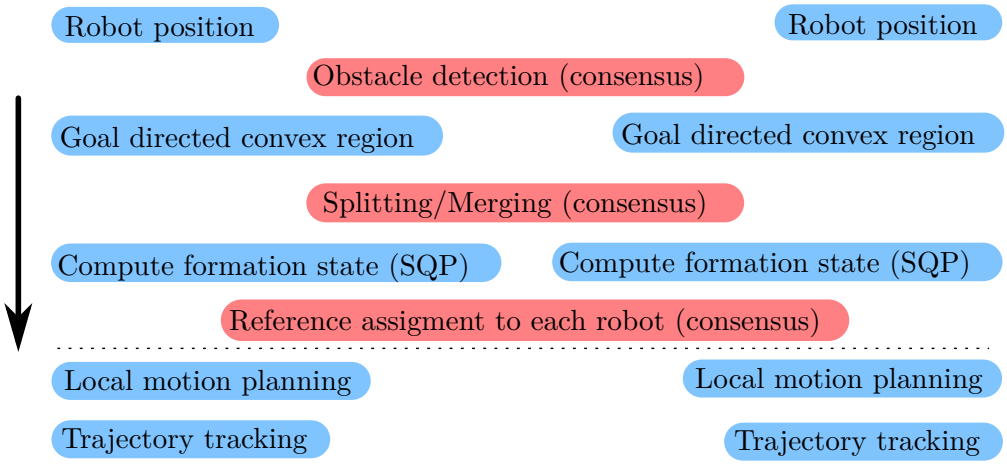


Figure 2-4: Overview of the layered approach presented.

Chapter 3

Conditions for Splitting and Merging

In order to implement splitting and merging behaviour for robot teams with an existing formation control approach, conditions under which this behaviour is desired need to be determined first.

3-1 Splitting

As explained in Section 1-2, the focus of this thesis lies in the splitting and merging of robot teams due to obstacle avoidance. As a consequence, the team splitting is supposed to be triggered if certain conditions are satisfied when an obstacle is encountered.

Firstly, it is assumed that robots avoid obstacles using one of the following avoidance actions as illustrated in Figure 3-1

- Obstacle avoidance to the right of the obstacle.
- Obstacle avoidance to the left of the obstacle.
- Obstacle avoidance above the obstacle.
- Obstacle avoidance below the obstacle.

This assumption is done as it reduces the computations later on.

Furthermore it is assumed an intended path of a robot exists. Such path does not take into account the other robots of the team and could be calculated using [29], [33] among others.

In a two dimensional environment, splitting of a robot team could be initialized by using the concept of path homotopy classes in the context of path planning. In [48], two trajectories or paths are defined to be in the same Homotopy Class if one can be smoothly deformed into the other without intersecting obstacles. Otherwise they belong to different homotopy classes.

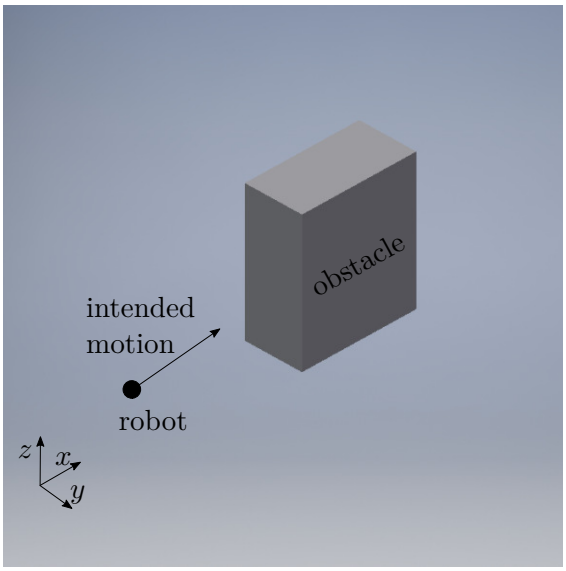
Using the term of an intended path combined with path homotopy classes, a team of robots is supposed to split, when the intended paths of robots correspond do different homotopy classes, as illustrated in Figure 3-2a.

In the three dimensional case however, two paths can still belong to the same homotopy class even though one path corresponds to obstacle avoidance on the left and the other one on the right. One solution could be to project the intended path onto the above defined avoidance planes. Using these planes, the concept of path homotopy could still be applied.

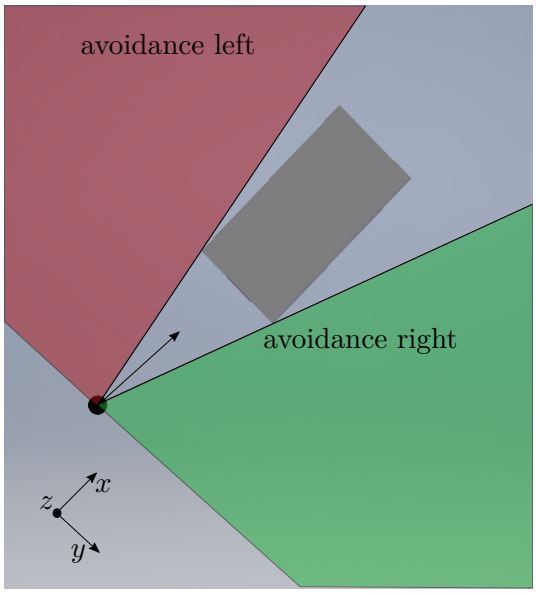
Nevertheless, using the concept of path homotopy requires the computation of intended paths for each robot. For solely determining the splitting of the team of robots, this seems computationally too expensive.

Therefore it is assumed that the intended paths can be sufficiently approximated by convex regions $\mathcal{P}$, which implicitly contain the obstacle avoidance through a heuristically computed intermediate goal and are reused for the computation of the optimal formation state. The volume of the intersection of two regions is used as a similarity measure and replaces the concept of path homotopy.

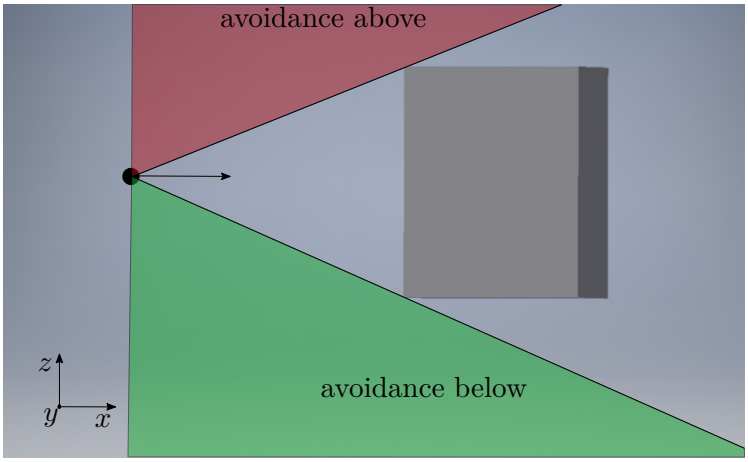
Splitting of a team of robots shall be observed if the size of one region intersection is zero. An example of the proposed concept is illustrated in Figure 3-3.



(a) Isometric view.

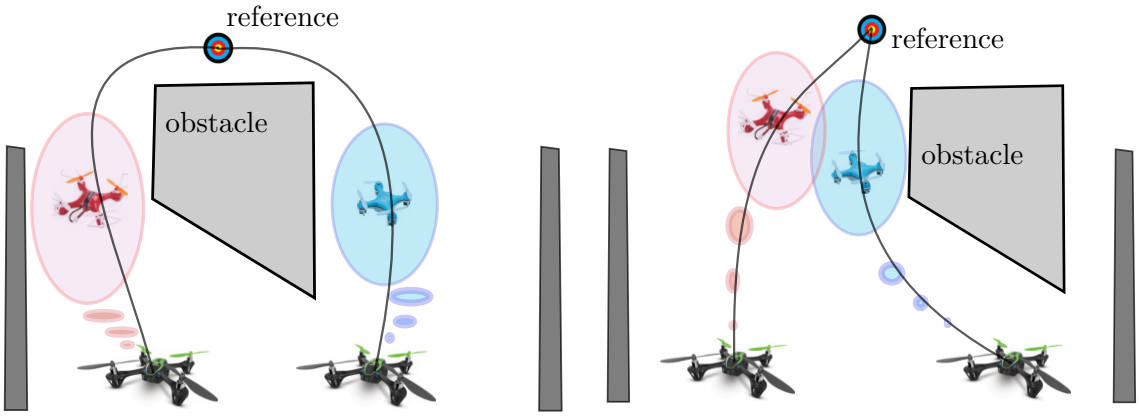


(b) Top view (horizontal avoidance plane).



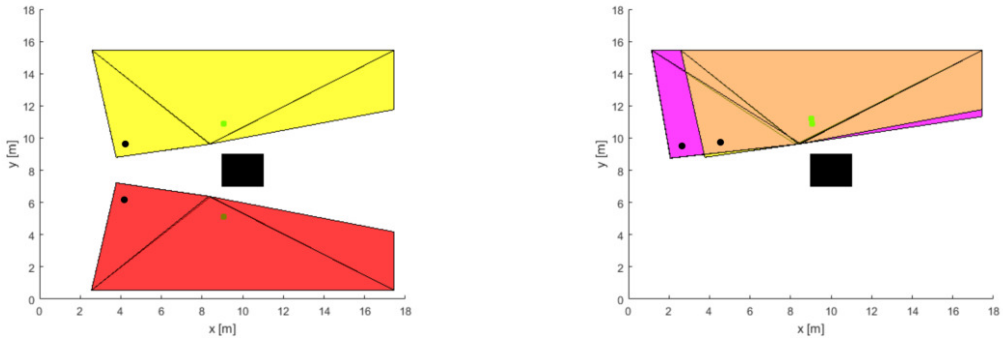
(c) Side view (vertical avoidance plane).

Figure 3-1: Classification of the free space according to obstacle avoidance actions.



(a) Splitting is desired. **(b)** Splitting is not desired.

Figure 3-2: Scheme of splitting due to different path homotopy classes. The intended path of each robot is represented by a grey line. **(a)** The desired paths of the quadrotors belong to different homotopy classes. **(b)** The desired paths of the quadrotors belong to the same homotopy class.



(a) Splitting is desired. **(b)** No splitting is desired.

Figure 3-3: Scheme of splitting due to the comparison of GDOFCR. Top view of the robots (black dots), intermediate goals (green dots), their corresponding GDOFCR (coloured polygons) and the obstacle (black). **(a)** The intersection of the two regions $\mathcal{P}_i$ is empty. **(b)** A non-empty intersection of the two regions $\mathcal{P}_i$ exists.

3-2 Merging

Generally speaking, the following merging behaviour is desired. The defined conditions for merging of robot teams are supposed to promote navigation in larger teams rather than scattered smaller teams, while ensuring safe navigation. Furthermore, they are supposed to lead to quick merging behaviour for multiple teams of robots after having split into multiple teams for obstacle avoidance.

While the above stated requirements give an idea of when merging behaviour is desired, the conditions need to be specified to allow for the implementation.

First, the limited sensing capabilities of the robots need to be taken into consideration. For teams k and l to merge, at least one pair of robots ($i \in \mathcal{I}_k$, $j \in \mathcal{I}_l$) have to be able to communicate with each other. This is not trivial, as limited communication range of the agents is assumed. Using the definition of inter-agent communication defined in Subsection 2-1-1, this translates to the necessary condition

$$\exists(i,j), \quad d_{i,j} = \|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2 \leq r_{\mathcal{C}}, \quad i \in \mathcal{I}_k,\, j \in \mathcal{I}_l. \tag{3-1}$$

Here, $d_{i,j}$ denotes the norm of the relative distance in between the pair of robots (i,j).

In case the merging is supposed to be dependent on the distance in between the centroids $\boldsymbol{x}_{k,c}$, $\boldsymbol{x}_{l,c}$ of the teams, an additional maximum inter-team distance $d_{c,max}$ can be set to prevent teams from merging which are already in communication range, but still too far from each other according to the operator. This leads to an additional necessary condition

$$d_{c,kl} = \|\boldsymbol{x}_{k,c} - \boldsymbol{x}_{l,c}\|_2 \leq d_{c,max}. \tag{3-2}$$

To prevent teams to merge, whose paths just intersect and quickly diverge afterwards, identical references are requested for now. Such a case is illustrated in Figure

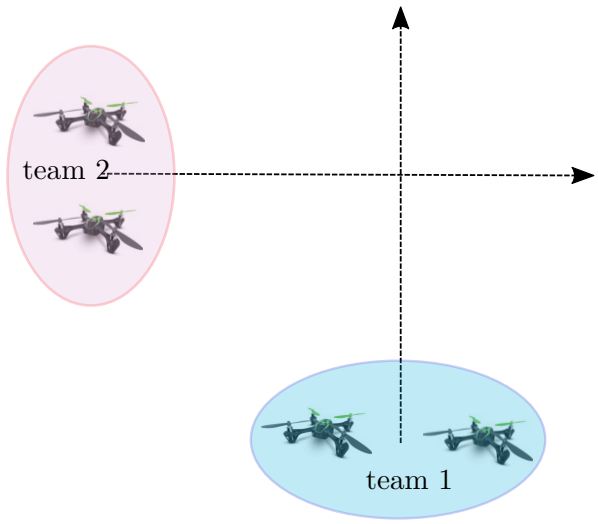


Figure 3-4: Example of intersecting paths (black dashed arrows), where no team merging is desired.

Furthermore it is necessary that the obstacle-free workspace $\mathcal{F}(t_0)$ is large enough to house the unity of both teams.

The usage of the convex obstacle free regions simplifies the mathematical definition of the above formulated necessary condition, which needs to be satisfied to allow for merging of two teams k and l.

The region $\mathcal{P}^k = \mathcal{P}_1 \cap \mathcal{P}_2 \cap \ldots \cap \mathcal{P}_{n_k}$ is used to represent the common obstacle free position-time space of team $k \in \mathcal{K}$. A region intersection $\mathcal{P}^{k,l} = \mathcal{P}^k \cap \mathcal{P}^l$ then represents the goal directed position-time space, which is reachable without collision for all agents within both teams.

In order for both teams to fit within the obstacle free space, the size $V(\cdot)$ of the region intersection $\mathcal{P}^{k,l}$ has to satisfy

$$V\left(\mathcal{P}^{k,l}\right) \geq V_{kl,min}, \tag{3-3}$$

where $V\left(\mathcal{P}^{k,l}\right)$ represents the size of $\mathcal{P}^{k,l}$ and $V_{kl,min}$ denotes the minimum required size for teams k and l to merge.

Chapter 4

Method

Following the outline of the approach in Section 2-3, this Chapter is dedicated to explaining all components of the method in depth.

4-1 Obstacle Consensus

As the splitting and merging is dependent on the comparison of goal-directed obstacle-free regions, it is important that these regions are computed with respect to the same set of obstacle free space for all agent within a team l of robots. As a consequence, all agents within a team must agree on a common set $\mathcal{F}(t_0) = \mathcal{F}_i(t_0) \quad , \forall i \in \mathcal{I}_l$.

This is done by performing a consensus step, which is synthesized in Algorithm 1. The robots do this in an iterative fashion.

Each robot i sends the difference set of obstacles $\Delta\hat{\mathcal{O}}_i(k) = \hat{\mathcal{O}}_i(k) - \hat{\mathcal{O}}_i(k-1)$ to all its neighbours, where k denotes the iteration step. In return, it receives the difference sets $\Delta\hat{\mathcal{O}}_j(k)$ of obstacles from all its neighbours. The difference sets are sent instead of the complete obstacle sets, in order to reduce the size of the data which is sent.

The new set of obstacles is then the junction of all received difference sets $\Delta\hat{\mathcal{O}}_j(k)$ with $\hat{\mathcal{O}}_i(t_0)$. This process is repeated for a number of communication rounds $d_{\mathcal{G}}$ equal to the diameter of the formation graph. Due to the assumption of the communication graph $\mathcal{G}_k$ being connected, the algorithm can be proven to converge in at most $d_{\mathcal{G}}$ communication rounds [2].

This results into consensus on the states of the static and dynamic obstacles visible to at least one member of the team.

Algorithm 1 Consensus on $\mathcal{F}(t_0)$

$\hat{\mathcal{O}}_i(1) = \hat{\mathcal{O}}_i(t_0)$, $\hat{\mathcal{O}}_i(0) = \emptyset$
for $k = 1, \ldots, d_{\mathcal{G}}$ **do**
    $\Delta\hat{\mathcal{O}}_i(k) = \hat{\mathcal{O}}_i(k) - \hat{\mathcal{O}}_i(k-1)$
    Send $\Delta\hat{\mathcal{O}}_i(k)$ to all $j \in \mathcal{N}_i$
    Receive $\Delta\hat{\mathcal{O}}_j(k)$ from all $j \in \mathcal{N}_i$
    $\hat{\mathcal{O}}_i(k+1) = \hat{\mathcal{O}}_i(k) \cup \Delta\hat{\mathcal{O}}_j(k) \quad \forall j \in \mathcal{N}_i$
end for
$\mathcal{F}_i(t_0) := \mathcal{W}(t_0) \setminus \hat{\mathcal{O}}_i(d-1)$

4-2 Obstacle-Free Convex Regions

The goal-directed obstacle-free convex regions (GDOFCR) $\mathcal{P}_i$ are computed similarly to [2], which is based on the iterative approach, named IRIS by [25]. The underlying algorithm alternates between a Quadratic Program (QP) and a Semidefinite Program (SDP).

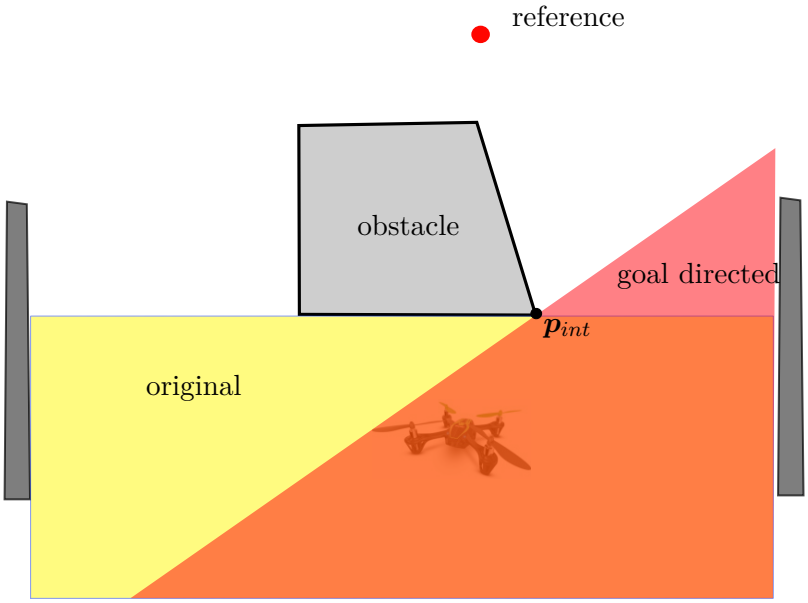


Figure 4-1: Projected regions computed by IRIS and goal directed computation.

The original method purely maximizes the size of the convex regions. In contrast to it, the GDOFCR include an intermediate goal, determined through a heuristic. This implicitly incorporates the obstacle avoidance action into the regions and leads to the regions making more goal progress. Consequently, it allows for the usage of the regions as a substitute of the desired path of each robot. The affect of the difference in the computation of the regions is illustrated in Figure 4-1.

The algorithm used to compute the GDOFCR is initialized with an ellipsoid $\mathcal{E}$, for which must hold $\mathcal{E} \subset \mathcal{P}_i$ for all iterations to satisfy the goal directedness of the regions. In order to define the initial ellipsoid $\mathcal{E}$, an intermediate goal $\boldsymbol{x}_{int} \in \mathbb{R}^4$ is computed beforehand, which contains the obstacle avoidance action.

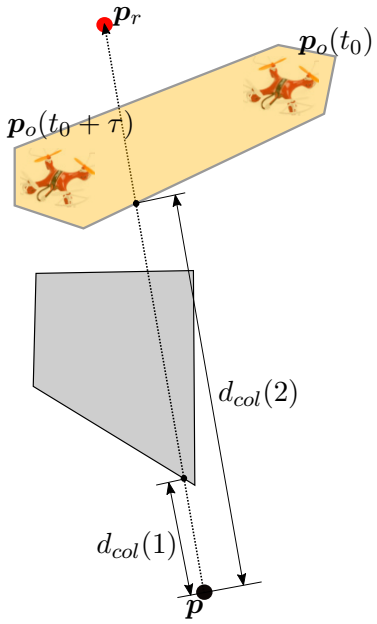


Figure 4-2: Top view. Storage of the intersection distances $d_{col}(i)$ for a static obstacle ($i = 1$) and one dynamic obstacle ($i = 2$).

Throughout this Section, the method of computing the convex regions is explained for a single agent. Analogously, the regions are computed for each agent i of every formation k. This allows to drop the indices referring to a specific agent from a specific formation. Furthermore, all computations are done at the current time denoted by t_0, allowing to drop the time index if not specified.

4-2-1 Collision Check

As explained above, an ellipsoid $\mathcal{E}_r$ serves as an initializer for the computation of the GDOFCR, as well as a stopping criterion. The intermediate goal shall only deviate from the reference $\boldsymbol{p}_r$, if the reference is not reachable without collision following a straight line. This condition helps to achieve the desired goal progress of the regions.

A collision check is therefore performed first to determine if the direct path towards the reference $\boldsymbol{p}_r$ intersects with any of the visible obstacles $\hat{\mathcal{O}}$. Due to the obstacle consensus step $\hat{\mathcal{O}}$ accounts for obstacles, which are located within the combined visibility volume at the current time. To add conservativeness when encountering dynamic obstacles, the convex hull of their occupied volume within $[0, \tau]$ is used.

The collision check is based on the analytic line clipping algorithm proposed by [49], for which the obstacles are represented by a set of linear equalities for each obstacle i in $\hat{\mathcal{O}}$.

If an intersection exists for an obstacle i and therefore a risk of collision is present, the euclidean distance in between $\boldsymbol{p}$ and the point of collision with obstacle i is stored in $d_{col}(i)$ and the check is repeated for every visible obstacle, as illustrated in Figure 4-2.

4-2-2 Intermediate Goal

After having performed the collision checks, the intermediate goal $\boldsymbol{p}_{int}$ can be computed. The reference $\boldsymbol{p}_r$ is used as the intermediate goal $\boldsymbol{p}_{int} = \boldsymbol{p}_r$, unless the collision check from Subsection 4-2-1 determines intersections with obstacles.

In case of an intersection with one or multiple obstacles, the obstacle with the smallest distance $d_{col}(i), \quad i = \arg\min_{i \in \mathcal{O}_{s,d}} d_{col}(i)$ is taken into account for the calculation of the intermediate goal.

The choice of only accounting for one obstacle in the computation of the intermediate goal is controversial. On the one hand, it allows for a fast computation and the GDOFCR takes into account all visible obstacles. On the other hand, this approach only allows for local obstacle avoidance and deadlocks can still occur. As the aim of this thesis includes the real-time performance of the algorithm, the computationally light method is pursued and methods which possibly allow for the formulation of optimality guarantees or, mathematical proofs for avoiding deadlocks are left for future work.

The intermediate goal $\boldsymbol{p}_{int}$ minimizes the heading deviation in between the direct path towards the reference $\Delta\boldsymbol{p}_r = \boldsymbol{p}_r - \boldsymbol{p}$ and the path towards the intermediate goal $\Delta\boldsymbol{p}_{int} = \boldsymbol{p}_{int} - \boldsymbol{p}$ for an intersection static obstacles.

For dynamic obstacles, the avoidance action is chosen based on minimizing the deviation from the preferred velocity $\boldsymbol{q}_{pref}$.

Intersection Static Obstacle

In case the considered obstacle for the computation of the intermediate goal is a static obstacle, a projected intermediate goal V_*^{pj} is computed for each avoidance plane $j = 1, 2$, where each projected intermediate goal has a cost $abs(\alpha^{pj})$ with which it is associated. The candidate with the lower associated cost is used to further determine the intermediate goal.

First, the obstacle in question is transformed into a right handed body coordinate system $\mathcal{S}_b$ with origin at $\boldsymbol{p}$, the local x-axis pointing into the direction of the reference . Afterwards, the obstacle is projected onto the local $x - y$ $(j = 1)$, respectively local $y - z$ $(j = 2)$ plane, as illustrated in Figure 4-3. This is done, as it simplifies the computations.

In the following, the calculations are outlined for the computation of V_*^{p1} in the $x-y$ plane and are done analogously for the other avoidance plane. A supporting scheme for the computation is illustrated in Figure 4-4.

The vertices of the obstacle projected onto the projection plane 1 are denoted by V_k^{p1} and the number of projected vertices is denoted by n_{v1}. The angle deviation for vertex k from the direct path towards $\boldsymbol{p}_r$ is used as the associated cost $\Delta\alpha^{p1}(k)$ and is computed as

$$\Delta\alpha^{p1}(k) = \frac{v_{ky}^{p1}}{v_{kx}^{p1}}. \tag{4-1}$$

Here, v_{kx}^{p1} and v_{ky}^{p1} denote the x and y value of the vertex V_k^{p1}. The vertices which can be used for the avoidance of the considered obstacle are the ones corresponding to the extrema

$$\begin{aligned} \Delta^{p1}_{max} &= max(\Delta\alpha^{p1}(k)), \\ \Delta^{p1}_{min} &= min(\Delta\alpha^{p1}(k)). \end{aligned} \tag{4-2}$$

The vertices define a non-convex collision avoidance constraint illustrated in Figure 4-5, where all projected vertices are within the constraint. All points within this plane satisfying the collision avoidance constraint can be reached approaching on a direct path, without colliding with the considered obstacle.

The projected vertex with the smallest cost

$$k^{p1}_* = \arg\min_k abs(\Delta^{p1}_{max}, \Delta^{p1}_{min}) \tag{4-3}$$

is used as the projected intermediate goal $V^{p1}_* = V^{p1}_{k^{p1}_*}$ of the projection plane $p1$.

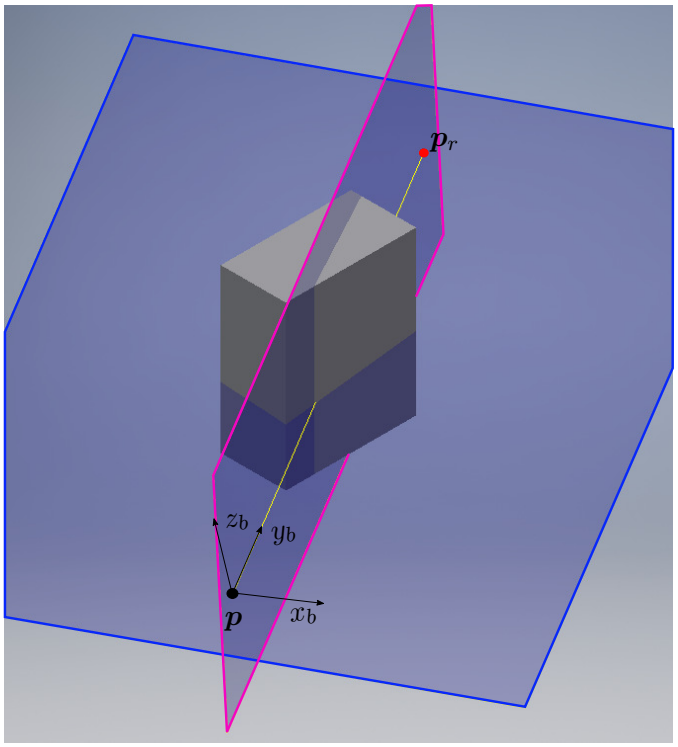
In case two vertices have the same minimal value $\Delta\alpha^{p1}(k^{p1}_*)$, the vertex with the larger value v^{p1}_{kx} is chosen as V^{p1}_*. This is done to increase the goal progress included in $\mathcal{E}_r$.

After calculating V^{pj}_* for the two projection planes, the projected intermediate goal vertex which satisfies

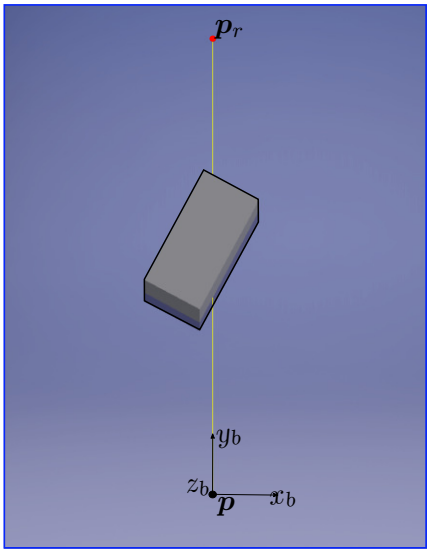
$$min\left(\Delta\alpha^{p1}(k^{p1}_*), \Delta\alpha^{p2}(k^{p2}_*)\right) \tag{4-4}$$

determines the optimal projected intermediate goal V^p_*.

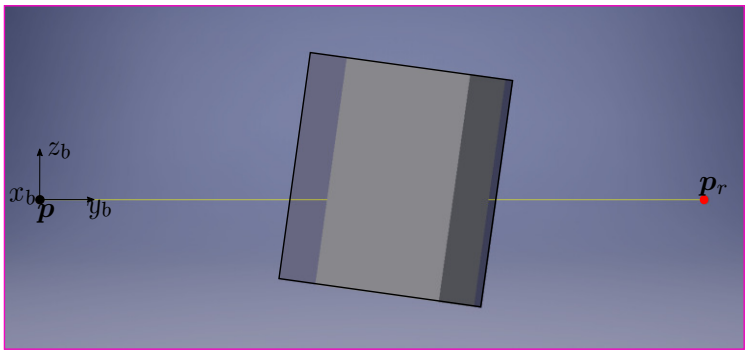
Due to the projection of the obstacle, also the optimal projected intermediate goal V^p_* corresponds to a point on a plane in the body coordinate system. Hence, it is transformed back into the world coordinates.



(a) Isometric view.



(b) Projected obstacle onto the $x_b - y_b$ plane ($j = 1$).



(c) Projected obstacle onto the $z_b - y_b$ plane ($j = 2$).

Figure 4-3: Projection of the obstacle onto the avoidance planes of the body coordinate system. $x_b - y_b$ plane (blue edging), $z_b - y_b$ plane (pink edging) and projected obstacle (black edging).

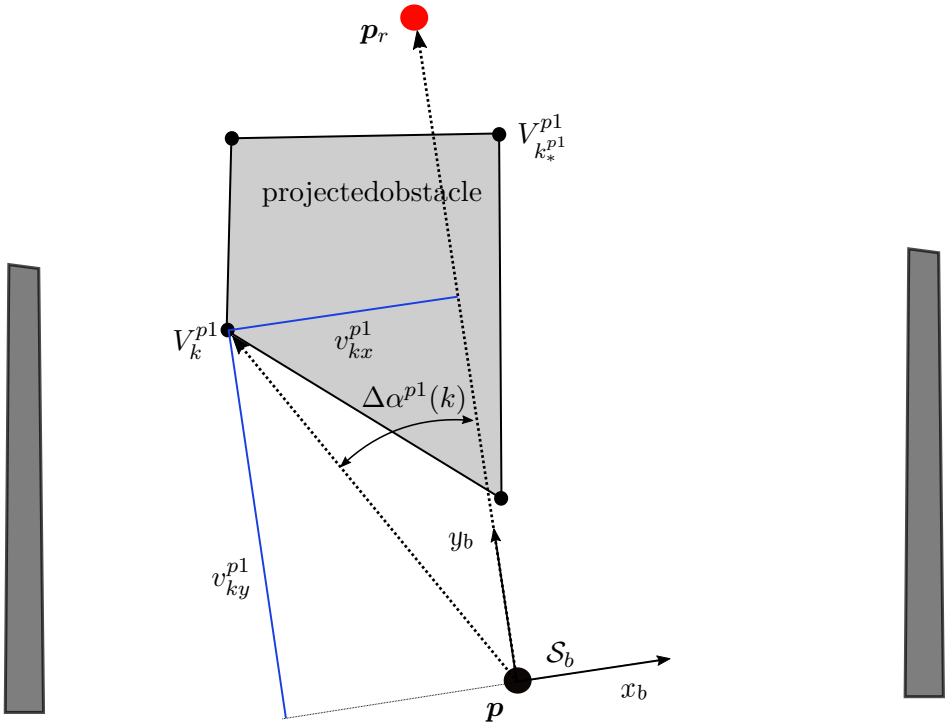


Figure 4-4: Scheme of the computation of the projected intermediate goal in projection plane 1.

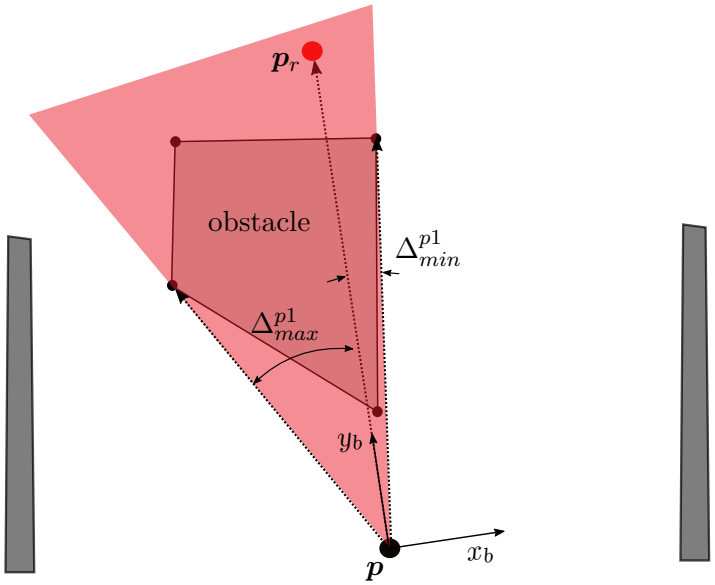


Figure 4-5: Usage of the vertices for obstacle avoidance. The shaded region defines the non-convex obstacle avoidance constraint.

Intersection Dynamic Obstacle

In case the considered obstacle for the computation of the intermediate goal is a dynamic obstacle, V_*^p cannot be determined as in the case for static obstacles. If the dynamic obstacle is considered to be static for each discrete time step, the intermediate goal and the avoidance action might change depending on the time instance chosen. This is due to the relative movement of the obstacle with respect to the agent.

As the avoidance action is fundamental for the obstacle avoidance, it is determined first and the intermediate goal $\boldsymbol{p}_{int}$ is calculated according to it.

The avoidance action is calculated using the approach presented by [50], which is based on the Velocity Obstacle (VO) principle by [29]. The method is formulated entirely in the velocity space and it computes one non-convex collision avoidance constraint for each obstacle to ensure collision avoidance during $[0, \tau]$.

Applying the above introduced approach to the dynamic obstacle considered, it results into a single non-convex collision avoidance constraint formulated in the velocity space. In order to determine the collision avoidance action, the non-convex constraint is approximated by four $i \in [1, 4]$ linear constraints of the form

$$\boldsymbol{n}_{oi}(\boldsymbol{q} - \boldsymbol{q}_o) \leq b_{oi}, \tag{4-5}$$

where $n_{oi} \in \mathbf{R}^3$, $b_{oi} \in \mathbf{R}$ and ensuring that once $(\boldsymbol{q} - \boldsymbol{q}_o)$ satisfies one of the linear constraints, it also complies to the original non-convex constraint. This leads to four half-spaces, each corresponding to one of the collision avoidance actions.

Afterwards, the approximated collision avoidance constraint is linearized by choosing one of the four linear constraints. The criterion to choosing the linear constraint is to maximize the constraint satisfaction for the preferred velocity $\boldsymbol{q}_{pref}$. This leads to solving

$$\arg\min_i \left(\boldsymbol{n}_{oi} \cdot (\boldsymbol{q}_{pref} - \boldsymbol{q}_o) - b_{oi}\right). \tag{4-6}$$

One could argue that a different preferred velocity for each agent should be computed, for example to the last computed reference position $\boldsymbol{r}_i$. When the formation cannot move into the direction of the goal however, it would also mean that the preferred velocity of each agent is not goal directed, which in preliminary simulations shows not to lead to satisfying results.

After having determined the collision avoidance action, $\boldsymbol{q}_{int}$ is calculated. The dynamic obstacle is considered at its position, which is closest to the robot $\boldsymbol{p}_{o,cl}$ within $[0, \tau]$. This position is calculated based on the assumption of constant velocity for both the robot $\boldsymbol{q}_{pref}$ and dynamic obstacle $\boldsymbol{q}_o$.

The time t_{min}, which determines $\boldsymbol{p}_{o,cl}$ corresponds to solving

$$\arg\min_t d_{ro}(t) = \arg\min_t \|\boldsymbol{p}(t) - \boldsymbol{p}_o(t)\|_2. \tag{4-7}$$

Eq. (4-7) can be solved analytically and leads to

$$t_{min} = -\frac{\boldsymbol{p}_{rel}^T \boldsymbol{q}_{rel}}{\boldsymbol{q}_{rel}^T \boldsymbol{q}_{rel}}, \tag{4-8}$$

where $\boldsymbol{p}_{rel} = \boldsymbol{p}_o - \boldsymbol{p}$, $\boldsymbol{q}_{rel} = \boldsymbol{q}_o - \boldsymbol{q}_{pref}$. Furthermore, the time is clipped by $t_{min} \in [0, \tau]$ to prevent time instances in the past and outside the prediction horizon.

Consequently the position of the dynamic obstacle is given by

$$\boldsymbol{p}_{o,min} = \boldsymbol{p}_o(t_{min}) = \boldsymbol{p}_o + t_{min}\boldsymbol{q}_o, \tag{4-9}$$

which gives all the necessary information to compute $\boldsymbol{p}_{int}$ similarly to the case for a static obstacle.

It is different however that the computations only need to be done for one projection plane until Eq. (4-2), as the avoidance action is already known. From here, the intermediate goal $\boldsymbol{p}_{int}$ follows straightforward with the precomputed avoidance action.

Even though the collision with an obstacle might only occur after the prediction horizon τ, it is assumed to lead to smoother trajectories if it is taken into account for the computation of the GDOFCR, as soon as the obstacles become visible to the formation. For obstacles with quickly changing velocities however, this could potentially lead to quickly changing GDOFCR and needs to be evaluated carefully.

4-2-3 Regions Computation

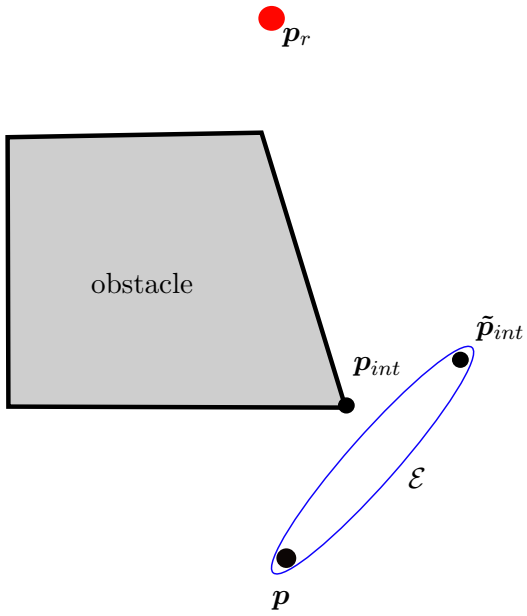


Figure 4-6: Translation of the intermediate goal and initial ellipsoid $\mathcal{E}_r$, shown in 2D scheme.

After having computed the intermediate goal $\boldsymbol{p}_{int}$ each robot computes the GDOFCR $\mathcal{P}_i$. As mentioned above, the computation of these regions builds on [2], which is based on the iterative method IRIS by [25]. The optimization steps of the original algorithm are as follows.

The first optimization algorithm generates a set of hyperplanes $Ax \leq b$ defining an obstacle free convex region $\mathcal{P}$. This is done by computing separating hyperplanes in between an ellipsoid $\mathcal{E}_r(k)$ and the common set of dilated obstacles $\hat{\mathcal{O}}$ via quadratic optimization,

where k corresponds to the iteration step. The ellipsoid is represented as a transformation of the unitcircle as $\mathcal{E}_r(\boldsymbol{C}_r, \boldsymbol{d}) = \{\boldsymbol{x} \in \mathbb{R}^4 | \boldsymbol{x} = \boldsymbol{C}_r\tilde{\boldsymbol{x}} + \boldsymbol{d} \wedge \|\tilde{\boldsymbol{x}}\| = 1\}$. The computed constraints are perpendicular to the surface of $\mathcal{E}_r(k)$ and furthermore intersect with the surface of the obstacles.

The second step computes the largest ellipsoid $\mathcal{E}_r(k+1)$ within $\mathcal{P}$, used for computation of the hyperplanes in the next iteration step. This corresponds to maximizing the determinant of $\boldsymbol{C}_r$ and is done by sequential quadratic programming. By alternating both optimization algorithms, the space grows monotonically until convergence.

The original method has a drawback however. While maximizing the including volume of the obstacle free region, the region itself may not lead to progress towards the goal.

In [2] the proposed solution is to stop the algorithm once the set of linear constraints defining $\mathcal{P}$ does not contain the points defining the initial ellipsoid $\mathcal{E}_r(1)$. In that case, the set of linear constraints from the previous iteration $k-1$ are used to define $\mathcal{P}$. The initial ellipsoid $\mathcal{E}_r(1)$ is determined by computing the smallest volume ellipsoid, which still contains the defining points $\boldsymbol{p}_{\mathcal{E},i_\mathcal{E}} \in \mathcal{E}_r(1) \forall i_\mathcal{E}$.

Here the method from [2] is refined using the robot position $\boldsymbol{p}$ and the intermediate goal computed in Subsection 4-2-2, to define the initial ellipsoid $\mathcal{E}_r(1)$ and therefore implicitly including an obstacle avoidance action in combination with more goal progress into the computed regions $\mathcal{P}_i$.

However, the point $\boldsymbol{p}_{int}$ lies on the surface of the obstacle. As it would lead to $\mathcal{E}_r(1) \not\subset \mathcal{F}$, $\boldsymbol{p}_{int}$ cannot be used directly. Hence, a rotation and translation is applied to the intermediate goal, as illustrated in Figure 4-6, according to

$$\begin{aligned} \boldsymbol{p}_{dir} &= \boldsymbol{R}_{int}(\boldsymbol{p}_{int} - \boldsymbol{p}), \\ \tilde{\boldsymbol{p}}_{int} &= \boldsymbol{p} + \boldsymbol{p}_{dir} + f_{int}\frac{\boldsymbol{p}_{dir}}{\|\boldsymbol{p}_{dir}\|_2}, \end{aligned} \tag{4-10}$$

where the translated intermediate goal is denoted by $\tilde{\boldsymbol{p}}_{int}$, the translation factor is represented by f_{int} and $\boldsymbol{R}_{int}$ is a rotation matrix. The rotation matrix is dependent on the avoidance action and further details can be found in Appendix A. Eq. (4-10) moves $\boldsymbol{p}_{int}$ away from the surface of the obstacle and translates it into the direction of $\boldsymbol{p}_{dir}$, as illustrated in Figure 4-6 in the 2D case. This helps the initial ellipsoid to satisfy $\mathcal{E}_r(1) \subset \mathcal{F}$ and furthermore preserves the goal progress.

As the regions computed are in position-time space, the time dimension has to be included into the definition of $\mathcal{E}_r(1)$. The robot position $\boldsymbol{p}$ is extended according to

$$\boldsymbol{p}_{\mathcal{E},1} = \begin{bmatrix} \boldsymbol{p} & 0 \end{bmatrix}^T. \tag{4-11}$$

In case of the intermediate goal, there exist two cases. In case of the of no obstacle intersection, or accounting for a static obstacle the prediction horizon τ is used. In case of accounting for a dynamic obstacle, the computed time $t_{min} \in [0, \tau]$ is taken. This is the case, as it represents the time instance which is used to compute $\boldsymbol{p}_{int}$. This leads to

$$\boldsymbol{p}_{\mathcal{E},2} = \begin{cases} \begin{bmatrix} \tilde{\boldsymbol{p}}_{int} & \tau \end{bmatrix}^T, \text{ if accounted for no/ stat. obstacle,} \\ \begin{bmatrix} \tilde{\boldsymbol{p}}_{int} & t_{min} \end{bmatrix}^T, \text{ if accounted for dyn. obstacle.} \end{cases} \tag{4-12}$$

The points $\boldsymbol{p}_{\mathcal{E},1}$ and $\boldsymbol{p}_{\mathcal{E},2}$ are the most important points defining the initial ellipsoid $\mathcal{E}_r(1)$, as they are responsible for the orientation of the ellipsoid. The remaining points define the width of the ellipsoid and are calculated from the center of the ellipsoid.

This leads to regions earlier illustrated in Figure 4-1. As can be seen, the GDOFCR leads to more goal progress, while implicitly including the avoidance action corresponding to $\boldsymbol{p}_{int}$. As in dense scenarios, it might happen that already the first iteration to compute $\mathcal{P}$ fails to contain $\boldsymbol{p}_{int}$, as $\mathcal{E} \subset \mathcal{F}$ is not mathematically ensured. This is due to the fact that only one obstacle is considered for the computation of $\boldsymbol{p}_{int}$. In that case, the conservativeness of the regions is decreased by reducing the obstacle dilation factor α_d first. If still no valid region can be obtained, $\boldsymbol{p}_{\mathcal{E},2}$ is moved closer to $\boldsymbol{p}_{\mathcal{E},1}$ according to

$$\boldsymbol{p}_{\mathcal{E},2}(k+1) = \boldsymbol{p}_{\mathcal{E},1} + (\boldsymbol{p}_{\mathcal{E},2}(0) - \boldsymbol{p}_{\mathcal{E},1}) \cdot \alpha_c,\ \alpha_c = 1 - k \cdot 0.1. \tag{4-13}$$

Here, the moving factor is represented by $\alpha_c \in (0, 1]$ and k denotes the iteration of the relaxation. After the application of each relaxation step, the computation of $\mathcal{P}_i$ is reinitialized. Note that the splitting and merging behaviour may suffer by reducing α_c. This is the case, as the computed GDOFCR might not correspond to a specific intermediate goal any longer.

4-3 Splitting and Merging of the Formations

With the GDOFCR being computed for every agent $i \in \mathcal{I}_j$ of every robot team $j \in \mathcal{J}$, all the necessary preliminaries are set for the team to check the conditions defined in Chapter 3, if to split into multiple teams or merge with another robot team.

In order to simplify the problem, the following additional specifications are set.

- Only one splitting or merging action is allowed to be performed per computation instance per robot team j.
- Splitting always leads to $n_{spl} = 2$ robot teams.
- Merging is allowed for $n_{merg} = 2$ robot teams.

While these simplifications are assumed to decrease the performance of the approach in certain situations, the algorithm can easily be extended later.

An overview of solving the problem of splitting and merging is given in Algorithm 2, where the computations are outlined for one formation. However, they are equally valid and performed by all robot teams $k \in \mathcal{K}$.

As can be seen, a robot team first checks for splitting. This is due to the fact that each robot team k needs a non-empty common region $\mathcal{P}^k = \mathcal{P}_1 \cap \mathcal{P}_2, \ldots, \cap \mathcal{P}_{n_k}$ for the computation of the optimal formation state, as will be explained in Section 4-4. A non-empty common region $\mathcal{P}^k$ can be assured by checking for splitting first, as the formation splits if that is not the case.

Algorithm 2 Checking for splitting and merging for robot team k

```
Check splitting
if Splitting desired then
    Split team
    end
else
    Check merging
    if Merging with a team l is desired then
        Request to merge with robot team l
        if Robot team requests to merge with team k then
            Merge with team k
        end if
    end if
end if
```

4-3-1 Splitting

Algorithm 3 Checking splitting of team k

```
for All robots i ∈ R_k do
    Compute additional regions A_{i,add}
end for
Consensus on Convex regions P_i, A_{i,add} ∀i ∈ R_k
Distributed computation of the intersection graph G_int
if ∃V_ij = 0|i ≠ j then
    for All robots i ∈ R_k do
        Compute graph partition
    end for
    Regroup in new teams according to the graph partition
end if
```

The splitting of one formation is outlined in Algorithm 3 and applied to all robot teams $k \in \mathcal{K}$.

First, one additional linear constraint is computed for every agent $i \in \mathcal{R}_k$. This results into sets of regions

$$\mathcal{A}_{i,add} := \left\{ \boldsymbol{x} \in \mathbb{R}^4 | \boldsymbol{a}_{i,add}^T \boldsymbol{x} \leq b_{i,add} \right\}, \tag{4-14}$$

where $\boldsymbol{x}$ denotes a point in the position-time space and $\boldsymbol{a}_{i,add}$, $b_{i,add}$ represent the additional constraint for robot i.

Afterwards, a distributed network flooding is performed in order to achieve consensus on the information of the convex regions and additional regions of all team members $i = 1, \ldots, n_k$. Each agent i then computes the intersections $\mathcal{P}_{i,j} = \tilde{\mathcal{P}}_i \cap \tilde{\mathcal{P}}_j$, $\forall j \in \mathcal{R}_k \neq i$ with the other robots, where $\tilde{\mathcal{P}}_i = \mathcal{P}_i \cap \mathcal{A}_{i,add}$. The size V_{ij} of each intersection is stored in the edges of the intersection graph $G_{int} = (V_a, \mathcal{E}_{int})$. This graph is built in a distributed fashion and defined

by its adjacency matrix A_a. If there are empty intersections, the graph is split using [51] and the formation is split into two independent robot teams.

Additional Constraints

As explained above, intersections of the GDOFCR are used as a similarity measure of the intended paths of the robots. Using those regions alone however, region intersections $\mathcal{P}_{ij}$ belonging to different avoidance actions within the same avoidance plane can still occur. This would keep robots within the same team, which are supposed to split based on the defined conditions for splitting. Such an example is illustrated in Figure 4-7a, where the region intersection is marked in red.

As a consequence, an additional constraint is computed for every agent i to overcome this problem. The constraint is formulated as

$$\begin{aligned} \boldsymbol{a}_{add} &= -\begin{bmatrix} \frac{\tilde{\boldsymbol{p}}_{i,int}-\boldsymbol{p}_i}{\|\tilde{\boldsymbol{p}}-\boldsymbol{p}_i\|_2}^T & 0 \end{bmatrix}, \\ b_{add} &= -\boldsymbol{p}_i^T \frac{\tilde{\boldsymbol{p}}-\boldsymbol{p}_i}{\|\tilde{\boldsymbol{p}}-\boldsymbol{p}_i\|_2}. \end{aligned} \tag{4-15}$$

This leads to a half space for each robot, as described in Eq. (4-14). Combining this constraint with the GDOFCR $\mathcal{P}_i$, the original regions get clipped behind the agents perpendicular to the vector $\tilde{\boldsymbol{p}}-\boldsymbol{p}_i$, leading to $\tilde{\mathcal{P}}_i = \mathcal{P}_i \cap \mathcal{A}_{i,add}$. The effect of the additional constraint is illustrated in Figure 4-7b. Here, the intersection $\mathcal{P}_{12}$ from Figure 4-7a gets clipped by the additional constraints (grey).

Note that these additional constraints are only used for the splitting of the formation and not for the computation of the optimal formation state $\boldsymbol{z}$.

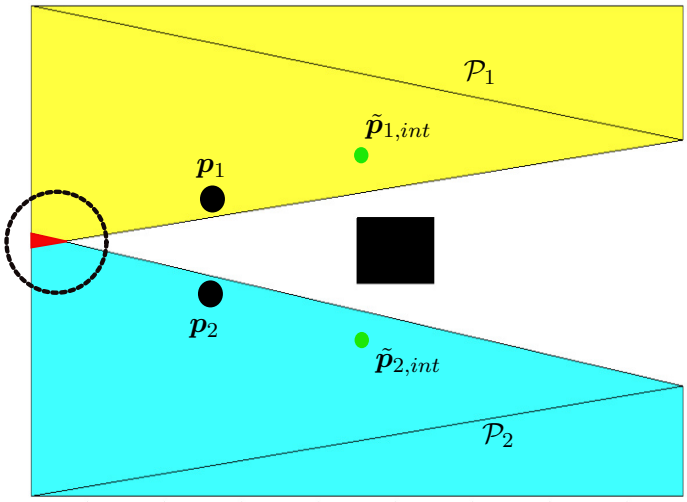
Consensus on Convex Regions

As explained above, it is necessary for each agent to have information of all regions $\mathcal{P}_i$, including the additional regions $\mathcal{A}_{i,add}$. For this purpose, a network flooding algorithm is used as outlined in Algorithm 2 to achieve consensus on the regions $\mathcal{P}_i$ and $\mathcal{A}_{i,add}$.

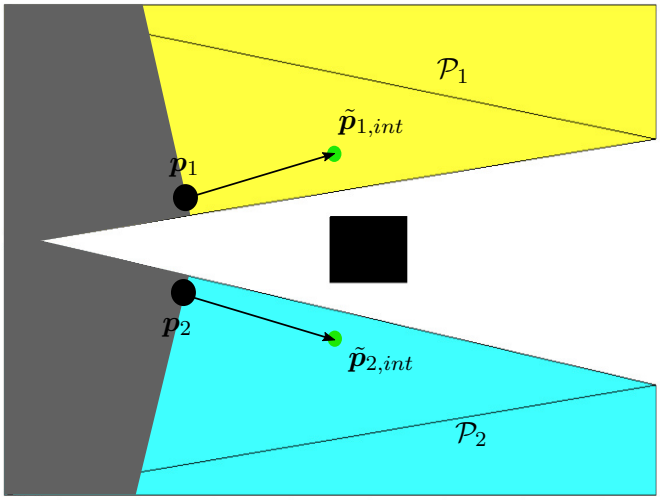
Algorithm 4 Exchange of $\mathcal{P}_i$ and $\mathcal{A}_{i,add}$

1: $\mathcal{P}_{i,t}(1) = \{\mathcal{P}_i\}$, $\mathcal{P}_{i,t}(0) = \emptyset$, $\mathcal{A}_{i,t}(1) = \{\mathcal{A}_{i,add}\}$, $\mathcal{A}_{i,t}(0) = \emptyset$
2: **for** $k = 1, \ldots, d_{\mathcal{G}}$ **do**
3: Compute $\Delta\mathcal{P}_{i,t}(k) = \mathcal{P}_{i,t}(k) \setminus \mathcal{P}_{i,t}(k-1)$, $\Delta\mathcal{A}_{i,t}(k) = \mathcal{A}_{i,t}(k) \setminus \mathcal{A}_{i,t}(k-1)$
4: Send all new sets of regions $\Delta\mathcal{P}_t(k)$, $\Delta\mathcal{A}_{i,t}(k)$ to all $j \in \mathcal{N}_i$
5: Receive $\Delta\mathcal{P}_{i,t}(k)$, $\Delta\mathcal{A}_{j,t}(k)$ from all $j \in \mathcal{N}_i$
6: Store received regions as $\mathcal{P}_{i,t}(k+1) = \mathcal{P}_{i,t}(k) \cup \Delta\mathcal{P}_{j,t}(k)$, $\mathcal{A}_{i,t}(k+1) = \mathcal{A}_{i,t}(k) \cup \Delta\mathcal{A}_{j,t}(k)$, $\forall j \in \mathcal{N}_i$
7: **end for**

In order to reduce the information sent through the network, only the sets containing new information $\Delta\mathcal{P}_t(k)$, $\Delta\mathcal{A}_{i,t}(k)$ are sent to the neighbours. Equal to the algorithm presented in Section 4-1, the convergence of Algorithm 4 can be shown.



(a) Undesired region intersection $\mathcal{P}_{12}$ (red) behind the agents.



(b) Additional constraints remove the region intersection.

Figure 4-7: Top view. GDOFCR $\mathcal{P}^1$ (avoidance action left) and $\mathcal{P}^2$ (avoidance action right).

Computation Intersection Graph

Once consensus on the convex regions $\mathcal{P}_i$, $\mathcal{A}_{i,add} \forall i \in \mathcal{R}_k$ is performed, the intersection graph $\mathcal{G}_{int}$ which is represented by its adjacency A_a can be built. In Figure 4-8, the relationship from in between the GDOFCR and the intersection graph is illustrated for a team of four quadrotor UAVs.

The intersection graph contains information on the size of the regions intersections

$$\mathcal{P}_{i,j} = \tilde{\mathcal{P}}_i \cap \tilde{\mathcal{P}}_j, \tag{4-16}$$

which is used as a similarity measure of the intended paths of the agents within a robot team k. In order to calculate the size of a region intersection $\mathcal{P}_{i,j}$, the determinant of the largest ellipsoid $\mathcal{E}_{ij,max}(\boldsymbol{C}_{ij,max}, \boldsymbol{d}_{ij,max})$ within $\mathcal{P}_{i,j}$ is used. This can be done, as the volume of an ellipsoid $\mathcal{E}(\boldsymbol{C}, \boldsymbol{d})$ is proportional to the determinant of $det(\boldsymbol{C})$ [25].

The computation of the elements within A_a are outlined in Algorithm 5.

Algorithm 5 Computation of Intersection Graph

```
1:
2: for All i ∈ R_k do
3:     split_i = 0
4:     Define nullmatrix A_{i,a} ∈ R^{n_k × n_k}
5:     for j > i do
6:         Compute P_{i,j} followed by E_{ij,max}(C_{ij,max}, d_{ij,max})
7:         Compute size V_ij = det(C_{ij,max})
8:         Save V_ij in A_{i,a}(i,j) = V_ij
9:         if V_ij = 0 then
10:            split_i = split_i + 1
11:        end if
12:    end for
13: end for
```

Due to the knowledge of the adjacency A_a being a symmetric matrix, it is sufficient that each agent i only computes the elements $A_{i,a}(i,j)$, $j > i$. Applying a similar consensus algorithm as in Algorithm 4, it allows for the distributed construction of a unique upper triangular matrix $A_{i,a}$ and consensus on $split = \begin{cases} 1, & if\ \exists i \in \mathcal{R}_k,\ split_i \neq 0 \\ 0, & else \end{cases}$. The complete adjacency matrix is finally obtained by

$$A_a = A_{i,a} + A_{i,a}^T. \tag{4-17}$$

Splitting of the team of robots in initialized, if $split \neq 0$.

Graph Partition

If there is an empty edge in $\boldsymbol{A}_a$, it results into $split = 1$ as explained above. As a consequence the team needs to split into two separate teams. The objective when splitting the team, is

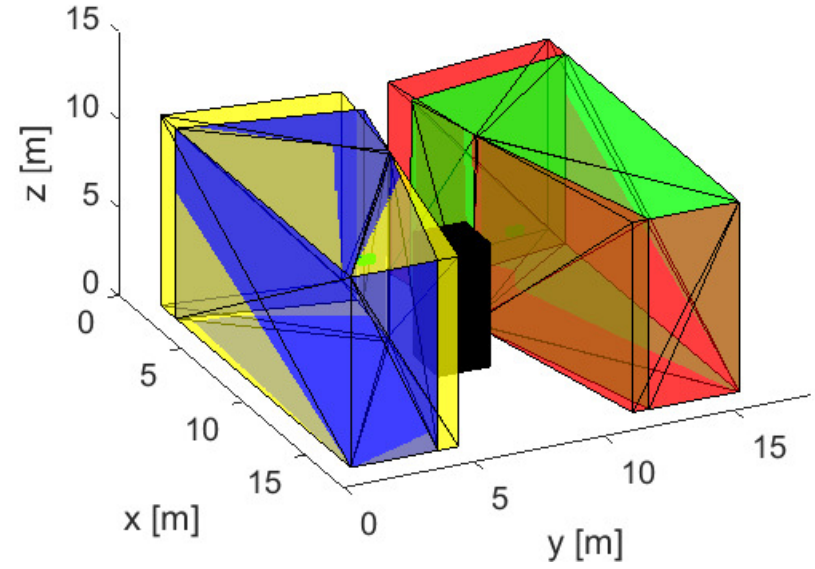
to minimize the size of the region intersections $\mathcal{P}_{i,j}$ belonging to robot pairs (i,j) that after splitting belong to different teams. In other words, the sum of the cut edges is supposed to be minimized while partitioning the intersection graph G_{int} into two graphs.

The graph partition problem belongs to the class of NP hard optimization problems. A large variety of approaches exist to solve the graph partition problem. An overview is given in [52].

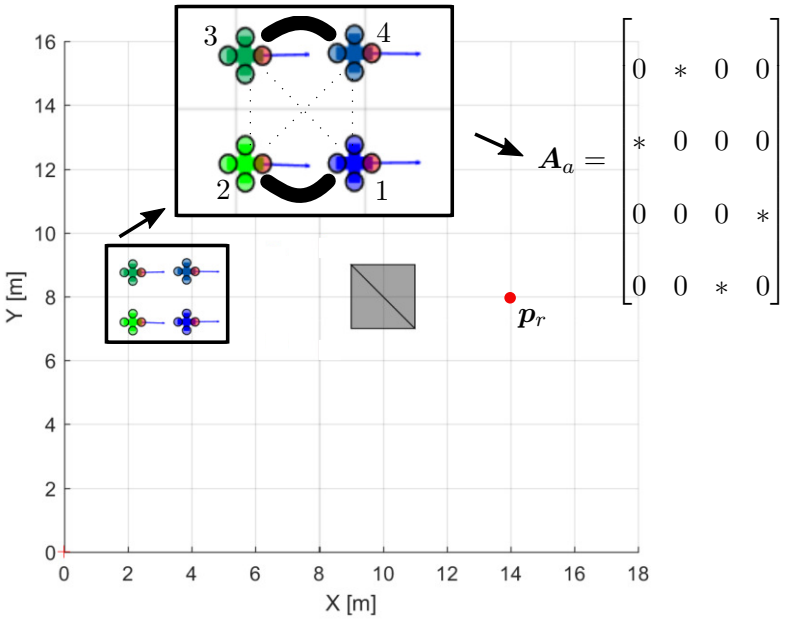
Here, each robot executes an implementation of [51], which is based on the concept of [53]. The graph partitioning algorithm uses the concept of spectral partition in combination with a k-means clustering algorithm to compute the graph partition, which minimizes the sum of the cut edges.

Even though each agent computes the graph partition individually, due to the consensus on $\boldsymbol{A}_a$, the result of partitioning the graph G_{int} into two independent graphs $G_{1,int}$, $G_{2,int}$, is the same for each robot within the same team.

Afterwards, the edges in the communication graph $\mathcal{G}_j$ are cut according to $G_{1,int}$ and $G_{2,int}$, in order to complete the splitting procedure of the formation.



(a) Isometric view of the GDOFCR $\mathcal{P}_i$ (coloured).



(b) Top view. Size of $\mathcal{P}_{ij}$ is represented by line thickness (doted line represents empty intersection). Resulting adjacency matrix A_a with non-zero elements being represented by $*$.

Figure 4-8: Scheme of progress from the GDOFCR to the intersection graph for a team of four quadrotor UAVs.

4-3-2 Merging

Algorithm 6 Check merging of team k

1: **for** All $i \in \mathcal{R}_k$ **do**
2: Determining the best team to merge with
3: **end for**
4: Distributed consensus on team to request merging with
5: **if** Solution of consensus step is non-empty, namely a team l **then**
6: Exchange merging requests with team l
7: **end if**
8: **if** Team l requests to merge with team k **then**
9: Merge with team l
10: **end if**

The merging procedure is outlined in Algorithm 6. Each robot i searches for other teams $k \in \mathcal{J}_i$, which satisfy all merging criteria. In case multiple teams satisfy the criteria, one team is chosen according to the cost explained below. Afterwards, distributed consensus is performed among the agents to agree on one formation, to request to merge with. In case no robot team $l \in \mathcal{J}_i, \forall i \in \mathcal{R}_k$ satisfies the merging criteria, the merging check ends here. Assuming a merging request exists to merge with formation l, the merging requests are exchanged in between teams k and l. If equally the merging request of team l responds to team k, both teams merge.

Determining the Merging Candidate

Algorithm 7 Team to request merging with according to agent i

1: Best formation $T_{i,best} = \emptyset$, best region intersection $\mathcal{P}_{i,best} = \emptyset$, size of best region intersection $V_{i,best} = 0$
2: **for** One agent j of each team $l \in \mathcal{J}_i$ **do**
3: Receive common convex region $\mathcal{P}^l$, reference $\boldsymbol{q}_{l,r}$, number of agents in team n_l and position of the centroid $\boldsymbol{x}_{l,c}$
4: **if** All conditions are satisfied **then**
5: **if** $V^{kl} > V_{i,best}$ **then**
6: $T_{i,best} = l$, $V_{i,best} = V^{kl}$, $\mathcal{P}_{i,best} = \mathcal{P}^{kl}$
7: **end if**
8: **end if**
9: **end for**

According to the above outline of the merging process, each robot $i \in \mathcal{R}_k$ of a team k determines the best team to merge with according to its limited communication range $r_{\mathcal{C}}$. The procedure is synthesized in Algorithm 7.

Therefore, each agent i within a team contacts one agent j of a different communication range and requests their common region $\mathcal{P}^l = \mathcal{P}_1 \cap \mathcal{P}_2, \ldots, \cap \mathcal{P}_{n_l}$, the reference $\boldsymbol{q}_{l,r}(t)$, in addition to the number of agents n_l in the team and position of the teams centroid $\boldsymbol{x}_{l,c}$. With this

information, it can be checked if all conditions for merging defined in Section 3-2 are satisfied for team l.

According to Section 3-2, the defined conditions for merging are satisfied once:

- The distance of the centroids of the teams satisfies $d_{c,kl} = \|\boldsymbol{x}_{k,c} - \boldsymbol{x}_{l,c}\|_2 \leq d_{kl,max}$, where $d_{kl,max}$ denotes the maximum allowed distance for merging.
- The references for the teams satisfy $\boldsymbol{q}_{k,r} = \boldsymbol{q}_{l,r}$.
- The size V^{kl} of the intersection of the common convex regions $\mathcal{P}^k$, $\mathcal{P}^l$, for the two formations $\mathcal{P}^{k,l} = \mathcal{P}^k \cap \mathcal{P}^l$, conform to $V^{kl} \geq V_{t,min} \cdot n_{k,l}$, where $V_{t,min}$ is the minimum required size for one robot and $n_{k,l} = n_k + n_l$ the total number of robots of both teams.

If all requirements are met by multiple teams, merging with a team with a larger region intersection V^{kl} is preferred. This results into at most one merging team candidate per robot within a team. The information $T_{i,best}$, $V_{i,best}$ and $\mathcal{P}_{i,best}$ are kept by robot i, in order to perform distributed consensus to agree on a robot team to request merging with.

Distributed Consensus on Merging Formation

After each agent of the team computed its personal preference, distributed consensus is performed among the agents $\forall i \in \mathcal{R}_k$ to agree on at most one robot team to merge with. An outline of the computations for agent i is illustrated in Algorithm 8.

Algorithm 8 Agreement on merging request

$\mathcal{P}_i(1) = \mathcal{P}^{kl}$, $\mathcal{P}_i(0) = \emptyset$, $\tilde{T}_{i,best}(1) = T_{i,best}$, $\tilde{T}_{i,best}(0) = \emptyset$
for $k = 1, \ldots, d_{\mathcal{G}}$ **do**
  $\Delta\mathcal{P}_i(k) = \mathcal{P}_i(k) \setminus \mathcal{P}_i(k-1)$, $\Delta\tilde{T}_{i,best}(k) = \tilde{T}_{i,best}(k) \setminus \tilde{T}_{i,best}(k-1)$
  Send $\Delta\mathcal{P}_i(k)$ and $T_{i,best}$ to all $j \in \mathcal{N}_i$
  Receive all $\Delta\mathcal{P}_j(k)$ and $\Delta\tilde{T}_{j,best}(k)$ from all $j \in \mathcal{N}_i$
  $m = \arg\max_{i,j} \left(V\left(\mathcal{P}_i(k)\right), \left(\Delta\mathcal{P}_j(k)\right)\right)$
  if $m = i$ **then**
    $\mathcal{P}_i(k+1) = \mathcal{P}_i(k)$, $\tilde{T}_{i,best}(k+1) = \tilde{T}_{i,best}(k)$
  else
    $\mathcal{P}_i(k+1) = \Delta\mathcal{P}_j(k)$, $\tilde{T}_{i,best}(k+1) = \Delta\tilde{T}_{j,best}(k)$
  end if
end for

This results into consensus on a request of team k to merge with a team $\tilde{T}_{best} = l$, which has the largest region intersection $\mathcal{P}^{kl}$ among all teams satisfying the merging requirements, if such team exists.

Assuming the above described team l exists, the merging request is exchanged with team l. If finally team l requests to merge with team k, both teams merge into one robot team.

4-4 Optimal Formation

After performing the splitting and merging, there exist several independent teams $k \in \mathcal{K}$. In this Section each agent i of team k computes the locally optimal target formation within $\mathcal{P}^k = \mathcal{P}_1 \cap \mathcal{P}_2, \ldots, \cap \mathcal{P}_{n_k}$. Linear feasibility constraints are added to furthermore constrain $\mathcal{P}^k$. This is done to ensure that the entire region $\mathcal{P}^k$ used for the optimal formation state is reachable for all robots within $[0, \tau]$ without exceeding the maximum allowed velocity q_{max}.

This is done according to the approach of [2], where a non-linear optimization problem is solved for each of the available template formations $f_t \in \mathcal{I}_f^k$.

$$\begin{aligned} \boldsymbol{z}_{f_t}^* &= \arg\min_z J_f(\boldsymbol{z}) \\ s.t. \quad & \mathcal{V}(\boldsymbol{z}, f) \oplus t_1 \subset \mathcal{P}, \\ & sd_0 \geq 2max(r, h), \\ & \|\boldsymbol{q}\|^2 = 1. \end{aligned} \tag{4-18}$$

Here, $\boldsymbol{z}_{f_t}^*$ is the optimal formation state vector for template formation f_t. The first constraint ensures that the position of all robots is within the obstacle free convex region $\mathcal{P}^k$. The second constraint prevents inter robot collision and the third constraint ensures the quaternion $\boldsymbol{q}$ to be of unit length. Due to the usage of the convex regions $\mathcal{P}^k$ to constrain the optimization problem, the computation is very efficient.

The cost function $J_f(\boldsymbol{z})$ to be minimized penalizes the weighted deviance from the reference $\boldsymbol{p}_r(t_1)$ at time $t_1 = t_0 + \tau$, a preferred size $\tilde{s}$ and a preferred orientation $\tilde{\boldsymbol{q}}_{rot}$. It is written as

$$J_f(\boldsymbol{z}) = \omega_t \|\boldsymbol{t} - \boldsymbol{p}_r(t_1)\|^2 + \omega_s \|s - \tilde{s}\|^2 + \omega_q \|\boldsymbol{q}_{rot} - \tilde{\boldsymbol{q}}_{rot}\|^2 + c_{f_t}, \tag{4-19}$$

where ω_t, ω_s, ω_q are weights, c_{f_t} is a predefined cost for formation f_t.

In [2], prior to solving the optimization problem, the agents perform distributed consensus on a common convex region $\mathcal{P}$. In contrast to this, the convex regions $\mathcal{P}_i \, \forall i \in \mathcal{R}_k$ are known to all agents within the formation from Section 4-3. Therefore each agent is able to compute the intersection $\mathcal{P}^k$ individually and the consensus step is not necessary.

Another difference to [2], the regions $\mathcal{P}^k$ computed here do not necessarily contain the current agent positions $\boldsymbol{p}_i(t_0) \, \forall i \in \mathcal{R}_k$. Nevertheless, the entire set $\mathcal{P}^k$ is reachable for every agent without collision based on the calculation of $\mathcal{P}_i \, \forall i \in \mathcal{R}_k$ in Section 4-2.

The constraint optimization in Eq. (4-18) was introduced by [24]. It can be solved with state of the art Sequential Convex Program (SCP) solvers. In this case the non-linear solver SNOPT [54] is applied. After solving the optimization problems, the best formation

$$\boldsymbol{z}^* = min(\boldsymbol{z}_{f_t}^* \quad \forall f_t \in \mathcal{I}_f^j) \tag{4-20}$$

is applied to determine the target robot positions $\left\{r_1^*, \ldots, r_{n_k}^*\right\}$.

The common regions $\mathcal{P}^k$ derived from the GDOFCR $\mathcal{P}_i$ in Section 4-2 tend to be smaller than the one computed as in [2]. This is due the inclusion of goal progress through obstacle avoidance. As a consequence, the optimization problem might become infeasible in dense environments. This is the case, if the computed region $\mathcal{P}^k$ is too small. In this case, the algorithm falls back on the regions computed as in [2] to compute the optimal target formation.

4-5 Position Assignment and Real-Time Control

The outcome of computing the optimal formation in Section 4-4 is a formation state $\boldsymbol{z}^*$ as in Eq. (4-20). With Eq. (2-3), it directly translates into a set of target positions $\left\{\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_j}\right\}$ for the robots. Hence, the target positions need to be assigned to the robots. Afterwards, the robots move towards their corresponding reference while avoiding collision.

4-5-1 Position Assignment

Robots within a formation are assigned to the target positions while minimizing the sum of squared distances travelled

$$\arg\min_{\sigma} \sum_{i \in \mathcal{I}_j} \|\boldsymbol{p}_i - \boldsymbol{r}_{\sigma(i)}\|. \tag{4-21}$$

Here, σ denotes the permutation matrix of the target positions.

There exist several distributed algorithms to arrive at the optimal solution, based on local interaction. Hence, the approach of [2] is followed. It makes use of a distributed simplex algorithm proposed in [6], which is proven to converge within finite time and has limited communication cost per iteration.

4-5-2 Real-Time Control

The result of the position assignment in Subsection 4-5-1 is a target position $\boldsymbol{r}_i^*$ for all robots within the formations. These positions are updated immediately after a new formation state is computed and the position assignment is completed. Then, the robots navigate towards their target position avoiding collisions locally with static obstacles, dynamic obstacles and other robots. This is achieved by computing collision-free local motions using state of the art receding horizon controller. The controller furthermore accounts for the dynamic model of the robots.

For the simulations the distributed approach of reciprocal velocity obstacles with motion constraints for aerial vehicles, introduced by [50], is used. For the experiments on the setup, the receeding horizon controller by [33] is applied, as its formulation allows for a simple connection to the used quadrotor UAVs.

Chapter 5

Simulation Results

Within this chapter, the approach presented in Chapter 4 is tested using simulation with teams of quadrotor UAVs. . Furthermore, the simulation environment used is .

The computations for formation control of robot teams including splitting and merging are done according to the approach presented as in Chapter 4. MATLAB serves as the simulation framework, where the nonlinear dynamic model of the quadrotors employed by [50] is used as it is verified with real quadrotors. More details on the dynamic model can by found [42] among others.

To compute the GDOFCR a modified version of IRIS [25] is used, which includes the functions to compute intersections of convex regions. Graph splitting is done using the implementation provided by [51]. The SNOPT solver [54] is used to solve the non-linear programs as in Eq. (4-18), which results into the optimal formation state. The toolbox Drake from MIT simplifies the handling of constraints and quaternions. It furthermore serves as an interface with SNOPT. The local motion planner and trajectory tracking controller used in the experiments corresponds to the onces presented by [50].

In order to validate the general performance of the algorithm, different simulation scenarios with static and dynamic obstacles are used for a formation of four quadrotor UAVs and compared to the results obtained by [2]. To validate the applicability of the approach to larger teams of robots, simulations with up to 16 quadrotor UAVs are provided. Afterwards, the robustness of the approach is tested by randomizing the workspace as well as changing several key parameters of the method.

Throughout all simulations, the optimization weights of Eq. (4-19) are kept at $\omega_t = 0.5$, $\omega_s = 1$ and $\omega_q = 2$. The maximum speed of the Quadrotor UAVs is limited to $q_{max} = 2\ \frac{m}{s}$, the optimal scaling parameter is $\tilde{s} = 1.5$ and the desired orientation is defined as $\tilde{\boldsymbol{q}}_{rot}$.

5-1 General Performance

In this Section, the general performance of the proposed method is going to be in focus. The performance measures used are:

- The inter-agent distances of agents within the same formation.
- The time spent split in multiple formations.
- The time of the formation being broken. This is considered to be the case, if the target positions for each robot of a team k do not satisfy $\{\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{n_k}\} \subset \mathcal{P}^k$, $\forall i \in \mathcal{R}_k$.
- Scaling with the size of the robot teams.

Furthermore, the rate of success for navigation towards a fixed reference is used. For trajectory tracking, the closest distance from the center of the formations towards the reference path is used instead. Finally it is checked, how the team splitting correlates to the computed avoidance actions of each agent.

The usage of the chosen performance measures allows for quantitatively evaluate the presented approach, as they contain the goal progress or path tracking accuracy, the splitting/ merging behaviour, as well as the stability of the formation. The approach is furthermore compared to the approach of [2].

5-1-1 Four Quadrotor UAVs

In the simulations with four quadrotor UAVs, the computations for splitting and merging followed by a new formation state are repeated every $f_f = 1$ s. The time horizon for the obstacle prediction is set to $\tau = 3$ s. Furthermore, a visibility distance of $r_{\mathcal{B}} = 6$ m and communication range of $r_{\mathcal{C}} = 3$ m are considered. The local motion planning algorithm for each of the quadrotors runs at a frequency of 5 Hz.

For formations up to four robots, the following default formation shapes are considered.

- 2 robots: line
- 3 robots: triangle (1st preference), line
- 4 robots: square (1st preference), diamond

Furthermore, the rotation in the optimal formation state is restricted to only allow for rotations in the $x - y$ plane.

Static Environment

Here, a team of four quadrotor UAVs (green and blue) flies in a 3D environment with static obstacles. The environment is illustrated in Figure 5-1 and the reference is kept fixed at $\boldsymbol{p}_r = [14, 8, 6]^T$. The initial position of the centroid of the team is varied linearly from $\boldsymbol{x}_c(0) = [2, 14, 6]^T$ to $\boldsymbol{x}_c(0) = [2, 2, 6]^T$ in steps of 1 m in y-direction, leading to 13 simulation scenarios.

In the first scenario highlighted, the initial position of the center of the team is located at $\boldsymbol{x}_c(0) = [2, 8, 6]^T$.

Figure 5-2 shows the trajectories of the team during $t = [0, 20]$ s. Figure 5-3 illustrates snapshots of the robot team at crucial time instances. Figure 5-4 includes the convex regions $\mathcal{P}^j$ used for the computation of the optimal formation state. The static obstacles are visualized by grey cuboids.

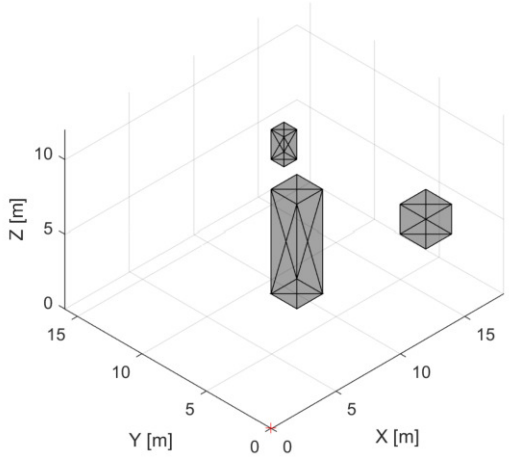


Figure 5-1: Isometric view of the static environment used in the simulations.

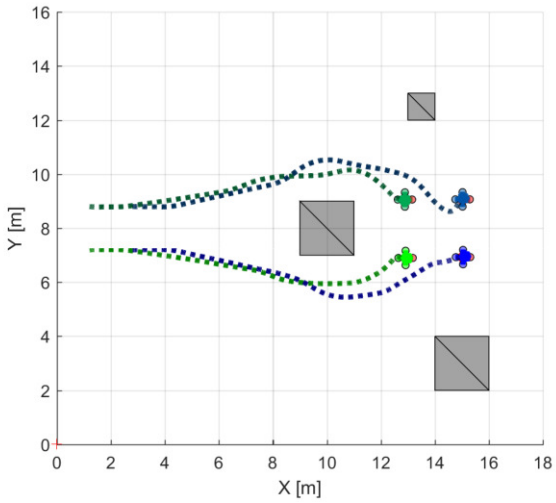


Figure 5-2: Top view. Trajectories of the Robots in between $t = [0, 20]$ s.

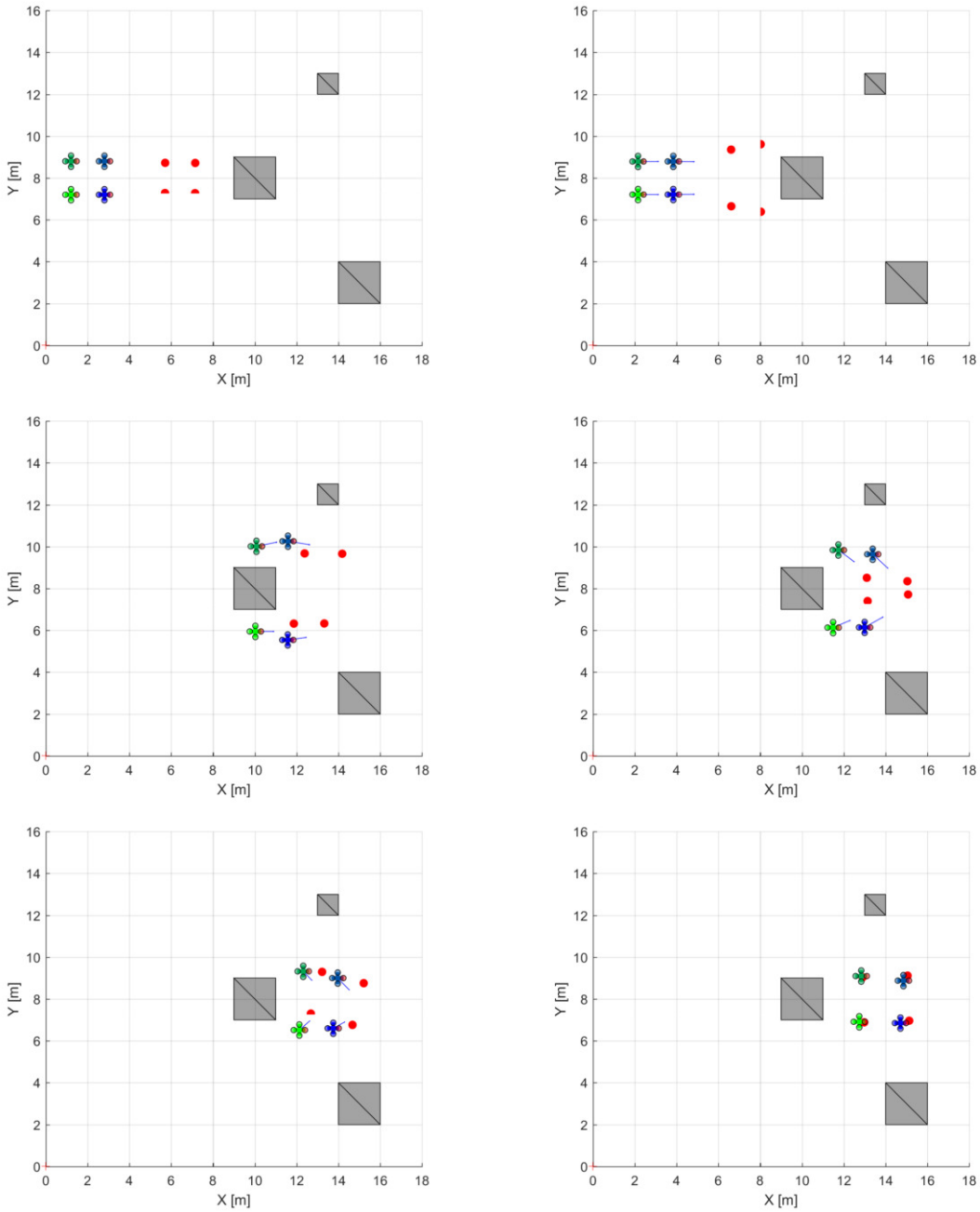


Figure 5-3: Top view. From left to right and top to bottom. Snapshots of robots and target positions (red dots) $\boldsymbol{r}_i^*$ at 0 s, 2 s, 10 s, 12 s, 13 s, 16 s.

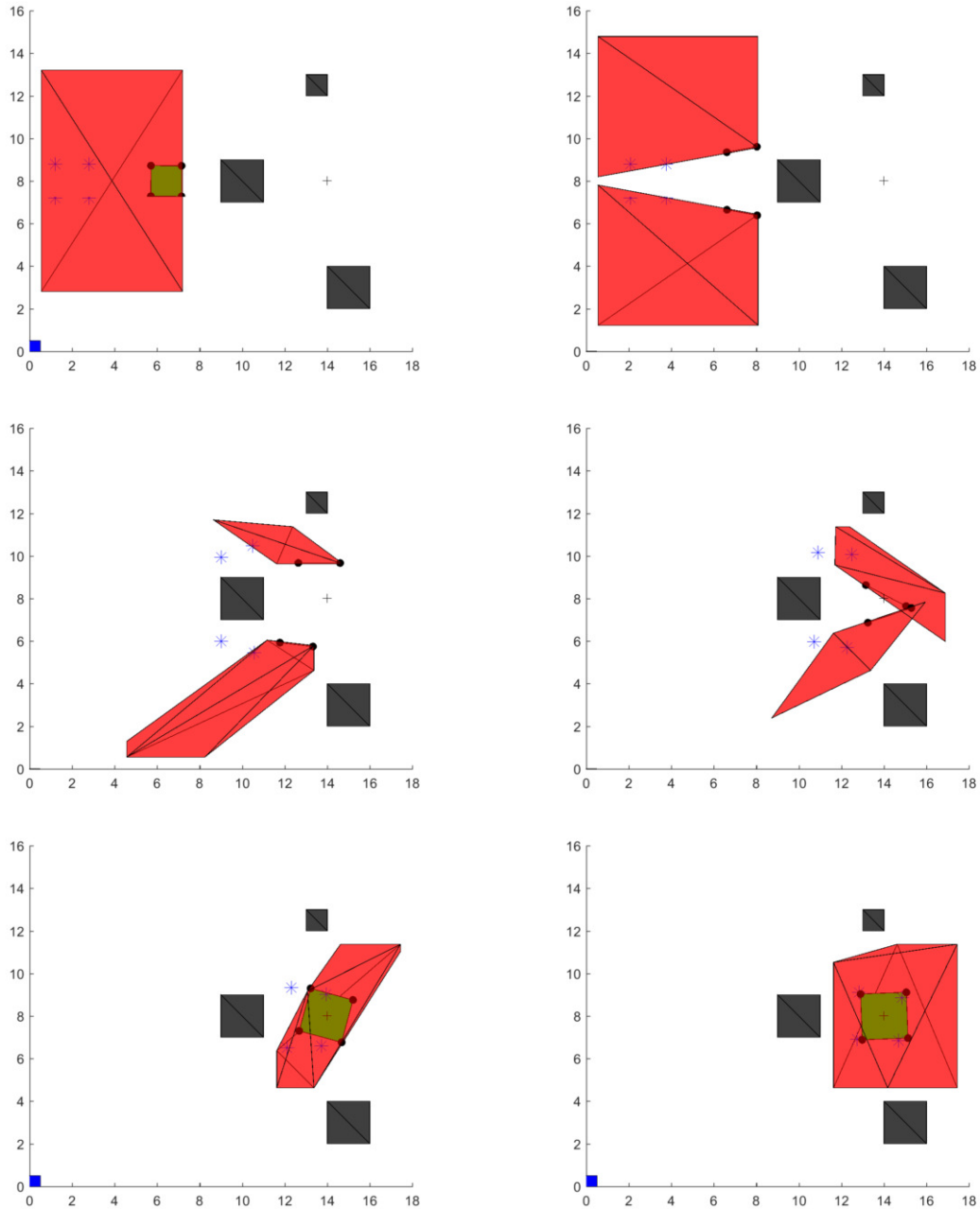


Figure 5-4: Top view. From left to right and top to bottom. Convex regions $\mathcal{P}^k$ (red), target formation (black dots) and robot positions (blue stars) at 0 s, 2 s, 10 s, 12 s, 13 s, 16 s.

The quadrotors start in an initial square formation and move towards the reference. As the visibility range covers the entire reachable space within $[0, \tau]$, the region $\mathcal{P}^1$ cannot overlap with obstacles that are out of vision. This is due to the introduction of the additional feasibility constraints as described in Section 4-4. Once the static obstacle comes into sight (2s), the robot team splits into two separate teams containing two members, where the resulting team members within the same team intend to avoid the obstacle according to the same avoidance action. The obstacles are avoided on different sides in the $x-y$ plane. As illustrated in Figure 5-3 there is an intersection of the regions $\mathcal{P}^k$ (12 s). However the teams are not in communication range yet and the intersection is furthermore not large enough to contain all robots. As the consequence, the teams do not merge yet. At 13 s, the teams merge, re-establish a square formation and reach the reference shortly after.

Similar illustrations of the simulation using the same parameters and the original approach can be found in Appendix C-1.

In the second highlighted scenario, the initial position of the center of the robot team is located at $\boldsymbol{x}_c(0) = [2, 7, 6]^T$.

Figure 5-5 shows the trajectories of the formations during $t = [0, 20]$ s. Figure 5-6 illustrates snapshots of the teams at crucial time instances and Figure 5-7 includes the convex regions $\mathcal{P}^k$ used for the computation of the optimal formation state.

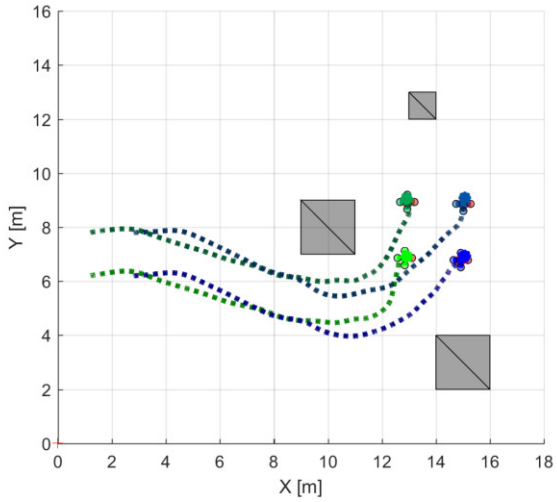


Figure 5-5: Top view. Trajectories of the Robots in between $t = [0, 20]$ s.

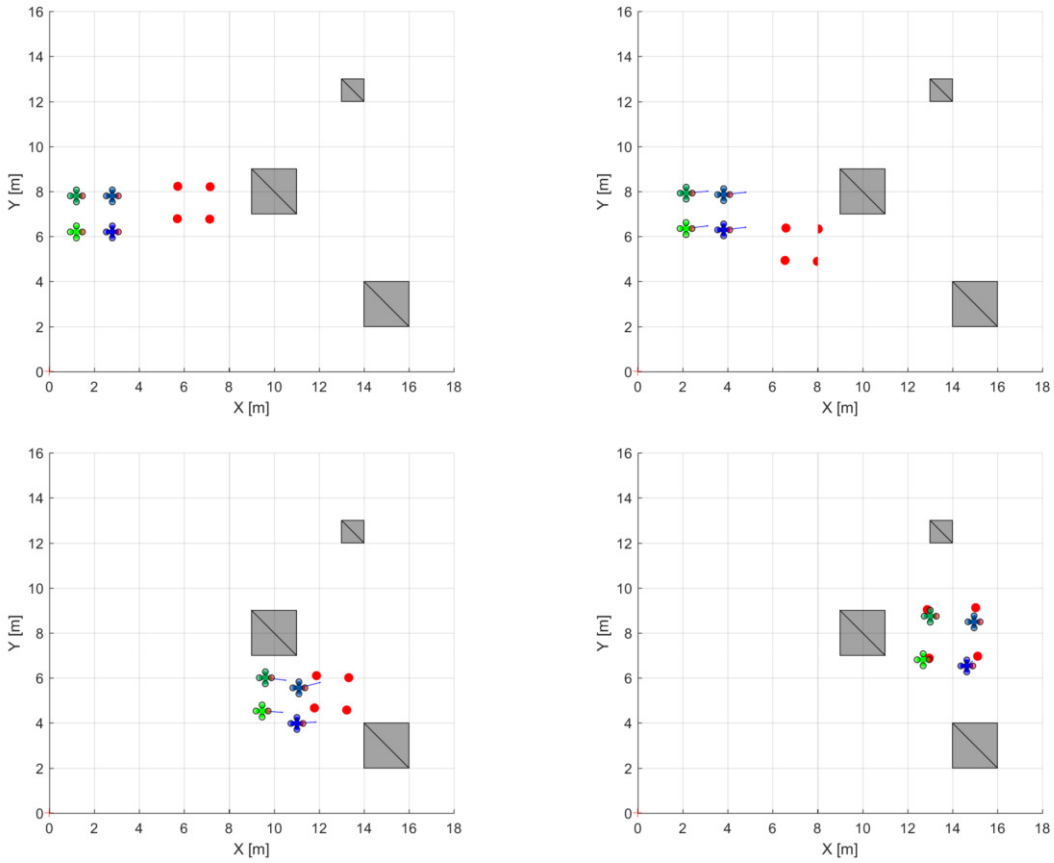


Figure 5-6: Top view. From left to right and top to bottom. Snapshots of robots and target positions (red dots) $\boldsymbol{r}_i^*$ at 0 s, 2 s, 10 s, 18 s.

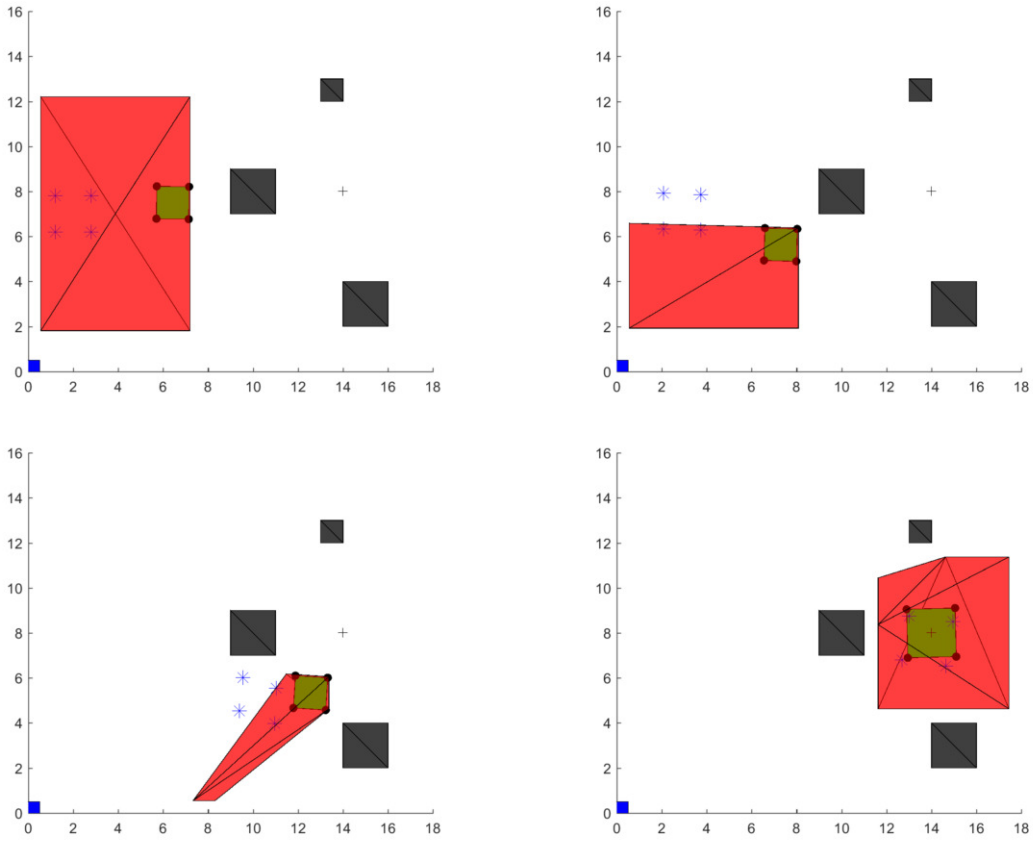


Figure 5-7: Top view. From left to right and top to bottom. Convex regions $\mathcal{P}^k$ (red), target formation (black dots) and robot positions (blue stars) at 0 s, 2 s, 10 s, 18 s.

Similarly to scenario one, the team starts in the square configuration and moves directly towards the reference, as the static obstacles are out of vision Figure 5-6 (0 s). Once the static obstacle is within the field of view and intersects with the direct path towards the reference, the robots avoid the obstacle. Different from the first scenario (2 s - 18 s), the formation does not split to avoid the obstacle and all robots avoid the obstacle to the right. This corresponds to the fact that all robots computed the same avoidance action for avoiding the large obstacle in the center of the workspace.

The performance measures of the 13 described simulation scenarios are illustrated in Table 5-1. As can be deducted from the table, the robot team navigating according to the proposed method reaches the reference for all initial positions. In contrast to this, the original approach leads to a deadlock scenario in 23 % of the initial conditions. This corresponds to the three scenarios with the initial positions $\boldsymbol{x}_c(0) = [2, 9, 6]^T$, $\boldsymbol{x}_{c,1}(0) = [2, 8, 6]^T$ and $\boldsymbol{x}_{c,2}(0) = [2, 7, 6]^T$, where the robot team approaches the large obstacle frontally. By comparing the average inter-agent distances, the only significant deviations from each other are found in the maximum inter-agent distance and the standard deviation. The larger maximum inter-agent distance using the proposed method is obtained in the scenario illustrated in Figures 5-2, 5-3 and5-4 with the initial position $\boldsymbol{x}_c(0) = [2, 8, 6]^T$. This is also the only case, where the quadrotor team splits in order to avoid the static obstacle within the 13 scenarios. The members in both teams correspond to the same avoidance action (right/left). The large inter-agent distance is observed, when the teams merge after being split for 11 s. The robots initially deviate from the square formation, but quickly converge towards the computed square shape, as illustrated in Figure 5-3 (13 s). The described behaviour is also the cause for the larger standard deviation of the inter-agent distances with the presented approach.

Table 5-1: Performance measures of 4 quadrotors in a static environment with a fixed reference.

Method	Average inter-agent distance [m]				Rate of success [%]
	mean	max.	min.	std. deviation	
New approach	1.94	3.11	1.50	0.39	100
Original approach	1.79	2.41	1.49	0.31	77

Dynamic Environment

In these simulation scenarios, the team of four quadrotor UAVs navigates in a similar 3D environment as used above. Additionally to the static obstacles, a dynamic obstacle is present as well, which moves with around $q_o = 0.5\ \frac{m}{s}$. The reference trajectory is of circular shape in the $x-y$ plane with radius $r_c = 4$ m and center at $\boldsymbol{c}_c = [10, 8, 6]^T$. Furthermore, the reference moves on the circle with a speed of $v_{traj} = 0.8\ \frac{m}{s}$. The time horizon $\tau = 3$ s, visibility range $r_{\mathcal{B}} = 6$ m and communication range $r_{\mathcal{C}} = 3$ m are the same as in the scenarios with the static environment. The initial position of the center of the formation is located at $\boldsymbol{x}_c(0) = [6, 8, 6]^T$.

Figure 5-8 shows the trajectories of the quadrotor teams during $t = [0, 25]$ s. Figure 5-9 illustrates snapshots of the teams at crucial time instances and Figure 5-10 includes the

common convex regions $\mathcal{P}^k$ used for the computation of the optimal formation state. The static obstacles are visualized by grey cuboid and the dynamic obstacle is visualized as a yellow quadrotor UAV.

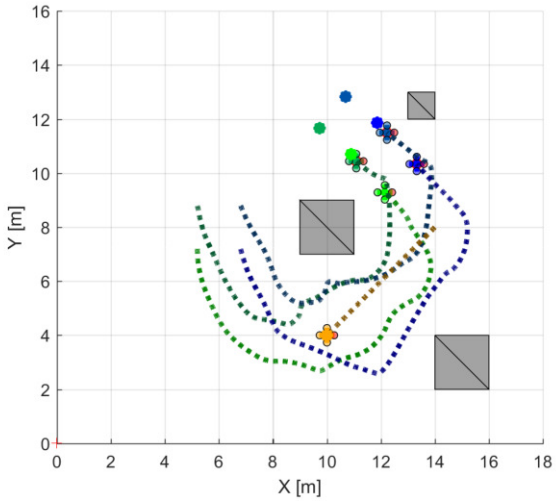


Figure 5-8: Top view. Trajectories of the robots in between $t = [0, 25]$ s.

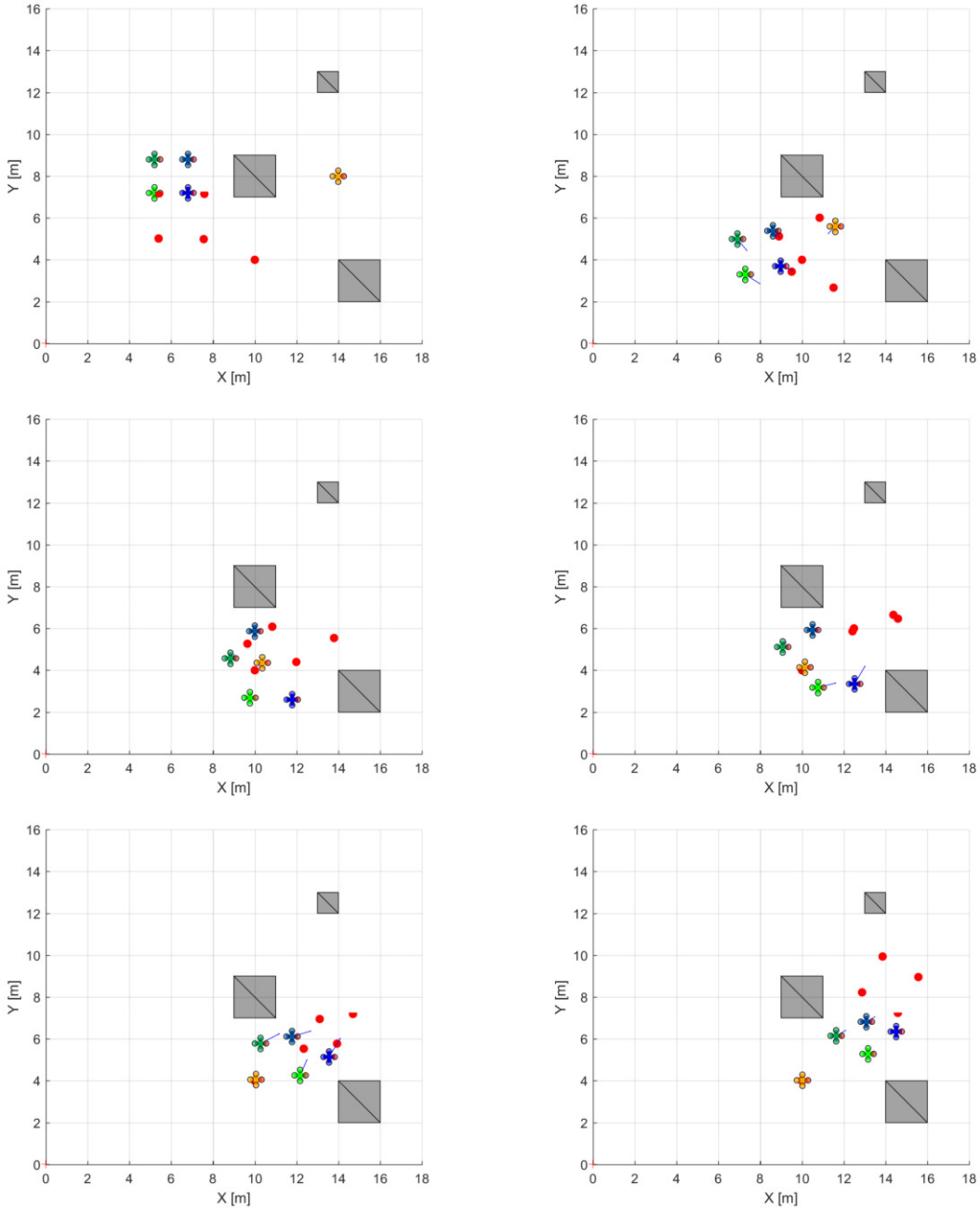


Figure 5-9: Top view. From left to right and top to bottom. Snapshots of robots and target positions (red dots) $\boldsymbol{r}_i^*$ at 0 s, 7 s, 11 s, 13 s, 15 s, 17 s.

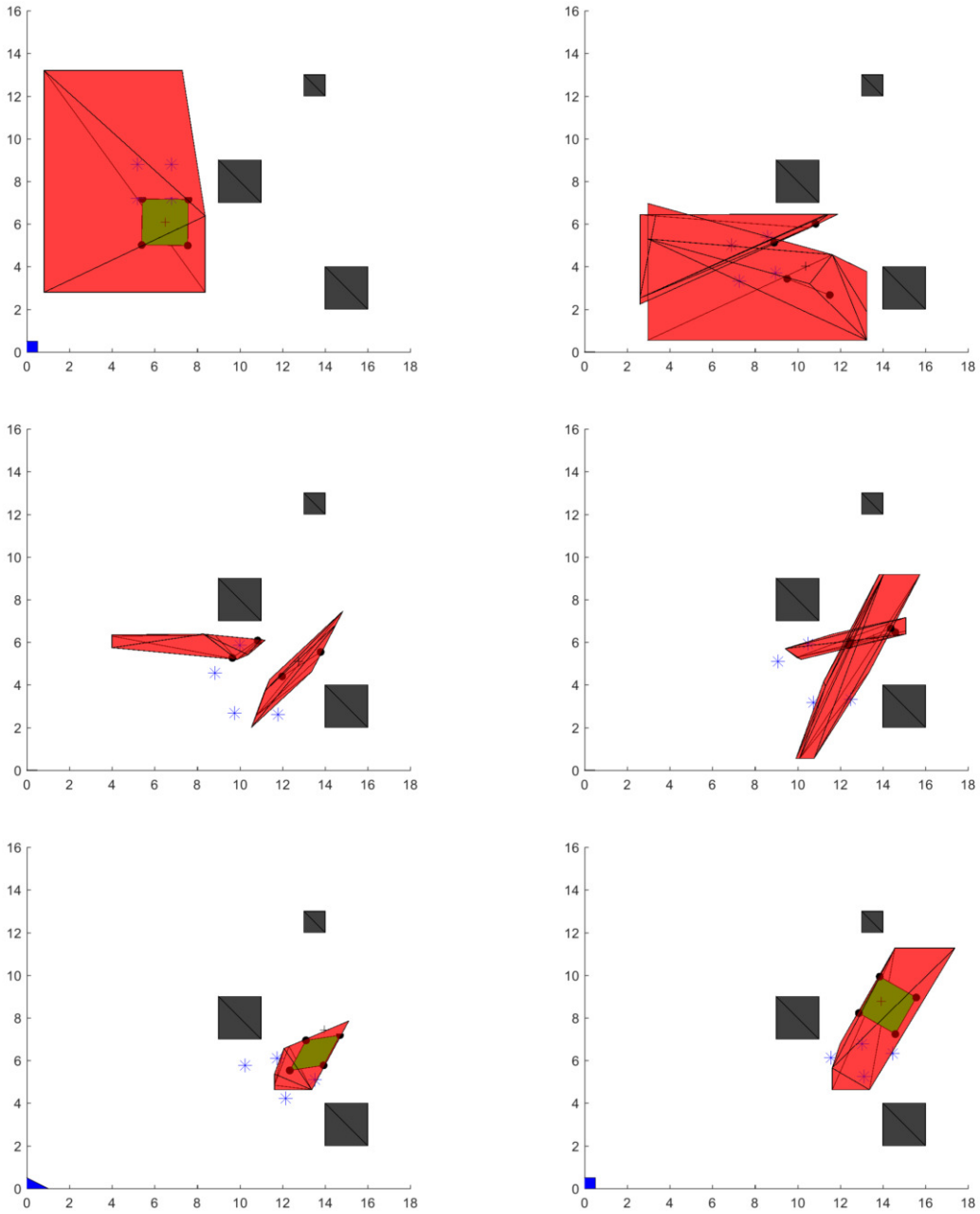


Figure 5-10: Top view. From left to right and top to bottom. Common convex regions $\mathcal{P}^k$ (red), target formation (black dots) and robot positions (blue stars) at 0 s, 7 s, 11 s, 13 s, 15 s, 17 s.

The four quadrotor UAVs start as one team in a square formation and follow the circular trajectory as can be seen in Figure 5-9. At $t = 6$ s, the team splits into two teams of two quadrotors each to avoid the dynamic obstacle. In Figure 5-10, an intersection of the common convex regions $\mathcal{P}^1$ and $\mathcal{P}^2$ is visible (6 s). Note however, that the intersection is not present in any of the region intersections $\mathcal{P}_{i,j}$ used for determining splitting of the team as in Subsection 4-3-1. This is due to the additionally introduced linear constraint. The robot teams manoeuvre around the dynamic obstacle, while avoiding collision with the static obstacles as well (7s - 13s). The teams merge back into one team at $t = 15$ s and move in diamond formation. This is due to the shape of the corresponding common region $\mathcal{P}^k$ as in Figure 5-10. With the convex region being larger, the team switches back into the square formation (17 s). Afterwards, the team continues to follow the reference, as can be seen in Figure 5-8.

Similarly to the static environment, the simulation scenario is repeated with the original approach. The illustrations of the simulation corresponding to the original approach, can be found in Appendix C-2.

The comparing performance measures are illustrated in Table 5-2. While the mean value of the inter-agent distances of both approaches are similar to each other, the maximum value of the original approach is significantly larger. This value occurs during the two seconds, for which the formation is broken and the dynamic obstacle is avoided using the local motion planner only. During these two seconds the robots still move as one team, but the formation shape is lost completely. This also explains the larger standard deviation in the inter-agent distances.

Using the proposed method, the team splits into two teams for 8 seconds to avoid the dynamic obstacle on one side each. This leads to the dynamic obstacle being avoided with the teams maintaining formation, however split into two teams. In addition, the team members of both teams correspond to the same computed avoidance action (right/left).

It is observed that the common obstacle free regions $\mathcal{P}^k$ computed using the presented approach tend to be smaller than the ones using the original approach. Furthermore, the computed optimal formation state more often includes a rotation around the vertical axis than in the original approach.

Table 5-2: Performance measures of 4 quadrotors in a dynamic environment and circular reference.

Method	Inter-agent distance [m]				Time split [s]	Time broken [s]
	mean	max.	min.	std. deviation		
New approach	1.97	3.41	1.26	0.44	8	0
Original approach	2.17	4.38	1.11	0.61	-	2

5-1-2 16 Quadrotor UAVs

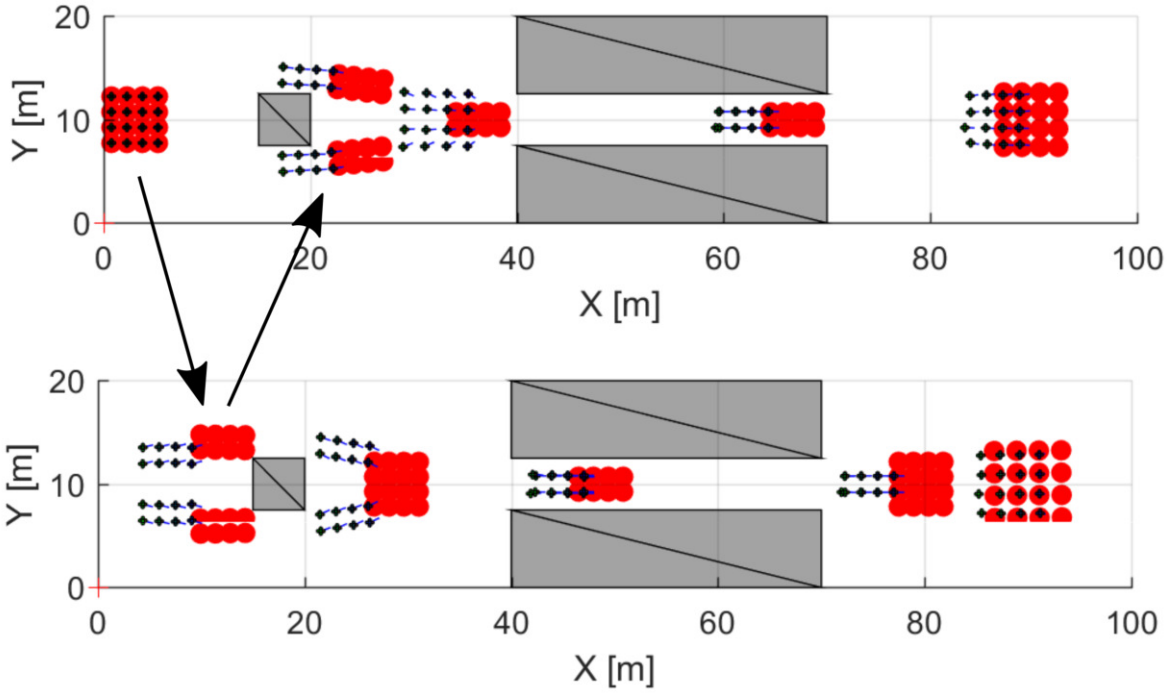
In order to evaluate the applicability of the method to a larger number team members, simulations with a team of 16 of quadrotor UAVs are performed. Due to the larger computational effort, the computations for splitting and merging followed by a new formation state are repeated every 2 s. The time horizon for the obstacle prediction is set to $\tau = 5$ s. Furthermore, a visibility distance $r_{\mathcal{B}} = 10$ m and communication range of $r_{\mathcal{C}} = 5$ m are considered. The frequency of the local motion planning remains at 5 Hz.

The template formations for up to 16 quadrotors can be found in Appendix B.

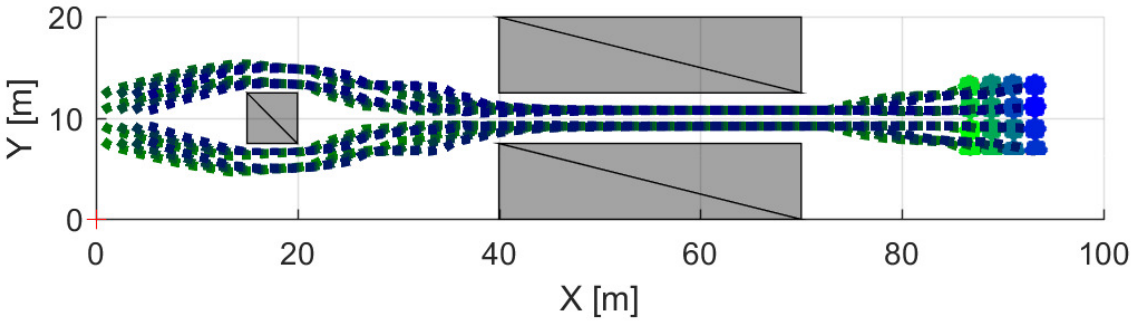
The optimal formation state is kept restricted to only allow rotations in the $x - y$ plane (horizontal orientation only).

In Figure 5-11 the simulation scenario containing several static obstacles is illustrated. Furthermore, it contains the top and side view of the trajectories. The center of the initial robot team is located at $\boldsymbol{x}_c(0) = [10, 10, 5]^T$. The reference consists of two setpoints. $\boldsymbol{p}_r(t) = [30, 10, 5]^T$ is active $\forall t \leq 40$ s. Afterwards, $\boldsymbol{p}_r(t) = [90, 10, 5]^T$, $\forall t > 40$ s.

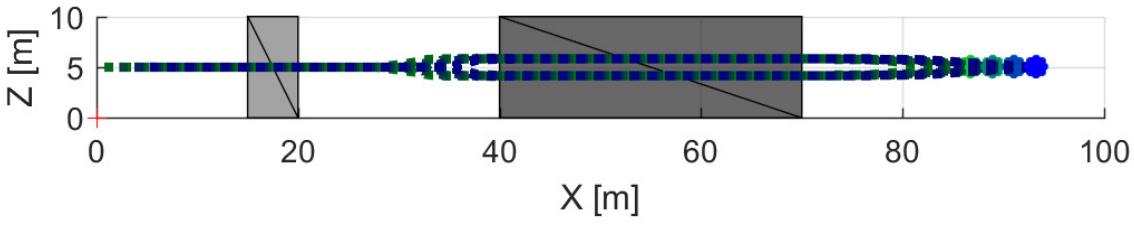
As can be seen in Figure 5-11a, the team starts in the preferred 4x4x1 formation. When the obstacle is encountered, the team splits into two teams containing eight members each using a 4x2x1 formation. The obstacle is avoided by each team individually on different sides. The computed avoidance actions of the robots within one team coincide. The teams merge back into one team and the preferred 4x4x1 formation as soon as they get into communication range, as the other conditions for merging are already met before. The team reconfigures into the 4x2x2 formation to squeeze through the narrow passage. During the transition, the local motion planner of the quadrotors avoid inter-agent collision. After leaving the narrow corridor, the team returns to its preferred 4x4x1 formation reaching the final reference after $t = 95$ s.



(a) Alternating top view of snapshots of quadrotors (black/blue) and target positions (red).



(b) Top view. Trajectories of the Robots in between $t = [0, 100]$ s.



(c) Side view. Trajectories of the Robots in between $t = [0, 100]$ s.

Figure 5-11: 16 quadrotors navigate in an environment with three static obstacles. The reference is $\boldsymbol{p}_r(t) = [30, 10, 5]^T, \forall t \leq 40$ s and $\boldsymbol{p}_r(t) = [90, 10, 5]^T, \forall t > 40$ s.

5-2 Robustness

In order to evaluate the robustness of the proposed method, key parameters of the method are changed and the environment is partially randomized. The simulations are done in combination with the dynamic environment from Subsections 5-1-1 and a circular trajectory. In addition, the static environment from Subsection 5-1-1 with a static reference is used for the robustness evaluation with respect to parameter changes as well.

These scenarios are chosen, as they contain splitting and merging behaviour due to static and dynamic obstacle avoidance, tracking of a time-invariant and time-variant reference position and reconfiguration of the formation used by the quadrotor team.

5-2-1 Parameters

The method presented makes use of several parameters, which are believed to influence the performance of the algorithm. These parameters are changed, one at a time from the default values as in Subsection 5-1-1 in order to evaluate their effect. The default values are:

- Frequency of computation for splitting/merging and locally optimal formation state $f_f = 1$ Hz.
- Communication range $r_{\mathcal{C}} = 3$ m.
- Time horizon $\tau = 3$ s.
- Visibility range $r_{\mathcal{B}} = 6$ m.
- Translation factor of the intermediate goal $f_{int} = 1$, as described in Eq. (4-10).

The results obtained with the static environment are illustrated in Table 5-3. Furthermore, the results with the dynamic environment are illustrated in Table 5-4. As can be seen, the formation does not break in any scenario, expect when the translation factor of the intermediate goal is reduced to 0.5 in the dynamic environment.

Frequency of Computing Splitting/Merging

First, the frequency of computing splitting and merging in combination with the locally optimal formation state is varied. Decreasing the frequency to $f_f = 0.5$ Hz it results in variations in the maximum inter-agent distances. In the static environment, it is larger than the default with 3.46 m and in the dynamic environment smaller than the default with 3.03 m. It is believed this inconsistency is caused by the sampling instances in combination with the environment.

By increasing the sampling frequency to $f_f = 2$ Hz, the maximum inter-agent distance, which is observed during time instance the teams merge, is consistently higher for the static and dynamic environment than the results obtained with the default parameters. This observation is expected, as the higher sampling frequency makes it more likely that an iteration of the presented method is executed immediately after all conditions for merging are satisfied.

Table 5-3: Performance measures using the presented method. A team of 4 quadrotor UAVs in a static environment with a fixed reference and varying parameters.

Changed parameter	Inter-agent distance [m]				Time split [s]
	mean	max.	min.	std. deviation	
Default	1.94	3.11	1.50	0.39	11
$f_f = 0.5$	2.13	3.46	1.53	0.52	11
$f_f = 2$	2.05	3.65	1.47	0.48	11
$r_{\mathcal{C}} = 2.5$	2.06	3.01	1.50	0.48	12
$r_{\mathcal{C}} = 6$	2.19	4.16	1.51	0.60	9
$\tau = 2$	2.04	3.05	1.42	0.47	10
$\tau = 5$	1.95	3.93	0.96	0.56	13
$r_{\mathcal{B}} = 2$	1.88	2.96	0.98	0.43	10
$r_{\mathcal{B}} = 10$	2.08	3.33	1.51	0.48	13
$f_{int} = 0.5$	2.06	3.14	1.53	0.48	11
$f_{int} = 2$	2.11	3.32	1.54	0.50	10

Communication Range

Next, the communication range $r_{\mathcal{C}}$ is taken into account. Reducing it below $r_{\mathcal{C}} = 2.5$ m causes the communication graph to be disconnected for the static and dynamic environment at times. As a consequence, the consensus steps within the approach fail to converge and the algorithm stops.

With a communication range of $r_{\mathcal{C}} = 2.5$ m, all values from the inter-agent do not differ significantly from the results achieved by using parameter settings in the static and dynamic case. Due to the smaller communication range, it is expected that the teams stay split longer than in the default parameter case. This is the case, as the teams need to be closer to each other to communicate and therefore merge. In the dynamic environment, the time $t_{split} = 12$ s spend split into two teams is with four seconds more than with the default settings. In the static case, the deviation is one second.

In the second variation the communication range is increased to $r_{\mathcal{C}} = 6$ m. In this case, the following observations are made. In the dynamic environment, non of the performance measures differ more than five percent from the default results. In the static case however, the teams merge two seconds faster than in the default case. This correlates with the maximum inter-agent distance being more than 1 m larger than the default case. As a consequence, the standard deviation in this case i higher as well.

In both environments, the obstacle intersecting with the path towards the reference is avoided by each team on a different side (left/right). The team members correspond to the same computed avoidance action.

Table 5-4: Performance measures using the presented method. A team of 4 quadrotor UAVs in a dynamic environment, circular reference and varying parameters.

Changed parameter	Inter-agent distance [m]				Time split [s]	Time broken [s]
	mean	max.	min.	std. deviation		
Default	1.97	3.41	1.26	0.44	8	0
$f_f = 0.5$	1.99	3.03	1.19	0.48	8	0
$f_f = 2$	1.98	3.50	1.31	0.43	8.5	0
$r_{\mathcal{C}} = 2.5$	1.96	3.69	1.35	0.40	12	0
$r_{\mathcal{C}} = 6$	1.99	3.46	1.20	0.42	8	0
$\tau = 2$	2.16	3.50	1.25	0.53	9	0
$\tau = 5$	2.04	3.82	1.39	0.53	10	0
$r_{\mathcal{B}} = 2$	2.05	3.58	1.13	0.45	7	0
$r_{\mathcal{B}} = 10$	1.99	3.30	1.33	0.43	8	0
$f_{int} = 0.5$	2.16	4.23	1.18	0.61	10	2
$f_{int} = 2$	1.98	3.33	1.14	0.44	9	0

Time Horizon

For a time horizon of $\tau = 2$ s, the following observations are made. In the dynamic environment, the team sometimes splits multiple times, leading to a maximum number of three teams. After avoiding the dynamic obstacle, the formations merge back into one team in all simulations. Furthermore, each agent avoids the obstacle according to its computed avoidance action. This phenomenon is not observed in the static environment. The inter-agent distances do not differ significantly form the default. However, the standard deviation and mean values are slightly higher in both cases.

Similar to the case of a reduced time horizon, the phenomenon of scattering into three teams is sometimes also observed for a time horizon of $\tau = 5$ s in the dynamic environment. In addition, the teams stay split 2 s longer in the dynamic and static case. Even though the maximum and minimum of the inter-agent distances are larger and lower, the navigation is safe and no collision occurs.

With the above exception, the obstacle avoidance in combination with team splitting occurs as observed with the default parameters.

Visibility Range

The visibility range is reduced to $r_{\mathcal{C}} = 2$ m first. Note that in this case, the common convex regions $\mathcal{P}^k$ can intersect with obstacles that are out of vision but reachable within $[0, \tau]$. This is due to the regions being constrained by the reachability condition and not full visibility of the reachable region. In the dynamic environment, the team successfully splits to avoid the obstacle. However, one robot visibly deviates from the direct path towards its target position due to interference of the local motion planner to avoid collision. Nevertheless, no collision occurs. This behaviour is not observed in the static environment. In both the static

and dynamic environment the teams stay split 1 s shorter than in the default cases. This behaviour is unexpected. It could be caused by obstacles only being taken into account when they are closer to the robots and therefore the target positions $\boldsymbol{r}_i$ being closer to the reference. Furthermore, the minimum inter-agent distance is, at 0.98 m, the smallest one obtained throughout the experiments.

When the visibility range is increased to $r_{\mathcal{C}} = 10$ m, almost all performance are similar to the ones obtained with the default parameters in the dynamic environment. In the static environment, the teams are split for 13 s, which is two seconds longer than in the default case.

In both environments, the obstacle intersecting with the path towards the reference is avoided by each team on a different side (left/right). The team members correspond to the same computed avoidance action.

Translation Intermediate Goal

The translation factor of the intermediate goal is first reduced to $k_{int} = 0.5$. In the dynamic environment, the team splits into two attempting to avoid the dynamic obstacle, as in the other cases one team on each side. However, the formation of one team breaks for 2 s. In that case, the obstacle is avoided using the local motion planner only. This results the maximum inter-agent distance to have the highest observed value with 4.23 m. The performance measures and the collision avoidance show no deviation from the default in the static environment.

Increasing the translation factor to $k_{int} = 2$, leads to similar results to using the default parameters.

In the static environment, the obstacle intersecting with the path towards the reference is avoided of each team on a different side (left/right). The team members correspond to the same computed avoidance action. In the dynamic environment, the above stated behaviour only meets the observations with the increased translation factor.

5-2-2 Randomizing the Environment

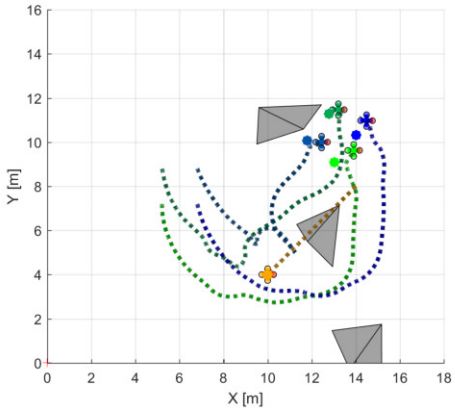
The simulation environment from Subsection 5-1-1 is used as the default environment. Then, the location of the center of the obstacles, as well as its vertices are randomized. As constraints the new center of the randomized obstacles lies within a sphere with a radius of $r_{sph} = 3$ m of the original ones. Furthermore, the vertices lie within a cuboid of the dimensions $d_x = 2$ m, $d_y = 2$ m and the height of the original obstacle. Here d_y and d_y define the length of edges of the cuboid in x and y-direction in the world coordinate frame. The dynamic obstacle remains as explained in Subsection 5-1-1 to provoke the splitting behaviour of the team. The initial position $\boldsymbol{x}_c$ and moving circular reference are kept as in the default scenario. The simulations are repeated for a total number of $N = 30$ randomized environments. An example of the randomized environment and the resulting trajectories is illustrated in Figure 5-12a

For all simulations, the robot navigation is save in the sense that no collision occurs. Due to the different shapes and positions of the obstacles, the obstacles are also avoided on top and

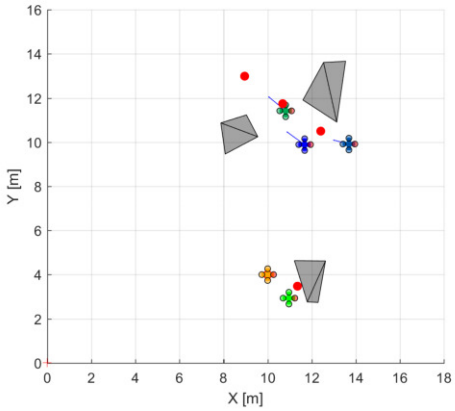
below at times. This shows, the obstacle avoidance does make full use of its given functions and works well in three dimensional scenarios.

During the simulations only one deadlock scenario is observed. It is illustrated in Figure 5-12b and occurs for a single agent that is trapped in between a static obstacle and the dilated dynamic obstacle which already reached its reference.

Furthermore, team scattering is observed in some scenarios resulting in up to four individual robot teams. However, the observed scattering behaviour is only temporary. The the teams finally merge back into a single large team and establish on of the allowed formations.



(a) Top view. Trajectories for $t = [0, \tau]$.



(b) Top view. Snapshot of the only deadlock scenario observed. It only affects a single agent.

Figure 5-12: Excepts from different simulations in randomized environments.

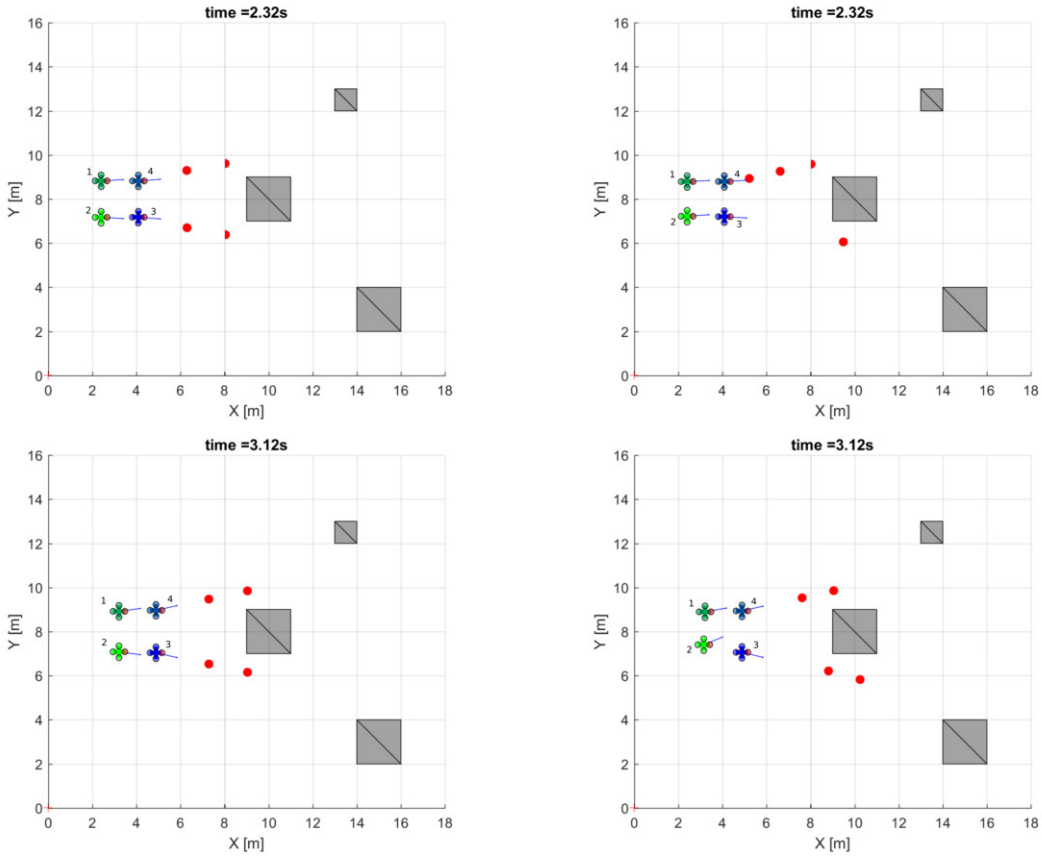
5-3 Necessity of Consensus about Obstacles

The proposed method contains a consensus step on static and dynamic obstacles within the field of view of the formation, as explained in Section 4-1. A valid question however is, if the splitting and merging behaviour also works successfully without this step, as one consensus step could be left out.

As a consequence, comparison in between the algorithm containing the consensus step and a version one without the consensus step is done. The team used consists of four quadrotor UAVs and navigates according to Subsubsection 5-1-1 from the initial position $\boldsymbol{x}_c = [2, 8, 6]^T$ to the static reference $\boldsymbol{p}_r = [14, 8, 6]^T$.

Snapshots of two teams, one with and the other one without obstacle consensus are illustrated in Figure 5-13. As differences are only observed in the splitting behaviour, the snapshots are limited to this process. More specifically, Figure 5-13a contains the team executing the consensus step on obstacles, while the team in Figure 5-13b does not agree on a common set of obstacles, with the agents being enumerated as visible in the Figures.

It can be observed in Figure 5-13a that the team splits symmetrically into two formations containing 2 robots and stays within these teams. In Figure 5-13b on the other hand, the



(a) Top view. Snapshots at 2 s, 3 s.

(b) Top view. Snapshots at 2 s, 3 s.

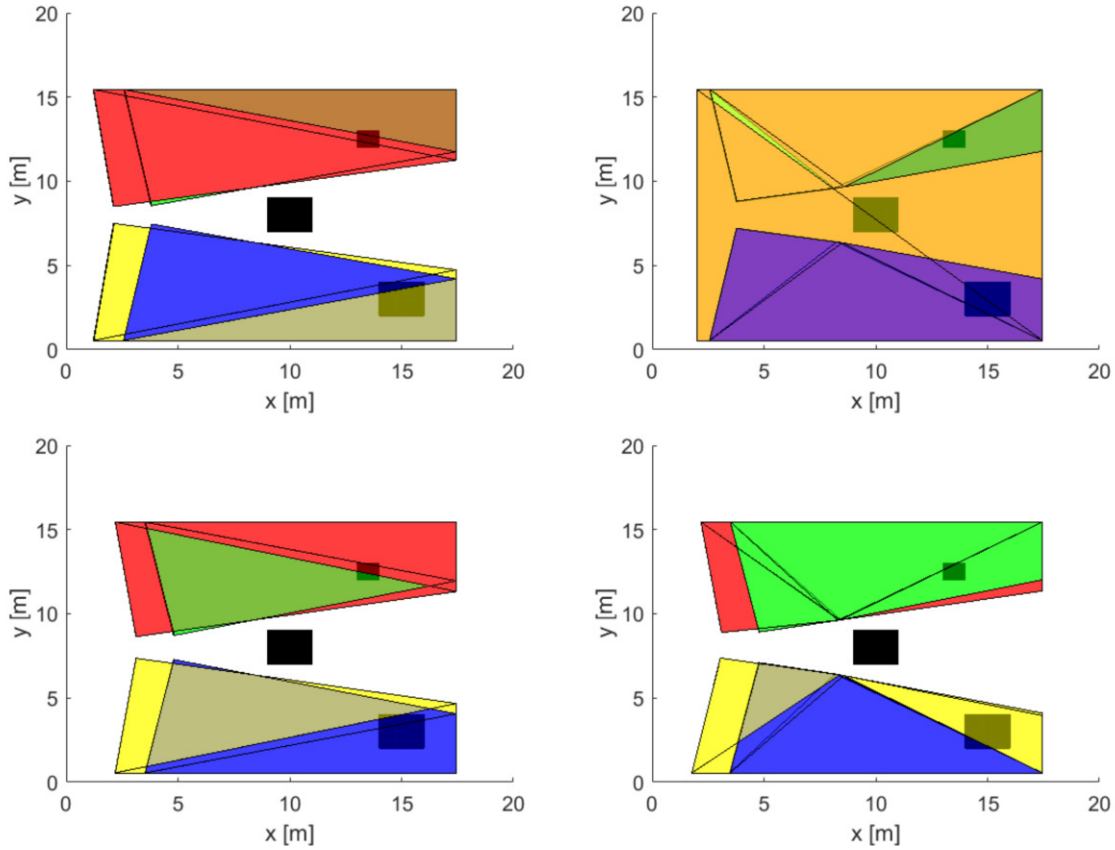
Figure 5-13: Comparison of the splitting behaviour of a formation of four quadrotor UAVs. **(a)** Consensus on obstacles. **(b)** No consensus on obstacles.

team first splits into two teams, where one contains 3 members and the other one 1 (2 s). At this time, only the row of robots closer to the obstacle have the obstacle within their field of view. Afterwards, the larger team splits once again and the single agent merges with the other team containing one agent into a team of two agents(3 s). At this time, all members have the obstacle within their field of view. Afterwards also the teams without the obstacle consensus step remain intact.

The cause for this observation can be found by looking at the illustrations of the GDOFCR in Figure 5-14. It is clearly visible for the team executing the obstacle consensus step in Figure 5-14a that the only region intersections are in between the pairs of agents (1,4) and (2,3). This leads to the region intersection matrix

$$\boldsymbol{A}_{cons} = \begin{bmatrix} 0 & 0 & 0 & * \\ 0 & 0 & * & 0 \\ 0 & * & 0 & 0 \\ * & 0 & 0 & 0 \end{bmatrix}, \tag{5-1}$$

which is used for the graph splitting. Here and in the matrix below, the symbol $*$ denotes



(a) Top view. Snapshots at 2 s, 3 s.

(b) Top view. Snapshots at 2 s, 3 s.

Figure 5-14: GDOFCR $\tilde{\mathcal{P}}_i$ during the splitting process, top (2 s) bottom (3 s). **(a)** Consensus on obstacles. **(b)** No consensus on obstacles.

non-empty entries.

Different to this, the regions in 5-13b allow for intersections in between all agents except the pair (3,4), which is the cause for the team splitting is initialized. The consequence is the intersection matrix

$$\boldsymbol{A}_{no_{cons}} = \begin{bmatrix} 0 & * & * & * \\ * & 0 & * & * \\ * & * & 0 & 0 \\ * & * & 0 & 0 \end{bmatrix}. \tag{5-2}$$

Due to the agents 1 and 2 not having knowledge of the obstacle, they compute large GDOFCR which intersect with the regions of all other agents. The resulting partition of the intersection matrix is therefore not as desired.

Similar behaviour is observed using simulation scenarios containing dynamic obstacles with four robots and a scenario with eight robots.

As already mentioned above however, when all agents have vision on the obstacle, the teams reorganize themselves and end up avoiding the obstacle similarly to the teams using the obstacle consensus step.

5-4 Computational Complexity

In order for the presented method to allow teams of quadrotors to navigate in formation in real-time, the computational complexity of the approach has to be investigated. As the experiments in Chapter 6 are conducted with a team of four quadrotor UAVs, this number of agents is taken as a reference value. Figure 5-15 illustrates the average computational times of the presented method for four quadrotor UAVs navigating in formation in the dynamic scenario as in Subsection 5-1-1. The computations are all done on a standard laptop (Quadcore Intel i7 CPU@2.8 GHz).

It can be seen that all steps are computed within 0.0335 s. This is including the assumption that all listed steps are executed within one iteration of the entire method. This however is not the case, as the graph partition is only executed if the checking for splitting determines team splitting. In that case, the checking for merging with other teams is skipped, as described in Section 4-3.

The accumulated time for all computation steps corresponds to the median of the computational time of a variation of [2] in experiments [1]. The result is promising for the real-time performance of the approach.

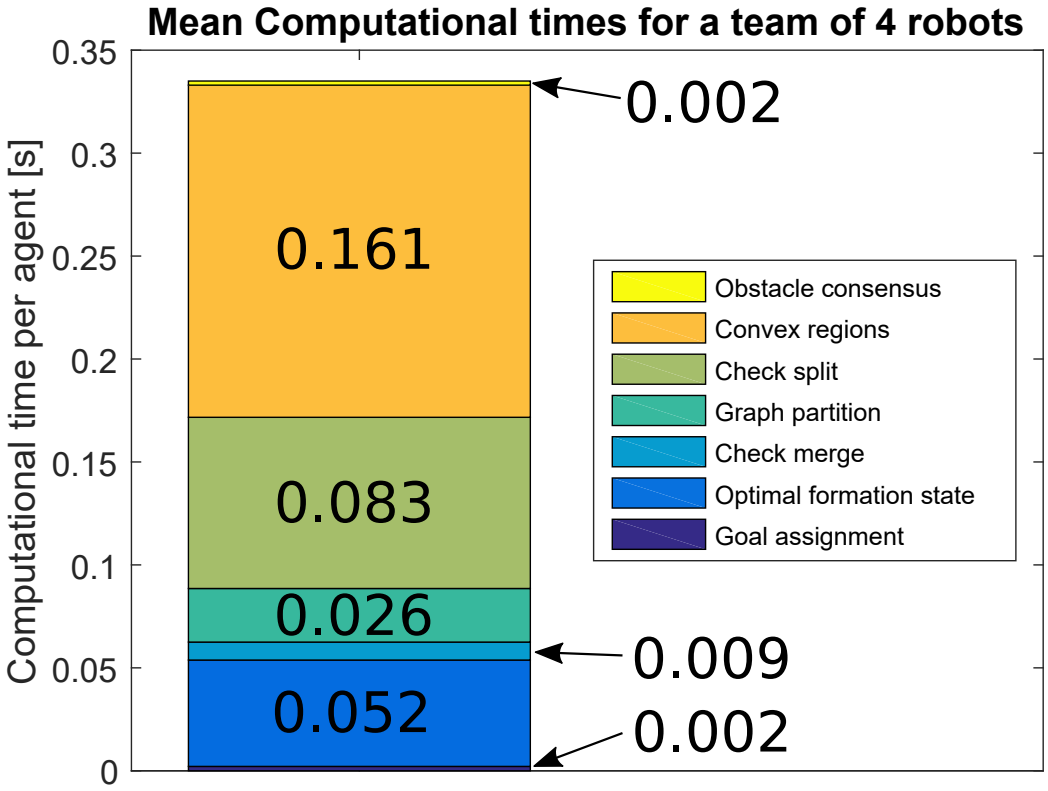


Figure 5-15: Computational times for a formation of four quadrotors.

5-5 Discussion

Within this Chapter, the presented approach is investigated regarding performance, robustness and computational complexity in simulations. In combination with the obtained simulation results, it allows for a first discussion of the approach.

The above presented content shows the simulation results of the presented method, which allows for distributed determining of team splitting and merging in combination with formation control, while taking into account limited visibility and sensing. The team splitting and merging is based on the comparison of GDOFCR, which are computed in combination with an intermediate goal that implies an obstacle avoidance action.

The simulation results in Subsection 5-1-1 show that the presented method achieved successful navigation in the presented static and dynamic environment for both a setpoint reference, as well as a time varying reference. While navigating towards the reference, the robot teams split and merge based on obstacle avoidance. It is observed that the team splits evenly in two teams of two members each. The members of each team correspond to the same avoidance action. This leads to the conclusion that indeed a simple intended path can be approximated by the GDOFCR. For paths that are highly non-linear it is questionable, if the assumption still holds to be true. In comparison to the method presented in [2], no deadlock scenario

is observed. Furthermore, the formation does not break in any scenario. The inter-agent distances are similar, with the exception to the maximum value. This value is higher in the presented approach and is observed when the teams merge.

In Subsection 5-1-2, the approach shows to perform well in a static environment with sixteen quadrotor UAVs.

From the robustness evaluation in Section 5-2 can be deducted that changes of key parameters of the method do not influence the performance directly. It is found that the method reacts the most sensitive to a reduction of the translation factor f_{int}. This can be explained due to the fact that the implied obstacle avoidance action included in the GDOFCR is easily is lost, when f_{int} is reduced and the intermediate goal does not extend to the obstacle. Furthermore the communication range is found to have impact on the merging behaviour and consensus convergence. This is due to the required communication in between different teams to merge and the assumption of a connected communication graph for the consensus algorithms to converge. Finally, the visibility range needs to be large enough to cover the reachable space within $[0, \tau]$. Otherwise, the collision avoidance included in the formation control cannot be not guaranteed. In a partially randomized environment, the approach shows to generally perform well, with only one deadlock situation observed. However, the occasional team scattering needs further investigation and seems to be a downside of this approach

The average computational time for four robots shows a similar update rate as reported in [1] during experiments. This is promising for the experiments with the quadrotor UAVs. Due to the distributed formulation of the algorithm, good scaling behaviour can be expected. For a proper evaluation of the scalability of the method with the number of agents, more simulations with teams of 16 quadrotor UAVs need to be performed

As the avoidance actions do not allow for staying in front of an obstacle, the computation avoidance actions sometimes leads to counter intuitive results, when the reference is within a static obstacle. This issue can be overcome by implementing a fifth avoidance action, which allows the agent to stay in front of the obstacle.

As previously mentioned, the size of the common region $\mathcal{P}^k$ tends to be smaller than in the original method. Different from the original method, the here computed GDOFCR of each agent do not need to contain the vertices v_i of the current formation. In simulations this results into the non-linear optimization for the optimal formation space to occasionally become infeasible. Due to the implemented fall back onto the original regions, this does not lead to problems however.

Throughout all simulations, the navigation is always safe in the sense that no inter-agent collision takes place and no collision between robot and environment occurs. Even though rarely, deadlock scenarios are still observed when the environment is partially randomized. This leads to the first limitation of the method. By taking into account only one obstacle for the computation of the intermediate goal $\boldsymbol{p}_{int}$ and computing it through a heuristic, this step is computationally light, but at the same time does allow for the formulation of optimality guarantees. Especially in dense cluttered workspaces this can lead to deadlocks. As an initial workaround, a simple A* graph search algorithm is implemented, to compute a collision free path for the team centroid towards the reference. As the graph search only determines one path collision free path for the entire team, it undermines the concept of the presented method,

which is based on the splitting and merging due to obstacle avoidance. Furthermore, it does not check for the feasibility of the path for the entire team. Hence, the initial workaround is not used any further. A more promising solution to this shortcoming could be to compute the GDOFCR according to multiple avoidance actions and afterwards compare the goal progress and/or their size in order to choose one region. A different solution involves the recurrent computation of a locally or globally optimal path for each agent using Rapidly Exploring Random Trees (RRT) or Probabilistic Roadmaps (PR). This path could then be used in order to determine the intermediate goal for the computation of the regions. This could allow the definition of optimality guarantees.

Furthermore, the approximation of highly non-convex workspaces through convex regions has it drawbacks as well. This is the case, as it reduces the usable obstacle free volume even further. The implementation of splitting and merging based on path homotopy in combination with a receding horizon method such as [4] could overcome such shortcomings. On the other hand, the representation of the non-convex space by convex regions speeds up the convergence of the optimization problems. In addition it is questionable, if navigation in formation is still desired in extremely crowded scenarios.

Chapter 6

Experimental Results

Within this Chapter, the presented method is investigated through experiments with four Quadrotor UAVs. The setup is described first, followed by the presentation of the experimental results. This Chapter concludes with the discussion of the experimental results.

6-1 Setup

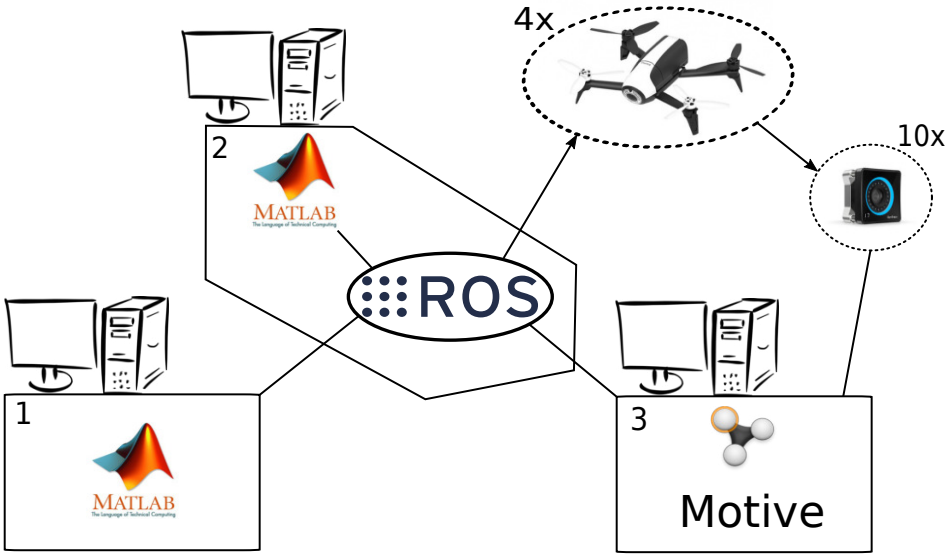


Figure 6-1: Scheme of the components involved in the experiments.

The experimental setup consists of several components, as illustrated in Figure 6-1. As can be seen, up to four quadrotor UAVs are used. Their positions are determined by using an optical motion capture system, consisting of 10 Optitrack Prime 17W cameras and the software Motive executed on a windows desktop computer (3). The formation control containing

splitting and merging of the robot teams is executed in Matlab using a standard laptop (1) (Quadcore Intel i7 CPU@2.8 GHz) and makes use of the position data aquired by the motion capture system. The computed reference positions for each quadrotor UAV serve as the inputs for an adaptation of the receeding horizon local motion planner presented in [33]. The local motion planner is implemented in Matlab, where the MPC is solved using FORCES Pro [55]. The computations are done on a standard desktop computer (2) (Dual core Intel i5 CPU@2.8 GHz). The different components are connected by the Robot Operating System (ROS), which also used to connect with the drones wirelessly. The reference commands computed by the local motion planner are sent to the quadrotor UAVs and executed by the internal controller.

6-1-1 ROS

The Robot Operating System (ROS) [56] is a open source meta-operative system for robots serves that as an interface to the quadrotor UAVs. At the same time, it allows for the connection of the different components illustrated in Figure 6-1. ROS software is organized in packages and nodes. These packages can contain ROS nodes, libraries, configuration files, third party software and other modules considered to be useful. A ROS node is a computation performing process. ROS nodes are able to communicate with each other through topics, which can be seen as data channels where nodes can listen to and furthermore publish messages on. The ROS packages used are:

- roscore
- mocap_optitrack [57]
- bebop_autonomy [58]

The roscore package is the prerequisite to running a ROS package and enabling ROS nodes to communicate with each other.

The ROS package mocap_optitrack is used to transform the data produced by the Motive, making it directly usable by Matlab. In this case, one pose is computed for each quadrotor UAV and dynamic obstacle if present. Velocities are determined by using the poses in combination with a Kalman filter internally in Matlab.

The ROS package bebop_autonomy serves as the interface to the quadrotor UAVs.

Using the Robotics System Toolbox within Matlab, Matlab creates its own ROS nodes. Hence, no additional ROS package is needed to interface with Matlab

6-1-2 Bebop Parrot 2

The experimental platform used consists of quadrotor UAVs of the brand Parrot, more specifically the model Bebop 2 which belongs to the entry level models within the consumer market. The quadrotor UAV is illustrated in Figure 6-2 and has a diameter of approximately 0.5 m. As mentioned above, the interface to one UAV is done through using a wifi dongle and the ROS package bebop_autonomy, which is based on the Software Development Kit (SDK)

Figure 6-2: The experimental platform, the Parrot Bebop 2.

offered by Bebop. This leads in total to the usage of four wifi dongles and four instances of bebop_autonomy. Due to the usage of the SDK, the state access of the UAV is limited.

The state of the quadrotor is described by its position $\boldsymbol{p} \in \mathbb{R}^3$, its velocity $\boldsymbol{q} = [\dot{p}_x, \dot{p}_y, \dot{p}_z]$ and its orientation in terms of roll Φ_q, pitch Θ_q and ψ_q. The control inputs to the system are the velocity of the quadrotor in the body-z axis $q_z = \dot{p}_z$, the desired roll angle Φ_q and the desired pitch angle Θ_q for the quadrotor and its angular speed around the body-z axis ω_z. The horizontal velocities are not directly controlled, but are a result from the pitch and roll angles.

The control inputs are then executed by the internal controller of the Bebop 2, which makes use of its Internal Measurement Unit (IMU) data. By using a different interfacing method, such as the paparazzi software package, freedom in the control design could be gained, in addition to the possibility to overwriting the IMU data. This could increase the performance and stability of the trajectory controller. On the other hand, the ROS package bebop_autonomy offers a well documented and quickly applicable solution to connecting multiple Bebop 2 drones to the computer, which makes it to be chosen for the experiments presented here. In addition to that, the anticipated manoeuvres are not aggressive and expected to be successfully handled by the internal controller.

6-1-3 Further Implementation Details

The formation control algorithm receives the current state of the drones and obstacles and computes a reference position for each drone accordingly to the here presented method. The computations are performed in a continuous manner, meaning as soon as one execution is finished the next iteration of the formation control algorithm starts immediately. Similar to the simulations, the communication radius of the drones kept at 3 m, the radius of the visibility distance is set to 4.5 with a time horizon of $\tau = 3$ s and a maximum allowed velocity of $q_{max} = 1.5\ \frac{m}{s}$.

The receding horizon local motion planner is used in combination with a horizon of $N_c = 20$ steps, at 0.005 seconds each. The frequency of the local motion planner is relatively constant lies around 10 Hz.

6-2 Experiments

Initial experiments with two to four quadrotor UAVs were performed at the Delft Center for Systems and Control (DCSC) laboratory of the Technical University of Delft, whose dimensions are 8m length x 3.4m width x 3m height. Especially the width of the usable workspace at the DCSC is challenging, as during obstacle avoidance little free space in between the drones an the workspace boundaries is observed. As a consequence, the experiments shown here were conducted at the Cyberzoo of the faculty for aerospace engineering, Technical University of Delft whose dimensions can be approximated by a cube with an edge length of 8m. An overview of the workspace and the world coordinate system is provided in Figure 6-3. Furthermore, the maximum allowed speed of the drones is reduced in the local motion planner to 1 $\frac{m}{s}$, as the inter-agent collision avoidance sometimes caused the drones to oscillate. In order to keep the regions $\mathcal{P}^k$ at a reasonable size, the maximum allowed velocity is not changed within the formation control.

Figure 6-3: Overview of the workspace for the experiments, including the definition of the world coordinate system (black).

During all the experiments presented here, the initial position of the centroid of the robot team is located at $\boldsymbol{x}_c = [-2.5, 0, 1.5]^T$. On command, the reference is changed from the initial position to $\boldsymbol{p}_r = [2.5, 0, 1.5]^T$.

Due to problems with the equipment and time constraints, position recordings of the drones and obstacles are only available for the dynamic environment.

6-2-1 Static Obstacle

Similar to the initial simulations, the first experiments presented here are conducted with one static obstacle in the center of the workspace. The static obstacle is virtually created with a height of 3 m. This is done to provoke collision avoidance to the side, which is more likely to lead to team splitting. The static obstacle visible in the snapshots is placed where the virtual obstacle is placed to visually underline the experiments.

In the first case, the static obstacle is located on the y-axis, but with an offset to the x-axis. It corresponds to the simulated environment, as illustrated in Figure 6-4. Furthermore, the

Figure contains the simulated trajectories of the drones. Snapshots of the corresponding experiment are illustrated in Figure 6-5. As can be seen, the static obstacle is avoided by the robot team on one side while maintaining the square formation. In simulations, the team quickly switches to the diamond formation, but this behaviour was not observed during the experiments. Finally, the team reaches its reference position.

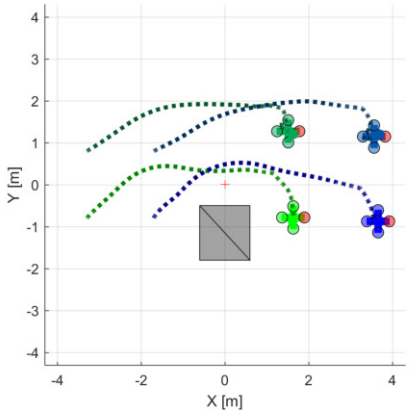


Figure 6-4: Top view. Overview of the environment with the simulated trajectories of the drones avoiding the obstacle on one side.

In the second case, the static obstacle is located at the origin. The simulated environment with the corresponding trajectories are illustrated in Figure 6-6. The snapshots of the experiment are illustrated in Figure 6-7. The quadrotor team starts in a square formation. As soon as the new reference is applied, the team splits into two teams of two members each. The obstacle is avoided on one side each. When approaching the reference, the two teams merge back into one team and re-establish the square formation. In some experiments, interference of the local motion planner is observed as some of the drones have very similar target positions $\boldsymbol{r}_i$ before merging.

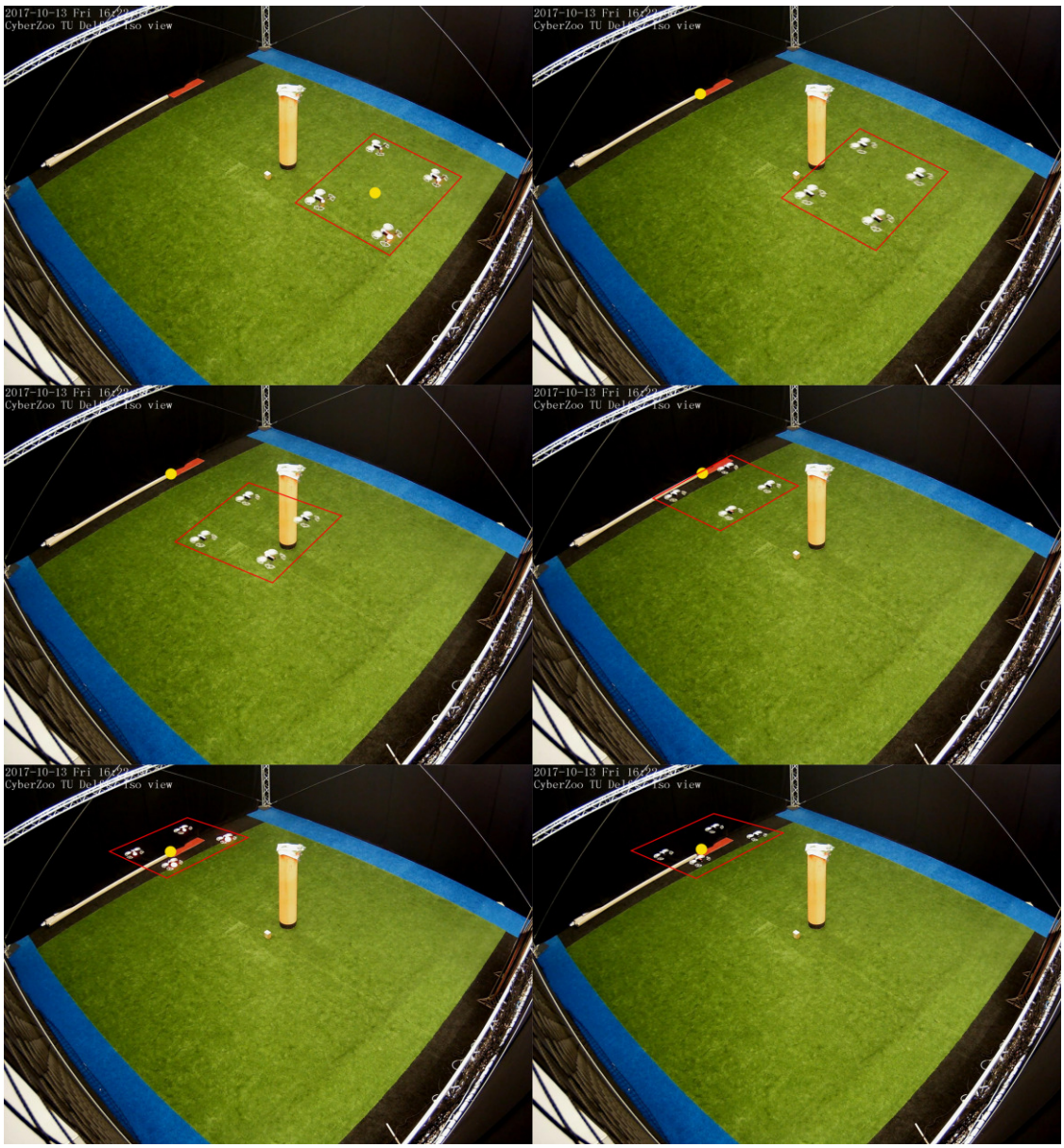


Figure 6-5: Isometric view. From left to right, top to bottom. Snapshots of the UAV team avoiding a static obstacle on one side. Reference in yellow. Formation shape in red.

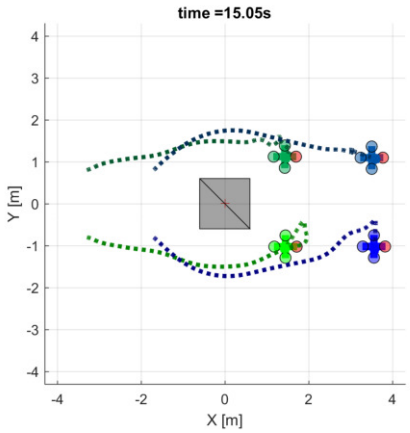


Figure 6-6: Top view. Overview of the environment with the simulated trajectories of the drones, containing splitting and merging.

Figure 6-7: Isometric view. From left to right, top to bottom. Snapshots of the UAV team avoiding static obstacle by splitting and afterwards merging. Reference in yellow. Formation shape as red lines.

6-2-2 Dynamic Obstacle

In order to test the formation control algorithm in a dynamic environment, a human serves as a dynamic obstacle. The person is equipped with a hat which is tracked by the motion capture system, equally to the quadrotor UAVs. In order to show the capabilities of the algorithm for team splitting and merging, the height of the human is virtually increased to $h_h = 3$ m. This provokes the quadrotor UAVs to avoid the human to the side and not on top. Within the formation control algorithm, the dilated human is represented by a hexagonal prism.

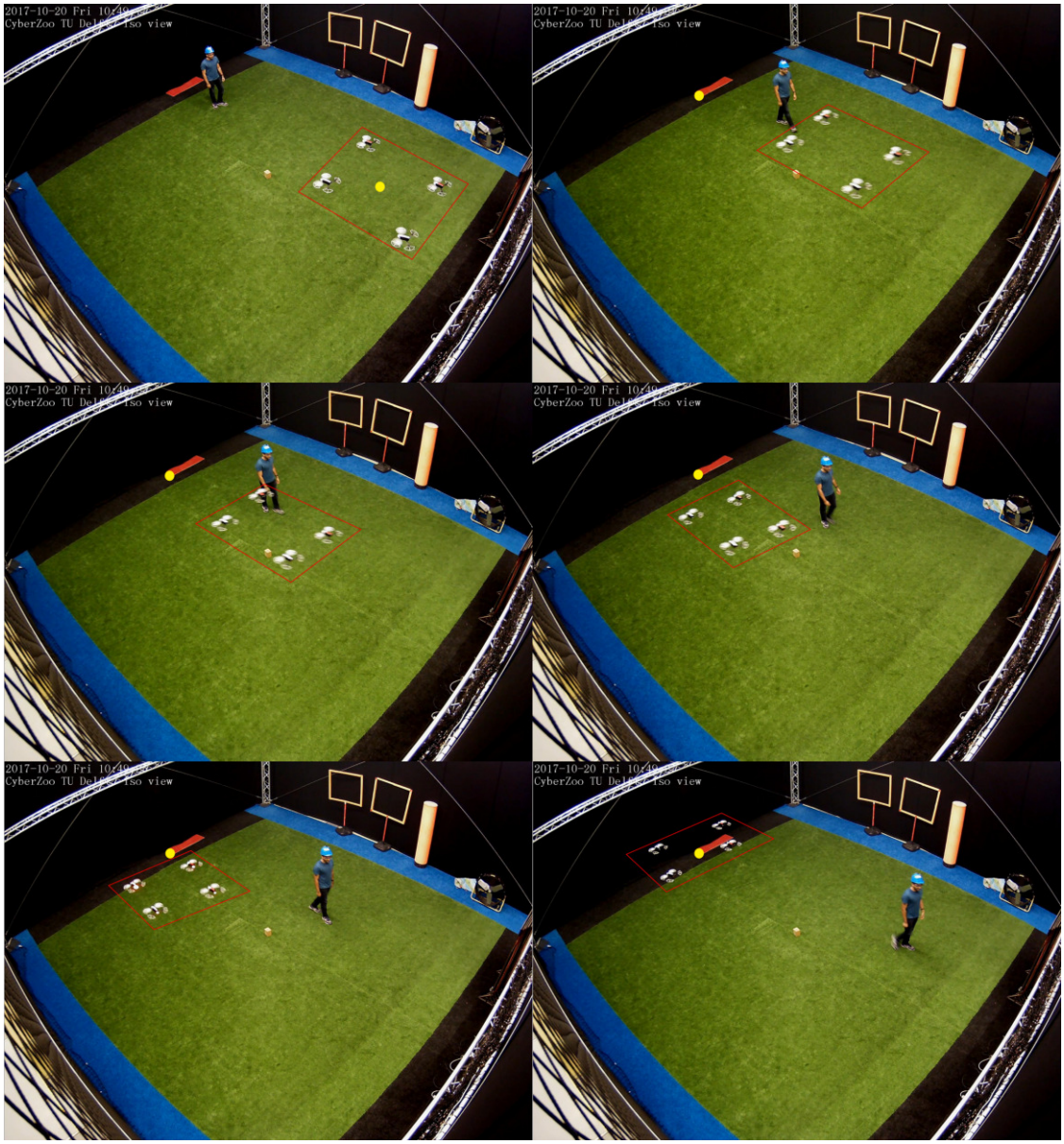


Figure 6-8: Isometric view. From left to right, top to bottom. Snapshots of the UAV team avoiding a moving human on one side. Reference in yellow. Formation shape in red.

Figure 6-8 illustrates snapshots from the experiments where a human moves with walking

speed in negative direction along the x-axis. While doing so, he is positioned with an offset to the x-axis. As can be seen from the snapshots, the quadrotor team first moves straight towards its reference position. As the person gets closer to the UAVs, the quadrotor team avoids the human to the left, while staying together in one team and keeping the square formation. After the human is avoided, the quadrotor team reaches its reference.

In a different experiment, the human walks again in negative direction along the x-axis. Different from the previous case, the human walks without an offset from the y-axis and therefore a heading towards the center of the quadrotor team.

As the snapshots of the described experiment show in Figure 6-9, the quadrotor team splits into two teams of two quadrotor UAVs each, after the setpoint reference is applied. The team members in both teams correspond to the same computed avoidance action (right/left). Both quadrotor teams avoid the approaching human safely, while making progress towards the goal. Afterwards, both teams successfully merge into one team and re-establish the square formation. In this particular case however, one quadrotor UAV does not reach its target position $\boldsymbol{r}_2$ (marked by a red dot) until the end of the experiment. During repetitions of the experiments containing successful splitting and merging of the quadrotor team, this peculiar behaviour of one robot not reaching its target position is not observed again.

While in simulations of the presented scenario the team directly splits into two when the new reference is applied, in experiments the team first attempts to avoid the human as one team, as illustrated in Figure 6-10a. This is due to no empty region intersection $\mathcal{P}_{i,j}$ even though the additional linear constraint is present. An illustration of these regions is given in Figure 6-10b.

As the resulting common region $\mathcal{P}^1$ is very small, the optimization problem for the optimal formation state is infeasible. The algorithm therefore falls back onto the original computation of the convex regions. This behaviour is not observed in experiments. In the next iteration of the algorithm however, the team splits according to the computed collision avoidance actions. The trajectories of the simulated scenario and the experiment are found in Figure 6-2-2. In Figure 6-11b it can be seen that while the team starts moving towards the reference, it moves into negative y-direction as well. Only afterwards, the team splits and the trajectories are similar to the ones simulated in Figure 6-11a.

Apart from the here presented scenarios, the experiments are repeated with a human approaching the robot team from different angles. During all the experiments, the approach shows to be safe. It means that no collision between the human and a drone is observed. The collision avoidance is either included in the formation control part or, if the computation of the target positions takes too long, the MPC running at a higher frequency avoids collision.

During the experiments, the mean value of an iteration of the formation control algorithm is found to be 0.64 s. The real computation time of each iteration varies from 0.28 s to over one second. This sometimes results into a noticeable delay in the computation of the target positions $\boldsymbol{r}_i$, especially when the human moves at higher speed and changes the direction of movement quickly.

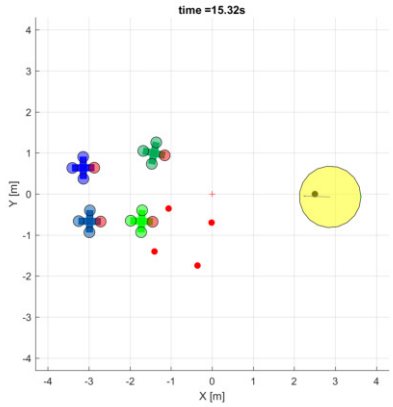
It is furthermore observed that a team occasionally scatters, when the team approaches the reference after the collision avoidance with the human.

The inter-agent collision avoidance in the modified version of [33] is not found to work as desired. Even though no inter-agent collisions are observed, the collision does not seem to be

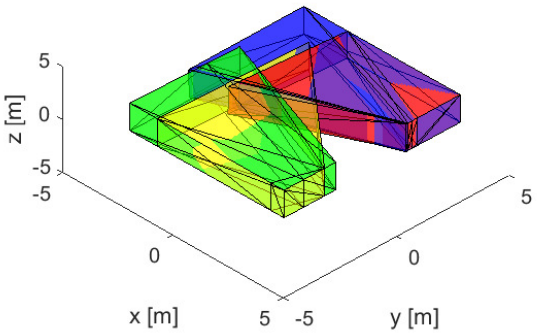
reciprocal, or future states are not taken into account properly. This is especially observable, when the drones move towards each other. This is the case, before the teams merge after being split.

Finally, a single drone sometimes quickly ascends towards the ceiling. This behaviour is believed to be caused by interference in between the sonar sensors, which operate at the same sampling frequency on all drones [1].

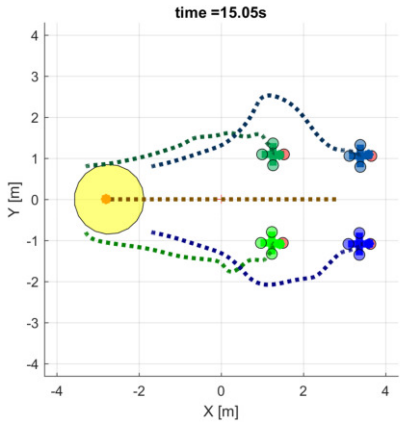
Figure 6-9: Isometric view. From left to right, top to bottom. Snapshots of the UAV team avoiding a moving human by splitting and afterwards merging. Reference in yellow. Formation shape as red lines. Target position for the drone out of position as a red dot (bottom right snapshot).



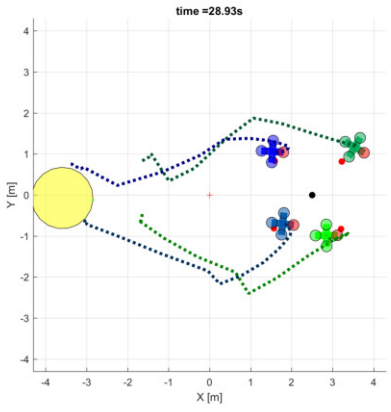
(a) Top view. Initial avoidance action of the drone team to avoid the human (yellow). Target position for the drones are represented by red dots.



(b) Isometric view. Constrained GDOFCR $\tilde{\mathcal{P}}_i$, with no empty intersections $\mathcal{P}_{i,j}$.



(a) Simulated trajectories.



(b) Recorded trajectories of the experiments.

Figure 6-11: A walking human is avoided by splitting and merging. Comparison of the simulated trajectories with the recordings from the experiments

6-3 Discussion

After showing promising results in simulations, the formation control algorithm, which contains splitting and merging due to obstacle avoidance, shows to be able of performing formation control in real-time with four drones of the type Parrot Bebop 2. During the experiments with a static and dynamic obstacle, the distributed approach is able to split/ merge teams and configuration according to the environment.

In order to further quantitatively evaluate the approach in experiments however, more experimental data with various references (constant/ time varying) needs to be collected and evaluated. In order to determine more limitations of the algorithm in practice, multiple obstacles should be included into the experiments.

Even though not observed in simulations, the computed additional constraint leading to $\tilde{\mathcal{P}}_i$ shows not to be complete in experiments. One solution could be to compute an additional linear constraint per convex region. It leads to promising results in simulations, but has to be validated in combination with experiments.

The highly varying update rate (mean computational time of 0.64 s) of the formation control algorithm in practice shows to be sufficient for moderate speeds of dynamic obstacles. A more efficient implementation could drastically improve the computational times and thereby reduce the reaction time of the algorithm.

In addition to lack of numerical experimental data, the problems regarding the inter-agent collision avoidance and the random ascent of single drones makes the evaluation of the approach in experiments more difficult.

Chapter 7

Conclusions and Future Work

Based on the formulated research question in Section 2-2, the main goal of this thesis is to incorporate efficient splitting and merging behaviour in combination with quadrotor formation control. This goal has been accomplished by developing, implementing, simulating and experimentally verifying a local concept of splitting and merging in dynamic environments, based on the comparison of Goal-Directed Obstacle-Free Convex Regions (GDOFCR) in combination with the formation control approach of [2].

Nevertheless, more detailed conclusions followed by suggestions for future work are presented below.

7-1 Conclusions

Within this thesis, specific conditions for splitting and merging of robot teams moving in formation are defined. The formulated conditions make use of the assumption that an intended path can be sufficiently approximated by a convex region. Splitting and merging of teams is determined by comparing the intersections of GDOFCR and is triggered by obstacle avoidance.

Team splitting and merging in combination with formation control has not received much research attention so far. Therefore, verifiable conditions for splitting and merging for teams in formation, as well as the approximation of an intended path by a convex region are both novel approaches. The intended path of each agent is simplified to an intermediate goal, which is computed heuristically in order to be computationally light. Even though the computed convex regions are globally obstacle free, the computed intermediate goal is computed based on a single obstacle. As a consequence, the presented method is local and does not allow for the formulation of optimality guarantees.

The computations are all done in a distributed manner while accounting for limited visibility and communication range of the agents. This possibly allows for the application of the approach to larger teams and better represents the real-world circumstances.

The algorithm is implemented in Matlab and evaluated quantitatively in simulations with in static and dynamic environments first. It shows, the defined conditions for splitting and merging promote, as intended, navigation in a large team, while the agents tend to avoid the obstacles according to their intermediate goal. Splitting and merging is observed according to the defined conditions. Therefore it is concluded that the assumption of the possibility to sufficiently approximate intended paths by convex regions is true for linear paths. Even though deadlocks can still occur, the algorithm shows to encounter such situation much less frequently than the original approach of [2]. Furthermore, the approach shows to be sufficiently robust with respect to parameter changes within algorithm and varying environments. The necessity of consensus on the set of obstacles is discussed.

The presented method is verified in experiments with four quadrotor UAVs of the type Parrot Bebop 2 and shows real-time performance in static and dynamic environments. However, the approach is also applicable to other robot platforms. For conducting the experiments, an experimental setup including a motion capture system, multiple drones, several matlab instances and ROS is re-created. In order to quantitatively evaluate the approach in experiments, more experimental results in different scenarios need to be collected however. Shortcomings of the algorithm are discussed and possible solutions are proposed.

7-2 Future Work

Although the concept is satisfying, future work is needed to furthermore improve the method and the experimental setup.

7-2-1 Method

The following suggestions regarding future work concern the method itself.

- The constraints defining $\tilde{\mathcal{P}}_i$ could be improved to increase the performance regarding team splitting.
- More experimental data needs to be collected to quantitatively evaluate the approach with experiments.
- Using a graph search method in combination with PR, or RRT can increase the performance of the algorithm, eliminate deadlocks and lead to global guarantees. However the real-time capabilities of the approach should stay in focus.
- The size of the GDOFCR could be used in order to determine the collision avoidance action in addition to the heuristic. This would not lead to team splitting, but instead would help the optimization problem for determining the optimal formation state to stay feasible.
- Other teams should be respected in the common region $\mathcal{P}^k$, in order to take load from the local motion planner.

- The applicability of splitting and merging due to path homotopy in combination with formation control established by an MPC should be investigated. It could overcome some of the shortcomings of the here presented framework.
- The approach could be extended to teams with different targets. This would allow for teams in proximity with similar paths to temporarily merge and thus occupying less workspace.

7-2-2 Experimental Setup

In order to improve the robustness of the experimental setup, future work should involve the work below.

- The inter-agent collision avoidance of the modified NMPC should be improved.
- The IMU data used by the drones to execute the given inputs should be overwritten by the data obtained by the motion capture system. This would prevent single drones from quickly ascending due to sensor interferences.
- The documentation of the experimental setup should be completed to allow for thorough knowledge transfer.
- The installation of a soft floor cover in the DCSC lab would decrease the risk of drones being damaged during experiments.

Appendix A

Rotation Intermediate Goal

Below, the rotation matrices for rotation of the intermediate goal, based on the obstacle avoidance action, are listed:

Rotation matrix corresponding to the avoidance left/right.

$$R_{l,r} = \begin{bmatrix} cos(\alpha_R) & k_{l,r}sin(\alpha_R) & 0 \\ -k_{l,r}sind(\alpha_R) & cos(\alpha_R) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A-1}$$

Rotation matrices for the avoidance on top, or below.

$$R_{int} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha_R) & k_{a,b} & sin(\alpha_R) \\ 0 & -k_{a,b}sin(\alpha_R) & cos(\alpha_R) \end{bmatrix} \tag{A-2}$$

The matrix is combined with the rotation matrix R_{yaw} that corresponds to the yaw rotation of the drone with respect to the inertial frame.

$$R_{a,b} = R_{yaw}^T R_{int} R_{yaw} \tag{A-3}$$

Appendix B

Default Formations

The used default formations in the simulations for more than four robots are:

- 8 robots
  - First preference: 2 x 4 x 1 rectangle
  - Second preference: 2 x 2 x 2 cube
- 16 robots
  - First preference: 4 x 4 x 1 square
  - Second preference: 2 x 8 x 1 rectangle
  - Third preference: 2 x 4 x 2 cuboid

Appendix C

Additional Simulation Results

Here, additional illustrations of simulation results based on the approach of [2] are provided.

C-1 Static Environment

Figure C-2 illustrates simulation results from the scenario described in Subsection 5-1-1 with the original approach. Furthermore, all simulation parameters are kept identical to ensure comparability.

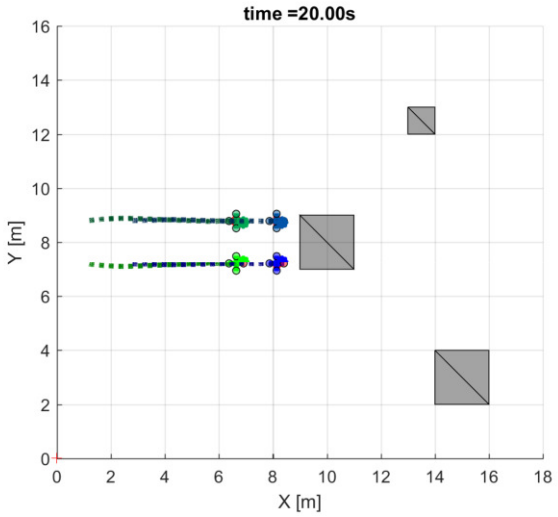
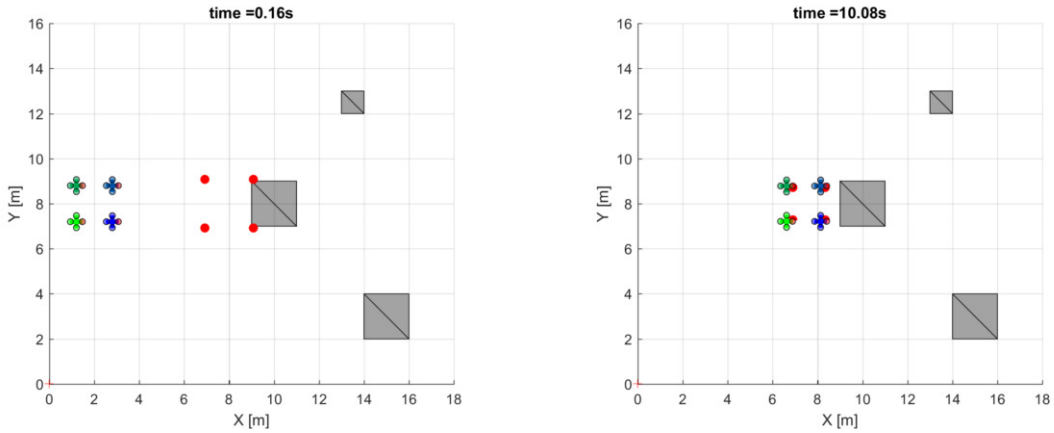
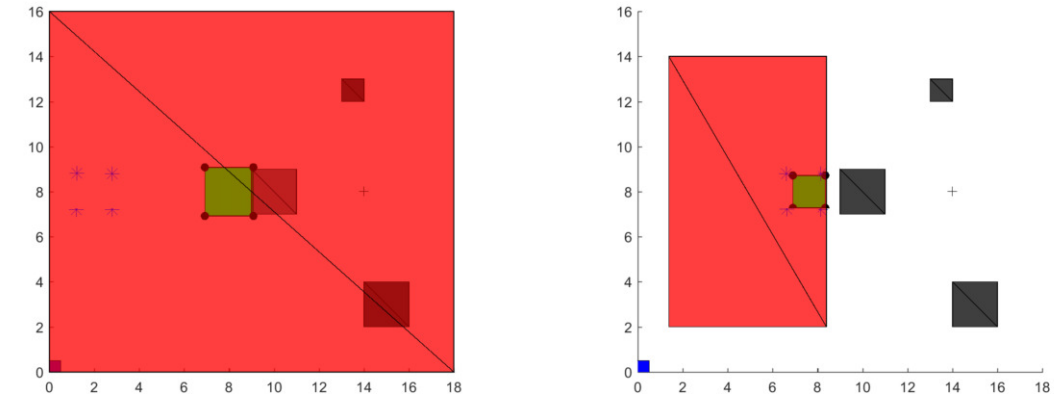


Figure C-1: Top view. Trajectories of the robots in between $t = [0, 20]$ s.



(a) Top view. From left to right and top to bottom. Snapshots of robots and target positions (red dots) $\boldsymbol{r}_i^*$ at 0 s, 10 s.



(b) Top view. From left to right and top to bottom. Snapshots of robots and target positions (red dots) $\boldsymbol{r}_i^*$ at 0 s, 10 s.

Figure C-2: Four quadrotors (green-blue) navigate in an environment with static obstacles (grey) towards a fixed reference $\boldsymbol{p}_r = [14, 8, 6]$.

C-2 Dynamic Environment

Simulation results from the scenario described in Subsection 5-1-1 with the original approach, are illustrated below. To ensure comparability, all simulation parameters are kept identical.

Figure C-3 shows the trajectories of the formations during $t = [0, 25]$ s. Figure C-4 illustrates snapshots of the formations at crucial time instances and Figure C-5 includes the convex regions $\mathcal{P}^j$ used for the computation of the optimal formation state. The static obstacles are visualized by grey cuboids and the dynamic obstacle is visualized as a yellow quadrotor UAV.

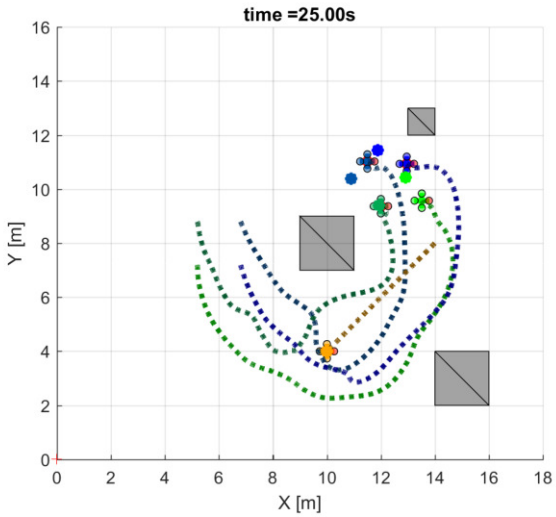


Figure C-3: Top view. Trajectories of the robots in between $t = [0, 25]$ s.

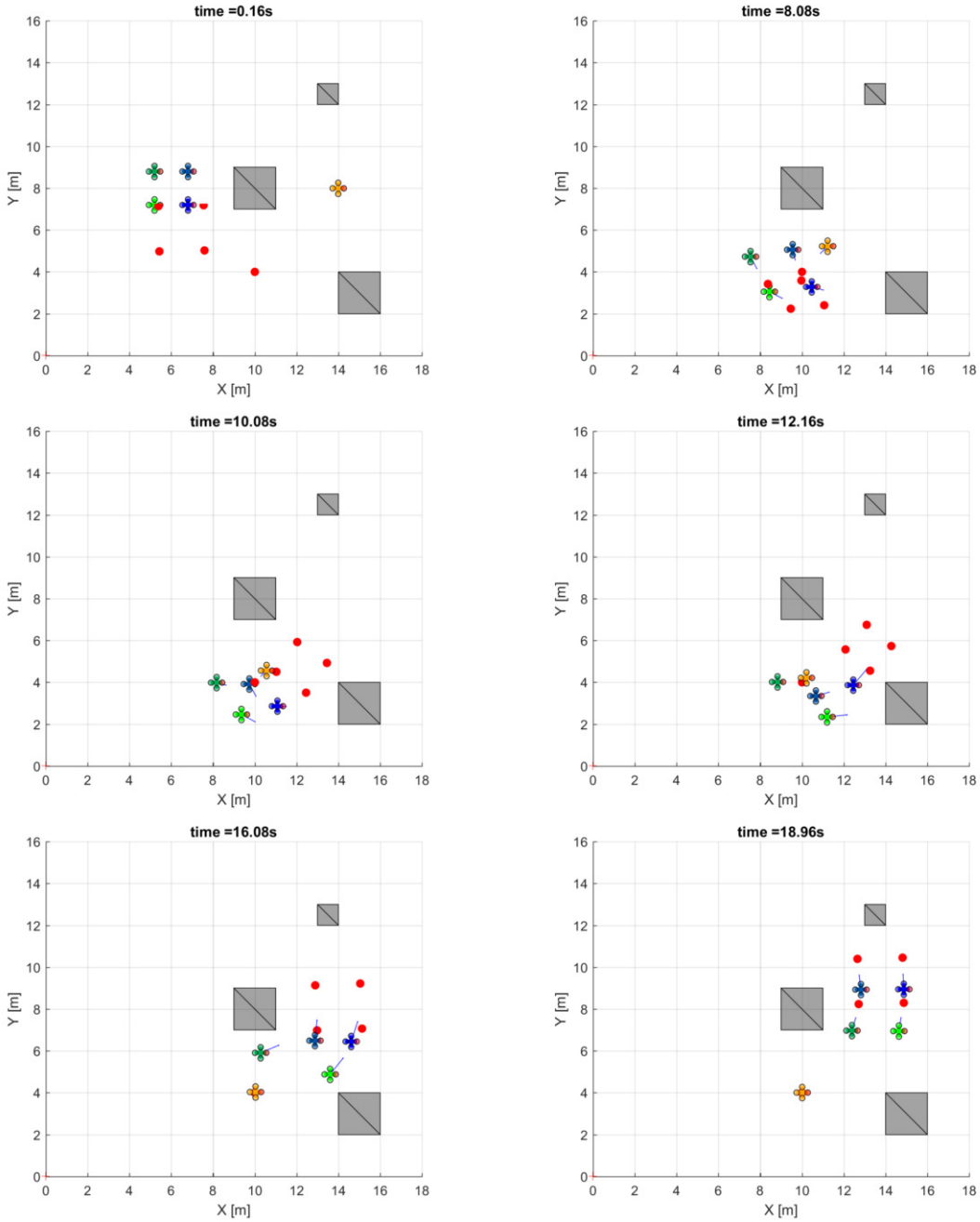


Figure C-4: Top view. From left to right and top to bottom. Snapshots of robots and target positions (red dots) $\boldsymbol{r}_i^*$ at 0 s, 7 s, 11 s, 13 s, 15 s, 17 s.

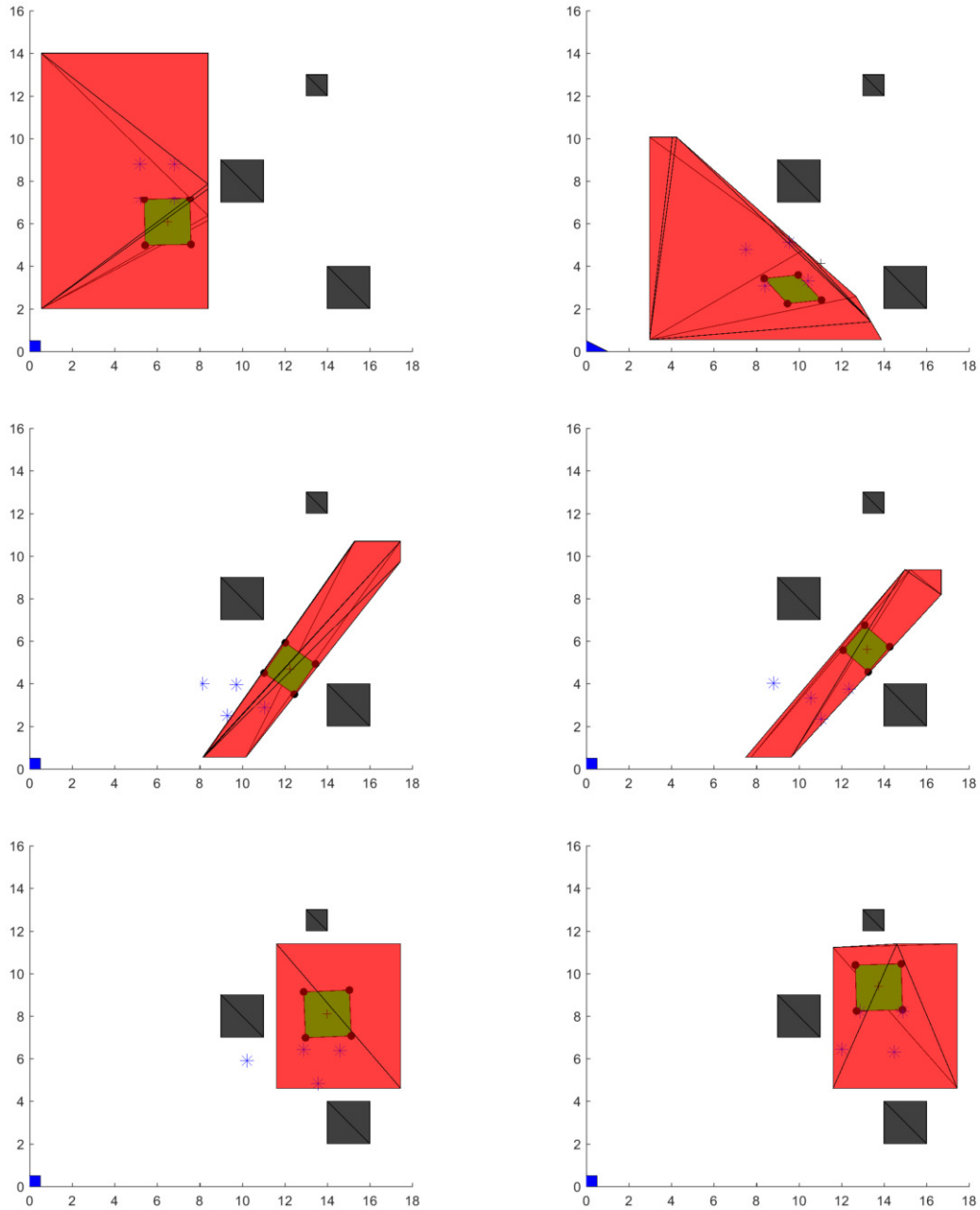


Figure C-5: Top view. From left to right and top to bottom. Convex regions $\mathcal{P}^j$ (red), target formation (black dots) and robot positions (blue stars) at 0 s, 7 s, 11 s, 13 s, 15 s, 17 s.

Appendix D

Instructions to Running the Simulations in Matlab

To run the Matlab code, the following prerequisites are necessary.

The file matlab/groupDistributed_sim.m in the folder mit-nhorca must run without an error. Furthermore, the file groupNavigation-sim/groupNavigation_sim.m in the folder mit-groupnavigation must run without and error. For both algorithms, instructions are available in the readme files.

Afterwards, the folders mit-nhorca and mit-groupnavigation need to be side by side. The file matlab/distr_splm_form.m should then run directly.

Bibliography

[1] J. Alonso-Mora, E. Montijano, T. Naegeli, O. Hilliges, M. Schwager, and D. Rus, "Distributed multi-robot formation control in dynamic environments [under review]."

[2] J. Alonso-Mora, E. Montijano, M. Schwager, and D. Rus, "Distributed multi-robot formation control among obstacles: A geometric and optimization approach with consensus," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5356–5363.

[3] T. Keviczky, F. Borrelli, K. Fregene, D. Godbole, and G. J. Balas, "Decentralized receding horizon control and coordination of autonomous vehicle formations," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 1, pp. 19–33, 2008.

[4] R. van Parys and G. Pipeleers, "Online distributed motion planning for multi-vehicle systems," in *2016 European Control Conference (ECC)*, pp. 1580–1585, IEEE, 2016.

[5] N. Michael, M. M. Zavlanos, V. Kumar, and G. J. Pappas, "Distributed multi-robot task assignment and formation control," in *IEEE International Conference on Robotics and Automation, 2008*, (Piscataway, NJ), pp. 128–133, IEEE, 2008.

[6] M. Bürger, G. Notarstefano, F. Bullo, and F. Allgöwer, "A distributed simplex algorithm for degenerate linear programs and multi-agent assignments," *Automatica*, vol. 48, no. 9, pp. 2298–2304, 2012.

[7] Z. Chen, T. Chu, and J. Zhang, "Swarm splitting and multiple targets seeking in multi-agent dynamic systems," in *2010 49th IEEE Conference on Decision and Control*, (Piscataway, NJ), pp. 4577–4582, IEEE, 2010.

[8] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006.

[9] K. Raghuwaiya, J. Vanualailai, and B. Sharma, "Formation splitting and merging," in *Advances in swarm intelligence* (Y. Tan, Y. Shi, and L. Li, eds.), vol. 9713 of *Lecture notes in computer science Theoretical computer science and general issues*, pp. 461–469, Cham and Heidelberg: Springer, 2016.

[10] P. Ogren, "Split and join of vehicle formations doing obstacle avoidance," in *Proceedings / 2004 IEEE International Conference on Robotics and Automation, April 26 - May 1, 2004, Hilton New Orleans Riverside, New Orleans, LA, USA*, (Piscataway, NJ), pp. 1951–1955 Vol.2, IEEE Operations Center, 2004.

[11] P. Ogren and N. E. Leonard, "A tractable convergent dynamic window approach to obstacle avoidance," in *Proceedings / 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Piscataway, NJ), pp. 595–600, IEEE Operations Center, 2002.

[12] T. Huang, M. Kapadia, N. I. Badler, and M. Kallmann, "Path planning for coherent and persistent groups," in *IEEE International Conference on Robotics and Automation (ICRA), 2014*, (Piscataway, NJ), pp. 1652–1659, IEEE, 2014.

[13] M. Kallmann, "Dynamic and robust local clearance triangulations," *ACM Transactions on Graphics*, vol. 33, no. 5, pp. 1–17, 2014.

[14] T. Keviczky, F. Borrelli, and G. J. Balas, "Decentralized receding horizon control for large scale dynamically decoupled systems," *Automatica*, vol. 42, no. 12, pp. 2105–2115, 2006.

[15] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 926–939, 1998.

[16] T. Balch and M. Hybinette, "Social potentials for scalable multi-robot formations," in *Proceedings / ICRA 2000, IEEE International Conference on Robotics and Automation*, (Piscataway, NJ), pp. 73–80, IEEE Service Center, 2000.

[17] L. Sabattini, C. Secchi, and C. Fantuzzi, "Arbitrarily shaped formations of mobile robots: Artificial potential fields and coordinate transformation," *Autonomous Robots*, vol. 30, no. 4, pp. 385–397, 2011.

[18] R. Olfati-Saber and R. M. Murray, "Distributed cooperative control of multiple vehicle formations using structural potential functions," *IFAC Proceedings Volumes*, vol. 35, no. 1, pp. 495–500, 2002.

[19] M. Egerstedt and X. Hu, "Formation constrained multi-agent control," in *2001 ICRA. IEEE International Conference on Robotics and Automation*, pp. 3961–3966, 21-26 May 2001.

[20] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors," *Autonomous Robots*, vol. 35, no. 4, pp. 287–300, 2013.

[21] D. Zhou and M. Schwager, "Virtual rigid bodies for coordinated agile maneuvering of teams of micro aerial vehicles," in *2015 IEEE International Conference on Robotics and Automation (ICRA 2015)*, (Piscataway, NJ), pp. 1737–1742, IEEE, 2015.

[22] H. G. de Marina, B. Jayawardhana, and M. Cao, "Distributed scaling control of rigid formations," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, (Piscataway, NJ), pp. 5140–5145, IEEE, 2016.

[23] Y. Chen, M. Cutler, and J. P. How, "Decoupled multiagent path planning via incremental sequential convex programming," in *2015 IEEE International Conference on Robotics and Automation (ICRA 2015)*, (Piscataway, NJ), pp. 5954–5961, IEEE, 2015.

[24] J. Alonso-Mora, S. Baker, and D. Rus, "Multi-robot navigation in formation via sequential convex programming," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4634–4641.

[25] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Algorithmic Foundations of Robotics XI* (H. L. Akin, N. M. Amato, V. Isler, and A. F. van der Stappen, eds.), vol. 107 of *Springer Tracts in Advanced Robotics*, pp. 109–124, Cham: Springer International Publishing, 2015.

[26] S. Boyd, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010.

[27] J. Bellingham, A. Richards, and J. P. How, "Receding horizon control of autonomous aerial vehicles," in *Proceedings of the 2002 American Control Conference, ACC*, (Piscataway, NJ), pp. 3741–3746 vol.5, IEEE Service Center, 2002.

[28] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 477–483.

[29] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[30] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research* (C. Pradalier, R. Siegwart, and G. Hirzinger, eds.), vol. 70 of *Springer Tracts in Advanced Robotics*, pp. 3–19, Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[31] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)*, (Piscataway, NJ), pp. 1917–1922, IEEE, 2012.

[32] A. Richards and J. How, "Decentralized model predictive control of cooperating uavs," in *2004 43rd IEEE Conference on Decision and Control*, (Piscataway, NJ), pp. 4286–4291 Vol.4, IEEE Operations Center, 2004.

[33] T. Naegeli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges, "Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization," *IEEE Robotics and Automation Letters*, p. 1, 2017.

[34] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[35] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *2000 ICRA. IEEE International Conference on Robotics and Automation*, pp. 995–1001, 24-28 April 2000.

[36] M. Nieuwenhuisen, M. Schadler, and S. Behnke, "Predictive potential field-based collision avoidance for multicopters," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XL-1/W2, pp. 293–298, 2013.

[37] G. Hoffmann, H. Huang, S. Waslander, and C. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, ([Reston, VA]), [American Institute of Aeronautics and Astronautics], 2007.

[38] M. Burri, J. Nikolic, C. Hurzeler, G. Caprari, and R. Siegwart, "Aerial service robots for visual inspection of thermal power plant boiler systems," in *2nd International Conference on Applied Robotics for the Power Industry (CARPI), 2012*, (Piscataway, NJ), pp. 70–75, IEEE, 2012.

[39] o. Falkenberg, J. Witt, U. Pilz, U. Welting, and H. Werner, "Model identification and h-infinity attitude control for quadrotor mav's," in *Intelligent Robotics and Applications*, pp. 460–471, 2012.

[40] P. Bouffard, A. Aswani, and C. Tomlin, "Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results," in *IEEE International Conference on Robotics and Automation (ICRA), 2012*, (Piscataway, NJ), pp. 279–284, IEEE, 2012.

[41] K. Alexis, C. Papachristos, G. Nikolakopoulos, and A. Tzes, "Model predictive quadrotor indoor position control," in *19th Mediterranean Conference on Control & Automation (MED), 2011*, (Piscataway, NJ), pp. 1247–1252, IEEE, 2011.

[42] T. Lee, M. Leoky, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se(3)," in *2010 49th IEEE Conference on Decision and Control (CDC)*, pp. 5420–5425.

[43] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2520–2525.

[44] D. Lee, H. Jin Kim, and S. Sastry, "Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter," *International Journal of Control, Automation and Systems*, vol. 7, no. 3, pp. 419–428, 2009.

[45] G. V. Raffo, M. G. Ortega, and F. R. Rubio, "Robust nonlinear control for path tracking of a quad-rotor helicopter," *Asian Journal of Control*, vol. 17, no. 1, pp. 142–156, 2015.

[46] S. Bouabdallah, "Design and control of quadrotors with application to autonomous flying."

[47] M. Erdmann and T. Lozano-Pérez, "On multiple moving objects," *Algorithmica*, vol. 2, no. 1-4, pp. 477–521, 1987.

[48] S. Bhattacharya, V. Kumar, and M. Likhachev, "Search-based path planning with homotopy class constraints," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence* (AAAI Press, ed.), pp. 1230–1237, 2010.

[49] M. Cyrus and J. Beck, "Generalized two- and three-dimensional clipping," *Computers & Graphics*, vol. 3, no. 1, pp. 23–28, 1978.

[50] J. Alonso-Mora, T. Naegeli, R. Siegwart, and P. Beardsley, "Collision avoidance for aerial vehicles in multi-agent scenarios," *Autonomous Robots*, vol. 39, no. 1, pp. 101–121, 2015.

[51] Hespanha, J., P., "An efficient matlab algorithm for graph partitioning."

[52] A. Buluç, H. Meyerhenke, I. Safro, P. Sanders, and C. Schulz, "Recent advances in graph partitioning," in *Algorithm Engineering*.

[53] R. Phillips and P. Kokotovic, "A singular perturbation approach to modeling and control of markov chains," *IEEE Transactions on Automatic Control*, vol. 26, no. 5, pp. 1087–1094, 1981.

[54] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM Journal on Optimization*, vol. 12, no. 4, pp. 979–1006, 2002.

[55] A. Domahidi and J. Jerez, "Forces pro: code generation for embedded optimization," 2016.

[56] "Robot operating system," http://www.ros.org/.

[57] "mocap_optitrack," https://github.com/ros-drivers/mocap_optitrack.

[58] "bebop_autonomy," https://github.com/AutonomyLab/bebop_autonomy.

Glossary

List of Acronyms

UAV	Unmanned Aerial Vehicle
MPC	Model Predictive Control
ADMM	Alternating Direction Method of Multipliers
MILP	Mixed Integer Linear Program
MIQP	Mixed Integer Quadratic Program
SCP	Sequential Convex Program
VO	Velocity Obstacle
PR	Probabilistic Roadmaps
RRT	Rapidly Exploring Random Trees
ROS	Robot Operating System
SDK	Software Development Kit
IMU	Internal Measurement Unit
LQ	Linear Quadratic
GDOFCR	Goal-Directed Obstacle-Free Convex Regions
DCSC	Delft Center for Systems and Control