UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

A Cooperative and Verifiable UAV Behavior for 3D Environments

Christoph Rasche

Dissertation in Computer Science

Submitted to the

Faculty of Computer Science,
Electrical Engineering and Mathematics,
University of Paderborn

in partial fulfillment of the requirements for the degree of

doctor rerum naturalium
(Dr. rer. nat.)

Paderborn, 2013

Supervisors:

Prof. Dr. Franz Josef Rammig,	University of Paderborn
Prof. Dr. Hannes Frey,	University of Koblenz · Landau
Date of public examination:	01.08.2013

Acknowledgements

First of all, I would like to thank my supervisor Prof. Dr. Franz Josef Rammig. His guidance and constructive feedback constantly helped me to improve this work. I would also like to thank Prof. Dr. Hannes Frey for vice-supervising this thesis.

Writing this thesis wouldn't have been possible without Dr. rer. nat. Bernd Kleinjohann, who offered me the opportunity to earn a doctorate in the first place and my group leader Dr. rer. nat. Lisa Kleinjohann, who always found the time to revise my work.

I am grateful to Dr. rer. nat. Willi Richert. His early advices reduced the time needed to create this work considerably. I also want to thank Claudius Stern for all the new ideas that popped up in our discussions and for the support during countless routine activities.

Further on, I thank my colleague Alexander Jungmann for taking on lots of daily work, as well as my colleague Jan Jatzkowski for revising this thesis and the guidance in expressing all the ideas in precise mathematical formulas.

Last but not least, I want to thank the bright students Benjamin Werdehausen, Jan Lutterbeck, Arne Jarek, Maarten Bieshaar and Richert Borkowski. They have always been there to lighten my workload such that I found time to write this thesis.

Zusammenfassung

Autonome Robotersteuerung ist eines der Hauptforschungsgebiete der Robotik. Eine Vielzahl an Ansätzen zur Kontrolle und Bewegung, sowohl einzelner als auch mehrerer Roboter, allein oder in Formation, existieren. Sie decken unterschiedliche Anwendungsgebiete ab und führen zu verschiedensten Ergebnissen.

Ein Ansatz zur vollständig autonomen Steuerung unbemannter Luftfahrzeuge (UAVs) wird in dieser Dissertation präsentiert. Die UAVs verfügen über die Fähigkeit zur dreidimensionalen Pfadplanung. Dadurch erreichen sie spezifische Positionen so dass Gebiete, einschließlich teilweise oder vollständig eingestürzter Gebäude, Tunnel, Brücken, etc. vollständig erkundbar werden.

Durch Verwendung multipler UAVs kann die benötigte Zeit zur Gebietserkundung reduziert werden, sofern Koordination und Aufgabenallokation verwendet wird. Der vorgestellte Ansatz funktioniert mit einem einzelnen UAV und sorgt für eine schnellere Abarbeitung aller Aufgaben bei Verwendung mehrerer UAVs. Diese arbeiten verteilt und vollständig dezentral. Redundante Aufklärung wird durch Informationsaustausch reduziert. Das System basiert auf der Potentialfeldtheorie, kombiniert mit harmonischen Funktionen, wobei Pfade durch ein Gradientenverfahren erzeugt werden.

Zusätzlich werden Formationsflüge zur Kooperation eingeführt. Diese basieren ebenfalls auf Potentialfeldern, in Kombination mit der Bifurkationstheorie. Die Art der Formationen kann durch einfache Parameteranpassungen verändert werden. Dieses Verfahren wird mit harmonischen Potentialfeldern zur vollständigen Gebietserkundung in Formation kombiniert.

Viele Methodiken zur Steuerung von UAVs berücksichtigen hindernisfreie Räume sowie zwei Dimensionen mit einer festen Höhe. Zusätzlich existieren einige Ansätze mit denen teilweise einzelne Hindernisse in Formation umflogen werden können. Der in dieser Dissertation vorgestellte Ansatz geht weit über derartiges Verhalten hinaus. Es wird ein einzigartiges System, das den UAVs das gleichzeitige Erzeugen mehrerer Formationen zur Erkundung komplexer und sich dynamisch ändernder dreidimensionaler Umgebungen erlaubt, vorgestellt. Der Ansatz ist vollständig dezentral und kann mathematisch verifiziert werden. Dadurch ist ein vorhersagbares Verhalten sichergestellt, das eine große Bandbreite an Einsatzmöglichkeiten eröffnet.

Abstract

Autonomous robot movement is one of the main research topics in robotics. Various approaches for control and movement of single and multiple robots, alone or in formation, from one positions to another exist. They cover diverse applications and lead to different results.

A system for completely autonomous unmanned aerial vehicle (UAV) control is presented in this thesis. It provides the UAVs with capabilities for three-dimensional path planning in order to fly to specific positions and to entirely explore environments including partially or completely collapsed buildings, tunnels, bridges, etc.

Using multiple UAVs decreases the time needed to explore an environment completely when the problems of coordination and task allocation are solved. The presented approach works well with a single UAV. Increasing the number of UAVs leads to a faster execution of all relevant tasks. The UAVs work in a distributed and completely decentralized manner without any central coordination instance. Redundant exploration is reduced by inter-UAV communication. The system is based on artificial potential field theory combined with harmonic functions. The simplicity of the gradient method is used to compute feasible paths based on a potential field. This leads to efficient trajectories with specific target positions, which are the essential behavior for exploratory navigation of complex environments.

Formations have been introduced to reach a high degree of cooperation. They are created through the use of artificial potential fields, based on bifurcation theory. The shape of a formation can easily be changed through the adaptation of single parameter values. A combination with the harmonic potential field is conducted to enable the UAVs to explore environments in formation.

Many UAV approaches consider an obstacle free space and merely consider two dimensions with a fixed height. Some techniques enable certain formations. They might also be able to avoid single obstacles while preserving a formation. The presented approach leaves such approaches behind and provides a unique system, wherein UAVs are able to create several flexible formations simultaneously in order to entirely explore complex and dynamically changing three-dimensional environments. The approach is completely decentralized and a mathematical verification ensures safe UAV behavior in order to address a wide range of applications.

Table of Contents

Chapter 1

Introduction

At bottom, robotics is about us. It is the discipline of emulating our lives, of wondering how we work.

ROD GRUPEN, Discover Magazine, June 2008

Many robotic approaches are ascribed to natural animal behaviors, e. g., from ants [11], [12], bees [13], fish, birds [14], and humans [15]. Using the knowledge of nature offers a wide range of potential benefits. Nature gained experience in creation over the course of millions of years, while countless species arose and became extinct. Species with particular abilities survived longer and evolved over time. This selection process led to highly efficient species.

Investigating evolution showed that only the fittest species survive. In this context the term fittest contains several capabilities. One of them is the proficiency to adapt to environmental changes. Another is the ability to establish groups of different individuals working together. Groups of animals can conduct tasks, single individuals are incapable of, enhancing their probability to survive. One example for successful incorporation are fire ants (Solenopsis invicta). They link their bodies to build waterproof rafts [16]. This technique allows a colony to survive floods, while single ants would drown.

Sahin et al. [17] observed three functional properties, essential for the survival of animal societies:

1. *Robustness*: The society is able to guarantee an operational status despite disturbances and the failure or malfunction of single members.

2. *Flexibility*: Each member is able to coordinate its behavior in a way that enables the society to process several tasks simultaneously.

3. *Scalability*: The society can operate under a wide range of members and also supports a large number of individuals, without influencing the performance considerably.

Robotic systems, composed of several robots, as well as animal colonies supporting these functional properties can fulfill tasks of higher complexity than their single individuals would be able to. The functional properties should always be taken into account for the design of complex and distributed robot systems with autonomously acting members.

The development of distributed robot control systems is not an entirely new topic. It reached increasing attention by researchers in the 1980s and thenceforth a countless number of different types of robots and robot groups have been developed. These systems consist of diverse kinds of robots and work in a wide range of applications. Not all developed systems are autonomous and distributed. Several are directly controlled by humans and many have a central coordination instance. There is often no need to support the three mentioned properties as, e. g., by industrial robots, working in a static environment to fulfill only one single basic task highly efficiently. Otherwise robots work cooperatively in dynamically changing environments, e. g., to find victims after disasters. Such robots can benefit by supporting the properties.

Nowadays, robots conquer more and more different environments. Diverse requirements, partly based on the kind of environment in which the robots are employed must be taken into account. Robots working in outer space need other skills than robots working in deep sea missions [18]. Based on the single requirements, different types of robots, like unmanned underwater vehicles (UUVs) [19], animal-like robots [20], or even robots able to play soccer [21] have been designed. They often work on their own to fulfill complex tasks. But it is also possible that the robots work together in swarms [22], where in the majority of cases each individual only has simple rules and a limited knowledge about its environment. Due to the differing capabilities of the robots it is not possible to design a single system, able to work efficiently with all kinds of robots.

A wide research area is swarm intelligence. In this area robot cooperation is achieved through simple rules only. In comparison to swarm intelligence approaches, the behavior of the presented unmanned aerial vehicle (UAV) system relies not only on simple rules. The robots compute complex behavior including plans. They are equipped with sensors to detect their surroundings and with suitable actors, able to react to environmental changes. Also they have enough computational power to compute complex behavior online, while every UAV has a model of the currently known world on which the behavior is based.

These days, several commercial UAVs achieve increasing interest in research and in real world applications. Figure 1.1 shows three of the currently available UAV platforms, developed by the companies Parrot [23], Draganflyer [24] and Microdrone [25]. The presented UAVs are used amongst others in education and research.

Related to the existing platforms, the UAVs considered for the presented system are helicopter-like. One advantage of such UAVs is their ability to take off and land at much more positions than fixed wing UAVs as they have no need for a runway.

A basic requirement for the design of a distributed robot system is autonomous robot movement. Helicopter-like UAVs are able to turn on the spot and therefore non-holonomic

Parrot AR - Drone 2 [26] *Draganflyer X8 [27]* *Microdrone MD4 1000 [28]*

Figure 1.1.: *A small selection of currently available commercial UAVs from the companies Parrot, Draganflyer and Microdrone.*

constraints can be neglected during the design of a first movement approach. Nevertheless, such constraints should be taken into account afterwards to compute smooth trajectories.

The use of cooperative robots, UAVs, as well as unmanned ground units supporting humans can lead to a significant easement in terms of processing several different tasks. In turn, these tasks can address a wide range of applications and might be performed by homogeneous, as well as heterogeneous robot groups. The following section gives a brief introduction to a selection of tasks realizable by groups or respectively by swarms of robots.

1.1. Motivation

Today's robots are used in a wide range of applications. One example are industrial robots that mostly execute simple tasks in a highly efficient manner. Another focus of robotics is on several different robots executing complex tasks. During recent years robots copying human and animal behavior received increasing interest, which in turn often work in a self-adaptive manner, where they adapt their behavior in order to properly react to changing environmental influences.

Beside the mentioned systems UAVs also received increasing interest during the last years. Currently they are primarily used in police and military applications, as well as in education and research. A future application might be the support of several kinds of different rescue operations.

A big disadvantage of current real world UAV control systems is their lack of autonomy. Often, at least one human operator is needed to control single UAVs. Figure 1.2 shows a modern control station for a Reaper MQ 9 UAV [29]. A pilot and a sensor operator are needed for the control of each single Reaper. The distance between the operator and the UAV, which can be thousands of kilometers, leads to delays. Combined with the limited sensor information the persons controlling the UAVs receive, this leads to a heightened probability of crashes.

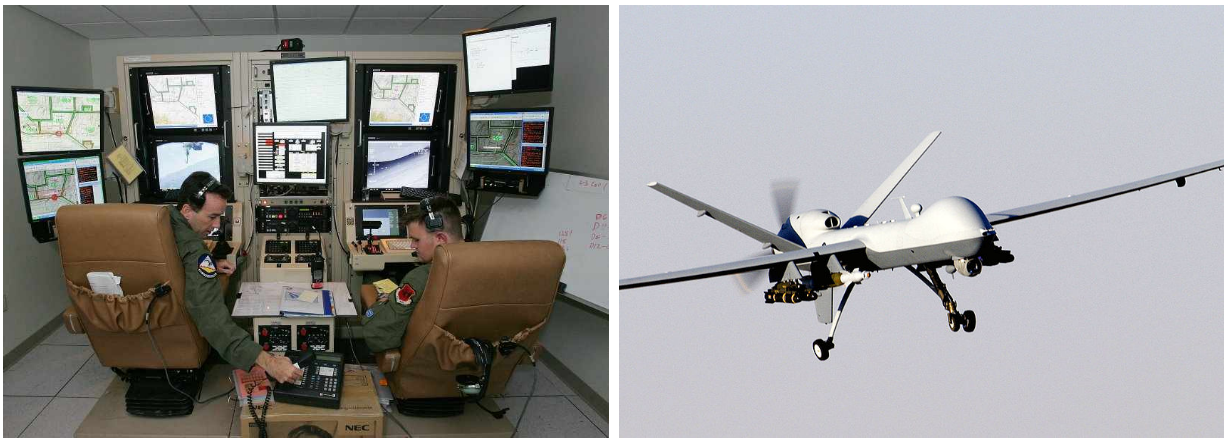


Figure 1.2.: *Ground control station [29] for a Reaper MQ 9 UAV [30] and the corresponding UAV.*

One way to overcome these disadvantages could be a system, able to autonomously control several UAVs at once. Controlling UAVs should then be performed decentralized to avoid single points of failure and bottlenecks. It should also be possible to adapt the UAV behavior autonomously, in order to properly react to changing environmental influences. For the use in real world applications, a verification of the resulting system must be possible.

The inefficient control of current UAV systems and the problems arising therefrom lead to this thesis' objectives. Thus, one of the main objectives is the design of a completely autonomous multi-UAV system. The UAVs should have capabilities to process different tasks like, e. g., environmental exploration. Sophisticated task execution should efficiently support ground units with information about the operational area.

A system, able to autonomously explore entire areas fast and efficiently has many applications. Popular examples are the provision of an overview after disasters or the support of search and rescue operations to find missing people, like children or older, perhaps bedridden persons. Those people are often last seen at a particular place at a certain time. Based on this information it is possible to determine an area in which these people are most likely to be found. Monitoring of crowds is also a potential application in order to provide the police with information, usable to relax tense situations before they get out of control, leading to panic and sometimes death.

Another interesting application is the exploration and measurement of unknown or only roughly known terrains. Such tasks have been conducted several times during recent years. A measurement of the Mariana Trench was performed in 2011 [31] and a measurement of the gravity field of the moon in unprecedented detail was executed in 2012 [32]. Upcoming tasks might be, e. g., exploration and measurement of Mars. It would be very beneficial if these tasks could be conducted using multiple space probes able to process the tasks self-coordinated and completely autonomously in the future. A system with these capabilities has to be verified at least in the sense that it always leads to a specified behavior. This is necessary, as on the one hand unspecified behavior during search missions can harm people.

On the other hand the time required by operators to react to unspecified behavior might be too long due to communication delays.

A cooperative robot platform can also be used as a mobile platform to measure gas concentration, wind, temperature and humidity [33]. It is possible to improve the air quality based on the data these robots measure at different locations inside a building. Another aspect is the analysis of gas clouds to achieve information about the composition and concentration of possible poisonous particles of a cloud. The achieved information can be used to warn residents if necessary and to supply appropriate pharmaceuticals.

In such scenarios a cooperative coordination of multiple robots can improve the results compared to the use of robots lacking coordination capabilities. Based on this idea an approach is presented where each single UAV is able to process most of the required tasks by itself, but is also capable to connect with the other robots to increase the efficiency of the overall system. The UAVs are able to reach consensus about several objectives such as the determination of specific UAVs for single tasks execution, when it is not beneficial to process tasks simultaneously using all UAVs. Additionally, the UAVs are able to autonomously determine their own part during task execution, while multiple UAVs are cooperatively executing tasks. One focus is efficient exploration of different environments and to increase efficiency by increasing the number of participating UAVs. To achieve the requirements, the UAVs must perform cooperative behavior such as formation flights if it leads to faster exploration of some of the environment's subareas. The complete objectives fulfilled by the presented system are described in the following section.

1.2. Objectives and Contribution

The overall objective of this thesis is the design of a multi-UAV system able to support ground units in a wide range of applications. Beside the aforementioned three functional properties robustness, efficiency and scalability, the system has to provide the following capabilities:

- An intelligent behavior for self-adaptation to environmental changes.
- Completely autonomous and distributed behavior.
- Efficient and complete terrain exploration and monitoring.
- Simultaneous multi-task creation, allocation, and execution.
- Dynamic establishing and dissolving formations.
- Verifiable behavior.

A dedicated UAV control software, providing several benefits has to be implemented to face these objectives. All algorithms and methods used to control the UAVs should be testable in a safe way. A simulation of the resulting UAV behavior is needed to avoid damage and an evaluation of the designed algorithms in several different environments is required, to ensure that the system works not only in a specific environment.

As mentioned before, a “good” system has to adapt itself to environmental changes. This has to be reached as the UAVs should be able to fly in unknown environments. Therefore, they must have capabilities for obstacle detection and proper reaction to newly detected obstacles. Hence, the possibility to represent environmental changes during UAV movement is required in order to test the self-adaptation capabilities of the UAVs.

Another objective is that different types of UAVs should be supported. Thus, the developed algorithms have to work in different environments and for different kinds of UAVs. Finally, the current status of the system, as well as parts of previous UAV behavior must be visible to human operators in order to identify wrong and inefficient behavior.

The UAVs have to work in a self-organizing manner due to the autonomy of the resulting system. They must also execute their behavior completely decentralized to reach the three functional properties robustness, flexibility and scalability. The single UAVs need to be able to entirely explore different terrains. Consensus reaching with other UAVs not only about the sections of the terrain each UAV has to explore but also about the distribution of given tasks is also necessary. All tasks have to be organized and processed by the UAVs themselves, without a central coordination instance.

In order to design a system, which is able to solve these objectives several requirements have to be fulfilled. First of all an internal environment representation is necessary. Safe movement inside various terrains and complete accessibility within each environment is required to guarantee complete exploration. The UAVs must be able to move to specified positions in the terrain in order to start with the most important tasks. Due to the use of multiple UAVs, a coordination is necessary to reduce redundant exploration and task processing resulting in an efficiency increase.

It has to be ensured that the UAVs work in a safe manner, meaning that they never collide with other UAVs or obstacles within the environment under any circumstances. A collision normally leads to a cascade. This can harm people standing or in case of casualties more likely lying beneath the UAV without a chance to move away from falling parts. To avoid such behavior and to guarantee that the expected behavior is executed, all algorithms used by the resulting system must lead to a mathematically verifiable UAV behavior.

The UAVs are able to take several different formation shapes in order to achieve a coordinated movement. These formations can be established any time and ought to be highly dynamic, considering the number of participants. A formation is dissolved when a further benefit is unlikely. The UAVs rely their decentralized behavior on the inputs of a global positioning system. Beside others, it is used to check whether the complete terrain is explored and to let the UAVs exchange relevant information regarding the environment.

The presented system is unique in its capabilities and combines Laplace's equation [34] and bifurcation theory [35] to reach coordinated UAV flights in a three-dimensional space, consisting of several complex obstacles in order to reach a common goal. Compared to many other robotic approaches it is possible to show the stability of the emergent behavior. The presented system is described in detail in the following chapters, while the next section simply gives an outline.

1.3. Outline

An introduction to autonomous systems from the beginnings up to state-of-the-art approaches is presented in the following chapter. Thereafter, the basic methods, used for the design of the presented system are introduced. An overview of the system and methods used to create and internally represent different environments is given in the further course of the thesis. The algorithms used for path planning and formation flights work with this representation. Exploratory navigation using formations leads to a higher degree of cooperation. An evaluation of the system is presented after the description of the designed system to show that the approach not only works in theory. It is investigated whether the approach is computable and works with discrete arithmetic. Finally, a conclusion and an outlook on future work is given.

Chapter 2 *(Evolution of Autonomous Systems)* introduces to the evolution of autonomous systems from the beginnings up to state-of-the-art robotic approaches. The first autonomous and decentralized acting systems were animals, such as bees, ants, fish and birds, created by nature. Observations inspired researchers to come up with different approaches for autonomous systems based on animal behavior. The chapter distinguishes single robot control and cooperative robot control. Popular techniques designed by various researchers are presented for both, single robot control and cooperative robot control. A few famous robot control areas are introduced as well. The system presented in this thesis is finally contrasted to state-of-the-art approaches.

Chapter 3 *(Cooperative Path Planning Foundations)* describes the basic approaches, used to design the presented system. Foundations for path planning, cell decomposition and potential field theory for single UAV flights and flights in formation are covered. Problem statements for the single objectives the system has to obtain are also given. Afterwards, the mathematical basics, used to verify the resulting system are presented.

Chapter 4 *(Architectural System Design)* mainly treats the basics necessary to efficiently compute feasible paths. First, an overview of the entire architecture is presented. The main focus lies on the world representation, based on an octree. This tree-based structure has to be highly dynamic and efficient as described in the chapter. Thereafter, the UAV configuration is introduced and camera based environmental exploration is described in detail.

Chapter 5 *(Decentralized UAV Flights)* addresses the artificial potential field approach for UAV flights of the presented system. A mathematical problem statement is provided at the beginning of the chapter, followed by state-of-the-art solutions. Afterwards, a potential field, represented by a harmonic function is described in detail. The chapter also shows some methods to reduce the computational effort, followed by a description of path computation and the technique used for exploratory navigation. The underlying mathematics of the presented approach are also verified.

Chapter 6 *(Behavioral Formation Flights)* discusses the methodology used for formation flights. A behavioral approach based on artificial potential fields is designed. The chapter first gives a problem statement and presents some state-of-the-art solutions. A combination of the potential field, presented in Chapter 5 with the formation approach is also described. A visualization of the resulting potential fields is given afterwards to provide a better impression of the designed path planning system. A selection of possible formations shapes is then presented. Formation handling in terms of requirements, formation flights, and formation rotation in three dimensions is also addressed. The underlying mathematics is verified at the end of the chapter.

Chapter 7 *(Cooperative Behavior)* describes the inter-UAV behavior. It is based on task creation and task allocation. The UAVs are able to change their tasks anytime and thus establish, dissolve, and change formations. Task allocation is based on a decentralized market place approach and the UAVs use a role system based on division of labor. Single tasks and their properties are also introduced, as well as situations in which the UAVs create tasks by themselves. The entire system works completely decentralized leading to a more complex handling.

Chapter 8 *(Evaluation)* shows the results of several test runs, conducted to evaluate the system. Diverse conditions are introduced resulting in known and unknown environments, as well as in UAVs able to fly in formation and UAVs unable to. A terrain, based on real world data, extended by complex three-dimensional structures served as the work environment for the UAVs. Performance issues, concerning the computational time of the entire system presented in this thesis are considered. In the further course of the chapter, outcomes for exploratory navigation are compared using single UAVs and formation flights. Thereafter, test runs regarding the UAV behavior and the task allocation behavior are presented. A conclusion regarding the results is also given.

Chapter 9 *(Summary and Outlook)* summarizes the presented approach and highlights the system's main advantages. The contributions of this thesis are also presented. Finally, suggestions for future improvements and additionally required approaches for the use in real world scenarios are provided.

Chapter 2

Evolution of Autonomous Systems

Several decades ago, researchers started to think about techniques for autonomous robot movements from one position to another. Systems for cooperative movement of various different robots have been designed. Lots of the resulting path planning approaches are biologically inspired as the evolution of animals provides good insights into possible design paradigms. Such methodologies reached increasing interest with the introduction of cooperative robot control based on behavior observed from animal swarms. The aim of this chapter is to provide an introduction to path planning for single and multiple agents, as well as to coordination methodologies of groups of autonomous agents.

The first decentralized acting swarms of autonomous agents were designed by nature millions of years ago. Nowadays, the objectives of the UAV system explained in Section 1.2 are solved by nature on a daily basis. Over the course of millions of years, evolution has brought forth, among many failures leading to species extinction and coining the term *survival of the fittest*, species with highly optimized task processing capabilities. This was recognized in the early 1990s by several researchers, e. g., [36], [37], [38]. Thenceforth, behavior methodologies observed from animal colonies were increasingly introduced to design single-robot systems and multi-robot systems. A research area called swarm intelligence arose. The developed robot swarms consist of single robots with basic rules for behavior computation. Due to their cooperation, they are able to solve tasks, single members are not capable of solving. This led, beside others, to great improvements in the design of autonomous multi-agent systems.

The considered UAVs do not necessarily form a robot swarm if the classical definition of robot swarms, in which a swarm consists of a large number of relatively simple physically embodied agents, is taken into account. The presented system works with a single UAV, as well as with a large number of UAVs. The maximum useful number of UAVs is mainly based on the capabilities of the UAVs and the size and properties of the environment. Too many UAVs will start to disturb each other as, e. g., shown in Chapter 8. The UAVs are able to solve complex tasks by themselves due to their computational power. However, swarm approaches are used in the resulting system to increase efficiency. Some behaviors, observed from ants, bees, fish and birds, which are interesting for the resulting UAV system are presented in the following section.

2.1. Autonomous Systems in Nature

Many animals work in a cooperative manner to process daily tasks, like foraging. These cooperations can consist of a couple of animals, as well as of millions of individuals. Examples are ant colonies, bee swarms, schools of fish and flocks of birds. Foraging is the main task animal swarms have to solve. In order to fulfill this task they have to explore the surroundings of their nest to find food sources.

The individuals of big societies specialize themselves to provide the colony with the ability for simultaneous task execution. Task allocation and the coordination of a large number of individuals is necessary for such animals to fulfill complex tasks in order to survive. They work together to build complex structures, like bee hives or termite nests or to transport big prey in groups as ants do. The single members have thereby only a limited knowledge and perception of their environment and of the overall task they process. Many of them communicate in an indirect manner.

Nevertheless, their behavior seems to be robust even with uncertainty in dynamically changing environments. The results achieved by cooperation are normally well beyond the capabilities a single individual can reach. It seems that the key mechanism is self-organization of the individuals [39]. This elated researchers to study animal behaviors in order to get new ideas for the development of robust and adaptive algorithms, able to handle robot societies. Simultaneously biologists regard robotics as a useful test bed to check and validate their theories [40].

Big improvements have been made in robotic research by adopting and adapting behaviors of ants, bees, birds and fish. Another feature adopted from animal swarms is the division of labor, used to specialize individual robots. Furthermore, animal behaviors have been utilized to design evolutionary algorithms that solve different optimization problems. They have also been used to design algorithms for traffic routing, networking, games, industry, robotics, economics and the general design of artificial, self-organized, distributed problem solving devices. Karaboga and Akay [13] presented a detailed overview of approaches, based on observations of bee behavior. An interesting point of ant behavior is their use of pheromones, as presented in the following section.

2.1.1. Ants

One of the most famous biologically inspired approaches in robotics is ant colony optimization, introduced in the early 1990s by Dorigo et al. [11], [12]. The approach is inspired by the foraging behavior of ants. The worker ants of a colony explore the area surrounding their nest in a random fashion. If an ant finds a food source, it evaluates the quality and quantity of that source and carries some food back to the nest. Thereby, the ant deposits a chemical pheromone trail on the ground during movement. The quality and quantity of the food source leads to a certain pheromone concentration. This pheromone trail leads other ants of the colony to the food source. If the source contains food too big to be transported by a single ant, they show their capability to solve coordination tasks. A group of ants collectively moves the food to the nest.

It has been shown [36] that this kind of indirect ant communication allows the ants to set up a shortest path between their nest and the food source. Another interesting observation is that after some time major paths are established, which enables the ants to lead homecoming ants directly to the nest. Based on such a behavior it should be possible to design a system able to work without the need of a global positioning system.

Based on ant foraging behavior, several algorithms and methods for different problems have been designed [41]. The foraging behavior shows similarities to the presented UAV behavior (cf. Chapter 5) designed amongst others to entirely explore various environments. One difference is that the ants stop exploration when they find a food source in order to bring food to their nest, while the UAVs continue to explore the terrain after they targeted some of their objectives. Further behavior, utilizable by UAVs for exploratory navigation has been observed from bees and is presented in the following section.

2.1.2. Bees

The communication behavior of bees is an interesting aspect. When a bee finds a food source it moves back to its hive and start dancing. Dependent on the distance to the food source, three different dances are performed: round dance, waggle dance or tremble dance. A round dance does not contain directional information and is conducted if the food source is less than 100 m away. The bee performs a waggle dance if the source is further away. The length of the dance is related to the distance to the source, whereby longer distances cause shorter dances. Additionally, the dance provides directional information related to the sun. If a bee perceives a long delay while unloading the food, it performs a tremble dance.

Bees also dance in order to perform non-conflictive, unified, decision making within the swarm. If the swarm needs a new nest, multiple scouts explore the surroundings in order to find suitable places. These scouts meet and exchange information regarding the quality of places they found by dancing. Based on the information provided by the dances, they choose the location for their new hive. Such a simple communication structure can also be used in robotics for information exchange without much overhead.

Another interesting observation is that bees use a map like organization of spatial memory for homing, foraging and other flights based on viewpoints and landmarks. Such an organization can also be used to entirely explore areas, without the need of a global positioning system.

Adopting this method would make, e. g., an exploration task more robust (cf. Section 2.2.3). Problem solvers based on bee swarm intelligence have been introduced from the beginnings of the 2000s [13].

Bees also cooperate to build new hives and to defend these. Further interesting cooperations are formation flights. They have been observed from birds and fish, as described in the next section.

2.1.3. Flocks of Birds and Schools of Fish

Behaviors observed from flocks of birds and schools of fish are shown in Figure 2.1. They can be used to provide a group of agents with abilities for movements in desired formations and to preserve them while moving. The animals use simple movement rules, based on the distances to their neighbors [14]. Formation flights have several applications, like the exploration of hazardous areas and the transportation of large objects.

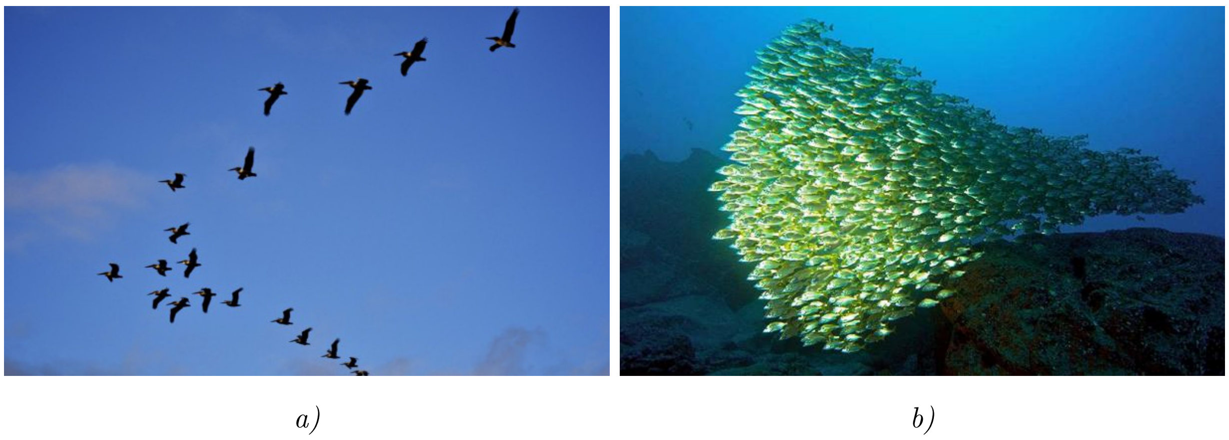


Figure 2.1.: *Image a) shows a flock of birds [42] and b) shows a school of fish [43].*

One famous example, based on the movement behavior of schools of fish and flocks of birds for robotic systems is the work conducted by Kennedy and Eberhart [44] in 1995. They have defined the particle swarm optimization (PSO) approach.

Flocks of birds and schools of fish inspired researchers to design methods for the movement of several robots without high computational effort. Such methods are, e. g., used by researchers in the area of dynamic robot control in distributed systems. A single fish or a single bird in a swarm moves independent to the goal the entire swarm wants to reach. It just positions itself at specified distances to its neighbors. The distances depend on the current environmental situation. If a school of fish is surrounded by predators, the distance decreases. This simple positioning also has the advantage of fault tolerance. If several agents fail, e. g., they were eaten by a shark, the entire school of fish is still able to move forward without leaving fish behind.

One application for flocks of birds is to cover large distances. Therefore, the birds form a flock with a designated leader, enabling them to save energy by minimizing drag for most of the participants. The leader, who still has to overcome the complete drag, changes from time to time. Due to these changes, each bird saves energy and is able to cover the distance.

In robotics, a formation behavior independent of a leader and not treating the formation as a single object is called behavioral formation approach. This is one of the three main approaches used to move a group of robots, as described in Section 3.4. The UAVs implement such a behavioral approach to fly in formation (cf. Chapter 6).

The methods presented so far treat animal swarms as a cluster of homogeneous agents. In order to increase efficiency, several animal swarms use the so called division of labor approach to create specialized individuals to simultaneously solve different tasks, as described in the next section.

2.1.4. Division of Labor

The use of biologically inspired self-organization and division of labor has reached increasing interest through the work of Benabeau et al. [45]. Division of labor describes the ability of colonies to execute several tasks simultaneously with the help of specialized agents. The agent's flexibility of both, responding to external and internal changes is a noticeable aspect of this approach.

Each large animal colony uses division of labor. It is primarily applied to create several types of workers and results in the ability to process various tasks simultaneously. The kind of division of labor can, e. g., change with the size of the colony, as well as with environmental changes.

Beside the division of labor considering only worker animals another division takes place in nature. Ants and termites have big soldiers and small workers. A further example is a honey bee swarm [13]. Such a swarm mainly consists of three kinds of bees, with one kind subdivided into two varying kinds:

1. Queen bees are responsible for reproduction.
2. Drones are male bees with only one task, to fertilize a new queen.
3. Workers fulfill diverse tasks, based on their age and the needs of the colony.
    a) Hive workers are young bees, responsible for solving all tasks inside the hive.
    b) Foragers are older workers, responsible for food supply.

A bee swarm formed on the basis of these different kinds of single members allocates all tasks dynamically and adapts itself in response to environmental changes. Young worker bees, as well as young worker ants mainly live inside the colony or hive and are responsible for all tasks, like feeding the offspring, until they reach a certain age. Then they change their behavior and start to move outside of the hive to execute foraging tasks. However, this task allocation is not fixed. If more hive workers or more foragers are needed a recruitment from the other camp of workers takes place.

It can also be beneficial to distinguish diverse types of robots. In case of robot groups a role assignment may occur and certain robots solve specified tasks, other do not. This is similar to the division of labor example, in which only workers are considered. Additionally, evermore systems consisting of heterogeneous robot societies are designed, wherein robots have varying capabilities making them more or less suitable to solve various tasks like, e. g., in ant colonies where big ants defend the nest and smaller ants are responsible for foraging. Based on this animal behavior a cooperative UAV behavior has been designed (cf. Chapter 7).

Several approaches for autonomous robot control are based on animal swarm behavior, briefly introduced in this section. Beside adopted swarm behaviors many other approaches for robot control exist. The following section gives a brief overview of methods used to control single and multiple robots.

2.2. Autonomous Robotic Systems

Some of the most famous approaches designed during the last decades and used for robot navigation are briefly presented in this section. These approaches can be combined with coordination patterns to move multiple robots in an efficient way. Additionally, two research fields, responsible for the design of several approaches are also presented. These fields are called wilderness search and rescue (WISAR) respectively simultaneous localization and mapping (SLAM). As the field of autonomous robot control is really broad—from the movement of single parts of robots from one position to another to the movement of complex robots to the movement of robot swarms to execute complex tasks cooperatively—this section is far from being complete.

The control of multiple robots is often based on control strategies for single robots, extended by multi-robot control methodologies. Thus, the next section starts by introducing single robot control.

2.2.1. Single Robot Control

One of the earliest approaches for the design of autonomous robots and still in use [1], [46] is reinforcement learning, introduced by Sutton et al. in 1981 [47]. Later on Sutton and Barto gave an overview of the field in [48]. This research area addresses a wide range of robotics applications, considering amongst others robots that learn to move and to reconfigure themselves if necessary, e. g., because they have lost a leg [49]. Reinforcement learning is also used to design robots able to ride a bike [50] or to provide computers with the capability to play backgammon [51], [52] on the level of human world masters. Additionally, it enables computers to solve complicated tasks of high dimensionality like elevator dispatching [53], [54] and gives them the ability to challenge human experts in games like jeopardy [55] or chess [56], [57].

In 1987 Reynolds [58] was one of the first, inspired by flocks of birds to come up with a simulation for autonomous robot behavior. He used a heuristic to reach rule-based movement of single simulated bird-like "bird-oid" objects called boids in order to produced qualitative results with only three simple steering behaviors—separation, alignment and cohesion. His work was extended by Heppner and Grenander in 1990 [37]. In 2003 and in 2004 it was applied to control autonomous UAVs in formation by Crowther [59], [60]. These researchers were able to demonstrate the possibility to design systems, in which swarm behavior is reached through the use of simple rules.

In 1986 a paradigm shift in the design of robotic systems took place due to the introduction of the subsumption architecture by Brooks [61]. He created a system, in which robot movement is based only on sensor inputs and communication. His completely reactive approach is able to work without the need of an internal world model. Such a world model was necessary while traditional artificial intelligence methodologies were used. The subsumption architecture is related to finite state automaton, modeling robots as finite state machines, using Markov dynamics to describe the swarm behavior. A layered set of behaviors is used in subsumption, where top level behaviors subsume low level behaviors. These approaches led to qualitative behaviors, like flocks or dispersion.

At the same time Brooks introduced the subsumption architecture another approach for autonomous robot control was designed by Khatlib [62]. He presented the artificial potential field (APF) method and used it to design an obstacle avoidance system for manipulators and mobile robots. The basic idea is that the robots can be guided by forces pulling them to the target position and simultaneously repelling them from obstacles. APF combines several behaviors leading to a superimposed behavior represented by a resulting potential field. This theory is extensively studied for autonomous path planning, e. g., in [63], [64], [65] and [66] for single robots and, e. g., in [67], [68], [69], and [70] for multi-robot systems.

Many approaches for autonomous robot control lack the possibility to be mathematically verifiable. One advantage of the APF approach is that it is able to generate low order dynamic systems. Additionally, it can often be used to mathematically prove the stability of the emergent behaviors, instead of applying traditional algorithm validation.

In 1989 another popular approach was presented by Arkins [71], called Motor Schema. It is similar to the APF and produces appropriate repulsive and steering commands for a robot. It works completely reactively based on simple rules or schemes like move to target or avoid obstacle. These schemes are independent from each other and can be processed concurrently. Nowadays, this method is, e. g., used by the robot head Mexi (Machine with extended emotional intelligence) to move its head [72].

One widely used approach for the control of UAV groups has been adopted from industry. Receding horizon control (RHC) or model predictive control is a possibility to move a group of robots in a given formation. It works with linear and nonlinear control dynamics and is mainly an optimization approach calculating responses on given inputs. Duan and Liu [73], e. g., use this method in combination with PSO to investigate the formation flight problem in complicated environments. Furthermore, Zhou et. al. [74] realized an obstacle avoidance system using this approach.

Beside reactive movement of robots as it is designed, e. g., using Motor Schema planning approaches have also been designed. If the environment in which the robot moves consists of streets, the road map method [75], [76] can be used to plan paths the robot has to follow. This approach is also used by car navigation systems. The basic idea is to design a road map reflecting the connectivity of streets. After the creation of such a road map, path planning can be performed by connecting the start and the target position of the robot with the road map. A path can then be found by simply following the path in the road map.

Several additional approaches designed in other research areas like mixed integer linear programming [77] combined with RHC [78], quantum particle swarm optimization [79], genetic algorithms [80], and many more have also been adopted for the use of robot control. Graph based approaches using, e. g., Dijkstra's algorithm [81] or the A^* algorithm [82] to find optimal paths are also widely used in path planning applications.

The system presented in this thesis needs a verification of the resulting behavior in order to be adaptable for real world applications. The UAVs have to guarantee several behaviors, like obstacle avoidance and target reaching. Thus, the artificial potential field approach has been selected as basis of the resulting system.

The presented methodologies were mainly developed to control and to move single robots. To design a multi-robot system, they have to be extended by some sort of cooperative behavior. The following section gives a brief overview of some famous coordination strategies used to control multiple robots at once.

2.2.2. Cooperative Behavior

An organizational structure should lead to cooperative multi-robot behavior. Countless approaches able to coordinate several UAVs have been designed during the last decades. Several summaries, e. g., from Horling and Lesser [83] exist.

A multi-robot system should be organized in some way, e. g., by a collection of roles, relationships and authority structures. One of the earliest designs is hierarchical organization presented by Fox [84] in 1979. This approach arranges the robots in a tree like structure.

A robot's knowledge of the global view depends on its position in the tree. The deeper the position of a robot in the tree, the lesser knowledge the robot has. The single robots are connected by the tree structure and interact only with robots they are connected to.

Another kind of organization are holarchies. A holarchy consists of partially autonomous holons, from the Greek words holos (whole) and on (part). The term was introduced by Koestler [85] in 1967 for a unified descriptive theory of physical systems. It is based on the nested, self similar organization that such systems possess. For robotics this approach was used, e. g., by Fischer [86] to create an organization through explicit modeling or implicit division of labor in real world systems as holons. In a holarchy robots are separated in groups, whereby each group is responsible for processing specific tasks, similar to ants where the group of young ants works inside the hive and the group of older ants performs foraging.

An approach popular in game theory is coalitions. Given a set A of agents or robots each subset of A is a candidate for a coalition. In general, a coalition is goal oriented and has a short life. The robots create coalitions dynamically to process a task and dissolve it afterwards. Each robot in a coalition tries to maximize its own benefit. It can be extended to longer living agreements [87] and is based on trust. Responding to dynamic task environments multiple coalitions [88] can be established simultaneously.

A similar sort of organization, called teams was introduced in 1981 by Fox [89]. A team consists of a number of cooperative agents. These agents have an agreement to work together to reach a common goal. Teams can also be established dynamically, but compared to coalitions, maximizing the utility of the entire team rather than the one of the individual members is performed.

Long lived cooperations are called congregations. They consist of a flat organization structure. A congregation is formed amongst robots with similar or complementary characteristics. This motivates them to congregate for solving multiple tasks [90] without the use of groups or global rewards.

Compared to the static approaches presented so far, a society is a dynamic organization with a changing number of participants. It is a long living social construct and acts as an environment through which the participants meet and interact. One example for a society is a permanently operating environment called "agent world" [91]. Robots in a society can be heterogeneous and do not necessarily process the same task. The society provides a common domain.

Based on a governmental system federations with different varieties have been designed. A central government exists in this organization. It makes decisions regarding various aspects and thus reduces the autonomy of single robots. The robots are also subdivided into roles like delegates, facilitators, mediators, and brokers [92]. Based on those roles different tasks are performed by the robots.

Market places are producer consumer based organizational structures. A single market place consists of buying robots, selling robots, and third parties called auctioneers. Tasks can be acquired at a market place structure. Buying robots place bids for tasks they want to process and an auctioneer determines the winning robot based on the bids. This modeling facilitates real world market economies [93]. Due to this similarity, several research results from human economics have influenced this approach [94], [95].

One decentralized approach is the matrix organization. This paradigm allows several managers or peers to manage the single robots. It is similar to the management of human beings. The humans or robots receive hints and tasks from their managers, co-workers, colleagues, etc. An interrelationship can have its own objectives, importance, and characteristics [96]. The term matrix was achieved by a grid-based view of the participants. One design method can be a matrix where the managers are the rows and the robots are the columns, whereby these sets may overlap. A check is then used to decide whether an authority relationship exists.

Additionally, approaches combining several organizations exist. So called compound organizations can be used to combine several differing organizational methodologies. They can overlap, allowing the combination of various organization paradigms. Thus, robots in a group can be organized in a potentially other way within a larger context. The robots can take different roles in response to diverse organizational demands [83].

The methods presented in this section are mainly centralized approaches and most of them are based on division of labor (cf. Section 2.1.4). Robots assigned to differing roles process diverse tasks monitored by a coordination instance. The UAVs should be able to compute intelligent and efficient behavior without a central control instance. The market place approach is used as basis. It has been decentralized to create a system where the UAVs offer bids. Not only a single instance decides about the allocation of a UAV to a task but all other UAVs, as described later in Chapter 7.

Robots, organized using one of the mentioned structures can work in various applications. Countless applications and research areas for autonomous multi-robot control exist. Beside others, two control areas called WISAR and SLAM have become famous during the last decades and are beside a popular robot contest called RoboCup introduced in the following section.

2.2.3. Robot Control Areas

One famous area of robot control is simultaneous localization and mapping (SLAM) [97] wherein robots explore unknown areas and simultaneously build maps, using their sensors. Most state-of-the-art applications work in two-dimensional environments and some of them have additional capabilities in order to find casualties. A few applications create three-dimensional maps using laser range finders, e. g., mounted in a way that the laser range finder is able to tilt [98]. Due to the arrival of Microsoft's Kinect camera [99], researchers got a new and cheap possibility to execute these tasks in three dimensions, based on vision algorithms.

Other popular robot control areas are wilderness search and rescue (WISAR) [100], [101] and urban search and rescue (USAR) [102], [103]. USAR often takes place after natural and man-made disasters in order to find several victims in a given area. WISAR is conducted most times to locate one single victim inside a large search region. The use of multiple robots can lead to an increase in efficiency for archiving the goals of WISAR and USAR if they work cooperatively. Additionally, the robots must be able to detect victims during exploratory navigation.

To test, promote and compare several approaches for intelligent and efficient movement and robot behavior the RoboCup [104] organization was founded in 1997. Every year several contests take place in which robots have to play soccer in different leagues, find victims in disaster areas [103], or support people accomplishing their domestic applications [105]. These contests include robot simulations, as well as real world robots. This is one of several contests used to test and present state-of-the-art approaches for cooperative robot design.

The UAVs of the presented system should be able to solve WISAR and USAR tasks. An introduction of SLAM methods can improve the robustness of the presented approach in real world applications. Self-localization can be combined with a global positioning system to improve the estimated positions of the UAVs. The better this estimation is the better are the results achieved by the planning approach. The methods of the designed system can also be used to improve robots for challenges like the RoboCup.

Beside the famous areas of use and the methodologies for control and movement of single and multiple robots designed from the 1980s up to now, several researches still work in this field. The following section gives a brief overview of state-of-the-art robots and control strategies. Diverse requirements must be met to actually design a cooperative robotic application. Real robots should exist and path planning and cooperative behavior must be possible. A formation approach is required to actually be able to move in formation. Normally, researchers take only parts of the mentioned requirements into account. Thus, the following section is subdivided to handle the single research areas explicitly.

2.3. State-of-the-Art

Robot movement depends not only on path planning. Today's robots have various drives like wheels, chains, legs, arms, heads, etc. They are also designed to move in different environments, like in deep sea, in deep space, in subterranean environments, in overground environments, etc. If such autonomous systems move at high speed, environmental influences, like drag, must be taken into account. Thus, the systems are often designed as self-optimizing system like, e. g., [2] and [106].

Single researchers often focus on only one of the topics relevant for the UAV system presented in this thesis. Most of them consider complex path planning or formation control with simple formation movements. The design of an efficient behavior is another research area addressed by the UAV system. Hence, this section is subdivided into several subsections considering the current research of the topics separately. It is not only restricted to topics relevant for the resulting system to provide a bigger picture of today's applications. The section starts with an overview of current robotic systems and shows an overview of state-of-the-art path planning and formation handling approaches thereafter. It concludes with state-of-the-art robot coordination approaches.

2.3.1. Robots

One important part of robot movement, beside path planning, is path following. If the target platform is restricted, e. g., due to holonomic constraints, or to non-holonomic constraints [107], a path planner has to consider these restrictions in order to compute feasible paths. The restrictions can occur in several different forms dependent on the robots.

Using robots in heterogeneous environments leads to an increase of needed properties. Some robots can move relatively slow but must conduct their movement precisely. Other robots, like the robot *Cheetah*, shown in Figure 2.2 a), use new approaches for leg design [20] to move at a maximum speed of 28.96 km/h. To avoid damage, it must, e. g., be guaranteed that no abrupt direction changes take place when the robot moves at high speed. Cheetah is a prototype and primarily intended to be a model for the design of new robots with abilities to move in open field and to transport equipment.

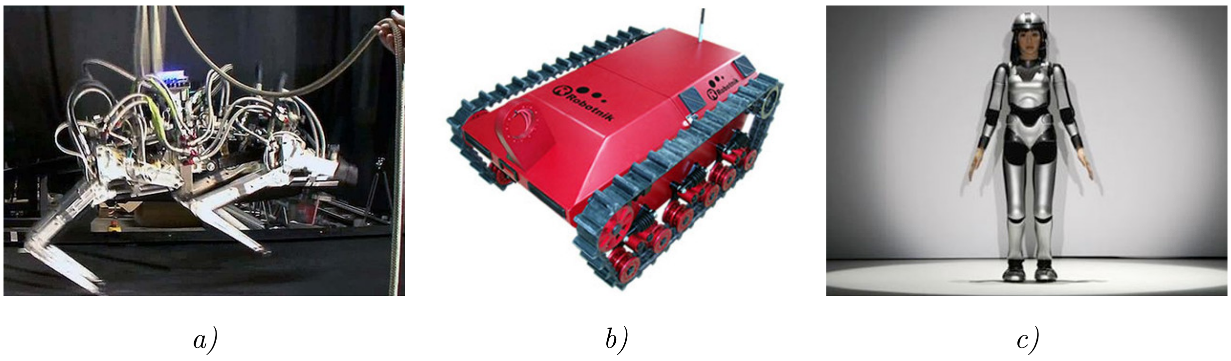
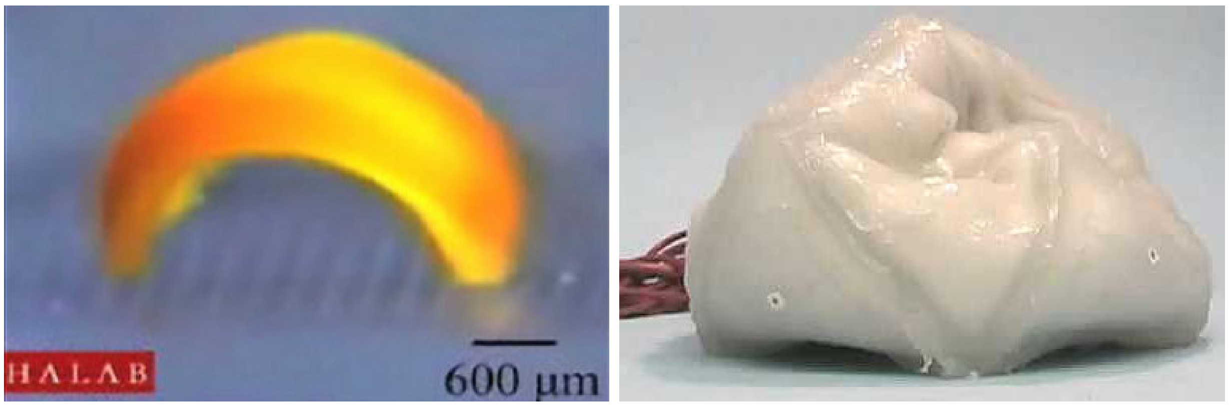


Figure 2.2.: *Figure a) shows the currently fastest robot* Cheetah *[108], b) shows a chain based rescue robot, designed by the company Robotnik [109] and c) shows the humanoid robot HRP-4C [110], developed by Yamaha.*

The robot presented in Figure 2.2 b) is a chain-based robot designed by the company Robotnik [109], with the intention to create a rescue robot. The focus of this robot is to overcome rough terrains, such as stairs to reach a high mobility. It is also used to support SLAM applications. Wheel based robots can also work in rough terrain to deploy goods.

Figure 2.2 c) shows a humanoid robot built by Yamaha [15]. It can sing and is a step towards service robots. Such robots are relevant for the support of humans in service questions as they will be more easily accepted by humans than non-humanoid robots. Beside the presented



a) *b)*

Figure 2.3.: *Two example robots without mechanical parts. A worm like robot from the Waseda University Tokyo is shown in a) [111] and the ChemBot presented at the DARPA is shown in b) [112].*

ground robots several additional kinds exist, e. g., wheeled robots. Autonomous cars are also a famous example. Several big companies, as well as research facilities work on cars able to autonomously navigate through daily traffic.

New types of robots without mechanical parts have been designed. These robots can, e. g., morph themselves from one design to another and are able to move. Figure 2.3 a) shows a worm-like robot able to move through electrical impulses. The robot in Figure 2.3 b) is able to change its form. Among other things, this might lead to new approaches for obstacle avoidance as future robots might not need to move around an obstacle but also have the capabilities to transform themselves in order to overcome obstacles.

The company Festo goes beyond the simple observation of animals. They want to develop new movement approaches for their future products. Birds, penguins, and jellyfish are some examples that have been rebuild, as shown in Figure 2.4, to prompt biologically inspired ideas. They focuses on all kinds of observed animal motion and wants to rebuild it in detail. The expert knowledge they achieve by designing and building these robots is used to improve their products. Thus, the observed behavior contributes directly to the ease of human work.

Beside the multicopter UAVs presented in Figure 1.1 a wide range of additional UAVs exists. Small UAVs used to spy on people, as well as large ones for exploration and monitoring have been built during recent years. The Neo S-300 UAV, shown in Figure 2.5 a) is build by the company Swiss UAV. It is a helicopter-like UAV and can be used, e. g., to provide goods due to its maximal payload of 35 kg and a possible mission radius of 50 km. Such UAVs are considered in the presented system for the support of ground units in diverse applications.

Another kind of UAVs are fixed wing UAVs as the RQ-170 Sentinel, presented in Figure 2.5 b). It is a fixed wing UAV equipped with stealth technology and can be used, e. g., to explore areas without being discovered. A fixed wing UAV has other constraints than a helicopter-like

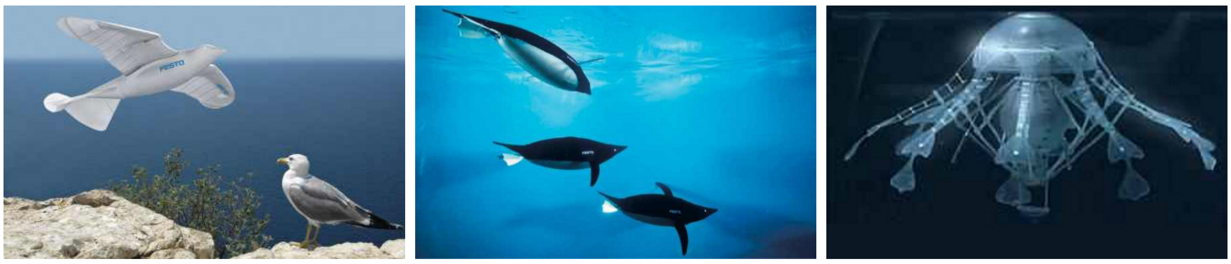


Figure 2.4.: *Biologically inspired bird [113], penguin [114], and jellyfish [115], build by the company Festo.*

UAV leading to different trajectories it can follow. It also needs a runway to start and land and cannot hover. An advantage of such UAVs is that they have better flight characteristics leading to larger mission radii and higher maximum speeds.

The robots illustrated in Figures 2.2 to 2.5 work in diverse environments and have differing movement capabilities. Only a small range of available robots and possible applications is presented in this section. A path planner for these robots must consider the capabilities of the robots to ensure that they are able to follow the computed paths.

Nowadays, it is possible to buy commercial robots, which are able to plan paths by themselves, e. g., home cleaning robots. Most of the planning approaches used for these robots are relatively simple and conduct a spiral-form movement with basic obstacle avoidance behavior based on try and error. Nevertheless, robots using laser technology to scan the surroundings and then computing efficient paths to cover the complete area exist. Similar technologies are used to design autonomous lawnmowers, able to mow entire meadows on their own. They also store the position of their charging station and move back to the station if they run out of power. Afterwards, they complete the mow.

As mentioned before, the robots illustrated so far have diverse capabilities, work in different environments, and have differing drives. This leads to various requirements for path planning,

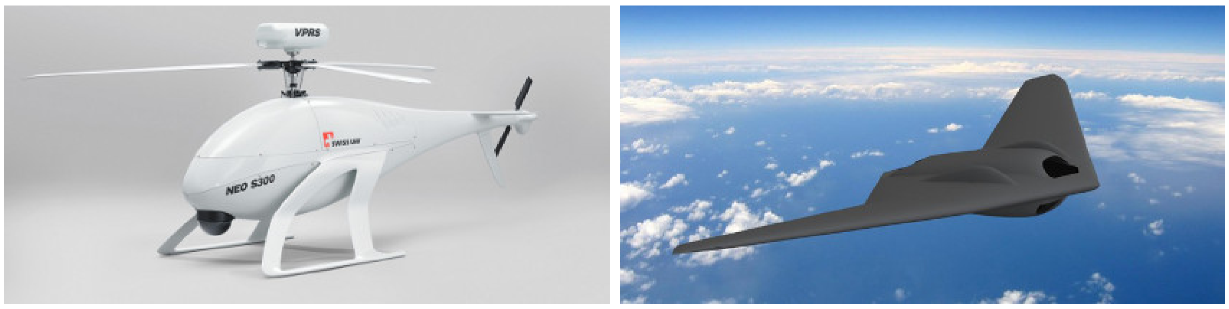


Figure 2.5.: *Figure a) shows the NEO S-300 [116] from Swiss UAV and b) shows the fixed wing stealth UAV RQ-170 Sentinel [117] designed for the U. S. military.*

formation movement, and cooperative behavior. The topic of flight control is neglected by the resulting system presented in this thesis, while the following sections present general state-of-the-art approaches for the mentioned objectives. Additionally, today's approaches similar to the ones used by the UAVs are presented in Sections 5.2, 6.2 and 7.2. The following section briefly addresses general path planning approaches with the main focus on UAV path planning.

2.3.2. UAV Path Planning

Countless techniques for path planning exist. They are mainly based on the roadmap, cell decomposition and potential field approaches, as described in Section 3.1. Diverse additional methodologies are used to design state-of-the-art UAV path planning systems. Examples are Voronoi diagrams combined with multi-objective fuzzy optimization [118] or combined with ant colony optimization [119], linear programming [120], multi objective optimization by receding horizon control [121], neural networks [122], [123], simulated annealing [124], dynamic programming [125], or genetic algorithms [126]. In order to not digress from the main topic, only one UAV path planning system is selected and described in more detail.

A real world approach for autonomous navigation is presented by Ross et al. [127]. They designed a system able to navigate a Parrot AR drone [26] (cf. Figure 1.1) maintaining a constant velocity of 1.5 m/s through natural forest environments. The UAV flies completely reactive and detects obstacles in the camera images. Supervised learning was used to provide the UAV with the ability to mimic expert pilot's choices of actions. The aim was to teach a linear controller of the left-right velocity how to avoid obstacles represented by trees of differing size. The altitude and velocity of the UAV was fixed during one test. This resulted in a controller able to map RGB images to flight commands. Combining their experiments, the UAV flew 3.4 km and was able to avoid more than 680 trees. The narrow field of view given by the UAV camera caused most of the crashes. This can be reduced, e. g., by creating a world model and using a planning approach combined with a much wider field of view. Finally, Ross et al. were able to teach flight behavior for complex environments to a real UAV.

As presented, a wide range of approaches for path planning of single UAVs exists. Researchers try to port some of them in order to control formations as described in the following section.

2.3.3. UAV Formation Flights

Also several approaches to autonomously fly in formation exits. They are mainly subdivided into leader-follower, virtual leader and behavioral approaches, as described later in detail in Section 3.4.1. Example techniques for today's UAV formation flights are receding horizon control [128], [129] and neural networks [130]. They are mainly based on a positioning system and inter-UAV communication while they try to preserve the distances between the UAVs in order to fly in formation. Such a neighboring distance approach is similar to the behavior of schools of fish (cf. Section 2.1.3) with the difference that most times one UAV is chosen to lead the entire formation.

The presented techniques are also used for single UAV path planning approaches. Adapting them to control formations increases their complexity. Yang et al. [131] try to overcome this issue by decentralizing formation flights based on a subdivision of the formation into single UAVs.

A real world approach for formation flights has been presented by Kushleyev et al. [132]. A team of 20 micro quadrotors is used and an external localization system provides positioning data. The team of quadrotors is subdivided into groups and each group can be controlled separately. The UAVs of a group are treated as a single individual and only one path, all robots of a group follow is computed. Known three-dimensional environments have been used to perform the formation flights based on an extension of mixed integer linear programming, called mixed integer quadratic programming, to compute a trajectory for each group or each quadrotor in a subregion of the environment. This technique maps trajectory planning to an optimization problem. The results show that the quadrotors are able to change the shape of the overall formation by changing the position of groups in the formation in order to avoid obstacles.

The methods presented so far consider either single UAV trajectories or flights in a single formation. To reach an efficient behavior with the possibility to simultaneously process several tasks and to create different formations, a cooperative behavior is needed. State-of-the-art approaches leading to such behaviors are presented in the following section.

2.3.4. Cooperative UAV Behavior

A few of today's UAV cooperation techniques are presented and shortly described in this section. Further approaches based on market places and auctions are presented in Section 7.2.

One state-of-the-art approach for cooperative UAV swarms has been presented by Madey et al. [133]. They use dynamic data-driven application systems (DDDAS) in combination with engineering guidelines to control an entire UAV swarm. The three objectives near real-time dynamic command and control, improved automation mission planning, and dynamic real-time re-tasking are addressed in their work. DDDAS uses the sensor data reached from the UAVs for the prediction of future needs in order to compute reasonable control commands.

Renzaglia et al. [134] use cognitive-based adaptive optimization for surveillance with multiple UAVs. They take unknown three-dimensional environments into account. Their objective is to provide a scalable and efficient technique to navigate the UAVs into a formation that locally optimizes surveillance coverage. They model coverage as multi-objective optimization problem of functions with an unknown explicit form. While continuously optimizing the UAV positions, they are able to increase the size of the covered area.

Other cooperative state-of-the-art approaches are based, e. g., on ant colony optimization [135], consensus-based task allocation [136] and market place approaches [137]. A disadvantage of most cooperative approaches is that they depend on communication. Sujit and Sousa [138] try to overcome this issue. They consider market place cooperation strategies with partial communication failures.

A cooperative UAV approach for constructions has been presented by Lindsey et al. [139]. They use real quadrotors to construct 2.5-dimensional tower-like structures. These structures consist of strata of identical cubes. A Vicon camera system [140] is used to track the UAVs and to estimate the positions and orientations of the bins and the base of the constructed object. Finite state automata coordinating the concurrent UAVs' actions. The main tasks of the UAVs are to pick up parts, transport them to the structure, and to assemble them.

The robots and approaches presented in this section briefly reflect current design and research. Nearly all of today's approaches that work with multiple UAVs in real world environments use a camera-based tracking system to provide the UAVs with global positioning data. Overcoming the limitations of such a need will be a topic of further research during the next years. Swarm approaches, like the pheromone based foraging behavior of ants (cf. Section 2.1.1) could result in possible solutions. The map-like organization of spatial memory of bees (cf. Section 2.1.2) in combination with SLAM (cf. Section 2.2.3) techniques to determine the current positions might also be a possibility to overcome the need of tracking systems.

This thesis does not cover all kinds of possible robots and focuses on UAVs. Beside the specialization of most researchers on one of the presented topics, additional differences between the presented approaches and this thesis exist. They are described in the next section.

2.4. Contrasting the Thesis

Most approaches use formation flights to simply move a single formation from an initial position to a target position. Compared to this the UAVs presented in this thesis can create different formations simultaneously to entirely explore complex three-dimensional environments. This includes flights into buildings to explore single rooms and flights through tunnels, etc. The formations are also highly scalable in the number of participating UAVs.

Many other approaches work with a fixed height and therefore have lower obstacle avoidance capabilities. They also do not consider highly dynamic environments. The UAVs designed in this thesis are able to react quickly to environmental changes (cf. Chapter 8). Each UAV also has the capability to fly to predefined target areas and to entirely explore an environment on its own.

Additionally, the UAVs work cooperatively and completely decentralized. They are able to allocate tasks by themselves and to create new tasks if this leads to further benefits or avoids damage. Exploratory navigation with reduced redundancy is also supported leading to a highly complex UAV behavior. The designed approach supports the essential properties scalability, robustness, and flexibility introduced in Chapter 1.

Another contrast to various other approaches is that all algorithms used in this thesis are mathematically verifiable in terms that they lead to the expected UAV behavior. The path planning algorithm guarantees obstacle avoidance and the formation algorithm ensures that the UAVs fly to the desired target positions while preserving a formation. Superpositioning of these approaches still guarantees that the UAVs will never touch an obstacle while flying to the expected positions.

2.5. Summary

The design of simple agents working together to solve complex tasks started millions of years ago by nature. Based on the experience nature gained in species design, several researchers have developed new methods for robot movements and task procession. Nowadays, approaches to redesign nature exist. Festo, e. g., redesigned animals in order to learn more about natural approaches and to adopt beneficial approaches for their products.

Robotic development reached increasing interest in the 1980s and several different approaches have been designed. A few of the most famous approaches are presented in this chapter. Additionally, some of the most famous paradigms to create robot organizations with abilities to efficiently solve relevant tasks are introduced. Most of these approaches have been developed to work with a central coordination instance. Such an instance leads to a bottleneck and a single point of failure. To avoid these disadvantages and to become independent of control personal in order to, e. g., accomplish tasks in deep space or deep sea, the market place approach has been selected and extended for the system presented in this thesis. The resulting market place works in a completely decentralized way.

Nowadays, countless robots and methods to move any kind of robots exists. Researchers started to dare new ways in robot design to overcome the limitations of mechanical robotic systems. They showed that robots consisting mainly of compressed air are able to carry more than one hundred kilograms. Two robots without mechanical parts are also presented in this chapter.

This chapter can only give a brief and incomplete overview of robotic behavior and robotic movement as this area is really large with thousands of researchers working in the field for decades. Therefore, countless methods for autonomous robot behavior exist. Only a few of them are used for the design of the resulting system. The foundations of the used approaches are described in detail in the following chapter.

Chapter 3

Cooperative Path Planning Foundations

One of the main objectives (cf. Section 1.2) is to design a system capable of exploring almost any kind of three-dimensional terrains. The ability to establish dynamically changeable formations is a further objective. A wide range of methods from different research areas can be used for designing such a system. Another requirement is that the resulting behavior has to be mathematically verifiable. This reduces the number of applicable methods.

The approach presented in this thesis combines different research areas whose foundations are introduced in the following sections. The chapter is structured as follows: First, the three main methods for path planning are introduced, followed by a general problem formulation for path planning. Based on this problem formulation the configuration space is introduced and the two main space types, occupied space and free space are defined.

The system uses cell decomposition in combination with artificial potential field theory described in the following sections. Special potential fields based on harmonic function theory and bifurcation theory are introduced in detail. Additionally, some requirements that must be met to fly in formation are presented.

Hereafter, the foundations for intelligent UAV behavior are specified followed by the basics for task creation and task allocation for multiple robots. Finally, a verification of the approach must be possible as mentioned before. Lyapunov's second method is used to verify the resulting behavior and thus introduced in the further course of this chapter. It concludes with a summary.

A world model is used by the UAVs to compute feasible paths. The data structure for the representation of the world model is one of the most important design choices. It affects the path planning methods in several ways. Not only the resulting paths but also the computation times needed for the single calculations result at least partially from the data structure. Several alternatives for each of the selected methods of the presented system exist. Some of the most famous approaches are introduced and some of them are described briefly. The next section shows the three main methods used for the design of path planning systems.

3.1. Path Planning

Path planning is a wide area of research. A countless number of path planning approaches for the movement of single robots, as well as for the movement of multiple robots and movements in formation are described in literature. Beside completely reactive approaches, the following three methods or a combination of them are most commonly used:

1. Road map method.
2. Cell decomposition.
3. Potential field theory.

The basic idea of the road map method is to create a map reflecting the connectivity of the obstacle free parts the environment is composed of. If such a road map exists, path planning is relatively simple. Only the initial and the target location of a robot must be connected to the road map. A feasible path is given by the path in the road map if both locations can be connected to it. Otherwise, no feasible path exists. The method can be combined with other methods as, e. g., shown by Ozaki et al. [141]. They use cell decomposition in combination with a road map graph to move a camera in three dimensions in order to follow given subjects. The road map graph is achieved by hierarchical cell decomposition leading to a description of the collision free camera movement space. The shortest path is finally computed by the A^*-algorithm [82].

The road map method can also be easily applied to navigation systems in order to find paths along roads from the start position to a destination. Therefore, a graph can be constructed by modeling the positions of all crossroads and road forks as nodes and the connecting roads as edges. Differing paths can then be computed by different weighting of the single edges, e. g., to find the fastest route or the shortest route.

The system presented in this thesis is based on a combination of cell decomposition and artificial potential field theory. This combination is described in more detail during the following sections. A subdivision of the complete terrain into several subareas is conducted by an octree to achieve cell decomposition. Afterwards, a so called potential value ϕ is computed for each of the resulting leaves. Based on these values, a feasible path can be computed. This chapter also provides a detailed description of the basic methods the resulting system depends on and mentions additionally relevant approaches.

The UAVs do not only explore the terrain. They also have to fly to predefined areas with high priority. To reach an increase in efficiency, the ability to establish formations exists. Exploratory navigation is treated as a series of several consecutive flights to unexplored areas until each area has been explored by UAVs' equipment. Coordination of the UAVs is required in order to decrease the time a group of UAVs need for an entire environmental exploration.

3.1.1. Problem Statement

Various approaches have been designed based on the three methods for path planning mentioned before. In the context of the presented system, path planning is based on the following environment:

Let $\mathcal{U}$ be a UAV. It moves in an Euclidean space $\mathcal{W}$, represented as $\mathbb{R}^N\colon N \in \mathbb{N}$ (for the UAVs is $N = 3$). Additionally, for $j = 1, \dots, r, r \in \mathbb{N}, \mathcal{O}_j \subset \mathcal{W}$ are fixed three-dimensional objects $\mathcal{O}_j \in \mathbb{R}^3$, distributed in $\mathcal{W}$. Every single $\mathcal{O}_j$ is an obstacle. Assume that the geometry and the position $p_\mathcal{U}$ of $\mathcal{U}$ is known. The $\mathcal{O}_j$ can be known a priori, but they don't have to be. Further on, it is assumed that $\mathcal{U}$ is not restricted in its movement through kinematic constraints. This leads to the following definition:

Definition 3.1. Path planning: *Given an initial position $p_\mathcal{I}$ and direction plus a target position $p_\mathcal{G}$ and direction of $\mathcal{U}$ in $\mathcal{W}$. Calculate a path ρ consisting of a consecutive sequence of positions and directions of $\mathcal{U}$ under avoidance of contact with any $\mathcal{O}_j$. The path has to start at $p_\mathcal{I}$ and must end at $p_\mathcal{G}$. Return an error if such a path does not exist.*

Beside a position and direction, the UAVs have additional properties, such as a role and tasks to be solved. Not only the position but the configuration of a UAV is needed for an efficient and cooperative multi-UAV system in order to take all properties into account. A configuration includes the relevant properties of a UAV. All path planning problems have in common that a path from the initial configuration $q_\mathcal{I}$ to the target configuration $q_\mathcal{G}$ has to be computed. This leads to the concept of the configuration space [142].

3.1.2. Configuration Space

The configuration space $\mathcal{C}$ considers the properties of a UAV $\mathcal{U}$ within the space $\mathcal{W}$. $\mathcal{U}$ is represented by a single configuration $q_i \in \mathcal{C}$, which is a tuple $(q_1, \dots, q_n)$. The n independent variables can, e. g., be the position $p_\mathcal{U}$, the role $\mathcal{R}_\mathcal{U}$, and tasks $\mathcal{T}_\mathcal{U}$ of $\mathcal{U}$. A UAV is considered to be a point $p \in \mathcal{W}$ without a volume. The point p is equal to the position of $\mathcal{U}$ in $\mathcal{W}$. So, UAV $\mathcal{U}$ has a configuration $q_\mathcal{U} \in \mathcal{C}$ consisting of all relevant properties of $\mathcal{U}$, e. g., the current position $p_\mathcal{U}$ and its direction. The actual volumes of the UAVs are considered by the environment representation as described later in Section 4.2.2.

Definition 3.2. Configuration space: *Let n be the number of configurations q_i a UAV $\mathcal{U}$ can take. The configuration space is then defined as:*

$$\mathcal{C} = \bigcup_{i=1}^{n} q_i. \tag{3.1}$$

Following the definition of Barraquand et al. [142] it is possible to generate a geometrical application able to map each configuration q to a corresponding point p of the terrain. This map

$$\begin{aligned} X\colon \quad & \mathcal{C} \to \mathcal{W}, \\ & q \mapsto p = X(q) \end{aligned} \tag{3.2}$$

is called the forward kinematic map.

To actually determine a position of a UAV $p_{\mathcal{U}}$, a Cartesian coordinate system $\mathcal{F}_{\mathcal{W}}$ is embedded in $\mathcal{C}$. Now a configuration of $\mathcal{U}$ is the specification of the position and direction of $\mathcal{U}$ having regard to $\mathcal{F}_{\mathcal{W}}$, plus its additional properties. $\mathcal{C}$ is discretized like the real space using cell decomposition. Two main space types are considered that lead to a specific discretization: Occupied space and free space. They are explained in the following section.

3.1.3. Occupied Space

The workspace of $\mathcal{U}$ contains a finite number of obstacles $\mathcal{O}_j$ with $j = 1, \ldots, r$. Obstacles can, e. g., be skyscrapers, mountains or other UAVs. Each obstacle $\mathcal{O}_j \subset W \in \mathbb{R}^N$ is represented in $\mathcal{C}_{\mathcal{O}} \subset \mathcal{C}$ of those configurations in which a UAV takes a position within an obstacle $\mathcal{O}_j$.

Definition 3.3. Occupied space: *Let n be the number of configurations $q \in \mathcal{C}$. The occupied space is defined as:*

$$\mathcal{C}_{\mathcal{O}} = \bigcup_{q_i \in \mathcal{C}} \exists 1 \leq j \leq r\colon \{X(q_i) \in \mathcal{O}_j\}. \tag{3.3}$$

Thus, the occupied space is defined as the union of all configurations that include a position inside an obstacle. These configurations are forbidden because they result in damage not only to the UAVs. Body injuries can occur if people are close by. The approach used to design the UAV system presented in this thesis can be mathematically proven in terms that no UAV will ever take one of these configurations and thus guarantees the avoidance of $\mathcal{C}_{\mathcal{O}}$.

A creation of the occupied space is possible if the environment is known a priori. The UAVs should also be able to fly in partially unknown, as well as in completely unknown environments and explore them over time. In such a case the occupied space is set to $\emptyset$ a priori and obstacles have to be mapped into the space as soon as they are detected. This leads to a change of the distribution of occupied and free space over time. The UAVs must take such changes into account in order to find feasible and safe paths from $q_{\mathcal{I}}$ to $q_{\mathcal{G}}$. Based on the definition of the occupied space, the free space can be defined as explained in the next section.

3.1.4. Free Space

The UAVs are not able to take every configuration in $\mathcal{C}$ due to the existence of obstacles. Using the definition of $\mathcal{C}_{\mathcal{O}}$ it is possible to calculate the allowed configurations of the UAVs, as shown in Equation 3.4. This subset of configurations in which the UAVs do not take a position within an obstacle is called free space $\mathcal{C}_{\mathcal{F}}$.

Definition 3.4. Free space: *Again, let n be the number of configurations $q \in \mathcal{C}$. The free space is defined as:*

$$\mathcal{C}_{\mathcal{F}} = \mathcal{C} \setminus \mathcal{C}_{\mathcal{O}}. \tag{3.4}$$

Each configuration $q \in \mathcal{C}_{\mathcal{F}}$ is called a free configuration. Following this modeling a collision-free path from an initial configuration $q_{\mathcal{I}}$ to a target configuration $q_{\mathcal{G}}$ is defined as a consecutive map $\rho\colon [0,1] \to \mathcal{C}_{\mathcal{F}}$ with $\rho(0) = q_{\mathcal{I}}$ and $\rho(1) = q_{\mathcal{G}}$. Furthermore, the computation of discrete paths is based on cell decomposition and explained in the following section.

3.2. Cell Decomposition

The main idea of cell decomposition is to subdivide $\mathcal{W}$ and respective $\mathcal{C}$ into disjoint areas called cells or leaves. An important part is the connectivity graph G [143]. It is a non-directed graph and represents the adjacency relation between the cells. The nodes of the graph are the cells of the free space. Two nodes are connected if and only if (iff) the corresponding cells are adjacent. A path can then be constructed by connecting the initial configuration to the target configuration through the center points of the intersections of every two successive cells.

Each subarea of the environment is finally represented by a leaf that contains all relevant information like position and dimensions of the area, positions of obstacles, etc. Cell decomposition approaches can be classified as exact or approximative methods [107]. Exact approaches normally need a much higher computational complexity to be performed than approximative ones, which can be conducted, e. g., by bounding volumes [144], grids [145], bsp trees [146], k-d trees [147], quadtrees [148], or octrees.

Grids are due to their simplicity one of the most commonly applied approaches. A decomposition of $\mathcal{W}$ into equally sized subareas is reached using a grid. One disadvantage of this approach is that neighboring areas with equal properties are not considered. A combination of, e. g., several neighboring areas $l_i \subset X(\mathcal{C}_{\mathcal{O}})$ would decrease the computational effort of the approach. Neighbors of a current area are all areas adjacent along a face, an edge or a vertex if a grid and a three-dimensional environment are considered. A combination of neighboring areas with equal properties to larger areas, so that only the bigger area has to be taken into account for path computation, can lead to an increase in efficiency.

The presented UAV approach decomposes $\mathcal{C}$ with the aid of an octree. This is a common data structure for the subdivision and representation of three-dimensional areas. The nodes of an octree are divided into eight equally sized child nodes. Each child of a node with the dimensions x, y, z has the dimensions $x/2, y/2, z/2$. Such a subdivision leads to relatively small nodes, even at a low tree depth.

Figure 3.1 shows the concept of an octree. The root node encloses the entire environment and is located at level 0. It is subdivided into eight child nodes, located at level 1. The first and the sixth node at level 1 are also subdivided. The figure shows the resulting terrain subdivision at the single levels on the left hand side and the corresponding octree on the right hand side. For the system presented in this thesis, the subdivision is based on the position and angles of the UAV, the properties of the camera equipment the UAV uses to explore the terrain, as well as the positions of obstacles. It is described in detail in Section 4.3.1.

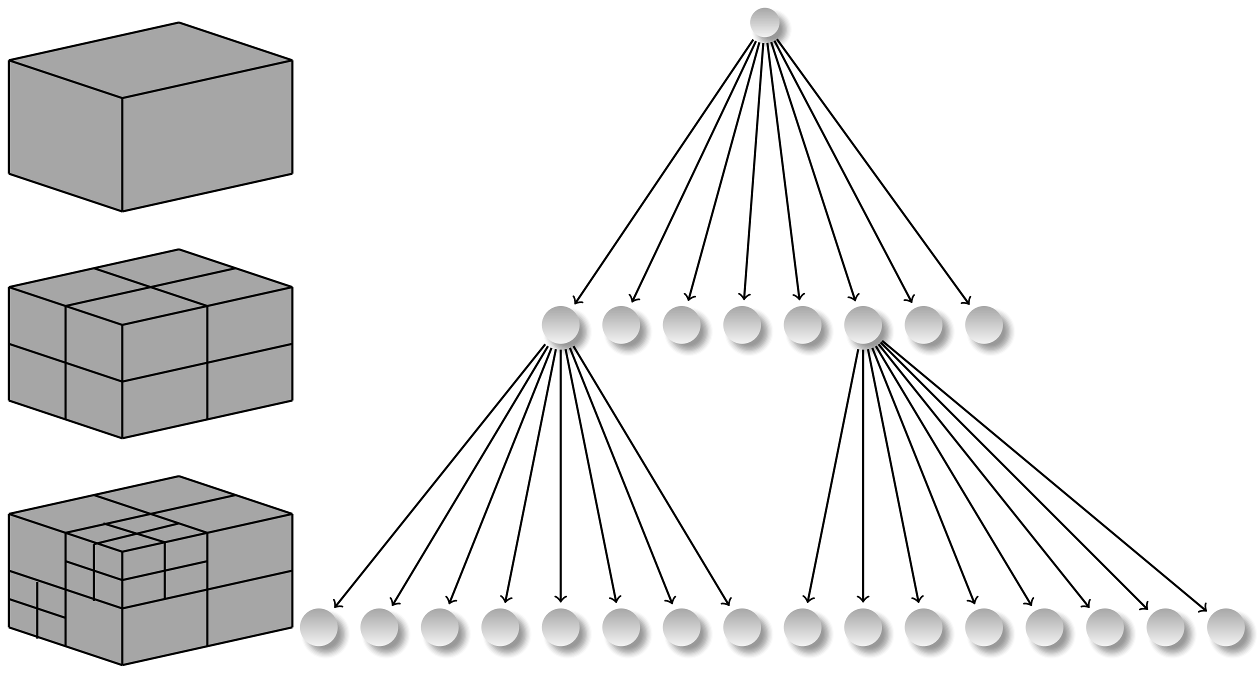


Figure 3.1.: *Subdivision of a cube into octanes and the corresponding octree. The octree has a maximum level of 2.*

In general, decomposition is conducted as follows. First, the octree divides the complete space into eight equally sized subspaces. These subspaces are recursively divided into eight further subspaces. Subdivision stops when the enclosed space of a leaf l_i is of homogeneous type or a predefined maximum breakdown is reached. Leaves generated due to the predefined maximum breakdown are called leaves at maximum level. Homogeneous means that the space consists of only one subspace of $\mathcal{C}$, e. g., $\mathcal{C}_{\mathcal{O}}$. If the maximum breakdown is reached and the space is not homogeneous, another approximation takes place. Each leaf partly consisting of occupied space is completely marked as occupied space. If the space consists of different subspaces of $\mathcal{C}_{\mathcal{F}}$, it is assigned to the type of space the leaf constitutes mainly of. The complete terrain is finally represented by the union of the resulting n leaves $l_i, i = 1, \ldots, n$ of the octree:

$$\mathcal{W} = \bigcup_{i=1}^{n} l_i. \tag{3.5}$$

Each leaf l_i has several neighboring leaves $l_j, j = 0, \ldots, m$. The octree must change over time to represent, e. g., newly explored areas or newly detected obstacles. The UAVs must take such changes into account to compute safe and feasible paths. This also leads to a change of the neighborhood conditions during UAV flights.

Definition 3.5. Neighborhood: *Two leaves l_i and l_j are neighbors $\{l_i, l_j\} \in \mathcal{N}$ in direction $\mathcal{D}, \mathcal{D} = 1, \ldots, 26$, iff l_i corresponds to the smallest block adjacent to l_j in direction $\mathcal{D}$.*

The 26 neighboring directions $\mathcal{D}$ mentioned in Definition 3.5 are given by the definition of neighborhood. Two leaves can be adjacent along a face (6 directions), along an edge (12 directions), or along a vertex (8 directions). This kind of neighborhood is called Moore neighborhood [149]. As the leaves have different dimensions, the number of neighbors per direction can vary between 0 neighbors (the leaf is located at the terrain border) and 4^{o-1} neighbors with $o \in \mathbb{N} \setminus \{0\}$ be the maximum level of the octree.

Determining neighboring nodes of an octree node is of higher complexity than for a grid. It normally takes logarithmic time to find neighbors in an octree while this operation can be conducted in constant time for a grid. This disadvantage is important to mention as the UAVs need to determine neighboring nodes with high frequency. The operation is required for potential field calculations and for path computation. An algorithm able to find a single neighbor node in time $\mathcal{O}(\log_8 n)$, with n representing the number of leaves of a complete octree has been designed. The idea of the algorithm is described in detail in Section 4.2.4 and the algorithm is presented in Appendix A.1. Tests have shown that the disadvantage is more than balanced by the achieved reduction of the number of leaves needed for potential field calculations (cf. Chapter 8).

The major advantage of an octree over a grid is that areas not relevant for the current path computation can be combined to bigger leaves. An efficient implementation of node combination and subdivision leads to a system able to represent really large areas without a high increase in computation time and memory consumption. Based on cell decomposition, a discrete collision free path leading a UAV from $q_{\mathcal{I}}$ to $q_{\mathcal{G}}$ can be defined as follows:

Definition 3.6. Discrete Path: *A discrete path ρ_d starting at the initial leaf l_i and ending at the target leaf l_n with length n is:*

$$\rho_d = \{l_1, \ldots, l_n \colon \forall l_i \in \mathcal{C}_{\mathcal{F}}, \forall \{l_i, l_j\} \in \mathcal{N}, i = 1, \ldots, n-1, j = 2, \ldots, n\}. \tag{3.6}$$

A path as defined in Definition 3.6 can be computed and followed by a UAV. Based on the achieved decomposition, it is possible to compute values for each leaf leading to feasible paths. One approach for the computation of such values is artificial potential field theory, introduced in the next section.

3.3. Potential Field Theory

Artificial potential fields for motion planning have been introduced by Khatib [62] in 1986. This technique steers a manipulator in a field of artificial forces. The main idea is to affect the UAV through linear superpositioning of fictive forces or their potentials ϕ to obtain paths. The UAVs' target positions are modeled as attractive poles and obstacles are modeled as repulsive poles. To adequately move the UAVs, a field of forces affecting the complete terrain has to be computed. Cell decomposition for the subdivision of the entire terrain is often combined with artificial potential field theory. The single ϕ can then be calculated for each leaf or cell created through the cell decomposition approach. This leads to differing potential values for each area, which are used to model forces. Two basic types of forces based on two types of potential fields are commonly used for path planning:

1. Attractive forces: They represent targets and pull the UAVs towards them.
2. Repulsive forces: They represent obstacles and push the UAVs away from them.

To obtain the proper modeling of an artificial potential field, the attractive forces have to fulfill two basic requirements:

1. The complete configuration space must be affected and
2. the forces must always lead to the target configuration q_G.

Figure 3.2 a) shows an example of an attractive potential field. The target configuration of a possible UAV is at the center of the potential field. All surrounding configurations have a higher potential value. Following the descent gradient $-\nabla$ leads the UAVs from any initial configuration to the designated target configuration.

Suppose that the first condition for attractive forces is unaccomplished. In this case, areas without acting forces exist. A UAV would never move if its initial configuration is in such an area. Another problem occurs when a UAV flies into such an area because no force affects the UAV and it would then usually stop flying and reaching the target configuration becomes impossible.

The second condition is crucial to ensure that the target configuration can be reached. If forces not leading to the target configuration exist, the UAV can get affected by these forces and it can become impossible for it to reach the target. Several possibilities to guarantee forces as demanded exist. One solution is to adjust the potential values like a cone. The determining values can, e. g., depend on the Euclidean distance to the target. The attractive potential field shown in Figure 3.2 a) results from the following equation:

$$\phi_{attractive} = 1 - e^{-\left(x^2+y^2\right)}. \tag{3.7}$$

The variables $x, y \in [-2; 2]$ denote the distances to the center point $(0, 0)$ of the two-dimensional attractive potential field.

Repulsive forces are also of common use. Their duty is to ensure obstacle avoidance. They must also fulfill two requirements:

1. Only a specified surrounding area of the obstacle must be affected and
2. they must always lead away from the obstacle.

Figure 3.2 b) shows an example of a repulsive potential field. The obstacle is modeled to be at the center of the potential field. All surrounding potential values are less than the potential value of the obstacle and they decrease with distance to the obstacle. Following the descent gradient will lead the UAV away from an obstacle modeled this way.

The first condition for repulsive forces ensures that obstacles do not affect the complete configuration space. Affecting the complete configuration space can change the trajectories of the UAVs such that they take unnecessary detours. The second condition ensures that UAVs are never pulled towards an obstacle. Repulsive potential fields can be based, e. g., on the Euler's number e like the repulsive potential field shown in Figure 3.2 b). It is obtained through the following equation:

$$\phi_{repulsive} = \mathrm{e}^{-\sqrt{x^2+y^2}}. \tag{3.8}$$

The variables $x, y \in [-2; 2]$ denote the distances to the center point $(0, 0)$ of the two-dimensional repulsive potential.

The potential fields shown in Figure 3.2 can be superpositioned through a simple addition as, e. g., described in [150]. A path can follow the descent gradient within the resulting potential field. Using the descent gradient, a UAV compares the potential value given at its current configuration with the potential values of all neighboring configurations. The configuration with the lowest potential value is then selected as next configuration. This action is repeated until no configuration with a lower potential value than the current configuration can be selected. A superpositioned potential field keeps the UAV away from obstacles modeled by repulsive potential fields and simultaneously pulls it towards the target configuration based on the attractive potential field.

The UAVs considered for the presented approach are helicopter-like. Hence, it is not necessary to take non-holonomic constraints [107] into account as the UAVs are able to hover and turn

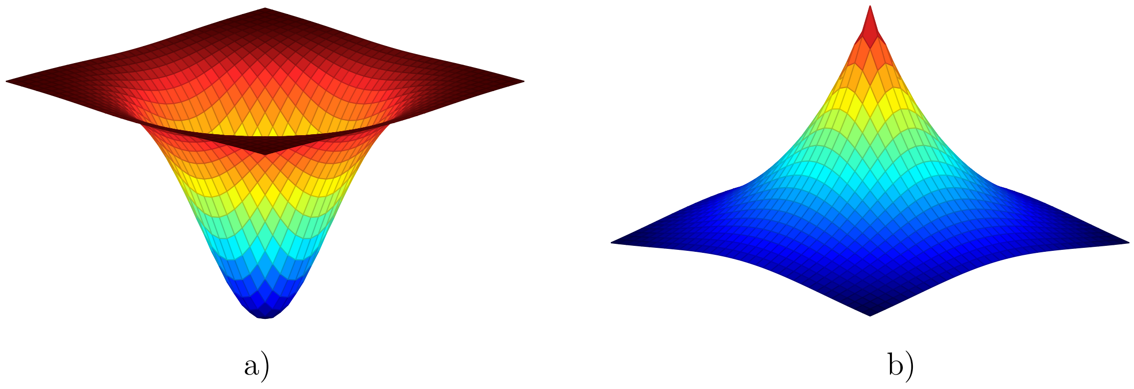


Figure 3.2.: *An attractive potential field is illustrated in a). It pulls the UAVs towards the center of the potential field. A repulsive potential field is presented in b). It repels the UAVs from an obstacle.*

at given positions $p_{\mathcal{U}}$. If other kinds of robots, like fixed wing UAVs or cars are considered, a feasible path might be obtained, e. g., by reducing the neighboring leaves the drone can reach dependent on the non-holonomic constraints. Another possibility is the use of informed search algorithms, like an adapted A*-algorithm [151]. The special kind of potential fields computed by the system presented in this thesis and described in the following section provides properties for the motion planning of non-holonomic systems [152].

Artificial potential fields computed as presented before come along with several problems, especially in case of complex and highly dynamic environments. The basic movement in such a field of forces is often conducted using a gradient method. Obstacles being around the UAVs trajectory may cause the occurrence of so called local minima or local equilibria. These equilibria are one of the most important problems.

The term minimum is of common use in potential field theory. In bifurcation theory (cf. Section 3.4.2) the term equilibrium is of common use. The aim of an attractive potential is to move the UAV to the desired stable equilibrium position that corresponds to the minimum potential value. The approach used to compute paths the UAVs can follow, always assigns the minimum potential value to zero. To provide a constant nomenclature, the term equilibrium is always used hereinafter.

If a potential field as introduced above is considered, a local equilibrium occurs when a single potential value is less than all surrounding potential values. Let l_i be a node. So called flat regions occur when no neighboring node of l_i has a lower potential value than l_i. In both cases the driving force vanishes and the UAV gets trapped. This violates the second requirement for attractive forces and should be avoided.

Several methods to avoid and to get out of such local equilibria and flat regions exist as, e. g., changing the potential field [153] if the UAV gets trapped or using algorithms to leave the equilibria, like simulated annealing [154]. Another method is to avoid the creation of local equilibria and flat regions. The use of harmonic functions [34] provides a possibility for the creation of local equilibrium-free potential fields. A potential field has no local equilibrium and no flat region, iff for each unbound leaf l_i with potential value ϕ_{l_i} a leaf $l_n, \{l_i, l_n\} \in \mathcal{N}$ with potential value $\phi_{l_n} < \phi_{l_i}$ exists, i. e.:

$$\forall \phi_{l_i} \exists \phi_{l_n} : \{l_i, l_n\} \in \mathcal{N}, \phi_{l_n} < \phi_{l_i}. \tag{3.9}$$

3.3.1. Harmonic Functions

Harmonic functions for artificial potential field calculations have been introduced first by Sato [155] in 1987. He formally used them as tools for motion planning. An English version of the Japanese article can be found in [156]. Connolly and Burns [34] were one of the first who applied the approach for robot movement in 1990. One advantage of these functions is that one can prove the absence of local equilibria if they are used to represent a potential field. Harmonic functions lead to the possibility of calculating globally consistent and collision-free paths. They satisfy Laplace's equation in n dimensions. A Laplace's equation is an elliptic second-order partial differential equation and is named after Pierre Simon Laplace. The resulting mathematical problem is to find a twice-differentiable real value function of variables such that:

$$\nabla^2 \phi^h = \sum_{i=1}^{n} \frac{\partial^2 \phi^h}{\partial x_i^2} = 0, \tag{3.10}$$

where $\nabla^2 \equiv \nabla \cdot \nabla \phi^h$ is the Laplacian operator. The notation of Equation 3.10 is for Cartesian coordinates whereby h denotes that it is a harmonic function. The solutions of such differential equations are called harmonic functions. It is necessary that ϕ^h is a twice continuously differentiable function and that the sum of its second partial derivatives is zero. Additionally, ϕ^h must be strictly increasing or decreasing dependent on the length of the path to the target. In the obstacle free case this is given by the distance to the target. The value of ϕ^h is given on a closed domain Ω in the configuration space $\mathcal{C}$ (cf. Section 3.1.2) and satisfies the mean-value property (cf. Definition B.1) [157], [158], the min-max principle (cf. Definition B.2), [158] and the uniqueness principle (cf. Definition B.3), [34], [158], [159]. The min-max principle ensures that the potential function has its maximum and minimum values at its boundary points. The resulting system has its boundary points at those leaves of the octree that contain obstacle and target areas. The min-max principle also guarantees that no local equilibrium has a potential value less than the potential value of a global equilibrium. Furthermore, no saddle point has a greater potential value than an obstacle leaf. Figure 3.3 shows an example of a harmonic function, given by $\mathrm{e}^x \cdot \sin(y)$ and defined on $\mathbb{R}^2$ on the domain $[-2; 2]$.

One disadvantage of harmonic function based potential field calculations occurs if a superpositioning of several potential fields shall be performed. Any linear combination of two harmonic functions is also harmonic and a solution of Laplace's equation [158]. But Connolly

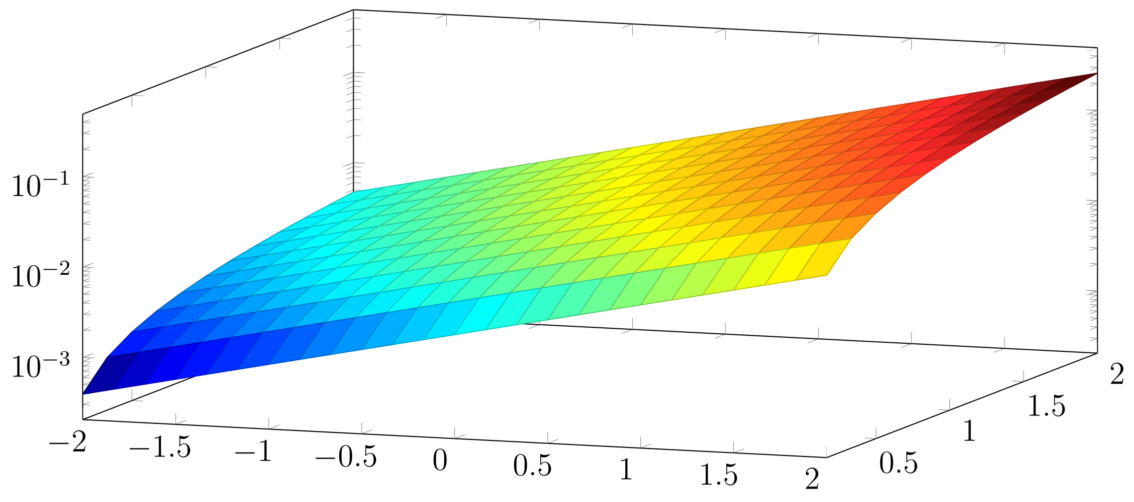


Figure 3.3.: *An example of a two-dimensional harmonic function computed using the function* $\mathrm{e}^x \cdot \sin(y)$, *which is harmonic for the two-dimensional case.*

states [34] that there is no guarantee that obstacles are avoided in complex or dynamic environments when several harmonic functions are superpositioned. The potential values in the neighborhood of an obstacle depend not only on this obstacle's potential, but also on every other obstacle's or possible target's potentials. If the configuration or the strength changes, the path of a UAV can get infinitely close to the obstacle. The only unswayable structure is a point. From this it follows that only points can be safely modeled as obstacles if superpositioning takes place.

To overcome this disadvantage superpositioning of several potential fields based on harmonic functions is not conducted in the presented approach. Instead a modeling as optimization problem takes place. First, the single leaves of the octree are set to be bound or unbound leaves. Leaves, which are target leaves or occupied leaves are called bound leaves and get assigned to special values based on the selected boundary condition. All other leaves are called unbound leaves. Thereafter, an approximation of a harmonic function for the potential values, which are not bound points, is computed. This approximation is finally successively relaxed. The relaxation also takes the repulsive potential values into account and brings the resulting function closer and closer to a harmonic function. Relaxation stops when path planning is feasible. The approach is described in detail in Chapter 5.

Various boundary conditions leading to different modes of behavior exist. Two of the most famous are the Dirichlet Boundary Condition and the Neumann Boundary Condition (cf. Definition B.7). Related to [160] the UAVs are considered to be points. Using the Neumann Boundary Condition the start points is assigned to a high value while the target point is assigned to a low value. The obstacle potential values are treated as homogeneous Neumann boundaries. The Neumann Boundary Condition constraints the normal component of ∇ to be zero at the boundaries. Using the Dirichlet Boundary Condition the obstacle potentials are assigned to fixed, high values while the target potentials are assigned to fixed, low values. Karnik et al. [160] compared both conditions. They showed that paths based on harmonic potential fields achieved by the use of the Neumann Boundary Condition tend to graze the boundary. The Dirichlet Boundary Condition has been chosen for the UAV system presented in this thesis as contact with obstacles must under all circumstances be avoided.

Connolly [161] has shown that harmonic functions lead to paths without spurious local equilibria. Combining harmonic functions with the Dirichlet Boundary Condition leads to a value-restricted configuration space, important for calculations with discrete arithmetic. The presented approach binds target areas to the potential value $\phi_{\mathcal{G}}^h = 0$ and occupied areas to the potential value $\phi_{\mathcal{O}}^h = 1$ to respect the Dirichlet Boundary Condition.

According to [34], every harmonic function defined on a compact region $\Omega = \partial\Omega \cup \dot{\Omega}$ satisfies three properties:

1. Analyticity: Each harmonic function is analytic (cf. Definition B.4).
2. Polar: Be $q_{\mathcal{G}}$ a target configuration with the constraint $\phi_{\mathcal{G}}^h = 0$. Set all obstacle configurations $q_{\mathcal{O}}$ to some constant $\phi_{\mathcal{O}}^h = c$. All harmonic functions satisfy the min-max principle [158], so ϕ^h is polar [162]. From this it follows that $q_{\mathcal{G}}$ will be the point at which ϕ^h attains its equilibrium value within Ω and the negative gradient always leads to $q_{\mathcal{G}}$.
3. Admissibility: If c is set to 1, ϕ^h will be admissible in the sense of the designed approach. This is a simple normalization.

Additionally, harmonic functions have several valuable properties [34], [161]:

- Soundness (cf. Definition B.6) up to discretization errors.
- Fast surface normal computation.
- The ability to exhibit different modes of behavior (grazing vs. avoidance).
- Robust control in the presence of unanticipated obstacles and errors.
- Lack of spurious local equilibria.
- The possibility of linear superpositioning.
- Robustness with respect to geometrical uncertainty.
- Continuity and smoothness of configuration space trajectories.

In this thesis a path planner is called sound when it is guaranteed that a path from each initial configuration to each target configuration will be computed if such a path exists (cf. Definition B.6). It is possible that a coarse discretization avoids a path computation, as shown in Figure 3.4, where the gray area represents an obstacle. Each leaf containing a part of the obstacle is treated to be in $\mathcal{C}_{\mathcal{O}}$ in order to ensure collision avoidance as the UAVs always fly from the center point of a node to the center point of a neighboring node. Using a coarse

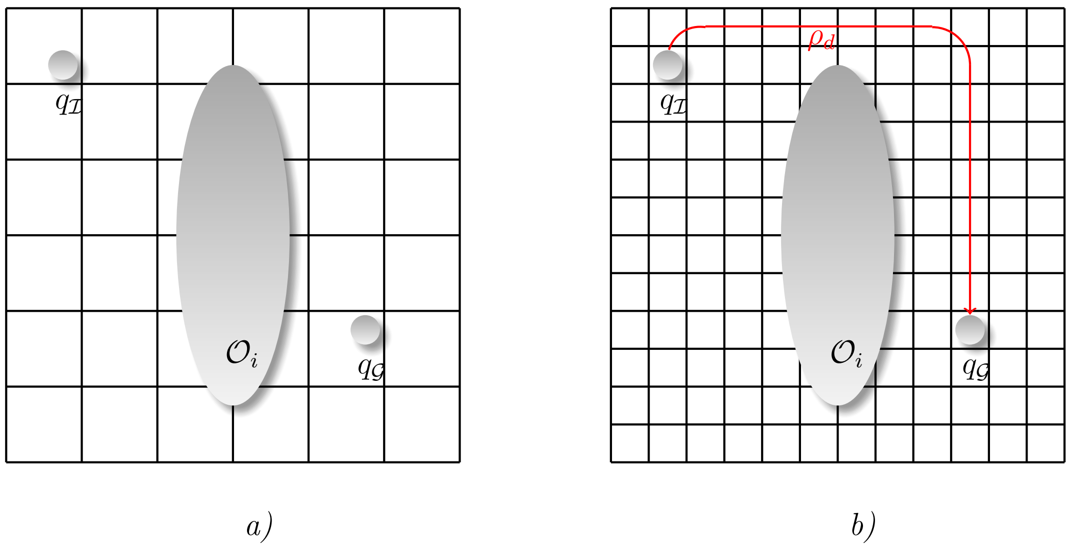


Figure 3.4.: *Using a coarse discretization as shown in a) can make path planning impossible. Refinement of the discretization as shown in b) leads to a feasible path ρ_d.*

discretization as shown in Figure 3.4 a) makes path planning from $q_{\mathcal{I}}$ to $q_{\mathcal{G}}$ impossible. If the discretization is refined, as shown in Figure 3.4 b) path planning from the initial configuration to the target configuration becomes feasible. Discretization errors cannot be compensated by harmonic functions.

To overcome the discretization problem, shown in Figure 3.4, the environmental subdivision is based on the dimensions of the UAV. A leaf at maximum level can encompass only one UAV at a time without collisions in at least one direction. A UAV also fits into one leaf at maximum level.

The used boundary conditions determine the mode of behavior mentioned before. The Dirichlet Boundary Condition raises obstacle regions to a constant high potential, while target regions are set to a low potential value. The resulting potential values of the free space are constrained by $\nabla^2\phi = 0$. Thus, ∇ is aligned with the surface normals of the obstacle regions and will tend to repel the UAVs from the obstacles. The Neumann Boundary Condition [160] does not keep obstacles at a constant potential value but the derivative of ϕ is constrained such that $n \cdot \nabla\phi = 0$ with n be the surface normal function for the obstacles. This causes $\nabla\phi$ to be tangential to obstacle surfaces and the UAV paths will gaze any obstacle they encounter [161].

Originally, potential fields based on harmonic functions have been used only with a fixed map and one single known target. But one can show that they also work well in dynamic environments with multiple UAVs and targets [3].

A UAV control system, based on the methods introduced so far is able to compute several paths consecutively to move the UAVs in different environments. It also has the ability to guarantee obstacle avoidance and sound path planning if the approximation is close enough to the resulting harmonic function. These are the basic requirements, needed to perform complete explorations. If information exchange regarding newly explored nodes is possible, the efficiency of exploratory navigation can be increased when the number of UAVs increases [4] due to implicit coordination. A further increase in efficiency can be reached through the establishment of a higher coordination level. One example for high level coordination are formation flights. The ability to establish formations can be reached by another kind of potential fields, introduced in the following section.

3.4. Formation Flights

The design of a system that allows UAVs to compute discrete paths ρ_d, starting at initial configurations $q_{\mathcal{I}}$ to every reachable target configuration $q_{\mathcal{G}}$ is possible, based on the previously described methods. An additional cooperation of multiple UAVs is needed to increase efficiency. Such a cooperation requires methods for information exchange.

Combining inter-UAV communication with the previously described methods leads to the basic requirements, needed to entirely explore $\mathcal{C}_{\mathcal{F}}$, using multiple UAVs. Performing information exchange reduces the time needed to explore a complete terrain if the number of participating UAVs increases [3], [5], [6]. An indirect coordination is conducted if the UAVs obtain information about parts of the terrain explored by other UAVs.

A higher degree of coordination can also lead to a more efficient task handling, e. g., for tasks like exploratory navigation of a given terrain in the aftermath of a disaster or in search and rescue missions, as shown in Chapter 8. Task handling is considered to be more efficient, when the time needed to solve tasks decreases. Explicit cooperation given, e. g., through formations leads to a higher degree of coordination. A formation of multiple UAVs has to be established, e. g., to solve a transportation task containing items too large to be handled by a single UAV. In order to give the UAVs of the presented system the ability to solve such tasks, a methodology for formation creation is described in this section. Simultaneous formation flights can lead to a further increase of efficiency. Some requirements must be met to actually fly in formation and it must be chosen, which kind of formation approach should be supported by the resulting system as described in the following section.

3.4.1. Requirements

As mentioned before, formation flights need direct UAV coordination. Several requirements must be fulfilled to fly in formation. The UAVs must be able to actually establish a formation in decentralized manner. They can conduct an agreement, e. g., by reaching consensus about where and under which constraints a formation shall be created. For this purpose information exchange is necessary. Inter-UAV communication based, e. g., on UDP/IP to exchange all necessary information is used by the presented system's UAVs.

The system to be designed is a dynamic, nonlinear system. To ensure that the UAVs are able to explore the terrain using formations, it must be guaranteed that the target of the formation always provides a stable equilibrium (cf. Section 3.6.1). The formation may break apart during movement without this guarantee. Using bifurcation theory as described in the following section, enables the application of Lyapunov's second method (also called Lyapunov's direct method, cf. Section 3.6.2) for the analysis of system stability without the need to explicitly solve all relevant equations [163] to obtain such a guarantee.

When a formation is created, the UAVs have to move from $q_{\mathcal{I}}$ to the desired $q_{\mathcal{G}}$ while they have to simultaneously preserve the formation shape. Three basic approaches to solve this task exist:

1. The *leader-follower approach*, where one UAV takes the part of a leader and the other UAVs follow this designated leader.

2. The *virtual leader approach*, where the entire formation is treated as a single virtual UAV and follows a moving point.

3. The *behavioral approach*, where each UAV has several desired behaviors, including formation preserving, target seeking, collision/obstacle avoidance, etc.

Each of these methodologies has its advantages and disadvantages. The most frequently used approach is the leader-follower approach. The advantage of this approach is the possibility to convert the problem of maintaining a formation to a standard tracking problem if information about the leader is available. One big disadvantage is missing robustness with respect to a leader's possible failure.

Formation handling based on the virtual leader approach is easier to achieve than it would be with other approaches as the complete group is treated as a single object. One disadvantage is that the formation is only able to perform synchronized maneuvers leading to a more challenging obstacle avoidance.

The most dynamic approach for UAV formation flights is the behavioral one. The UAVs need the capabilities to reach consensus about where and how a formation can be established in order to work completely decentralized. Areas that the formation has to explore next must be identified by the UAVs themselves. Each UAV $\mathcal{U}_i$ computes its own path $\rho_{\mathcal{U}_i}$ in respect to the formation and the terrain properties, like obstacles. Furthermore, each path of a UAV $\rho_{\mathcal{U}_i}$ depends on all other UAVs' positions, independent of their formation affiliation if a formation is established.

During exploratory navigation it is not always beneficial to use all UAVs forming only one single formation. Due to this, the formation approach should be a possible extension of the previously described three-dimensional potential field approach, used by each single $\mathcal{U}_i$ to explore environments. For path planning of UAVs participating in a formation the three-dimensional potential field can be extended to a four-dimensional potential field.

This extension is possible through bifurcation theory. Potential values computed using bifurcation theory can simply be added to the previously calculated potential field. One advantage of a potential field based on bifurcation theory is that the resulting formation shape can be changed by simply adjusting single parameters of the underlying equation, resulting in the potential values. It is also possible to show that global equilibria, stable in the sense of Lyapunov are created every single time. Path planning formation flights are still possible using the descent gradient. Thus, bifurcation theory, presented in the next section has been used to create UAV formations.

3.4.2. Bifurcation Theory

Bifurcation theory is part of catastrophe theory which again is part of chaos theory. In catastrophe theory a change of a function parameter leads from stable solutions to a point where no stable solution exist. The change occurs at the so called bifurcation point of a function. The system changes at this point from order to chaos. One famous example for such a function is the cusp catastrophe [164]. It is defined as $V = x^4 + \gamma x^2 + \delta x$ with γ and δ being the parameters of the parameter space. The equilibrium surface of a two-dimensional cusp catastrophe is shown in Figure 3.5. It is based on the canonical form of the potential function for the cusp catastrophe, defined as

$$-V(x; a, b) = ax + \frac{1}{2}bx^2 - \frac{1}{4}x^4. \tag{3.11}$$

The equilibrium points of Equation 3.11, given as a function of the control parameters a and b are solutions of Equation 3.12.

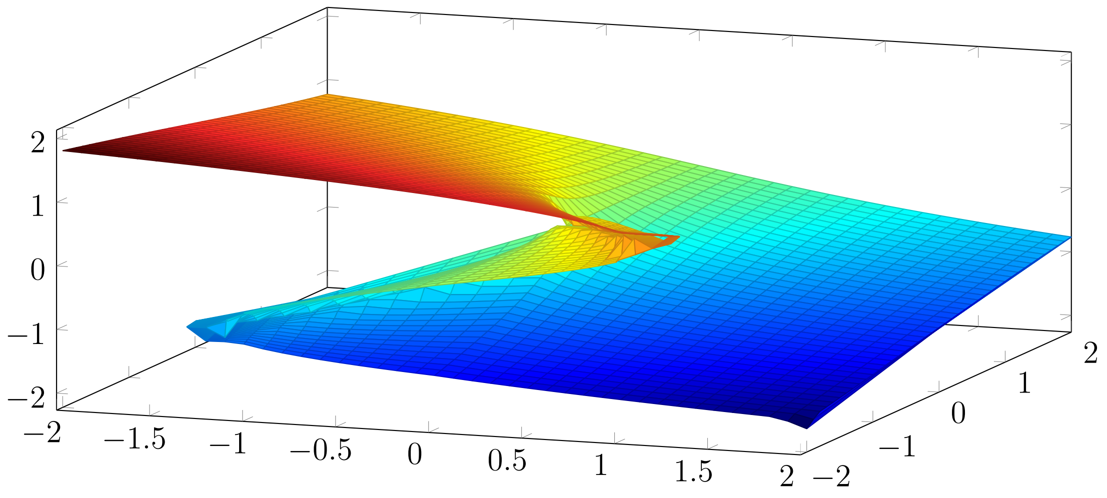


Figure 3.5.: *Schematic illustration of the Cusp equilibrium surface*

$$0 = a + bx - x^3 \tag{3.12}$$

Equation 3.12 leads to a single solution if $\delta = 27a - 4b^3$—known as *Cardan's discriminant*—is greater than zero. If $\delta < 0$ three solutions, illustrated in Figure 3.5 exist [165]. The floor of the figure is given by the two-dimensional (a, b) coordinate system. The presented cusp equilibrium surface can be considered as some sort of response surface, whereby the height predicts the values of x, given the values for the control variables $a, b \in [-2; 2]$.

A cusp catastrophe is common if one considers a fold bifurcation extended by a second parameter δ. If a parameter change takes place, a curve of points (γ, δ) is created, in which stability is lost. Compared to one equilibrium, now two equilibria exist to which the system can jump. For periodically changing δ a jump back and forth is generated. This works only for $\gamma < 0$. The closer γ becomes to zero the smaller the hysteresis curves become, until they vanish for $\gamma = 0$.

If δ is set to be constant and γ changes a Pitchfork bifurcation [164] is observed in the symmetrical case $\delta = 0$. If γ decreases one stable equilibrium changes into two stable and one unstable equilibria at the point where the system passes $(\gamma, \delta) = (0, 0)$. This is an example for a violation of symmetry. The abrupt change vanishes far away from the cusp point and only a second possible solution appears. The described changes, regarding the number of equilibria can be used to design a potential field for formations, in which the shape of the formation changes if the number of equilibria changes.

Harmonic functions used for path planning have to be strictly increasing or strictly decreasing with the length of the path to the equilibrium. Compared to that, bifurcation functions lead to a change of number of equilibria if the bifurcation parameter changes. This allows the creation of multiple global equilibria. Dynamic bifurcation theory concerns changes occurring in the structure of limited sets of solutions of differential equations when parameters are varied in a vector field [166].

Related to [35] bifurcation theory attempts to explain several phenomena, discovered and described in science over centuries. All with a common cause: A specific physical parameter crosses a threshold and this event forces the system into a new state, differing substantially from the preceding. Bifurcation is a parameter of non uniqueness in nonlinear systems. Popular local bifurcations are Hopf bifurcation, Saddle-node bifurcation, Transcritical bifurcation and Pitchfork bifurcation [164].

Given a dynamic system, a bifurcation takes place when a minor, smooth change made to a system's bifurcation parameters causes a sudden qualitative or topological change in its behavior. A popular example is an upright standing stick fixed at the bottom and equipped with a weight μ at its head. The angular deviation is denoted by υ.

As long as the weight is small enough, $\upsilon = 0$ is a stable and balanced condition, i. e., for small deviations the stick adjusts itself back to its upright position $\upsilon = 0$. The balanced condition becomes unstable at a given weight μ if it is continuously increased. Simultaneously, two new balanced conditions evolve as the stick bends down to the left or to the right. The change from one stable to two stable and one unstable balanced conditions is called bifurcation. In this example it is a Pitchfork bifurcation.

In general, the local stability properties of equilibria or other invariant sets change at a bifurcation. Figure 3.6 shows an example using the two-dimensional bifurcation function $\mu|x + b \cdot y| + \alpha|x^3 + b \cdot y^3|$ for $x, y \in [-1; 1], b = 0$. The number of stable equilibria changes due to the change of the bifurcation parameter μ from one to two if $\mu \geq 0$ changes to $\mu < 0$, as presented in Figure 3.6.

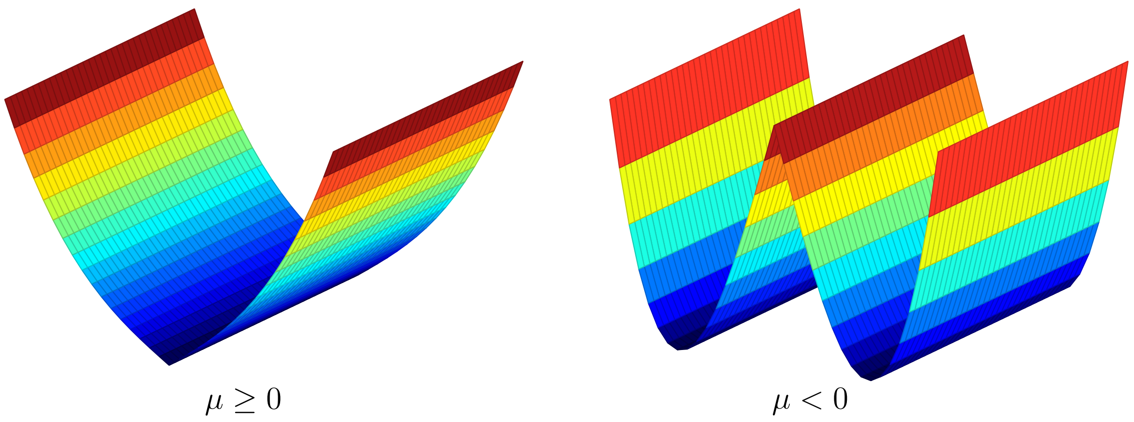


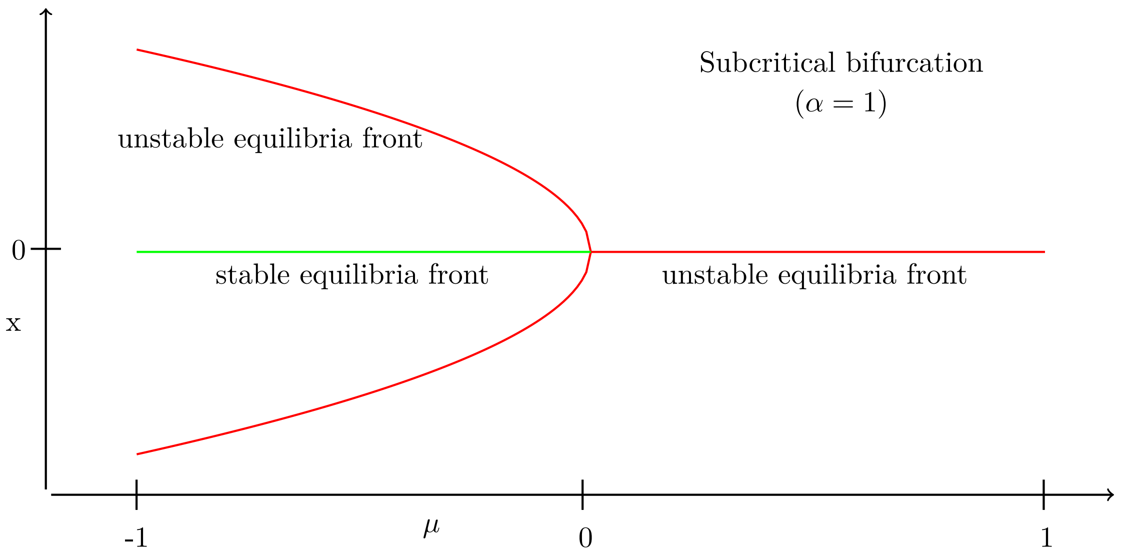
Figure 3.6.: *Change of the number of stable equilibria, resulting from a change of the bifurcation parameter μ.*

The bifurcation function finally used by the resulting UAV system for the creation of dynamically changing formation shapes (cf. Chapter 6) is based on the Pitchfork bifurcation. This bifurcation is a definite type of a bifurcation of a nonlinear system with normal form

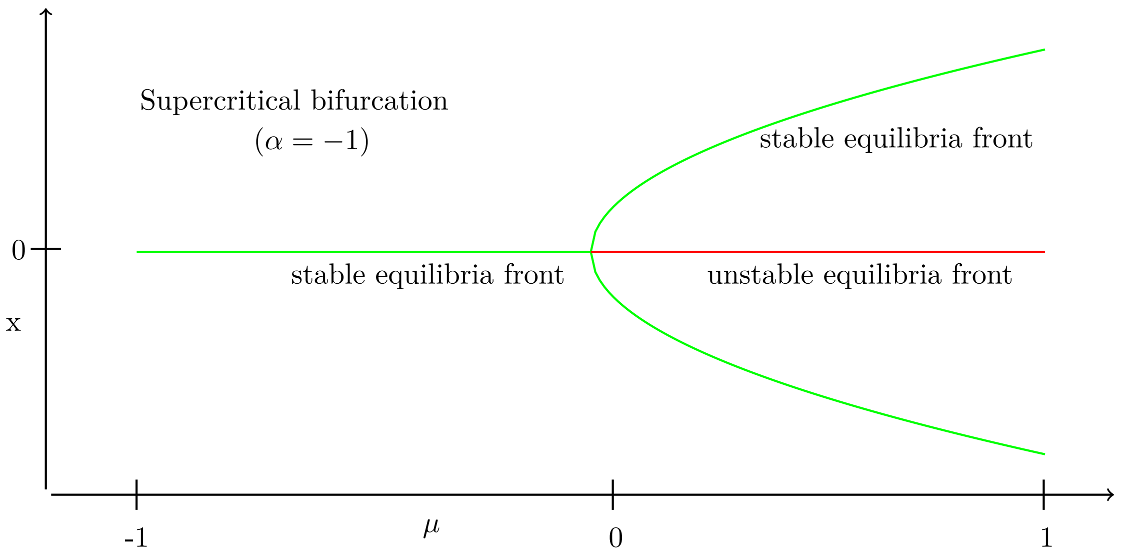
$$\dot{x} = \frac{dx}{dt} = \mu x + \alpha x^3, \tag{3.13}$$

whereby μ is the bifurcation parameter and α can be used to change the bifurcation from a subcritical bifurcation ($\alpha > 0$) to a supercritical Pitchfork bifurcation ($\alpha < 0$), as shown in Figure 3.7 with $\alpha = \pm 1$. A bifurcation is called subcritical when the bifurcating periodic or quasi-periodic solution is unstable. If the periodic or quasi periodic solution is stable, the bifurcation is called supercritical.

Figure 3.7 illustrates the equilibria points computed using Equation 3.13 with given $\mu \in [-1; 1]$ at the x positions. The green lines represent the stable equilibrium positions and the red lines represent the unstable equilibrium positions. These equilibria can be computed as follows: Considering the subcritical case presented in Figure 3.7 a) for $\mu < 0$, $x = 0$ is a stable equilibrium and $x = \pm\sqrt{-\mu}$ are unstable equilibria. If $\mu \geq 0$ is set the equilibrium at $x = 0$ is unstable. For a supercritical Pitchfork bifurcation shown in Figure 3.7 b) with $\mu \leq 0$ one stable equilibrium exists at $x = 0$. If $\mu > 0$ is set, one unstable equilibrium at $x = 0$ and two stable equilibria at $x = \pm\sqrt{\mu}$ exist.



a)



b)

Figure 3.7.: *Change of the bifurcation from a subcritical a) to a supercritical b) Pitchfork bifurcation by changing α. Stable equilibria are represented by green lines and unstable equilibria are represented by red lines. For the subcritical case the stable equilibrium is at $x = 0$ and $x = \pm\sqrt{-\mu}$ are unstable equilibria if $\mu < 0$. $x = 0$ is an unstable equilibrium if $\mu \geq 0$. For the supercritical case $x = 0$ is a stable equilibrium if $\mu \leq 0$. $x = 0$ is unstable and $x = \pm\sqrt{\mu}$ is stable if $\mu > 0$.*

It is possible to design an equilibrium at the same position as the global equilibrium received by a harmonic function (cf. Section 3.3.1). This allows the creation of a superpositioned equilibrium leading the UAVs towards it with the advantages of the harmonic potential field by simultaneously providing a formation potential field forcing the UAVs to create a formation with the equilibrium position as the center point.

It is also possible to create equilibria that are able to overwrite the one from the harmonic function. This allows the change of a formation with a single equilibrium point to a formation with several equilibrium points. The equilibrium of the harmonic potential field will then be between the equilibria of the bifurcating potential field and thus still leads the UAVs towards the equilibrium points.

Bifurcation theory allows an establishment and change of equilibria by simple parameter adaptation. Thus, it is possible to change the positions of the UAVs in such a way that they create formations as presented in Figure 3.6. The case that $\mu \geq 0$ would lead to a line formation and the case that $\mu < 0$ would lead to a double line formation.

Bennet and McInnes [167] have shown that the establishment of a formation using bifurcation theory can be verified by proving system stability in the sense of Lyapunov. They have also shown that a multi UAV system still works if the UAVs only have information about their neighboring UAVs. This is important as it is unrealistic for many real world applications' UAVs to always communicate with all others.

The foundations presented in this chapter so far can be used to create a multi-UAV system, able to compute paths, to explore terrains, and to establish dynamically changeable formations. The designed system, based on the potential field approach is scalable, flexible and robust. These properties are considered to be important for an autonomous multi-UAV system, as described in Chapter 1. The use of bifurcation theory creates a flexible formation system that allows to change between different formation shapes anytime by simple parameter variations to command an entire formation.

Exploring terrains using only a single formation does not seem to be very beneficial. Increasing efficiency is possible if the UAVs can create several formations simultaneously and if they are able to process different tasks. It should also be possible to explore small subareas like, e. g., rooms in buildings using only one UAV, while other UAVs simultaneously fly in formation to monitor large areas. To provide the resulting system with these abilities, a cooperative behavior including task allocation is needed. The basics for the resulting system behavior are presented in the following section.

3.5. Cooperative Behavior

To reach an increase in efficiency of the overall performance, a support of different kinds of UAV coordination approaches can be utilized. Redundant exploration of areas, as well as redundant tasks processing should be avoided. Additionally, the UAVs have to work cooperatively and consensus considering different decisions must be reached. An establishment of formations should also be performed if it leads to an increase in efficiency.

To achieve such a coordinated behavior, several requirements must be met. The UAVs have to exchange information and they need abilities for task creation and task allocation. This section introduces the basic methodologies the UAVs use to achieve the desired coordination. It is subdivided into two subsections, one explaining the basics and requirements for task creation and the other explaining the foundations and requirements for task allocation.

3.5.1. Task Creation

One part of the behavior the UAVs rely on is task creation. They should create tasks by themselves if these tasks lead to an increase in efficiency or if they avoid damage. If it seems, e. g., beneficial to explore some areas of the environment in formation, at least one UAV should notice this and start the creation of a task for exploration in formation. Task creation is based on environmental information. The more information exists, the more efficient task creation is possible.

Another point is cooperative obstacle avoidance. If, e. g., a formation flies into a valley and the UAVs can fly inside this valley only one after another, a task to dissolve the formation should be created. Additionally, if a formation has to surround, e. g., a mountain, it can be beneficial if the formation is divided into two, with one flying, e. g., counterclockwise and the other clockwise around the mountain.

The most important point is that tasks have to be created by the UAVs themselves if the current UAV configuration will lead to future damage. Such configurations are given when the connection to the other UAVs fails or if the fuel level of a UAV reaches a critical quantity. In such cases, new tasks must be created and the corresponding UAVs have to process these tasks with high priority before all other tasks.

After the creation of tasks in order to increase efficiency and to avoid damage, an allocation and execution of these tasks must take place. Therefore, the following section introduces to the field of multi-robot task allocation.

3.5.2. Multi-Robot Task Allocation

A wide range of methodologies for multi-robot task allocation exist. Many task allocation approaches are based on the division of labor of natural swarms (cf. Section 2.1.4). Division of labor can be achieved, e. g., by introducing a role system. Other approaches consider homogeneous robots and centralized, as well as decentralized task distribution (cf. Section 2.2.2).

To achieve an efficient task allocation, it is necessary to define the term efficiency. It can be defined based on one, as well as on several objectives the task allocator has to optimize. The better the resulting optimization is, the more efficient the system becomes. Some widely used objectives to optimize task allocation exist. Mosteo and Montano [168] summarize several of them as follows:

- Minimize the costs of the worst agent.
- Maximize the utility of the worst agent.
- Minimize the sum of individual costs.
- Maximize the sum of indirect utilities.
- Minimize the average costs per task.
- Maximize the overall performance.

It has been shown [169] that the distributed task allocation problem is NP-complete if the goal is to minimize the time needed to process all tasks. The problem can be mapped for a single UAV to the Traveling Salesman Problem (TSP). If several UAVs are involved, it can be mapped to multiple TSP.

Researchers work on different approaches for multi robot task allocation. An example for a division of labor based task allocation approach, where a simple reinforcement of response thresholds is transformed into a decentralized adaptive algorithm for task allocation, is presented by Benabeau et al. [38]. Furthermore, Gerkey and Mataric, e. g., analyzed several approaches [170] maximizing the overall performance.

The method used by the UAVs for distributed task allocation is based on the market place approach (cf. Section 2.2.2). To establish a basic market place, three types of robots or agents are normally needed. One kind are selling agents. These sellers can be robots or third party applications, which require specific tasks to be performed. The remaining robots are then considered as buying agents. Based on the optimization objectives the buying robots calculate offers and place them. Additionally, a third entity exists. The auctioneer takes these offers and is responsible for comparing them and for determining a winning agent. This winning agent will finally execute the task.

A market place as it has been introduced first is a centralized approach with one central auctioneer. The approach has been extended by the introduction of division of labor (cf. Section 7.3.2) to support heterogeneous robot groups, placing bids, based on their capabilities. To overcome the disadvantages of centralized approaches, such as bottlenecks and single point of failures, the market place approach has also been decentralized. This decentralization has been obtained through the distribution of the decision process over all UAVs in order to replace the auctioneer (cf. Section 7.4.2).

Compared to many other task allocation approaches, e. g., [171], [172], [173], the resulting system has not only one of the aforementioned values to optimize. Additionally, each task has diverse properties the UAVs have to take into account (cf. Section 7.3.1). The execution of some tasks is more important than the execution of other tasks in order to avoid damage. Additionally, some tasks require several UAVs to be performed. Due to their equipment, certain UAVs fit differently to certain types of tasks. These additional requirements (cf. Section 7.3.3) must be met by the system described in this thesis. The UAVs have, e. g., only limited fuel and it has to be guaranteed that they never run out of it. To take these additional requirements into account, some tasks must be more beneficial for the UAVs than others.

An increase of the gain can be obtained by weighting the single tasks using priorities. Based on the priority, a task becomes more or less beneficial to a UAV. Such a way of weighting is also used by operating systems to distribute and sort the single tasks in queues for the available CPUs as, e. g., shown by Baumberger et al. [174].

The foundations presented so far allow the design of a decentralized multi-UAV system with abilities for environmental exploration, task allocation, flights in formation, etc. In adopting these approaches to real world applications a first step is to ensure that the designed system is verifiable. Several methods to verify software and for mathematical algorithms exist. The following section presents the basics one can use to verify the resulting system.

3.6. Verification

A multi-agent system for real world environments must guarantee that the behavior of the single agents and the behavior of the overall system works as expected. This is necessary as unpredictable behavior can lead to crashes and several other dangerous conditions endangering the safety of human beings. To avoid unexpected behavior, a system has at least to be verified in the sense that the system always behaves in the way specified.

Typically, mathematical verification is used for system validation. Using potential field theory, a verification can be obtained by showing that the system always relaxes into stable equilibria. The positions of these stable equilibria result from the desired target configurations and are equal to the positions of them. Dynamic system theory can be used to ensure the stability of the resulting system by showing that conditions stable in the sense of Lyapunov and explained in the following section are always reached. A system always relaxes into an equilibrium if the it is globally asymptotically stable. Lyapunov's methods extended by the LaSalle theorem can be used to prove this behavior.

3.6.1. Lyapunov Stability

Lyapunov stability is given if a small change of forces does not lead to new conditions. A common example is a pendulum with a rod and a weight. It has a stable and an unstable equilibrium in the sense of Lyapunov. The stable equilibrium is reached if the weight is hanging straight down. It always relaxes into this equilibrium after affecting forces have vanished. The unstable equilibrium is reached if the weight is pointing straight up. The pendulum will not necessarily relax into this position after an affecting force has vanished. Lyapunov stability is defined, e. g., in [175] and additionally in [176] by Lyapunov. A detailed definition is given by Shankar Sastry et al. [177] in the context of robot manipulation.

Following the definition from [177], a dynamical system which satisfies

$$\dot{x} = f(x,t), \quad x(t_0) = x_0, \quad x_0 \in \mathbb{R}^n \tag{3.14}$$

is considered. Let $x^* \in \mathbb{R}^n$ be a position, while t denotes the time. The position x^* is an equilibrium of Equation 3.14 if $f(x^*,t) \equiv 0$. The equilibrium is called locally stable if all solutions with initial conditions in the neighborhood of x^* remain near x^* for all times. Additionally, x^* is called locally asymptotically stable if x^* is locally stable and all solutions starting near x^* tend towards zero when t goes towards infinite.

It can be assumed that the equilibrium of interest occurs at $x^* = 0$. This can be reached by shifting the origin of the system [177]. An appropriate shifting of the origin can be performed for each equilibrium point in order to study the stability of several equilibrium points. Then, stability in the sense of Lyapunov is defined as follows:

Definition 3.7. Lyapunov stable: *An equilibrium position $x^* = 0$ of Equation 3.14 is said to be Lyapunov stable at $t = t_0$ if*

$$\forall \epsilon > 0 \ \exists \delta(t_0, \epsilon) > 0 \textit{ such that } \forall t \geq t_0 \colon \|x(t_0)\| < \delta \Rightarrow \|x(t)\| < \epsilon. \tag{3.15}$$

Equation 3.15 shows that a solution starting within a distance δ to the equilibrium always remains within a distance ϵ. This must be true for all $\epsilon > 0$. Trajectories starting close to the origin are not obliged to tend to the origin asymptotically following Definition 3.7. Additionally, the stability of the system is only defined at a time instant t_0. It is necessary to guarantee that the equilibrium position does not lose stability in order to eliminate unexpected behavior. Therefore, the concept of uniform stability is introduced in the following.

A uniformly stable equilibrium x^* with a given δ from Definition 3.7 is not a function of t_0. To the contrary, it must be ensured that Equation 3.15 holds true for all t_0. Furthermore, asymptotic stability is introduced in the following to define uniform asymptotic stability.

Definition 3.8. Asymptotic stable: *Given an equilibrium $x^* = 0$ from the system of Equation 3.14. It is called asymptotically stable at $t = t_0$ if*

1. *$x^* = 0$ is stable and*
2. *$x^* = 0$ is locally attractive. This means that*

$$\exists \delta(t_0) > 0 \colon \|x(t_0)\| < \delta \Rightarrow \lim_{t \to \infty} x(t) = 0. \tag{3.16}$$

Based on Definition 3.8, uniform asymptotic stability is defined as follows:

Definition 3.9. Uniform asymptotic stable: *A system is called uniform asymptotic stable if*

- *$x^* = 0$ is uniformly stable and*
- *$x^* = 0$ is uniformly locally attractive, i. e., that $\exists \delta$ independent of t_0 for which Equation 3.16 holds. Additionally, convergence in Equation 3.16 must be uniform.*

Definitions 3.7, 3.8, and 3.9 are only local definitions, i. e., they describe the system behavior close to an equilibrium. An equilibrium x^* is called globally stable if it is stable for all initial conditions $x_0 \in \mathbb{R}^n$ [177].

Lyapunov introduced two methods to demonstrate stability. These are the direct and the indirect method. Additionally, the direct method has been extended by LaSalle to show asymptotic stability under extended conditions. All methods are described in the following sections starting with Lyapunov's direct method.

3.6.2. Lyapunov's Direct Method

The direct method (also known as second method) allows the determination of stability of a system without explicitly integrating the Differential equation 3.14 [177]. The method generalizes the idea of it beings possible to study the rate of energy change to ascertain stability if a "measure of energy" exists in a system. A "measure of energy" is thereby defined as follows. Given a ball B_ϵ of size $\epsilon > 0$ around the origin, $B_\epsilon = \{x \in \mathbb{R}^n \colon \|x\| < \epsilon\}$.

Definition 3.10. Locally positive definite: *A continuous function $V \colon \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}$ is called a locally positive definite function if for $\epsilon > 0$ and for a continuous, strictly increasing function $\alpha \colon \mathbb{R}^+ \to \mathbb{R}$,*

$$\forall x \in B_\epsilon, \forall t \geq 0 \colon V(0,t) = 0 \text{ and } V(x,t) \geq \alpha(\|x\|). \tag{3.17}$$

Functions defined by Definition 3.10 are locally like energy functions. Analogous to this definition, positive definite functions are globally like energy functions.

Definition 3.11. Positive definite: *A positive definite function* $V\colon \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}$ *is given if it satisfies the conditions of Definition 3.10 and*

$$\lim_{p \to \infty} \alpha(p) = \infty. \tag{3.18}$$

Additionally, so called decrescent functions exist in order to bound the energy.

Definition 3.12. Decrescent: *A continuous function* $V\colon \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}$ *is called decrescent function if for* $\epsilon > 0$ *and for a continuous, strictly increasing function* $\beta\colon \mathbb{R}^+ \to \mathbb{R}$*,*

$$\forall x \in B_\epsilon, \forall t \geq 0\colon V(x,t) \leq \beta(\|x\|). \tag{3.19}$$

Using the previous definition it is possible to determine stability for a system by studying an appropriate energy function [177]. Stability of the equilibrium point can be concluded by the basic theorem of Lyapunov if $V(x,t)$ is a locally positive definite function and $\dot{V}(x,t) \leq 0$.

Theorem 3.1. *Let* $V(x,t)$ *be a non-negative function with derivative* $\dot{V}|_{\dot{x}=f(x,t)} = \frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} f$ *along the trajectories of the system.*

- *The origin of a system is locally stable in the sense of Lyapunov if* $V(x,t)$ *is locally positive definite and* $\dot{V}(x,t) \leq 0$ *is locally in* x *and for all* t*.*
- *The origin of a system is locally uniformly stable in the sense of Lyapunov if* $V(x,t)$ *is locally positive definite and decrescent. Additionally,* $\dot{V}(x,t) \leq 0$ *is locally in* x *and for all* t*.*
- *The origin of a system is locally uniformly asymptotically stable if* $V(x,t)$ *is locally positive definite and decrescent. Additionally,* $-\dot{V}(x,t)$ *is locally positive definite.*
- *The origin of a system is globally uniformly asymptotically stable if* $V(x,t)$ *is positive definite and decrescent. Additionally,* $-\dot{V}(x,t)$ *is positive definite.*

What Lyapunov has realized is that it is possible to prove stability without the requirement to know the real physical energy if a Lyapunov function can be found to satisfy the above constraints. The global asymptotic stability of the origin is a consequence if $\dot{V}(x,t)$ is negative definite. This is often not the case but LaSalle has introduced a criterion for asymptotic stability if $\dot{V}(x,t)$ is only negative semidefinite as described in the further course of the chapter.

3.6.3. LaSalle Theorem

Lyapunov's direct method has been extended by LaSalle [178]. The resulting theorem allows the conclusion of asymptotic stability even if $-\dot{V}(x,t)$ is not locally positive definite. It can only be applied to autonomous or periodic systems. Let the solution trajectories of an autonomous system be

$$\dot{x} = f(x). \tag{3.20}$$

Further on let $\nu(t, x_0, t_0)$ be the solution of Equation 3.20 at time t. The trajectories start from x_0 at time t_0. Then, a ω limit set is defined as follows:

Definition 3.13. Limit set: *A set $S \subset \mathbb{R}^n$ is the ω limit set of a trajectory $\nu(\cdot, x_0, t_0)$ if a strictly increasing sequence of times t_n for every $y \in S$ exists such that*

$$\lim_{t_n \to \infty} \nu(t_n, x_0, t_0) = y. \tag{3.21}$$

Additionally, an invariant set is defined as follows:

Definition 3.14. Positive invariant: *A set $M \subset \mathbb{R}^n$ is called a positive invariant set if*

$$\forall y \in M, \forall t_0 \geq 0, \forall t \geq t_0 \colon \nu(t, y, t_0) \in M. \tag{3.22}$$

The ω limit set of every trajectory is closed and invariant [177]. Based on the previous definitions, LaSalle's theorem can now be stated.

Theorem 3.2. *Let a function $V \colon \mathbb{R}^n \to \mathbb{R}$ be a positive definite function. For this function $\dot{V}(x) \leq 0$ must hold on the compact set $\Omega_c = \{x \in \mathbb{R}^n \colon V(x) \leq c\}$. Further on define*

$$S = \{x \in \Omega_c | \dot{V}(x) = 0\}. \tag{3.23}$$

The trajectory tends to the largest invariant set inside S for $t \to \infty$, i. e., its ω limit set is included inside the largest invariant set in S. This means that if no invariant set other than $x = 0$ is contained in S, then 0 is asymptotically stable.

A global version of Theorem 3.2 may also be stated. Based on the sufficient conditions given by Theorem 3.1 it is possible to verify the system by showing that always a globally uniformly asymptotic equilibrium exists. The system always relaxes into this equilibrium if the origin of the system is uniformly globally asymptotically stable and then the behavior is always as expected. A disadvantage of the direct method is that no computable technique for the generation of Lyapunov functions exists. Another possibility to show that a system is stable is given by the indirect method of Lyapunov described in the next section.

3.6.4. Lyapunov's Indirect Method

Lyapunov's indirect method (also known as first method) uses the linearization of a system in order to determine stability of the origin [177]. Given the system

$$\dot{x} = f(x, t) \tag{3.24}$$

whereby $\forall t \geq 0\colon f(0, t) = 0$. Additionally, let

$$J(t) = \left.\frac{\partial f(x, t)}{\partial x}\right|_{x=0} \tag{3.25}$$

be the Jacobian matrix of $f(x, t)$ with respect to x and evaluated at the origin. From this it follows that for each fixed t, the remainder

$$f_1(x, t) = f(x, t) - J(t)x \tag{3.26}$$

approaches zero as x approaches zero. From this it is not given that the remainder approaches zero uniformly. Therefore, condition

$$\lim_{\|x\| \to 0} \sup_{t \geq 0} \frac{\|f_1(x, t)\|}{\|x\|} = 0 \tag{3.27}$$

is required. The system

$$\dot{z} = J(t)z \tag{3.28}$$

is referred to as the uniform linearization of Equation 3.14 about the origin if Equation 3.27 holds. The stability of the linearization determines the local stability of the original nonlinear equation if a linearization exists [177]. From this the theorem of stability by linearization can be stated as follows:

Theorem 3.3. *Assume that Equation 3.27 holds true for the system from Equation 3.24. Additionally, let $J(\cdot)$ from Equation 3.25 be bounded. Then, zero is a locally uniformly asymptotically stable equilibrium of Equation 3.24 if it is a uniformly asymptotically stable equilibrium of Equation 3.28 [177].*

Uniform asymptotic stability of the linearized system is required by Theorem 3.3 for the proof of uniform asymptotic stability of the nonlinear system. Let the system given in Equation 3.24 be time-invariant. Then Lyapunov's indirect method says that if the eigenvalues of

$$J = \left. \frac{\partial f(x)}{\partial x} \right|_{x=0} \tag{3.29}$$

are in the left half of the complex plane, the origin is asymptotically stable. The prescribed theorem shows that local uniform asymptotic stability of the original nonlinear system is implied by global uniform asymptotic stability of the linearization.

3.7. Summary

The foundations needed to design a verifiable three-dimensional path planning system supporting formation flights are described in this chapter. Additionally, basics considering the design of a cooperative behavior for task allocation with the aim of increasing efficiency is presented.

Formations are established in order to increase efficiency in terms of decreasing the time needed for task execution. The entire system is based on artificial potential field theory. A world model is used by the UAVs based on a dynamic octree as underlying data structure. It affects the used path planning and formation flight methods. The affection concerns not only the design and implementation of the algorithms but also the resulting computation times needed to compute feasible paths. It is therefore the most significant design decision.

Artificial potential fields are computed using harmonic function theory for path planning and bifurcation theory for formation flights. The descent gradient of the resulting potential field is finally used for path computation.

Coordination approaches are necessary to achieve an operative multi-UAV system and thus introduced in this chapter. Inter-UAV communication is used for information exchange. It allows the UAVs to inform other UAVs about newly created tasks and to exchange all relevant information regarding their current configuration $q_{\mathcal{U}} \in \mathcal{C}$. Tasks have to be allocated by the UAVs without any central coordination instance. The basic techniques used to achieve a task creation and allocation methodology are also presented in this chapter.

The methods described in this chapter allow the behavior of the resulting system to be verified using dynamic system theory. Verification is a first step for the use in real world scenarios where validated software is needed.

The following chapter introduces the overall system. It describes the interactions of the foundations introduced in this chapter to design a system able to plan efficient paths in a computable fashion. One focus lies on the data structure used for environmental representation. Additionally, inter-UAV communication and the final UAV configuration is described in detail. The exploration approach based on the camera equipment of the UAVs is also presented.

Chapter 4

Architectural System Design

The methods introduced in the previous chapter can be extended and combined to design a multi-UAV system with capabilities for distributed task allocation, path planning and formation flights. The UAVs presented in this thesis are able to behave completely autonomously and decentralized. A system with these abilities has several advantages and can address a wide range of applications without a central coordination instance no bottleneck and no single point of failure exists. It is also able to work in a safe and efficient way even in situations wherein user interaction is impossible.

The overall architecture and the details of the system are described within the following chapters. This chapter provides an overview of the entire approach and its architecture, as well as a description of the underlying data structure used to represent three-dimensional terrains. All algorithms explained in the following chapters, make use of this data structure. It directly influences the performance and abilities of the overall system. The UAV configuration is described in the further course of this chapter. The algorithmic used to compute newly explored terrain is also presented. To reach a coordinated and decentralized UAV behavior, information exchange between the UAVs is necessary. The technique used by the UAVs to communicate with each other is also introduced in this chapter. It finally concludes with a summary.

The following chapter deals with the algorithmic for path planning and exploratory navigation (cf. Chapter 5). Formation flights are presented in Chapter 6. All algorithms consider complex and dynamic three-dimensional terrains. Chapter 7 describes the resulting task creation and task allocation system to achieve a cooperative UAV behavior.

The methods of the resulting system can be mathematically verified. Such a verification proofs that the methods always lead to the desired behavior. It is shown, that the selected target configurations q_G are always stable in the sense of Lyapunov. From this, it follows that the system will relax into the desired q_G, which ensures that the UAVs are able to reach their targets.

An implementation has been carried out to show that the system works with discrete processor arithmetic and not only considers a continuous mathematical space. Several tests have been conducted (cf. Chapter 8) using the implementation. In order to reduce the computational effort of the approach, various approximations have been made. They are also introduced in the following chapters while the next section gives an overview of the entire system.

4.1. System Overview

GNU g++-4.7 [179] has been chosen as the programming language. The implementation is unitized to reach exchangeability if, e. g., a different kind of UAV has to be tested. Figure 4.1 illustrates an overview of the entire system, consisting of three main parts:

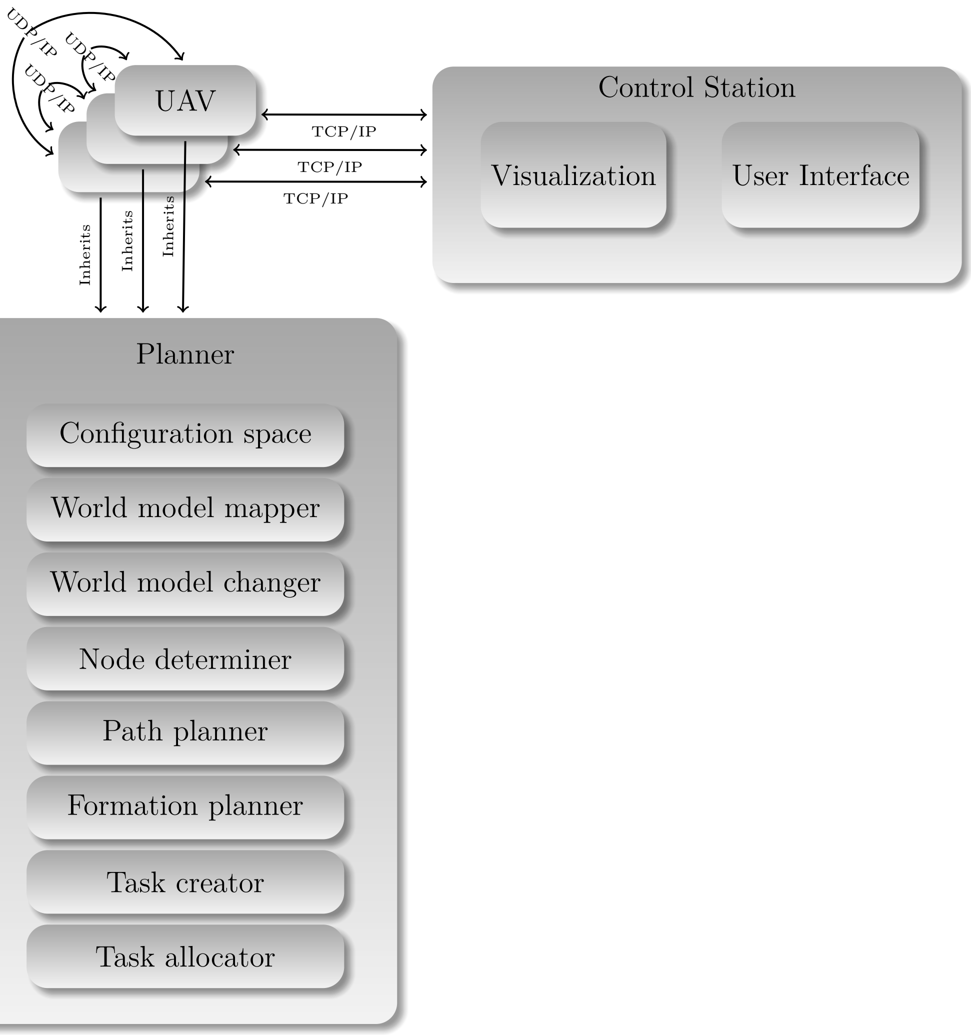


Figure 4.1.: *Overall architecture of the system. It consists of three parts. The programs* Control Station *and* UAV *and the library* Planner*.*

1. A *Control Station*, responsible for the visualization of the current system state. It also allows user interaction to create new obstacles or tasks and to pause the system if necessary.
2. A program *UAV* representing a single UAV. It implements inter-UAV communication and simulates UAV movement.
3. A library *Planner*, responsible for the internal representation of $\mathcal{C}$. Additionally, all algorithms needed for path planning, formation handling, task allocation, etc. are included in this library.

The UAV capabilities and also the UAV behavior result from the algorithms implemented in the library *Planner*. Thus the main focus of the chapter lies on this library. Nevertheless, the following section briefly describes the *Control Station*. It is responsible for terrain creation and to provide the UAVs with environmental information.

4.1.1. World Creation and Modeling

As mentioned before, the UAVs have to work completely decentralized. In order to reach this objective the *Control Station* cannot influence the decision making processes of the UAVs. The main duties of the *Control Station* are the provision of environmental information to the UAVs and the representation of the current system state in order to support ground units with up-to-date information regarding the environment. Additionally, prevention of personal injury and collision avoidance of real UAVs make it necessary to conduct first tests of the UAV algorithms in a safe way. A simulation environment provides such a safe test bed.

The representation of different terrains and the visualization of the current system state are also supported by the *Control Station*. Partial information about previous system states are provided, too. These information include the visualization of all previous trajectories of each UAV. Beside the system state of the terrain the single UAVs are visualized at their positions and it is possible to display their current configurations.

To obtain information about specific areas of the environment, the implemented scene camera can be moved in 3 dimensions and rotated along 2 axes. The current simulation can be paused anytime to identify problems. It is also possible to create airfields that the UAVs use in an emergency to land.

Different environments including obstacles and targets can be visualized. Environments can be created and dynamically changed anytime during simulations and environmental changes will always be sent to the UAVs. A dynamically changing number of UAVs is also supported. It is possible to save the current information regarding the terrain. This allows to store the start parameter of test cases. Hence, the results obtained in Chapter 8 are easily reproducible.

The *Control Station* renders scenes using the rendering engine WildMagic5 [180]. Amongst other things it allows three-dimensional polygon rendering. Environments can be created using a combination of heightmaps [181], models generated with 3ds max [182] and self-made meshes. Heightmaps are usually monochrome pictures. Figure 4.2 shows an example heightmap usable to create an underlying terrain. The brighter the single pixels of the heightmap are, the higher the resulting area of the terrain is. The values $v \in \{0, 1, \ldots, 255\}$ of the single pixel combined with by a bilinear filtering [183] and multiplied by a scaling factor to create smooth terrains.

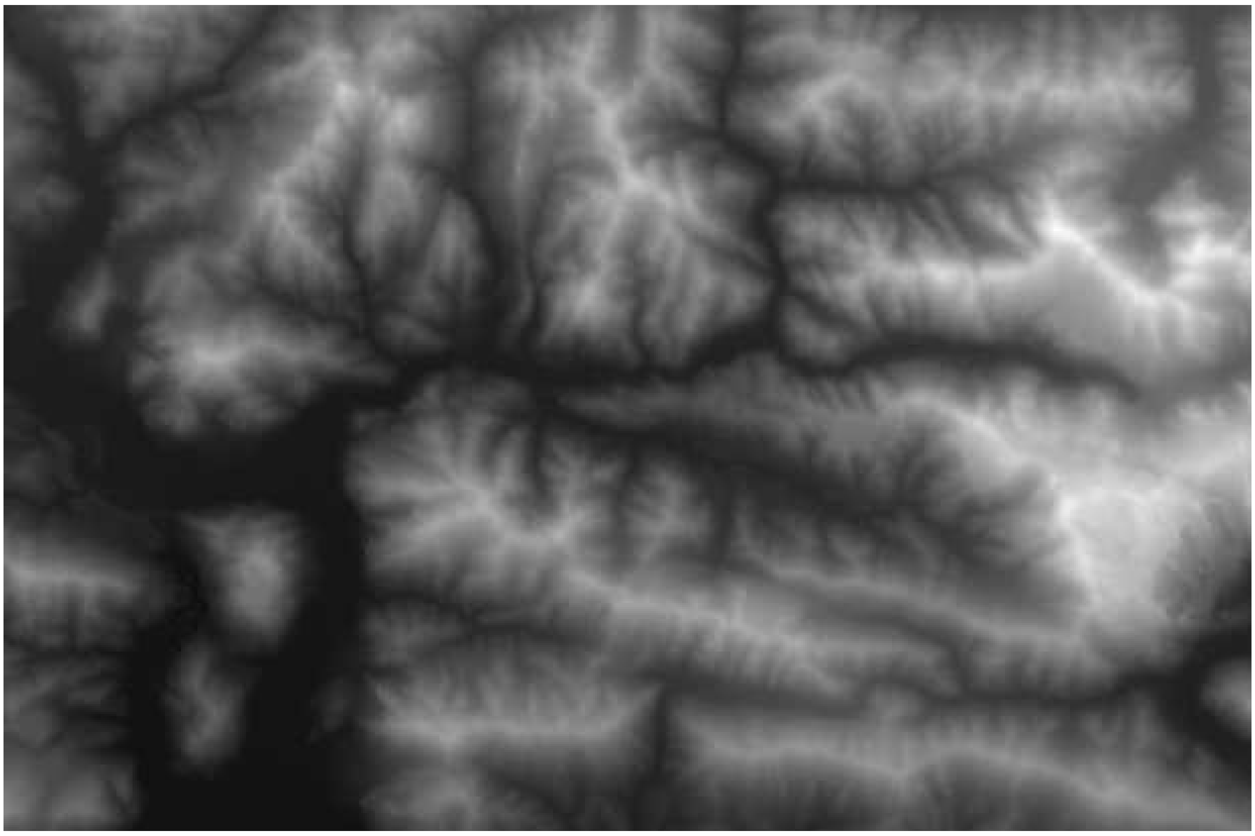


Figure 4.2.: *An example heightmap usable to create an underlying terrain.*

It is possible to create real terrains based, e. g., on maps offered by the NASA [184] or OpenStreetMap [185], which can be converted to heightmaps. 3ds max models are mainly used for the representation of complex structures and the UAVs. Self-made meshes can easily be created online during simulation, e. g., to test the response times the UAVs need to compute an appropriate reaction to newly detected obstacles. New tasks can be introduced to be able to consider, e. g., third party information about areas containing casualties with high probability. The UAVs decide on their own about the distribution and execution of given tasks.

A fictive terrain created and visualized by the *Control Station* is shown in Figure 4.3. The current state of the terrain is visualized in the center of the Program. The bottom shows the current configuration of a selected UAV including its role, current position, and angles, as well as its maximum speed, favorite altitude, and maximum altitude. The left hand side of the *Control Station* is used to control the current simulation. Targets and obstacles can be

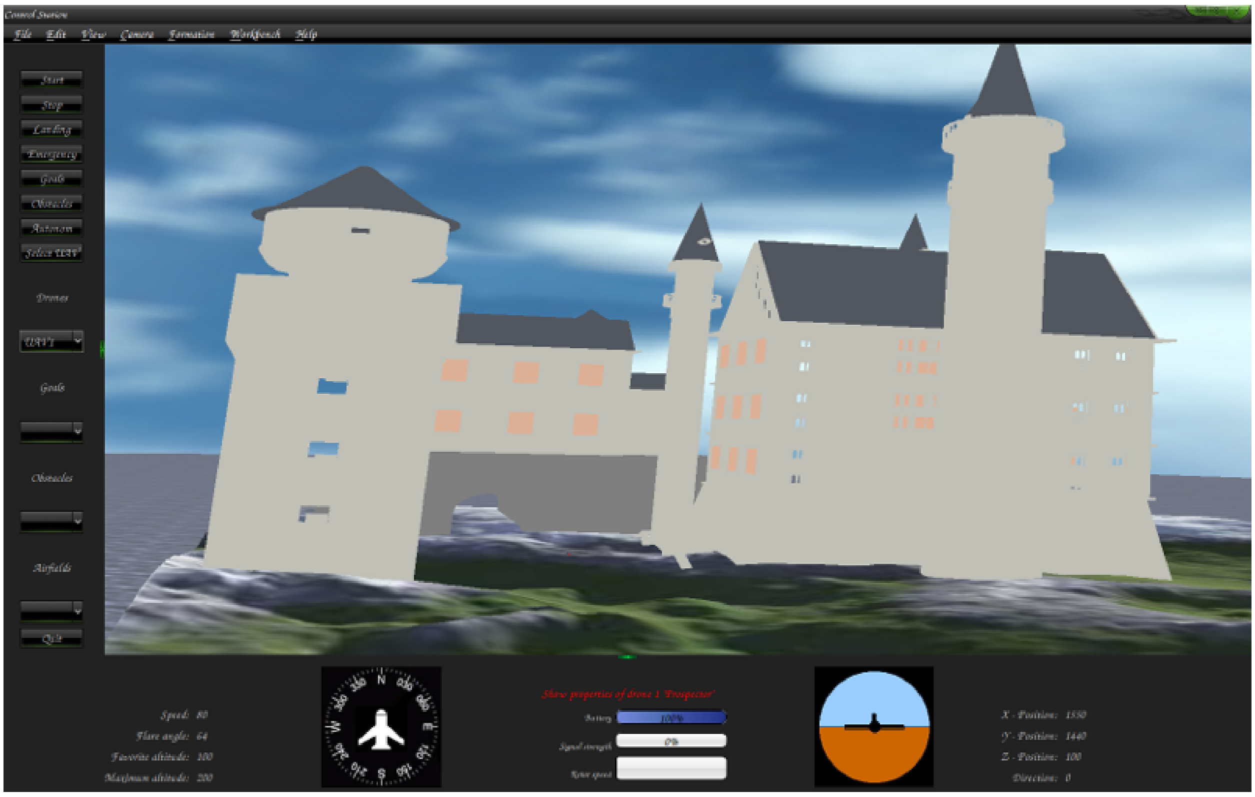


Figure 4.3.: *Visualization of a fictive terrain using the* Control Station.

defined and deleted and the simulation can be started, paused, and stopped. Additionally, all targets, obstacles and airfields set by human operators can be selected to show their properties. A list of all UAVs is also provided on the left hand side of the program. Further functionality, like the creation of terrains including models of buildings, etc. and enabling the visualization of current and previous UAV paths is provided by the menu.

Each object the environment consists of is internally represented as triangle mesh. A UAV receives all relevant triangle meshes every time new objects are detected, have changed, or when it connects itself to the *Control Station*. Based on this internal representation, it is possible to load and visualize a wide range of three-dimensional real world environments using triangulation algorithms as, e. g., the algorithm presented by Seidel [186].

The system's UAVs presented in this thesis establish inter-UAV communication by UDP/IP broadcasting, e. g., to reduce redundant exploration of areas. They also create a dedicated TCP/IP connection to the *Control Station*. When a UAV explores new areas of the environment, it sends information about the newly explored parts to all other UAVs and to the *Control Station*. The UAVs map all information they receive about newly explored areas into their own configuration spaces. The *Control Station* uses these information to update the visualization of the relevant areas. The UAVs compute their behavior based on the information they receive from other UAVs, regarding the current state and from the *Control Station*, regarding the environment.

UAVs are represented as own program inheriting the library *Planner*. This leads to an exchangeability of different kinds of UAVs. The program *UAV* is mainly responsible for providing communication sockets and to compute new UAV positions in order to simulate UAV flights. Therefore, it is neglected and only briefly described in the following section.

4.1.2. UAV Architecture

The program *UAV* is used for the representation of UAVs. It has been developed as a separate program and each instance of the program simulates one single UAV. Some of the main duties is inter-UAV communication and the establishment of a dedicated TCP/IP connection to the *Control Station*. These connections are used for information exchange. The program is also responsible for the simulation of UAV flights and to monitor the internal state. The internal state includes, e. g., the current fuel level. Monitoring the fuel level is necessary to avoid crashes. A threshold is introduced, which is described in detail in Section 7.4.1. It is based on the size of the environment, the velocity, and the tank capacity of the UAV. If the fuel level gets below this threshold, a new task is created sending the UAV to a gas station.

The most important part of the designed multi-UAV system are the algorithms and their interaction illustrated in Figure 4.4. They are responsible for behavioral computation and are completely implemented in the library *Planner*. Figure 4.4 shows an overview of the single parts of the behavior system including information exchange and simulated UAV flights implemented in the program *UAV*. The algorithms work cooperatively to achieve the objectives introduced in Section 1.2.

Another responsibility of the library is the internal representation of different environments. For this purpose a world model based on an octree is created. The configuration space $\mathcal{C}$ (cf. Section 3.1.2) is also part of the world model. Based on the configuration space, an exploration of the current environment can be conducted. The creation of formations and task procession is related to $\mathcal{C}$, too. The library also includes basic algorithms, like neighbor determination in octrees (cf. Appendix A.1), relevant for path planning and for potential field computations.

The octree is dynamically changeable in order to reflect changing up-to-date information of the environment, like newly explored areas and newly detected occupied areas. Furthermore, a task allocation and task creation system has been implemented to provide the UAVs with all necessary abilities to simultaneously execute different tasks.

The UAVs have to execute different algorithms continuously to reach the desired behavior. First, a task is selected as shown in Figure 4.4. Each task contains a target area. The octree represents this target area internally by assigning the leaves, which enclose the target area, to $\mathcal{C}_\mathcal{G}$. The UAV needs to fly to the target area in order to fulfill the selected task. A potential field represented by a harmonic function is computed for that reason (cf. Chapter 5). If the UAV flies in formation, an additional potential field based on bifurcation theory is calculated and superpositioned with the harmonic potential field as described in detail in Chapter 6.

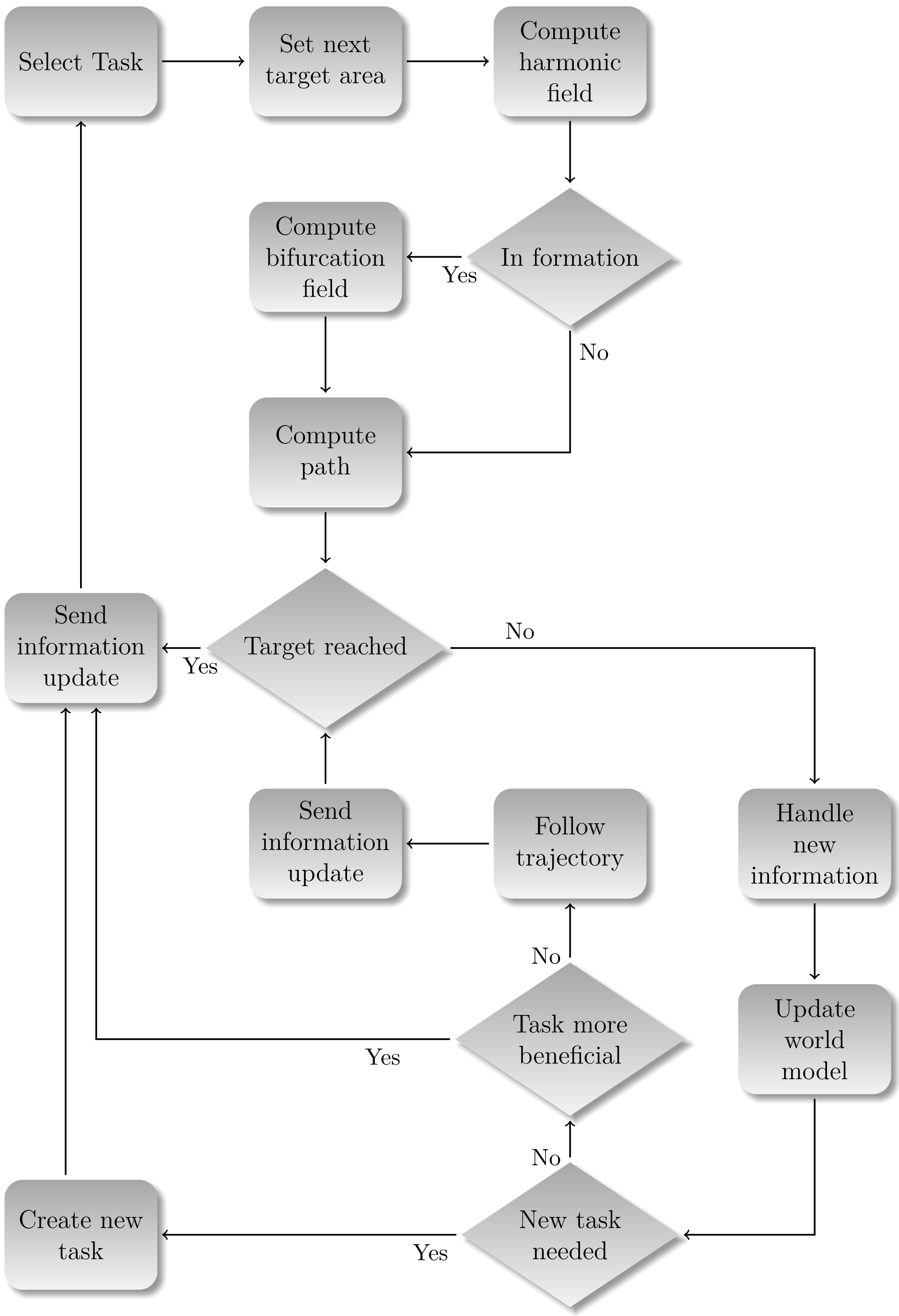


Figure 4.4.: *Graphical representation of the UAV behavior including the interaction of the methods.*

Based on the resulting potential field, a path is then computed using the descent gradient as described in Section 5.4.

The UAV continuously updates its internal state during the path following procedure. It first handles new information received from the *Control Station* and other UAVs. This information handling contains position updates from all other UAVs, storage of newly arrived tasks and newly detected obstacles, as well as deletion of fulfilled or vanished tasks and no longer existent obstacles.

Updating the world model is the next step. This update contains the computation of newly explored areas through both, the UAV itself and other UAVs, as well as the internal representation of newly detected obstacles and the deletion of vanished obstacles. Additionally, a combination of leaves from the octree takes place. The algorithm for leave combination is described in Section 4.2.1.

The UAV then checks whether it is beneficial or necessary to create a new task. The details of task creation are provided in Section 7.4.1. Such tasks might be, e. g., the creation of a formation for exploratory navigation if the UAV is inside a large area consisting only of free space. In that case, a new task will be created and selected. The task processed before is returned to the UAV's task list. Thereafter, the other UAVs and the *Control Station* are informed about the new task and the task reallocation conducted by the UAV. Informing the other UAVs about returned tasks ensures that these tasks can be processed soon by another UAV. Informing the *Control Station* ensures that human operators obtain up-to-date information.

If the UAV did not create a new task, it checks whether any task in its task list has become more beneficial than the current one. Other tasks in the task list can become more beneficial than the current one for several reasons. The UAV could get close to the target area of some task it can now process without noticeable delay. Another possibility is that a new task with higher priority than the current one has been received from another UAV or a third party application such as an operator. All possible reasons leading to task creation or task change are explained in Section 7.4. If some tasks in the task list become more beneficial than the current one, the UAV returns the current task into its task list and takes on the new task. Every time an unfinished task is returned, the other UAVs are informed to ensure efficient task processing. The UAV also informs the *Control Station* to provide up-to-date information to the human operators.

If no task has become more beneficial than the current one, the UAV computes its new position based on the current configuration and its velocity. It then sends an information update containing its new position and angles to the other UAVs and the *Control Station*. It also sends the new fuel level to the *Control Station* if it has been changed. Thereafter, the UAV checks whether its configuration is equal to the target configuration and as long as it is unequal, the path following procedure continues. If the configurations are equal and the current task is solved, the UAV sends information about the fulfillment of the task to all other UAVs and the *Control Station*, erases the task from its task list, and selects a new one.

The UAVs compute their behavior based on the environments created as described in this section. Hence, they need to internally store and represent all necessary properties of the terrains in an internal world model. Internal environment representation is performed using an octree as mentioned before. The decision of the data structure is one of the most important one as it affects all further design decisions. Information the UAVs receive from the *Control Station* and from other UAVs are mapped to this data structure. The octree and its functionality is described in detail in the following section.

4.2. UAV World Model

A data structure is needed to create world models of environments received from the *Control Station*. For this purpose a dynamic octree has been introduced. It subdivides the environment into several disjoint areas, whereby each leaf of the octree depicts one subarea of the terrain and the union of the leaves represents the entire terrain $\mathcal{W}$ (cf. Section 3.2). The leaves store additional properties, so that the union of the leaves finally represents the combination of all subspaces of $\mathcal{C}$ presented below.

A world model based on a tree structure has several advantages. Areas that are irrelevant for the computation of suitable behavior can be combined. An efficient combination allows the representation of large terrains with relatively low memory consumption. It also decreases the time needed for potential field calculations (cf. Chapter 5). All methods and algorithms designed for the resulting system take the octree as basic data structure into account. Hence, it influences all abilities of the resulting system. It also directly affects the computational effort needed for single calculations and therefore the overall performance of the entire system.

Each UAV works on its own configuration space $\mathcal{C}$ (cf. Section 3.1.2) in order to reach a proper representation of the environment and its own internal state. $\mathcal{C}$ consists of five disjoint subspaces, namely explored space ($\mathcal{C}_{\mathcal{E}}$), unknown space ($\mathcal{C}_{\mathcal{UK}}$), unexplored space ($\mathcal{C}_{\mathcal{UX}}$), occupied space ($\mathcal{C}_{\mathcal{O}}$) and target space ($\mathcal{C}_{\mathcal{G}}$), whereby

$$\begin{aligned} \forall i, j, i \neq j \colon \mathcal{C}_i \cap \mathcal{C}_j &= \emptyset, \\ \mathcal{C}_{\mathcal{E}} \cup \mathcal{C}_{\mathcal{UK}} \cup \mathcal{C}_{\mathcal{UX}} \cup \mathcal{C}_{\mathcal{G}} &= \mathcal{C}_{\mathcal{F}}, \\ \mathcal{C}_{\mathcal{F}} \cup \mathcal{C}_{\mathcal{O}} &= \mathcal{C}. \end{aligned} \tag{4.1}$$

A distinction between unknown space and unexplored space takes place. The UAVs are able to fly in a priori known environments, as well as in partially or completely unknown environments. Aging of data is also modeled. Explored space can become unexplored again if no further exploration of the space has been conducted within a predefined time span. Space is assigned to $\mathcal{C}_{\mathcal{UX}}$ if it has never been explored before or if the last exploration lies back more than a given time. Information about areas marked as unexplored space ensure that the space is in $\mathcal{C}_{\mathcal{F}}$. This information can, e. g., be based on third party applications. If information points out that unexplored space is also in $\mathcal{C}_{\mathcal{O}}$ it is directly assigned to $\mathcal{C}_{\mathcal{O}}$.

Unknown space describes areas without any information regarding their state. In order to be able to gather further information regarding these areas $\mathcal{C}_{\mathcal{UK}}$ is also considered to be part of $\mathcal{C}_{\mathcal{F}}$. This ensures that the UAVs try to explore these areas. Finally, each leaf of the octree is assigned to one of the five subspaces.

One issue not addressed in this thesis derives from the described kind of mapping. The UAV presented in Figure 2.5 a), e. g., has a mission radius of 50 km. Taking such UAVs into account can lead to large environments with high resolution. This would result in a high number of leaves resulting in excessive memory consumption. It is unnecessary to use all leaves of the octree for the computation of the potential fields, as described later in Section 5.3.4. Additionally, the UAVs plan paths in a way that the positions the single UAVs will visit next are known. Thus, only the part of the octree, needed to compute feasible potential fields should be stored in main memory and the parts necessary for the next potential field computation can be reloaded before they are needed. A simple solution for the storage of parts of the octree, e. g., on the hard drive can be the use of a database. More efficient solutions can be adapted from game theory as, e. g., the solution described in [187]. Today's game designer face the same issue as it is often impossible to store the entire game environment in the main memory.

The octree stores the positions of all known UAVs, currently relevant targets and obstacles. The mapping of the positions to the octree is based on the three-dimensional Cartesian coordinate system $\mathcal{F}_{\mathcal{W}}$ (cf. Section 3.1.2). The main task of the UAVs is to find discrete paths ρ_d (cf. Section 3.2) from their initial configurations $q_{\mathcal{I}}$ to the next target configurations $q_{\mathcal{G}}$. These paths have to lead to an exploration of the complete environment by simultaneous avoidance of all obstacles in a fast and efficient way. Mapping all relevant positions to the octree allows the UAV to reach this objective by using only information provided by the octree.

The UAVs execute different tasks beside exploratory navigation with certain priories (cf. Section 7.4.2). They work in a completely decentralized manner and have to coordinate their actions themselves, in order to reduce redundant exploration and to avoid redundant task execution. Additionally, a simultaneous execution of several different tasks is needed to reach an increase in efficiency of the overall system. All flight actions conducted in order to reach these objectives are based on discrete paths ρ_d. A single ρ_d consists of a finite sequence of points. These points in turn are related to the leaves of the octree and the space type the single leaves are assigned to. Thus, all operations regarding the octree need to be performed as fast as possible in order to design a high-performance system.

The UAVs fly inside a terrain and always explore areas through their cameras. This leads to environmental changes as areas of the environment unexplored or unknown before become explored and new obstacles might be detected. It is necessary that the UAVs represent all changes internally to achieve an adequate behavior. Thus, the octree changes dynamically, as described in the following section.

4.2.1. Representation of Environmental Changes

The surroundings of the UAVs are highly dynamic. This leads to the need of a highly flexible world model. Dynamic changes of the environment are internally mapped as follows. The UAV starts with the creation of a root node representing the entire terrain. It consists of the boundaries of the environment in three dimensions and is assigned to $\mathcal{C}_{\mathcal{UX}}$ or $\mathcal{C}_{\mathcal{UK}}$ depending on initial parameters.

The octree has a maximum level in order to reach an adequate subdivision. This level leads to a maximum number of subdivisions and is based on the dimensions of the terrain and the dimensions of the UAV (cf. Section 3.3.1). A leaf node at maximum level is big enough to contain a complete UAV. This is necessary as a UAV entering the node must avoid neighboring nodes, containing obstacles, under all circumstances. Every time a leaf is created, it is assigned to one of the subspaces.

The root node gets subdivided into eight equally sized child nodes as soon as the UAV enters the environment. Some of the resulting nodes are also recursively subdivided into eight equally sized child nodes until the resulting child nodes are of homogeneous type or until the maximum level has been reached. Here, homogeneous type means that a resulting leaf encompasses only one single space type. An assignment of the leaf to the corresponding space type is then performed.

Leaves of maximum level can also be heterogeneous. A leaf is heterogeneous if it encompasses multiple subspaces. This leads to an approximation. Leaves that encompass $\mathcal{C}_{\mathcal{O}}$ are always assigned to $\mathcal{C}_{\mathcal{O}}$. Leaves surrounding only subspaces in $\mathcal{C}_{\mathcal{F}}$ are assigned to the type of subspace they mainly consist of.

Finally, $\mathcal{C}$ is subdivided into $\mathcal{C}_{\mathcal{F}}$ and $\mathcal{C}_{\mathcal{O}}$ and the resulting $\mathcal{C}_{\mathcal{F}}$ is additionally subdivided into $\mathcal{C}_{\mathcal{UK}}$, $\mathcal{C}_{\mathcal{UX}}$, $\mathcal{C}_{\mathcal{E}}$ and $\mathcal{C}_{\mathcal{G}}$. The single space types are represented by different leaf types. The subdivision changes dynamically during runtime when the UAVs gather new information about the terrain. The possibility to reassign leaves to other space types is necessary in order to model all possible changes. Environmental exploration is achieved through cameras. An algorithm based on the ray tracing approach has been designed for the determination of newly explored leaves. It takes all required properties into account and is described in detail in Section 4.3.1. The algorithm subdivides all leaves the UAV can currently explore through its cameras to create leaves at maximum level. This subdivision takes place to ensure that large leaves are not completely assigned to $\mathcal{C}_{\mathcal{E}}$ if they are only partially explored. Afterwards, a check for the identification of newly explored leaves at maximum level is conducted. The resulting algorithm is described in pseudo code in Appendix A.3.

The mentioned update method for the assignment of newly explored leaves results in an increasing number of nodes. This rises the time needed for path calculations and formation flights dramatically [188]. To reduce the increase of the calculation times, an algorithm responsible for the combination of leaves has been designed. The algorithm considers that eight child nodes of a node are created one after the other for subdivision. The single child nodes are consecutively marked with values from 0 to 7. The algorithm is presented in pseudo code in Appendix A.4 and works as follows.

It moves recursively through the complete octree. Each time a leaf marked as 7th child is reached, the algorithm checks whether a combination is possible. It is more efficient to check only one child of each node as all children are combined at once. Checking the last leaf is done due to programming reasons. It is a simple way to avoid that the algorithm checks leaf nodes already combined and therefore nonexistent any more. Another way would be, e. g., to always check if the leaf node to be checked still exists.

The algorithm checks whether the seven siblings of the leaf are assigned to the same space type. A combination is performed if this is true and if aging is currently not considered. The points in time the last explorations of single leaves dates back, are also taken into account for leaves, assigned to $\mathcal{C}_{\mathcal{E}}$ if the scenario supports aging. The difference between these points in time has to be below a threshold for a combination. The threshold depends on the given time after a newly explored leaf is considered to be unexplored again.

Using a recursion ensures that the leaf nodes at maximum level are checked first and also combined first if possible. This leads to the advantage that a complete combination takes only one run through the tree.

Due to the fact that explored areas of the terrain can become unexplored again, the UAVs must have the ability to explore the environment over and over again to gather up-to-date information about the underlying terrain. Repeated exploration after some given time is also advantageous if, e. g., moving objects have to be found or if a changing area has to be monitored permanently without enough UAVs to cover the complete area at once.

Based on the properties of the octree explained so far, it is possible to represent environmental changes through octree changes. Additionally, the abilities to map and represent occupied areas of the environment must exist. The UAVs receive environmental objects as triangle meshes and map them into the octree as described in the following section.

4.2.2. Environment Mapping

As mentioned before, two different configuration space settings are selectable at the beginning of a simulation. The environment can on the one hand be treated as known a priori if, e. g., the main task is WISAR (cf. Section 2.2.3) and up-to-date information about the environment exists. The unexplored space is then assigned to $\mathcal{C}_{\mathcal{UX}}$. It is on the other hand possible to consider the environment as completely unknown at the beginning, in order to take big disaster applications, such as earthquakes or tsunamis, seriously changing the environment, into account. In this case the terrain is assigned to $\mathcal{C}_{\mathcal{UK}}$.

If the simulation considers the environment as unexplored at the beginning, the UAVs receive initial terrain information from the *Control Station* in the form of triangle meshes. Those are mapped into the octree as follows. Each triangle is recursively subdivided into two equally sized triangles. Subdivision stops if the resulting triangles are smaller than leaves of maximum level in each dimension. Afterwards, each leaf that encloses the positions of at least one triangle vertex is checked. These leaves are recursively subdivided until leaves at maximum level have been created. The resulting leaves are then checked. Each leaf enclosing at least one of the vertices is assigned to $\mathcal{C}_{\mathcal{O}}$. This ensures that each leaf intersected by a triangle is considered to be occupied independent of the original size of the triangle. The mapping is also executed when new obstacles have been detected.

As mentioned before, each UAV is represented as point $p_{\mathcal{U}}$ in $\mathcal{C}$. This representation neglects the dimensions of the single UAVs. It is necessary to consider the volumes of the UAVs in order to guarantee collision free paths. An obstacle growing [189] is conducted through an expansion of the occupied space by the dimensions of the UAV in three dimensions for that purpose. The approach implemented for obstacle growing starts with the creation of neighboring leaves at maximum level for all occupied leaves. The resulting neighboring leaves are then assigned to $\mathcal{C}_{\mathcal{O}}$. As mentioned, the maximum level is based on the dimensions of the UAV leading to resulting leaves, big enough to enclose the UAV completely. The UAVs do not take grown obstacles' areas into account for path planning. So, they can only be at the edge of a leaf, assigned to occupied space by obstacle growing and never be at the edge of a leaf containing an obstacle. Thus, a UAV overlaps with half of its size in a single dimension, which is equal or less to half of the size of the leaf. So, assigning these leaves to the occupied space ensures collision avoidance.

Figure 4.5 illustrates the method of obstacle growing in a two-dimensional environment. An obstacle is located inside the black leaf. Due to obstacle growing, the surrounding leaves are also considered to be occupied and marked gray. The path leads from $q_{\mathcal{I}}$ to $q_{\mathcal{G}}$ and takes only the center points of the nodes as route points and touches one of the leaves set as occupied. This leaf does not contain the obstacle and if the dimension of the leaf is bigger than half of the dimension of the UAV, a safe distance between the UAV and the obstacle is guaranteed as the point representing the UAV is located at the center of the UAV. The same holds true for the three-dimensional case.

The use of triangle meshes for terrain representation is not only practicable for simulations. If real UAVs receive their environmental information by third party applications or through sensors, like laser and cameras, it is always possible to compute triangle meshes from the input data, e. g., using the algorithm presented in [186]. This makes the approach universally applicable.

The described mapping can lead to areas unreachable for UAVs but assigned to $\mathcal{C}_{\mathcal{F}}$. Such areas are, e. g., rooms in a building the UAVs cannot enter. If a UAV selects an unreachable area as the one to be explored next, a feasible path computation is impossible. It is assured that a UAV will never select such unreachable areas as next target area in order to overcome this problem. An adaptation of classical flood fill algorithms [190] has been introduced for that reason. The algorithm starts at the leaf $l_{\mathcal{U}}$ that encloses the UAV. All neighboring leaves l_i with $\{l_i, l_{\mathcal{U}}\} \in \mathcal{N}, l_i \notin \mathcal{C}_{\mathcal{O}}$ are marked as reachable. The neighboring leaves l_j with

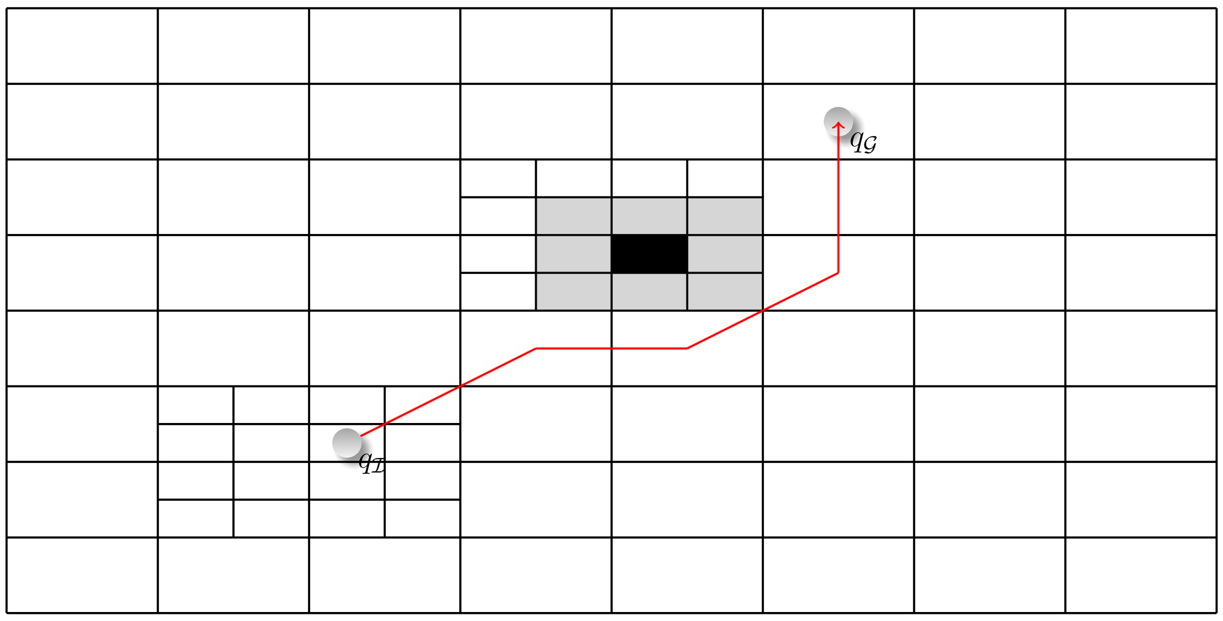


Figure 4.5.: *A two-dimensional sample path from $q_\mathcal{I}$ to $q_\mathcal{G}$ leading as close as possible to an obstacle. The obstacle lies in the black leaf while the gray leaves are also considered to be occupied through the use of obstacle growing.*

$\{l_i, l_j\} \in \mathcal{N}, l_j \notin \mathcal{C}_\mathcal{O}$ of those leaves are also marked as reachable and so on. This is iteratively executed until only previously marked leaves can be reached. Afterwards, each leaf $l_k \in \mathcal{C}_\mathcal{F}$ that is not marked as checked is assigned to $\mathcal{C}_\mathcal{O}$. The algorithm is presented and described in detail in Appendix A.2.

Finally, an example of a resulting octree, subdividing a fictive three-dimensional terrain is shown in Figure 4.6. The subdivision of the terrain using an octree even with a low maximum level leads to relatively small subareas. Thus, a high tree depth is usually not necessary.

The algorithms presented so far endow the octree with the abilities needed for environmental object representation. It is also dynamically changeable in order to represent the current system state. Finally, the octree needs abilities to relate the UAVs to the environment. Each node of the octree has several properties for the creation of such a relationship as described in the following section.

4.2.3. Node Properties

To reach a reasonable exploration of different environments, it must be guaranteed that the path planner is at least complete (cf. Definition B.5). Additionally, information gathered by the UAVs must correspond to meaningful positions. It is not very beneficial if a UAV finds, e. g., casualties during a rescue mission when the position of the UAV related to the environment and therefore the positions of the casualties are unknown. To specify the positions of UAVs and the leaves of the octree, a Cartesian coordinate system $\mathcal{F}_\mathcal{W}$ is embedded

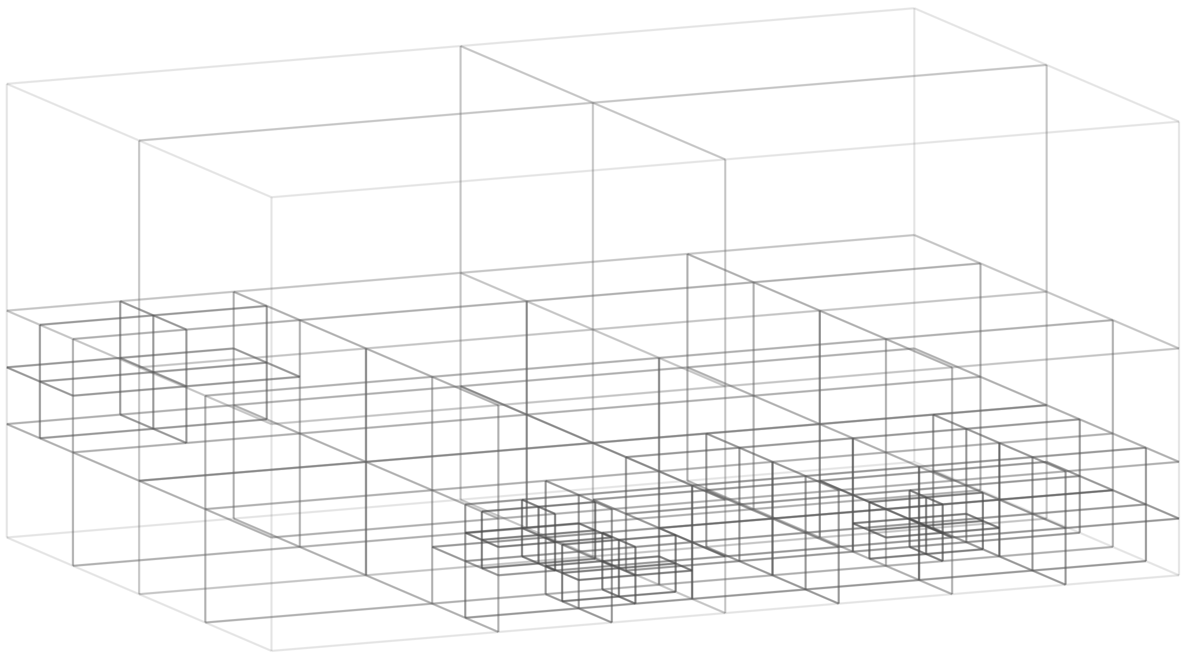


Figure 4.6.: *Possible subdivision of a fictive terrain through an octree with a maximum level of 3.*

into the configuration space $\mathcal{C}$. Each UAV has a position and direction associated to the coordinate system. Additionally, each node of the octree has six properties presented in Table 4.1. They are necessary for the computation of feasible paths and for the provision of a consistent representation of $\mathcal{C}$.

Property	Description
boundaries	Two three-dimensional points specifying the position and dimensions of the node.
center	Center position of the node.
space type	The space type the node is assigned to, e. g., $\mathcal{C}_{\mathcal{E}}$ or $\mathcal{C}_{\mathcal{O}}$.
level	The level of the node related to the octree. The root node is at level 0.
potential value	Current potential value of the node.
exploration time	The point in time of the last exploration.

Table 4.1.: *The six main properties describing the single nodes of the octree.*

The position and dimensions of each node are specified by the boundaries and are based on $\mathcal{F}_{\mathcal{W}}$. The boundaries denote the minimum positions (x_{min}, y_{min}, z_{min}) and the maximum positions (x_{max}, y_{max}, z_{max}) of each node. Based on these points the dimensions of each node can easily be determined. The boundaries are mainly used to compute the positions of new child nodes in a highly efficient way. Efficient subdivision is necessary in order to respect highly dynamic environments.

The center positions of each node are stored as well. This is redundant as it is possible to compute the center positions from the boundaries at all times. Nevertheless, they are needed frequently, e. g., for node determination (cf. Section 4.2.4), for potential field computation (cf. Chapter 5 and Chapter 6) and path computation in order to ensure collision avoidance (cf. Section 5.4). Computing the center positions each time they are needed, would increase the system's computational effort noticeably. Storing the positions leads to a higher memory consumption but the achieved reduction of the computational effort has been considered to be worth the higher consumption.

The property space type described in Table 4.1 represents the currently assigned subspace of the configuration space. As mentioned before, this assignment can change anytime during simulation and it has crucial influence on the potential field approaches described in detail in Chapters 5 and 6.

The level denotes the depth of the leaf inside the octree. The root node of the octree is level zero. The children of a node at level a are at level $a + 1$. This property is mainly used to lower the computational effort needed by the neighbor finding algorithm (cf. Appendix A.1) and the algorithm responsible for dynamic octree changes (cf. Appendix A.3).

The union of the potential values of the single leaves represents the current potential field. Path computation (cf. Section 5.4) and formation flights (cf. Chapter 6) are based on these potential values.

The resulting system has the ability to gather up-to-date information on each area over and over again. To efficiently determine areas with out-of-date information, each node stores the moment of its last exploration using the property exploration time. If the difference between the current time and the exploration time is above a given threshold, a reassignment of the space type to $\mathcal{C}_{\mathcal{UX}}$ is executed.

The methods introduced so far lead to an octree providing the basics needed to reach the objectives introduced in Section 1.2. Additionally, it is necessary to determine specific nodes of the octree efficiently in order to execute the algorithms of the resulting system frequently. The determination technique is described in the following section.

4.2.4. Node Determination

Determining specific nodes of the octree is a basic operation. It is extensively performed by the UAVs, up to several thousand times per second. Thus, it has direct influence on the performance of the entire system. Therefore, node determination should be executed as fast as possible.

The operation is e. g., necessary for the computation of a harmonic potential field. This computation often takes several thousand nodes into account (cf. Sections 8.3 and 8.4). For each node, all neighboring nodes in six directions have to be found in order to be able to compute the potential field. So, the algorithm designed for node determination has direct influence on the computation times presented in Chapter 8. The potential field calculations must be performed as fast as possible, to give the UAVs the ability to react to changing environments, e. g., to avoid moving obstacles.

Several additional operations need node determination. They are also based on different inputs. Only one algorithm has been designed to compute nodes. The single operations that need node determination are presented in Table 4.2.

Determine	Corresponding operation
neighboring nodes of a given node	Obstacle growing
	Potential field computation
	Path computation
existing leaves enclosing a specific position	Potential field computation
	Path computation
leaves at maximum level enclosing a specific position	Environment mapping
	Collision avoidance
	Octree updates

Table 4.2.: *The different types of operations that need node determination.*

It is easy to design an algorithm that finds specific nodes at level n inside an octree with a worst case performance of $\mathcal{O}(\log_8 n)$, with n describing the maximum possible number of leaves. Improving the worst case performance normally leads to higher memory consumption and complexity as, e. g., the algorithm designed by Aizawa et al. [191]. They have designed a $\mathcal{O}(1)$ algorithm able to find neighboring nodes in quadtrees. Therefore, they store several additional values for each node to actually reach this worst case performance. A high overhead of memory consumption for each single node is unacceptable considering the resulting system wherein several million nodes might exist. So, a tradeoff between memory consumption and computational complexity has to be found.

The resulting algorithm (cf. Appendix A.1) for node determination still has a worst case performance of $\mathcal{O}(\log_8 n)$. The idea is to reduce the number of steps needed to move from one node to another. Additionally, only instructions a CPU performs highly efficiently are used.

The first step is to design the octree in a way that moving from one node to another can be performed very fast. This has been reached through the use of pointer arithmetic to link each node to its parent and its eight child nodes if they exist. Hence, moving from one node to another is only a memory leap.

The algorithm always starts at the root node and moves down from one level of the octree to the next. It thereby takes a position $p \in \mathcal{F}_{\mathcal{W}}$ enclosed by the resulting node into account. This position is given, e. g., by the position of the UAV or a computed position enclosed by a node to be determined. The algorithm computes iteratively the corresponding child node of the current node that encloses p.

Computation of the correct child node is based on the way these child nodes are stored. Each node internally stores pointers to its children and assigns them to a value c_i. The correct value c_i for children $i = 0, ..., 7$, which encloses p has to be computed. Subdivision of a node and the storage of the children is always executed in the same fashion. So, each single subarea of the node is always enclosed by the child, with value i. Figure 4.7 illustrates the areas covered by the child nodes of a single node and their values i.

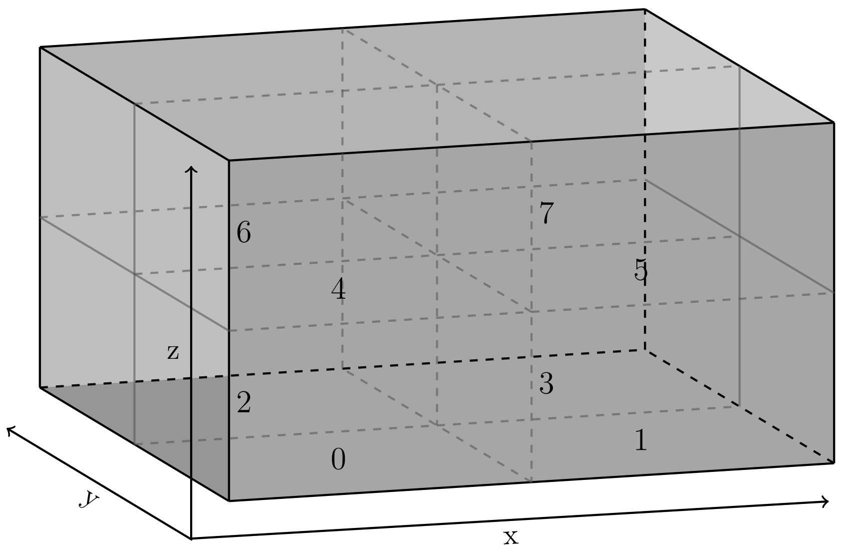


Figure 4.7.: *The values of c_i to which the eight child nodes of a given node are assigned to.*

The resulting child value c_i depends on the center position $l_{i,(x,y,z)}$ of node l_i. The single positions at the dimensions x, y, and z of the center point of node l_i are considered separately for node determination. They are represented by $l_{i,x}$, $l_{i,y}$, and $l_{i,z}$. The same is true for the point p enclosed by the child node and the positions are denoted as p_x, p_y, and p_z. Finally, the resulting child number is computed using Equation 4.2:

$$c_i = \chi_x + \chi_y + \chi_z, \tag{4.2}$$

whereby

$$\chi_x = \begin{cases} 1, & \text{if } l_{i,x} < p_x, \\ 0, & \text{else} \end{cases} \tag{4.3}$$

$$\chi_y = \begin{cases} 2, & \text{if } l_{i,y} < p_y, \\ 0, & \text{else} \end{cases} \tag{4.4}$$

$$\chi_z = \begin{cases} 4, & \text{if } l_{i,z} < p_z, \\ 0, & \text{else} \end{cases} \tag{4.5}$$

Executing Equation 4.2 and the corresponding Cases 4.3, 4.4 and 4.5 iteratively allows to move down the tree by starting at the root node in order to determine nodes enclosing p at a given level.

A specific position enclosed by the neighbor node has to be computed to find neighboring nodes of a given node. The dimensions of the single nodes at each level are fixed and given by the size of the environment. Offsets are computed a priori for each dimension and each level in order to find neighboring nodes of a given node. Let $d_{n,x}$ be the dimension of a node at level n in x direction and be $d_{max,x}$ the size of a leaf at maximum level in x direction. The offset o_x used to compute a point enclosed by a neighboring node at level n in positive x direction is then calculated by:

$$o_x = \frac{d_{n,x}}{2} + \frac{d_{max,x}}{2}. \tag{4.6}$$

The offsets for the other neighboring directions are computed analogously. They are multiplied by -1 in order to find neighbors in $-x, -y$ and $-z$ direction. 26 offsets are stored for each level of the octree in order to represent the 26 neighboring directions. Each offset is a vector consisting of an x, y, and z value. The offset is added to the center point of the node leading to the point p, that is taken into account by Equation 4.2, until the resulting neighboring node at level n has been found.

For potential field computations (cf. Chapter 5) not only neighboring nodes at the same level but all neighboring leaves independent of the level must be determined. The octree changes dynamically leading to neighboring nodes at different level. Due to the changes of the tree, it is not advantageous to store the neighboring nodes of each node persistently.

The neighbor finding algorithm first uses Equation 4.2 to compute the neighboring node at the same or a lower level as the current node. Afterwards, it is checked whether the resulting node is a leaf. The algorithm terminates if this condition is true. If it is false and if the neighboring node to be determined is a neighbor node along a vertex, only one resulting leaf exists. For neighboring nodes along an edge, always two neighboring child nodes of the nodes determined so far are recursively checked. To find neighboring nodes along a face, four neighboring child nodes are checked recursively. The nodes to be checked depend on the search direction. The algorithm is described in detail in Appendix A.1.

The octree presented in this section subdivides the configuration space $\mathcal{C}$ introduced in Section 3.1.2 into disjoint areas. The UAVs rely on this subdivision beside others for path computation. The octree provides only the basic functionality needed to create an operational, cooperative multi-UAV system. A feasible and efficient behavior has to take more into account than the current position of the UAV related to the octree. It has to respect the abilities of the UAV and thus use the current configuration $q_{\mathcal{U}} \in \mathcal{C}$ for decision making in order to compute the resulting UAV behavior. A UAV configuration is the combination of several properties described in the following section.

4.3. UAV Configuration

The decisions each UAV makes are mainly based on the UAV's unique configuration (cf. Section 3.1.2). As mentioned before, the union of all possible configurations leads to the configuration space $\mathcal{C}$.

A configuration $q_{\mathcal{U}}$ represents a single UAV. The current configurations of the UAVs are stored by all UAVs and by the *Control Station*. $q_{\mathcal{U}}$ is a 12-tuple $(q_1, \ldots, q_{12})$, whereby each element q_i represents one property of the UAV. The single properties are briefly described in Table 4.3.

Element	Description
q_1	A unique identifier used to distinguish the single UAVs
q_2	The maximum speed of the UAV
q_3	The current position in space ($p_{\mathcal{U}}$) of the UAV
q_4	The current yaw ($\beta_{\mathcal{U}}$), pitch ($\gamma_{\mathcal{U}}$) and roll ($\delta_{\mathcal{U}}$) angles of the UAV
q_5	The favorite altitude of the UAV
q_6	The maximum altitude the UAV can reach
q_7	A unique identifier of the current formation of the UAV
q_8	The role of the UAV used for task allocation
q_9	The currently executed task of the UAV
q_{10}	The current fuel level of the UAV
q_{11}	The communication status
q_{12}	The equipment of the UAV

Table 4.3.: *Brief description of the single entries of a UAV configuration $q_{\mathcal{U}}$.*

The first entry q_1 of $q_{\mathcal{U}}$ denotes a unique identifier. It is used to easily distinguish and track the single UAVs by the *Control Station* and other UAVs. This identifier makes it easy to monitor a single UAV over long time spans or to compare previous behaviors of a specific UAV with the current one.

The maximum speed q_2 is the maximum flight speed of a UAV. Taking heterogeneous UAVs into account this speed can differ from UAV to UAV.

The position q_3 of a UAV $p_{\mathcal{U}}$ denotes the current position of the UAV in the three-dimensional space. This position is the mentioned point representing the UAV (cf. Section 3.1.2) inside the configuration space and is located at the center of the UAV. It changes during flights and must be unique at each single point in time. Two UAVs can take the same position at the same moment only after a collision. The position could also be used to distinguish the UAVs. A distinction based on the single positions would increase computational complexity compared to a constant identifier due to the dynamically changing values.

The current positions of the UAVs can also be used to establish collision avoidance. In that case, each UAV would need to compare its position to the positions of the other UAVs. If the distance between two UAVs gets below a threshold, a collision warning is set up by the UAVs. Then they use the advantage of the potential field approach (cf. Section 5.3.1) to compute collision free paths as described in Section 5.4.

Beside their position the current yaw $\beta_{\mathcal{U}}$, pitch $\gamma_{\mathcal{U}}$ and roll $\delta_{\mathcal{U}}$ angle of the UAV q_4 are part of the configuration, too. The yaw angle denotes the current flight direction of the UAV related to $\mathcal{F}_{\mathcal{W}}$. The combination of these three angles and the current position plays a major role in the computations of the currently explored area, as described in the next section.

Each UAV is tempted to fly at a favorite altitude q_5 for exploratory navigation. This altitude is fixed for each UAV during a simulation. Camera equipments are used for the exploration of the environment. The quality of the ground pictures taken by the cameras is related to the current altitude of the UAV. Each UAV tries to fly at its favorite altitude in order to fit given quality requirements. Hence, the favorite altitude specifies the altitude of a UAV above the closest underlying obstacle or the terrain ground.

The maximum altitude q_6 is the maximum height the UAV can reach related to $\mathcal{F}_{\mathcal{W}}$. It is one of the values used to partition the terrain into $\mathcal{C}_{\mathcal{F}}$ and $\mathcal{C}_{\mathcal{O}}$. All parts of the terrain lying above the maximum altitude are assigned to $\mathcal{C}_{\mathcal{O}}$.

Different UAVs of the system presented in this thesis are able to create several formations simultaneously (cf. Section 7.4). Each formation is assigned to a unique identifier q_7, so that the different formations can easily be distinguished. q_7 is set to the unique identifier of the formation as long as the UAV is part of it. If the UAV takes no part in any formation, q_7 is set to 0.

The role system q_8 [6] is the implementation of the division of labor approach (cf. Section 2.1.4). Each UAV is always assigned to a role. This role is also fixed during a simulation and mainly based on the equipment of the UAV. Properties like maximum flight speed can also influence the role selection. It is mainly used by the decentralized task allocation system described in Chapter 7.

The UAVs have the capabilities to execute different tasks beside exploratory navigation. Different UAVs can conduct various tasks simultaneously. This is reached by a task list containing all known tasks and their current status. The UAV always processes one task q_9 in order to support ground units. Each task has exactly one target. A target can be a single point within the environment or an area the UAVs have, e. g., to explore or to monitor with high priority. Each target is assigned to exactly one task.

Another element of a UAV configuration is the fuel level q_{10}. Obviously, it is important that a UAV flies to a gas station and lands before it runs out of fuel. Each UAV creates a new task with its first initial position as target position if the fuel level reaches a threshold in order to land before it runs out of fuel. The threshold is based on the maximum flight speed of a UAV and the terrain size. It assures that the UAV has enough fuel left to fly over a distance equal to the diagonal d of the complete environment as described in detail in Section 7.4.1.

The communication status q_{11} of a UAV is important, too. If a UAV loses connection to the other UAVs, it is unable to fly in a safe manner. It then computes a path to the next airfield. This path keeps the UAV at an altitude unequal to the favorite and maximum altitudes of the other UAVs in order to avoid other UAVs flying at these altitudes.

The UAVs are heterogeneous in terms of equipment. Entry q_{12} represents the current equipment of the UAV. It can be equipped, e. g., with a gripper in order to perform side clearing operations. The diverse equipments of the UAVs are described later in Section 7.3.2. The UAVs are always equipped with at least one camera to explore the surroundings. The properties of the camera and its influence on exploratory navigation are explained in the next section.

4.3.1. UAV Camera Equipment

As mentioned before, the UAVs explore the terrain using a camera equipment. They always have one bird's-eye view camera to explore the underlying terrain. The UAVs also need to determine whether the airspace in their flight direction is in $\mathcal{C}_{\mathcal{F}}$ or in $\mathcal{C}_{\mathcal{O}}$, to be able to avoid collisions when they fly in unknown areas. In that case they are equipped with front cameras used to check a region in front of the UAVs. The size of the region they are able to explore depends on the used camera. This restricts the UAV flight directions such that they do not perform rear or sidewards flights. The areas a UAV can explore at a certain moment and the positions the cameras are mounted, have been schematically illustrated in Figure 4.8. The presented environment is an example terrain provided by the rendering engine WildMagic5 [180].

The position of the UAV $p_{\mathcal{U}}$ and its yaw $\beta_{\mathcal{U}}$, pitch $\gamma_{\mathcal{U}}$ and roll angle $\delta_{\mathcal{U}}$ are considered for the computation of the newly explored areas in order to create a proper modeling of camera based exploratory navigation. Taking the flare angle θ_d, the camera angle λ of the bird's-eye view camera, and additionally the flare angle θ_f of the front camera into account, leads to a more realistic computation. The single angles used to compute the newly explored areas of the terrain are illustrated in Figure 4.8.

An update algorithm, based on the mentioned values and described in Appendix A.3 using pseudo code has been designed to determine newly explored leaves. The algorithm computes four boundary points on the terrain ground enclosing a two-dimensional area the camera can monitor based on the single angles and the UAV position. It is then assumed that all leaves at the terrain ground are at maximum level and the center positions of the leaves enclosed

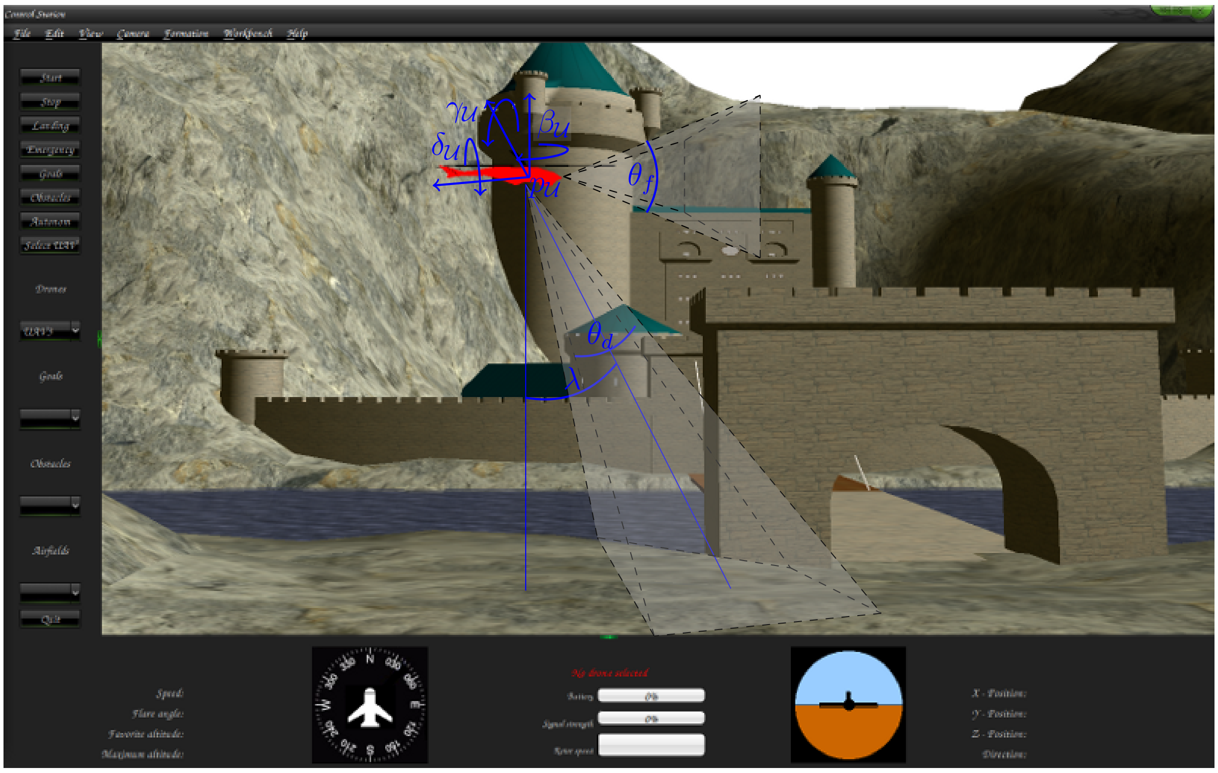


Figure 4.8.: *Schematic illustration of the areas a UAV explores through its camera equipment and the angles used to determine these areas.*

by the boundary points are computed. Afterwards, rays are sent from the camera position to the computed center positions of the leaves.

All nodes intersected by a ray are subdivided until leaves at maximum level have been created in order to avoid the assignment of large nodes to $\mathcal{C}_{\mathcal{E}}$. All nodes at maximum level have the same dimensions. This makes it possible to discretize the rays into positions, leading to all newly explored nodes. Classical ray tracing approaches [192] would take each leaf of the octree and check whether it intersects with the ray. This check is repeated for each ray. In contrast to such an approach, a fast determination of the newly explored leaves is achieved. A ray ends if it intersects a node assigned to $\mathcal{C}_{\mathcal{O}}$ or if it has left the environment. Exploration using the front camera is conducted analogously with a fixed distance the camera can explore instead of the terrain boundary. A more detailed description of the algorithm is presented in Appendix A.3.

An exploration at a given position concerning the octree nodes is presented in Figure 4.9. The illustration shows a simple example where $\beta_{\mathcal{U}}$, $\gamma_{\mathcal{U}}$, and $\delta_{\mathcal{U}}$ are zero. Additionally, no occupied nodes are considered. The UAV hovers at the center of the red node. The boundary points $b_1, \ldots, b_4$ are computed from this position. Based on these boundary points rays are sent to the terrain ground. The rays leading to positions enclosed by the boundary points are presented as violet lines and the rays to the boundary points as blue lines. Based on the

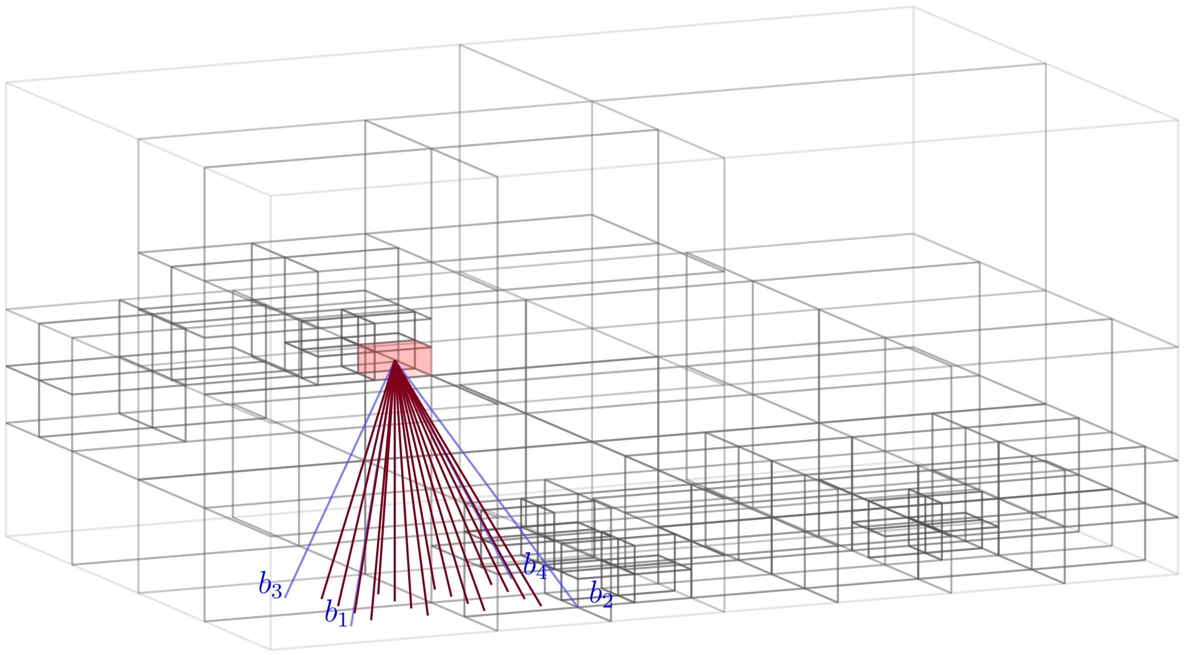


Figure 4.9.: *Schematic illustration of the exploration approach. The UAV is considered to hover at the red node. Rays sent to the boundary points b_i are blue. Rays sent to additional positions at ground level are violet. These rays will be discretized and the resulting points lead to the newly explored leaves.*

state as shown in the figure a discretization of the rays takes place and each node intersected by a ray will be subdivided to leaves at maximum level. The resulting leaves are then checked for whether an intersection with a ray takes place. In that case the leaves are assigned to $\mathcal{C}_{\mathcal{E}}$.

The introduced methods lead to a system wherein several UAVs are able to plan paths, explore environments, and allocate different tasks. An information exchange between the UAVs is necessary to avoid redundancies and to coordinate multiple UAVs in order to increase efficiency. Several possible methods for information exchange exist. Nature performs this, e. g., by dancing bees (cf. Section 2.1.2), through physical contact, or pheromones used, e. g., by ants (cf. Section 2.1.1). It is also possible to exchange information through sounds, gestures, and lights. A common method for information exchange in computer science are networks. Wireless networks are used by the UAVs to solve this task. The details of inter-UAV communication are presented in the following section, whereby the problems of real world dynamic ad-hoc networks are neglected and some possible solutions for the design of such communication systems are briefly pointed up.

4.4. Communication

Communication is always essential for the design of a cooperative system. Even animals in swarms need to exchange information between individuals (cf. Section 2.1) if they want to solve tasks single animals are not capable of. The UAVs must also have abilities to perform some sort of information exchange to reach a cooperative behavior. Compared to animal swarms, it is not possible to exchange information by contact as this would shorten UAVs' life span drastically. One common way to exchange information in computer science is provided by wireless networks. A dynamically moving UAV system wherein new UAVs occur and known UAVs vanish, can be considered as a dynamic ad-hoc network. Guaranteeing efficient information distribution in such a system is challenging.

Routing problems and efficient packet distribution along wireless sensor networks with dynamically changing nodes are not addressed by the resulting system. A simplification assuming that each UAV can exchange information with all other UAVs takes place. Some restrictions are mentioned later on (cf., e. g., Section 6.4.1). Nevertheless, a few approaches to overcome problems resulting from such a system are briefly presented.

Several researchers have designed efficient methods for packet distribution along dynamic ad-hoc networks. Hence, several possible solutions can be used for a system able to work in real world environments. Examples of possible methods have been presented by Warneke and Dannewith [193]. They have designed a load balancing algorithm for peer to peer networks. Frey and Stojmenovic [194] work on routing methodologies in multi-hop ad-hoc networks usable for information routing from one UAV over other UAVs to a target UAV. Scheideler et al. [195] work on methods to overcome interferences in wireless ad-hoc networks. Several additional approaches usable for information exchange between dynamically changing UAVs exist.

The UAVs need to exchange information for several reasons. The first one is that the UAVs have to receive terrain information. Real UAVs could achieve this information, e. g., by sensors like cameras and build a map based on the single camera pictures, as presented, e. g., in [7]. In simulation, another kind of information distribution must take place. The UAVs of the system presented in this thesis communicate with the *Control Station* via TCP/IP to receive initial information about the environment and also information about environmental changes, e. g., about new tasks. Information regarding the current configuration of the UAVs must also to be distributed in order to allow the *Control Station* to visualize an overview of the current system state.

Another reason why the UAVs should establish some sort of information exchange is that the entire system benefits from inter-UAV communication, like information they receive about areas newly explored by other UAVs. Additionally, it is beneficial to know the current configurations of the other UAVs to perform a more sufficient path computation. It is also necessary that the UAVs exchange their current positions if they have to fly in formation and if they want to guarantee collision avoidance. The UAVs would have no chance to allocate tasks efficiently in a decentralized manner without inter-UAV communication. They can achieve an information exchange, e. g., through the use of UDP/IP to broadcast information considering their own configuration.

4.5. Summary

This chapter first provides an overview of the entire system and then introduces its three main parts: the programs *Control Station* and *UAV*, as well as the library *Planner*. The relationships of the designed algorithms and their overall interaction implemented in the library *Planner* are presented in detail after a brief description of the *Control Station's* duties.

One of the chapter's focuses lies on environment representation. Each UAV has an internal world model for said purpose. Several different methods to represent environments exist. A representation using an octree as dynamic data structure to create a high-performance system and to dynamically react to environmental changes has been chosen. The representation directly influences the computational complexity of the entire system. An algorithm enabling the UAVs to dynamically change the octree at anytime in order to react to environmental changes is presented. Another algorithm allows the UAVs to map a wide range of environments into their internal world model. The determination of newly explored nodes is also described. Each of the octree's nodes has several properties presented in the further course of this chapter. Finally, an algorithm for node determination is introduced.

Afterwards, the UAV configuration is described in detail. One part of the configuration is the camera equipment that the UAVs use for exploratory navigation. UAV behavior is based on these configurations, as described in Chapter 7.

Some sort of information exchange is necessary to reach a cooperative UAV behavior. This topic is neglected for the resulting system and it is assumed that each UAV can communicate with each other. Such an assumption for communication is not realistic in real world environments. Several approaches, usable for packet routing and information distribution in the resulting multi-UAV system have been designed by researchers during the last decades. A few of them are briefly presented at the end of this chapter.

The UAVs can plan paths, fly in formation, and are able to perform decentralized task allocation using the presented architecture, as described in the following chapters. The next chapter explains the potential field approach for UAV flights, leading to the abilities needed to conduct entire explorations of complex and dynamically changing three-dimensional environments.

Chapter 5

Decentralized UAV Flights

Several approaches for autonomous robot movement considering single robots, as well as robot groups, exist (cf. Section 2.2). It is mandatory for a UAV application to ensure collision avoidance due to safety reasons. The resulting behavior of the UAVs must be verifiable if the UAVs support ground units and process different tasks on their own. The use of potential field theory can lead to a verifiable system. Using harmonic function theory to create an artificial potential field ensures collision avoidance. Additionally, the big disadvantage of the potential field theory—the occurrence of local equilibria—is avoided. The UAVs combine different techniques with path planning capabilities in order to reach a high degree of cooperation.

This chapter presents a three-dimensional artificial potential field approach wherein an approximation of a single harmonic function represents the entire potential field. The main idea is to map the problem of potential field computation to the Dirichlet Problem (cf. Definition B.8). The mapping is achieved by the use of the Dirichlet Boundary Condition (cf. Definition B.7). The resulting Dirichlet Problem can then be solved by a harmonic function [196].

The descent gradient $-\nabla$ of the resulting potential field is used to actually move a UAV from its initial configuration $q_{\mathcal{I}}$ to the required target configuration $q_{\mathcal{G}}$. One objective of the resulting system is that it must be able to entirely explore complex three-dimensional environments. This is achieved by the UAV ability to successively choose unexplored leaves $l_i \notin \mathcal{C}_{\mathcal{E}} \cup \mathcal{C}_{\mathcal{O}}$ of the octree, until all leaves of the free space are assigned to $\mathcal{C}_{\mathcal{E}}$.

The chapter first gives a problem formulation for environmental exploration. Afterwards, state-of-the-art approaches for exploratory navigation based on artificial potential fields are presented. Harmonic function theory is then briefly redescribed. The entire potential field is based on a single three-dimensional harmonic function (cf. Section 3.3.1). The computation of the potential field is a three step procedure introduced after the state-of-the-art section. First, the Dirichlet Boundary Condition is introduced. The second step computes a first approximation. This approximation is iteratively relaxed in a third step to converge the potential field gradually to the resulting harmonic function while the computation of a perfect harmonic function is unnecessary to find feasible paths. To reduce the computational effort, termination conditions to stop the iteration are introduced in the further course of this chapter. A fast computation of the potential field is reached by an efficient algorithm described thereafter. The gradient method used to compute paths is then presented, followed by the verification of the harmonic potential field approach. Finally, a summary of the chapter is presented.

5.1. Problem Statement

The formulation of the problem to entirely explore environments is as follows. Each UAV $\mathcal{U}$ is equipped with cameras for exploratory navigation. The cameras explore at each given time t a subarea A of the environment. Given $k \in \mathbb{N}$ UAVs $\mathcal{U}_j$. These UAVs have to successively compute discrete paths ρ_d (cf. Definition 3.6) such that each leaf of the octree $l_i \notin \mathcal{C}_\mathcal{O}$ has been at least once part of A. Thereby, each UAV follows only one path at a time and computes several consecutive paths.

Definition 5.1. Explored: *A terrain is considered to be completely explored, iff*

$$\forall l_i \in \mathcal{C} \setminus \mathcal{C}_\mathcal{O} \colon l_i \in \mathcal{C}_\mathcal{E} \tag{5.1}$$

Equation 5.1 means that the union of all explored leaves represents the complete free space. The path planning approach has to be sound (cf. Definition B.6) in order to reach this requirement. An additional property of the designed system is aging of information. Environmental areas explored a given time span ago can become unexplored again, i. e., leaves $l_i \in \mathcal{C}_\mathcal{E}$ can be assigned to $\mathcal{C}_{\mathcal{U}\mathcal{X}}$.

To find solutions for the mentioned problem, various approaches exist. Section 2.3.2 presents a selection of state-of-the-art methodologies for this. Additionally, several approaches for path planning based on artificial potential field theory exist. A small selection of today's approaches is presented in the following section.

5.2. State-of-the-Art

Despite the age of the potential field approach—it has been introduced by Khatib [62] in 1986—several state-of-the-art systems exist. This section presents a selection of such systems used for the movement of autonomous robots and drones. Current systems are mostly an extension of the original approach presented by Khatib.

Although all of the presented approaches are based on the main idea provided by the artificial potential field approach (cf. Section 3.3), no standard approach for the creation of such fields exists. Thus, four selected and very different ideas regarding the way of how to compute potential fields to solve the problem of path computation are presented in this section.

One approach are potential fields based on fuzzy logic as presented by Jaradat et al. [197]. They consider highly dynamic two-dimensional environments with not only static and dynamic obstacles, but also static and dynamic targets. The moving targets are modeled as attractive forces and are based on the relative position and velocity of the robot with respect to the target. The repulsive potential values are based on the relative positions of the robots to the obstacles. The forces are represented by expert "if, then" rules due to the use of fuzzy logic. 20 rules are used for the creation of attractive forces and a neuro-fuzzy network with learning capabilities and 121 rules is used for the creation of repulsive forces. Several tests were run based on MATLAB simulation scenarios and the moving targets have always been reached by simultaneously solving the local equilibrium problem (cf. Section 3.3).

An approach for the use of UAVs has been presented by Rivera et al. [198]. Their approach takes three-dimensional environments into account and is based on a two-step algorithm. The first step is the creation of a potential field for each obstacle under the use of sigmoid functions. A trajectory is computed in the second step using a technique of moving points that connect the UAV to the target. Each of these points follows the potential field and moves towards areas of free space. The points are computed as follows. First, the initial and the target configuration are connected by a straight line. Thereafter, initial points are placed along this line. Finally, the gradient based on the obstacles associated to these points is computed and moved towards the points. This is similar to a rubber band whereby the position of the band at single points is given by the obstacles and results in obstacle free paths. The presented algorithm overcomes local equilibria and unstable oscillations. It also works with arbitrary obstacle shapes.

It is also possible to combine swarm approaches with the potential field theory. A so called pheromone potential field has been designed by Piljae and Kurabayashi [199]. They use RFID tags to distribute pheromones over the environment through the interaction with mobile robots. The characteristics diffusion and evaporation of pheromones are used to compute partial differential equations. The digital pheromones are diffused by interaction with the robots. Each robot reads the currently readable pheromone values and calculates the following densities. The robot then moves around and finally creates a potential field by repeating this action.

Potential fields based on harmonic functions are also used nowadays by other researchers. Masoud [200], e. g., uses this approach to compute a path and to simultaneously create a reference velocity for a UAV. A methodology for the creation of harmonic potential fields is described in this chapter in detail. The velocity Masoud computed by the use of the harmonic potential field is compared to the velocity given by a so called virtual velocity attractor leading to another artificial force. The aim of his work is the creation of a flexible and easy to use, generic navigation controller that is able to support diverse kinds of UAVs and is also mathematically verifiable. In contrast to the approach presented in this thesis, a simultaneous planning and controlling of single UAVs has been achieved by Masoud. Controlling UAVs addresses the problem of computing flight speeds and flight angles of the UAVs based on the previously computed paths using some sort of a guidance law. This topic is neglected in this thesis and only path planning, formation flights, and task handling is addressed by the resulting system.

The UAVs of the presented system also create artificial potential fields in order to move from their initial configurations to their target configurations. They use potential fields represented by harmonic functions for that purpose, as described in the following section.

5.3. Harmonic Potential Field Computation

As mentioned before, an approximation of a harmonic function is used to represent a potential field [34]. In contrast to traditional potential field approaches, this kind of potential field computation has several advantages, like avoidance of local equilibria and smooth paths. A harmonic potential value ϕ^h, where h denotes that the potential value is part of a harmonic function, is assigned to each of the octree's leaves (cf. Section 4.2) in order to create such a potential field. The descent gradient $-\nabla$ based on the different ϕ^h values of the single leaves is used to find a path from the initial configuration $q_{\mathcal{I}}$ to the target configuration $q_{\mathcal{G}}$. Previous tests showed that it is possible to entirely explore different environments [5]. Each UAV calculates an approximation of a harmonic function describing the complete potential field, considering all obstacles and including other UAVs to achieve a decentralized behavior.

Harmonic functions are functions that satisfy Laplace's equation in n dimensions (cf. Section 3.3.1). Considering the three-dimensional case in Cartesian coordinates the problem is to find twice-differentiable real-value functions ϕ of real variables x, y, and z such that:

$$\nabla^2\phi = \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2} = 0. \tag{5.2}$$

It is possible to compute harmonic functions in several different ways. The UAVs presented in this thesis treat the potential field computation as an optimization problem. They first assign relevant leaves to fixed values to create the Dirichlet Boundary Condition (cf. Definition B.7) as described in the next section. The resulting Dirichlet Problem (cf. Definition B.8) is solved afterwards.

5.3.1. Dirichlet Boundary Condition

The UAVs compute the resulting potential field in a three step procedure. First, the potential field is modeled as a Dirichlet problem. This problem can be solved, using a harmonic function [196]. Two different kinds of potential values are introduced for the creation of the Dirichlet problem (cf. Definition B.8): bound and unbound values. The leaves $l_i \in \mathcal{C}_{\mathcal{O}} \cup \mathcal{C}_{\mathcal{G}}$

obtain bound potential values. The bound potential values $\phi^h_{\mathcal{G}} = 0$ and $\phi^h_{\mathcal{O}} = 1$ for these leaves are fixed constants, whereby h denotes that these are potential values of a harmonic function.

Leaves containing other UAVs are temporarily assigned to $\mathcal{C}_{\mathcal{O}}$. Assigning only these leaves to $\mathcal{C}_{\mathcal{O}}$ will not guarantee collision avoidance as it is possible that several UAVs fly at the same time to the same leaf. Such cases can be intercepted due to the introduced inter-UAV communication (cf. Section 4.4). The UAVs plan their paths and therefore all leaves a UAV takes on its way to its target configuration are known to the UAV. To avoid the described situation, the UAVs do not only exchange their current position but also the next route point they head for. The leaves enclosing these route points are also assigned to $\mathcal{C}_{\mathcal{O}}$ in order to guarantee collision avoidance.

As aforementioned, the leaves enclosing the target configurations obtain a potential value of 0. Harmonic functions satisfy the min-max principle (cf. Definition B.2). Thus, this value is the lowest potential value of the entire potential field. To fly towards such a value the UAVs have to use $-\nabla$.

Additionally, due to the min-max principle the bound values of the occupied leaves are the leaves with the highest potential values. Using the descent gradient will never lead to leaves with a higher potential value than the value of the current leaf. Thus, obstacle avoidance is guaranteed if the resulting approximation is close enough to a harmonic function.

A Dirichlet Problem is created through the assignment of the bound potential values. It can afterwards be solved by the creation of a harmonic function. An initial approximation of such a function is calculated as described in the following section.

5.3.2. First Approximation

The first step for solving the Dirichlet Problem is the computation of a first approximation of the resulting harmonic function. Such an approximation can, e. g., be the simple initialization of the potential values to fixed values. Prestes et al. [201], e. g., initialized all cells of the free space with the value 0. This is possible as the relaxation method introduced in the next section iteratively relaxes the potential field towards a harmonic function independent of the initial values. Nevertheless, a first approximation close to the resulting harmonic function should reduce the computational effort needed for relaxation. Thus, the equation used to compute a first approximation satisfies Laplace's equation. It is developed in a step wise manner in this section.

A strictly increasing function has been selected to compute the first approximation of a harmonic potential field $\phi^h_{x,y,z}$, where x, y, z denotes the three-dimensional case. A simple function strictly increasing is the Euclidean distance $\tau_{x,y,z}$ to the target configuration presented in Equation 5.3. Therefore, each UAV selects one leaf, which is part of its target configuration as next destination and computes the distances of the unbound leaves to the selected leaf.

$$\phi^h_{x,y,z} = \tau_{x,y,z} \tag{5.3}$$

The second property, required for Laplace's equation is that the function must be twice-differentiable and that the sum of the second partial derivatives vanishes overall. This can be achieved by taking 1 divided by $\tau_{x,y,z}$ from Equation 5.3, resulting in the well known Newtonian potential [202], shown in Equation 5.4. It is harmonic in the three-dimensional case.

$$\phi^h_{x,y,z} = \frac{1}{\tau_{x,y,z}} \tag{5.4}$$

Due to the Dirichlet Boundary Condition, a value restricted domain is given. Thus, the resulting potential values should be normalized in a way that $0 < \phi^h_{x,y,z} < 1$ in order to always ensure obstacle avoidance and soundness. If this is not the case the min-max principle is not fulfilled by the first approximation. The relaxation explained in the following section must then be executed until it is close enough to a harmonic function that it satisfies the min-max principle. This causes additional checks and therefore a higher computational effort. Equation 5.4 ensures that the resulting potential values for the unbound leaves are greater than zero and less than one, but the equation leads to decreasing potential values with increasing distances to the target. Increasing potential values with an increasing distance to the target are intended for the system presented in this thesis and are finally given by Equation 5.5. Therefore, potential values are defined as:

$$\phi^h_{x,y,z} = 1 - \frac{1}{\tau_{x,y,z}}. \tag{5.5}$$

The gradient ∇ obtained by the resulting potential field keeps the UAVs away from the target leaf. Therefore, $-\nabla$ represents a vector field pulling the UAV towards the target configuration. Equations 5.6 to 5.8 show the first partial derivatives of Equation 5.5 with respect to x, y and z. Equations 5.9 to 5.11 show the corresponding second partial derivatives with respect to x, y and z. As it can easily be seen, the sum of the second partial derivatives vanishes everywhere and Laplace's equation is fulfilled.

$$\frac{\partial \phi^h_{x,y,z}}{\partial x} = \frac{x}{(x^2+y^2+z^2)^{\frac{3}{2}}} \tag{5.6}$$

$$\frac{\partial \phi^h_{x,y,z}}{\partial y} = \frac{y}{(x^2+y^2+z^2)^{\frac{3}{2}}} \tag{5.7}$$

$$\frac{\partial \phi^h_{x,y,z}}{\partial z} = \frac{z}{(x^2+y^2+z^2)^{\frac{3}{2}}} \tag{5.8}$$

$$\frac{\partial^2 \phi^h_{x,y,z}}{\partial x} = \frac{-2x^2+y^2+z^2}{(x^2+y^2+z^2)^{\frac{5}{2}}} \tag{5.9}$$

$$\frac{\partial^2 \phi^h_{x,y,z}}{\partial y} = \frac{x^2-2y^2+z^2}{(x^2+y^2+z^2)^{\frac{5}{2}}} \tag{5.10}$$

$$\frac{\partial^2 \phi^h_{x,y,z}}{\partial z} = \frac{x^2+y^2-2z^2}{(x^2+y^2+z^2)^{\frac{5}{2}}} \tag{5.11}$$

Equation 5.5 is finally used to compute the first approximation of the harmonic potential field. Based on this potential field the UAVs are able to plan paths and entirely explore different three-dimensional environments.

Important information about the current environment, e. g., the height of single areas to determine $\mathcal{C}_\mathcal{O}$ can be obtained in several ways. In real world scenarios third party applications like maps are a good way to achieve at least approximative information. Additionally, the UAVs must have on-board sensors to ensure safe flights. These sensors can be used to explore unknown areas and to update the information available so far. During simulation, the *Control Station* is used for environment creation, environment storage, and for providing the UAVs with up-to-date information about environmental changes (cf. Section 4.1.1).

As the name suggests Equation 5.5 only computes a first coarse approximation of a harmonic function and does not take occupied areas into account. The first approximation is iteratively relaxed as described in the next section to reach an approximation that regards occupied space and is feasible for path planning.

5.3.3. Relaxation

The potential values computed by the first approximation so far are only based on the target position. The occupied areas have no influence to the unbound potentials. The common way in potential field theory used to consider the target, as well as the obstacles is the computation and superpositioning of two different potential fields based on attractive and repulsive potential values (cf. Section 3.3).

Superpositioning of harmonic functions representing repulsive and attractive potential fields leads to several problems. One is that paths that are computed and based on superpositioned harmonic potential fields can lead infinitely close to obstacles (cf. Section 3.3.1). A UAV that flies too close to an obstacle crashes. Such a behavior must be prevented under all circumstances. Thus, another type of potential field computation is needed.

The Dirichlet Boundary Condition ensures fixed potential values for the bound leaves. The first approximation leads to n equations resulting in n unbound potential values. The problem of relaxing the potential values for each leaf can also be performed by a system of equations. This system of equations can be solved, such that the occupied areas are also taken into account for the resulting potential field. Two popular algorithms leading to a solution are beside others the Gauss-Seidel method [203], [204] and the method of successive over-relaxation [204].

Prestes et al. [205], [201] have compared both methods empirically by computing several harmonic potential fields for path planning. Their results show that the Gauss-Seidel method leads to better results. Due to their results, the Gauss-Seidel method has been chosen to relax the given potential field for the UAV system presented in this thesis.

It is impossible to compute perfect harmonic functions due to given processor arithmetics. Only an approximation of a discrete sampling can be calculated. Numerical solutions for discrete samplings can be obtained relatively easy by finite differentiation methods as follows: For each point of a harmonic function $\phi^h_{x,y,z}$ the value is equal to the arithmetic mean of its surrounding points [206]. Let $\phi^h_{x,y,z}$ be a function satisfying Laplace's equation and let the single leaves of the octree be a representation of a discrete sampling of the three-dimensional potential field. The potential values of the left neighbor ϕ^h_l and right neighbor ϕ^h_r represent the potential values of the neighbors in x - direction. The potential values of the front neighbor ϕ^h_f and back neighbor ϕ^h_b represent the potential values of the neighbors in y - direction. The potential values of the lower neighbor ϕ^h_d and the upper neighbor ϕ^h_u represent the potential values of the neighbors in z - direction. By extending the work of Connolly et al. [34] these six values are used to derive second order central finite differences approximations for the second derivatives of $\phi^h_{x,y,z}$ using Taylor series expansion. The method for the derivation has been adopted from Smith [207], is described in Definition B.11, and results in the following three second order partial derivatives:

$$\frac{\partial^2 \phi^h_{x,y,z}}{\partial x} = \frac{\phi^h_l - 2\phi^h_{x,y,z} + \phi^h_r}{s_x^2} + O(s_x^4), \tag{5.12}$$

$$\frac{\partial^2 \phi^h_{x,y,z}}{\partial y} = \frac{\phi^h_f - 2\phi^h_{x,y,z} + \phi^h_b}{s_y^2} + O(s_y^4), \tag{5.13}$$

$$\frac{\partial^2 \phi^h_{x,y,z}}{\partial z} = \frac{\phi^h_d - 2\phi^h_{x,y,z} + \phi^h_u}{s_z^2} + O(s_z^4). \tag{5.14}$$

The terms s_x, s_y and s_z are step sizes to approximate the derivatives in directions x, y and z. Related to [34], the forth order error terms are negligible and can be discarded. The single derivatives presented in Equations 5.12 to 5.14 can be combined and Laplace's equation can then be written in discrete form as follows, iff $x = y = z$:

$$\nabla^2 \phi^h_{x,y,z} = \phi^h_r + \phi^h_l + \phi^h_f + \phi^h_b + \phi^h_u + \phi^h_d - 6\phi^h_{x,y,z} = 0. \tag{5.15}$$

Combining Equation 5.2 and Equation 5.15 and solving it for $\phi^h_{x,y,z}$ results in

$$\phi^h_{x,y,z} = \frac{1}{6}(\phi^h_r + \phi^h_l + \phi^h_f + \phi^h_b + \phi^h_u + \phi^h_d). \tag{5.16}$$

The introduction of the Dirichlet Boundary Condition fixes those $\phi^h_{x,y,z}$, which fall on $\partial\Omega$ (cf. Section 3.3.1). Thus, Equation 5.16 can be used to successively relax the first approximation of the potential values if the step sizes are equal in all directions. The equation is therefore executed several times for each leaf of the octree. The system is really large but techniques like the Gauss-Seidel method or successive over-relaxation can be used to solve it.

Due to the use of a dynamic octree, the neighboring leaves have different dimensions. This results in different distances to the neighboring nodes and thus in different step sizes. So, in contrast to the approach presented by Connolly et al. [34] the UAVs have to take the step sizes from the currently considered leaf to its neighboring leaves into account, too. Therefore, the work presented by Ames [208] for a quadtree and considered for different step sizes in two-dimensional environments by Zelek [209] has been adopted. The single step sizes are considered by Zelek as follows:

$$\phi^h_{x,y} = \frac{\tau_f\tau_b(\tau_l\phi^h_r + \tau_r\phi^h_l) + \tau_l\tau_r(\tau_b\phi^h_f + \tau_f\phi^h_b)}{\tau_f\tau_b(\tau_l + \tau_r) + \tau_l\tau_r(\tau_f + \tau_b)}, \tag{5.17}$$

where τ_l and τ_r represent neighbor distances in x - direction and τ_f and τ_b represent neighbor distances in y - direction.

Equation 5.17 has been extended to work in a three-dimensional environment and to take the considered neighboring leaves into account. The neighboring directions taken into account result in the von Neumann neighborhood [210] and are used to relax the potential values in order to reduce the computational effort by taking only six of the possible twenty-six neighboring directions (cf. Section 3.2) into account. The extension of Equation 5.17 leads to the resulting relaxation function shown in Equation 5.18.

$$\phi^h_{x,y,z} = \frac{\tau_u\tau_d\tau_f\tau_b(\tau_l\phi^h_l + \tau_r\phi^h_r) + \tau_u\tau_d\tau_l\tau_r(\tau_f\phi^h_f + \tau_b\phi^h_b) + \tau_f\tau_b\tau_l\tau_r(\tau_u\phi^h_u + \tau_d\phi^h_d)}{\tau_u\tau_d\tau_f\tau_b(\tau_l + \tau_r) + \tau_u\tau_d\tau_l\tau_r(\tau_f + \tau_b) + \tau_f\tau_b\tau_l\tau_r(\tau_u + \tau_d)} \tag{5.18}$$

This equation is computed iteratively several times for each unbound leaf, while a single iteration executes this equation once for each unbound leaf. Each iteration relaxes the potential field closer to a harmonic function. The actual values used by Equation 5.18 differ related to the used finite difference method. Using the octree and the Gauss-Seidel method the values taken into account for ϕ^h_l, ϕ^h_f and ϕ^h_d are the potential values newly computed in the current iteration, while the values for ϕ^h_r, ϕ^h_b and ϕ^h_u are the old potential values computed in the previous iteration.

The number of neighboring leaves of an octree leaf is not fixed. Several different neighbor leaves with different dimensions can exist. An approximation is introduced to reduce the complexity of the approach. Only the average distance τ_i from the current leaf l_i at position $p_{x,y,z}$ to the according neighbor leaves and the average potential value ϕ_i^h of the neighbor leaves in each direction are taken into account. The six directions of the von Neumann neighborhood are given by l and r representing neighbors in x - direction, f and b representing neighbors in y - direction and u and d representing neighbors in z - direction. The average potential value of the leaves in direction $\mathcal{D}$ is given by ϕ_i^h.

Considering a grid based data structure would lead to a more simple relaxation equation. Due to the different dimensions of the leaves the use of a tree based structure leads to an equation requiring much more computational effort. This is more than compensated by the abilities of the octree (cf. Section 4.2). The time needed to compute a potential field in the presented way is directly related to the number of unbound leaves. The number of iterations n_i needed to stabilize the potential field within a predefined precision $\epsilon < 10^{-s}, s \in \mathbb{R}^{>0}$ is approximately $n_i \approx \frac{1}{4}s \cdot n_l$ [188], whereby n_l denotes the number of leaves of the octree. The octree combines leaves unnecessary for current path computations and thus reduces n_l needed for potential field computations significantly as shown in Chapter 8.

Equation 5.18 is used to successively enhance the potential values of the unbound leaves. It is not necessary and due to architectural restrictions most times even not possible to compute a perfect harmonic function as a CPU does not have the abilities to compute infinitely long numbers. So, only an approximation valid for path computation is calculated. This is achieved by restricting the maximum number of iterations of the relaxation equation by three termination conditions that are introduced in the following section.

5.3.4. Performance Optimization

The highest computational effort of the potential field computation is undertaken in Equation 5.18. It is essential to decrease this equation's number of executions in order to compute paths online. Only discrete points of the harmonic function are approximated by the relaxation equation due to a processor with a given arithmetic accuracy and the fact that a perfect and continuous function is not necessary for path planning. An approximation of a harmonic function "good enough" to successfully compute feasible paths from $q_{\mathcal{I}}$ to $q_{\mathcal{G}}$ has to be calculated.

Three conditions have been introduced to reduce the necessary computation time. The execution of the relaxation stops, if one of the conditions is fulfilled and $-\nabla$ is used to compute a path. The conditions are:

1. No reachable local equilibria are left in the potential field.
2. The mean potential value change is less than a certain threshold.

3. The iteration depth is greater or equal to the number of unbound leaves multiplied by a certain threshold.

To count the number of reachable local equilibria, every node is first checked whether its potential value is less than the potential value of the leaf enclosing the UAV. If this is true, each neighboring node of the leaf is checked whether it has a potential value less than the current leaf. The leaf is not a local minimum in this case. The single neighbor leaves are required by the relaxation equation and have been computed before. The comparison of two values does not take long. So, counting the number of local equilibria can be performed relatively fast.

Condition one is fulfilled if no leaf with a potential value less than the leaf containing the UAV is a local equilibrium (cf. Definition B.9). The descent gradient will never lead the UAV to a leaf obtaining a higher potential value than the initial leaf. From this it follows that a leaf representing a local equilibrium will never be reached if it obtains a higher potential value than the leaf currently enclosing the UAV.

It is assumed that the changes of the single potential values are too small to expect the removal of the remaining local equilibria, if condition two is fulfilled. Finally, condition three guarantees the termination of the algorithm. It turned out during evaluation that these conditions are a good tradeoff between the time needed for computations and efficient environmental exploration (cf. Chapter 8). Changing the conditions would lead to increasing or decreasing durations necessary to compute a potential field. Using these conditions it is no longer mathematically guaranteed that UAVs will never get stuck in local equilibria. Thus, the A^*-algorithm [82] has been introduced as fallback option for path computations to a randomly chosen unexplored leaf if a UAV gets stuck in a local equilibrium.

The computation times depend directly on the number of nodes taken into account for the calculation of the potential field (cf. Section 5.3.3). The octree provides the ability to decrease the number of nodes needed to compute potential fields leading to feasible paths. Therefore, nodes further away than distance $\tau_{x,y,z}$ to the current position of the UAV are used for calculations instead of their leaves. This reduces the number of nodes dependent on the setting of $\tau_{x,y,z}$. It simultaneously forces more potential field computations to reach target configurations further away than $\tau_{x,y,z}$. The number of necessary iterations increases significantly with an increasing number of nodes used to compute the potential field. Thus, more computations with less nodes should be performed faster than a single computation with a high amount of nodes.

Another part of decreasing the computational times is provided by the implementation of the approach. An algorithm for efficient potential field computation has been designed. It benefits from the requirement that the octree has to be static during potential field computations. A description of the algorithm is presented in the following section.

5.3.5. Algorithm

The termination conditions introduced in the previous section lead to a tradeoff between the time needed to compute a potential field and accuracy of the harmonic function. The number of leaves considered by the algorithm has been reduced through the use of an octree instead of a grid. This also lowers the number of necessary iterations. Additionally, an appropriate implementation of the algorithm minimizes the computation time.

The algorithm used by the UAVs is presented in detail in Appendix A.5. It mainly benefits from the fact that the octree has to be static during potential field computations. Keeping the octree dynamic during potential field calculations can make it impossible to compute a potential field "good enough" for path planning. Always new points of the harmonic function represented by new leaves could occur and vanish making it impossible to approximate the potential field to a given accuracy.

The algorithm performs pre-computations while establishing the Dirichlet Boundary Condition. A pointer to each unbound leaf relevant for the current potential field computation is stored in a list together with pointers to all neighboring nodes. Coincidentally, the average distances to the neighboring nodes are added and the resulting values of $\tau_u\tau_d\tau_f\tau_b$, $\tau_u\tau_d\tau_l\tau_r$ and $\tau_f\tau_b\tau_l\tau_r$ are computed for each leaf and stored, too. These values are part of Equation 5.18. They are also constant for each node during the complete potential field computation. Storing these values leads to a rigorous reduction of the computational effort needed to compute an relaxation iteration.

A leaf must be selected to become part of the next target configuration if the current task of the UAV is exploratory navigation. During pre-computations and establishment of the Dirichlet Boundary Condition all leaves are checked whether they fulfill the requirement for being the next target leaf (cf. Section 5.4.1). The leaf closest to the UAV and fulfilling the requirements becomes the next target leaf. Tests considering different environments have shown good results using this selection [5].

The resulting list created by the pre-computations is then looped once and for each leaf in the list Equation 5.5 is computed. Afterwards, looping the list is iteratively conducted until at least one of the termination conditions introduced in Section 5.3.4 is fulfilled. The resulting potential values of the single leaves finally represent the potential field.

Nevertheless, it is possible to reduce the computation time even further. Multi-core processors can compute the potential field in a parallelized manner. Equation 5.18 is executed frequently and consists of many floating point operations. This predestines the equation to be computed by a graphics card optimized for fast floating point computations.

The resulting computation times for $\mathcal{C}$ in a concrete environment are presented in Chapter 8. Feasible path planning based on the computed potential field is possible as described in the next section.

5.4. Path Planning

The UAVs use the descent gradient of the resulting potential field in order to compute feasible paths. Each leaf of the octree that contains another UAV and each leaf other UAVs want to reach next is treated as $\mathcal{C}_{\mathcal{O}}$ to ensure collision avoidance. Inter-UAV communication is used to exchange information about those leaves.

The combination of the harmonic potential field and the octree allows the computation of a discrete path ρ_d. This path is computed using the descent gradient of the potential field and consists of a finite set of positions given by the center positions of the selected leaves, whereby only leaves $l_i, l_j \colon \{l_i, l_j\} \in \mathcal{N}$ lead to two consecutive positions in ρ_d. Areas, unreachable for the UAVs are treated as $\mathcal{C}_{\mathcal{O}}$ (cf. Section 4.2).

Figure 5.1 shows a two-dimensional potential field, which is an approximation of a harmonic function. A grid-based two-dimensional representation is used due to visualization reasons. Comparing the figure with a real harmonic function as, e. g., illustrated previously in Figure 3.3 shows that the presented function is only a rough approximation. It is smooth in the area of unbound leaves, but the unbound values differ much from the bound values. This is an evidence that the function is only a rough approximation.

The potential field has been achieved by computing the unbound values using the function $\log(\tau_{x,y,z})/\log(d)$ as first approximation and conducting five iterations of the relaxation equation for each unbound leaf. The value d denotes the diagonal of the terrain. The descent gradient of the potential field leads a UAV from $q_{\mathcal{I}}$ to $q_{\mathcal{G}}$ without taking a configuration $q \in q_{\mathcal{O}}$. Thus, the potential field is considered to be "good enough" for path planning. The resulting path ρ_d is illustrated by the black line in Figure 5.1.

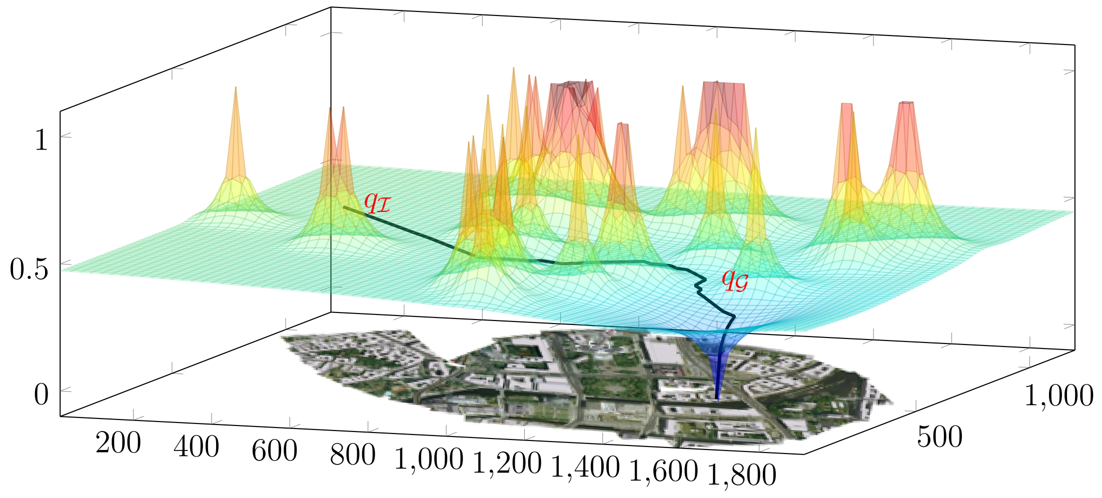


Figure 5.1.: *A two-dimensional potential field. It is an approximation of a harmonic function "good enough" for path planning.*

A fallback option is introduced in case of approximation and discretization errors leading a UAV into a local equilibrium. Section 5.5 shows that the approach is sound (cf. Definition B.6) if the approximation and discretization is not taken into account. So, selecting adequate approximations and a fine discretization leads to a sound path planning solution.

Collision avoidance is based on the occupied space, as described in Section 5.3.1. The leaves of the octree enclosing the UAVs, as well as the leaves enclosing the next route point of the UAVs are assigned to $\mathcal{C}_{\mathcal{O}}$. The current routes of the UAVs are deleted if the distance between two ore more UAVs gets below a given threshold and new routes with only one path point are computed until the distance between the UAVs gets above the threshold again. The threshold can be the sum of the dimensions of the UAVs multiplied by two. The presented collision avoidance is used for UAVs not flying in formation. If the UAVs fly in formation, an adaptation of this approach is used as presented in Section 6.4.2. The behavior for path planning is extended in order to achieve entire exploration of environments as described in the next section.

5.4.1. Exploratory Navigation

The environment is considered to be completely explored if condition $\mathcal{C}_{\mathcal{E}} = \mathcal{C}_{\mathcal{F}}$ is fulfilled. The UAVs select new leaves l_i as part of their next target configuration as long as leaves $l_i \in \mathcal{C}_{\mathcal{F}} \setminus \mathcal{C}_{\mathcal{E}}$ exist in order to reach a complete exploration.

Each UAV sends the position of the leaf it will explore next to the other UAVs. So each UAV knows which areas are explored next by fellow UAVs. The UAVs use this knowledge and do not choose leaves, selected by other UAVs as target leaves, to be explored next. This decreases redundant exploration especially if only a small part of the environment is unexplored.

As mentioned in Section 4.3, the camera pictures taken by the UAVs should fit given quality requirements. To achieve this, the UAVs do not explore the environment once by flying at their maximum altitude, which would be the fastest way. They first explore at their favorite altitude $q_4 \in q_{\mathcal{U}}$ (cf. Section 4.3), thereafter they explore at their favorite altitude times two and so on, until they reach their maximum altitude. Each leaf encompasses a three-dimensional area from $(x_{min}, y_{min}, z_{min})$ to $(x_{max}, y_{max}, z_{max})$. A leaf encloses the favorite altitude of a UAV iff $z_{min} < q_4 \leq z_{max}$. The UAVs first treat all leaves assigned to $\mathcal{C}_{\mathcal{UX}}$ and to $\mathcal{C}_{\mathcal{UK}}$ and additionally enclosing q_4 as target space for exploratory navigation. One of the bound target leaves is selected to be part of $q_{\mathcal{G}}$ for the computation of the first approximation for the remaining unbound leaves.

Exploration of the areas above the favorite altitude is necessary as the UAVs cannot assume that they have all environmental information and, e. g., bridges on which there are casualties during a rescue mission might exist above the favorite altitude. If all nodes are explored this way, nodes below the favorite altitudes of the UAVs are left. Such nodes can be inside of buildings, in tunnels, etc. and are explored last. This order is only a selection for the system presented in this thesis. It can be changed if, e. g., buildings should be explored first.

To actually explore the environment, a UAV selects the unknown or unexplored leaf l_i at the given altitude that is closest to its position and currently not selected by another UAV as next to be explored. Thereafter, it computes a potential field and uses the descent gradient of the field to create a path to the selected leaf. After path computation to the selected unexplored leaf l_i it checks whether the neighboring leaf of l_i in its flight direction is also unexplored or unknown. If this is true, it adds the leaf to its path and then checks the following leaf until an explored or occupied leaf is reached. This decreases the number of computations needed to entirely explore an environment.

As described in Section 5.3.4, not only the leaves of the octree but the nodes further away from the UAV than distance $\tau_{x,y,z}$ are used for potential field computations in order to reduce the number of nodes used for the computation. Such nodes contain different space types. They are treated as explored if they contain explored space. The nodes are considered as unexplored space if they contain only unknown and unexplored areas. After potential field computation, the leaves of these nodes are assigned to a potential value of one if they are occupied, to a potential value of zero if they are unexplored or unknown and to the value of the parent node in all other cases. The descent gradient used for path planning takes only leaves into account. Assigning the occupied leaves to a potential value of 1 still ensures obstacle avoidance. Explored leaves are assigned to the same potential value leading to flat regions. A path ends in such a region and a recalculation of the potential field takes place. The UAV is then inside the flat region such that the single leaves of this region are taken into account for the next potential field computation and the flat region vanishes.

Flying at the favorite altitude is only possible if the UAVs have terrain information. In case that they must explore an unknown environment, they compute the favorite altitude for the leaves to be explored next based on their current knowledge. Therefore, they assume that the height of the ground below the unknown leaves is equal to the height of the ground below the UAV. This might not be true but it is most times a good assumption as the UAVs select the unknown leaf closest to their position.

The UAVs can be equipped with front cameras, as mentioned in Section 4.3.1. Using these cameras, they are able to detect obstacles intersected by their trajectory. A UAV detecting such an obstacle maps it into its world model and deletes the current path if the path intersects with the obstacle. This then forces a new path computation taking the newly detected obstacle into account. Additionally, the UAVs have to check which parts of the environment are reachable. Thereby, each time their path is intersected by newly explored obstacles, they check for the reachable environment as described in Section 4.2.2 using the algorithm presented in Appendix A.2.

A simplified example for exploratory navigation is presented in the following. It is assumed that the underlying environment is flat. In this case, the UAVs fly always at a consistent altitude, as long as leaves are unexplored at the corresponding height. This allows a representation in two dimensions to facilitate the example. Considering only two dimensions results in a quadtree, as each node is decomposed into four child nodes if the height is neglected.

A single UAV is considered to conduct exploratory navigation in an a priori known environment. Its position is marked in Figures 5.2 and 5.3 by a red cross. Additionally, an occupied areas has been introduced. It is represented as black part of the environment. The resulting decomposed environment before exploration takes place is illustrated in Figure 5.2 a).

The UAV first starts with obstacle growing. Thereafter, it checks for areas unreachable due to given occupied nodes. This results in a large occupied area shown in the upper right of Figure 5.2 b). As soon as all occupied areas are internally represented by the UAVs' data structure exploratory navigation starts. It is assumed that the UAV is able to explore the node it is enclosed by, as well as all neighboring leaves for which the distance between the UAV and the neighboring node is less than half of a maximum level leaf's corresponding dimensions. Based on this exploration capabilities, the UAV is able to explore four leaves while hovering at the start position. Explored areas are illustrated gray while unexplored areas are white in the figures. The explored leaves are not combined as the UAV is enclosed by one of them and leaves enclosing a UAV are never combined due to several reasons.

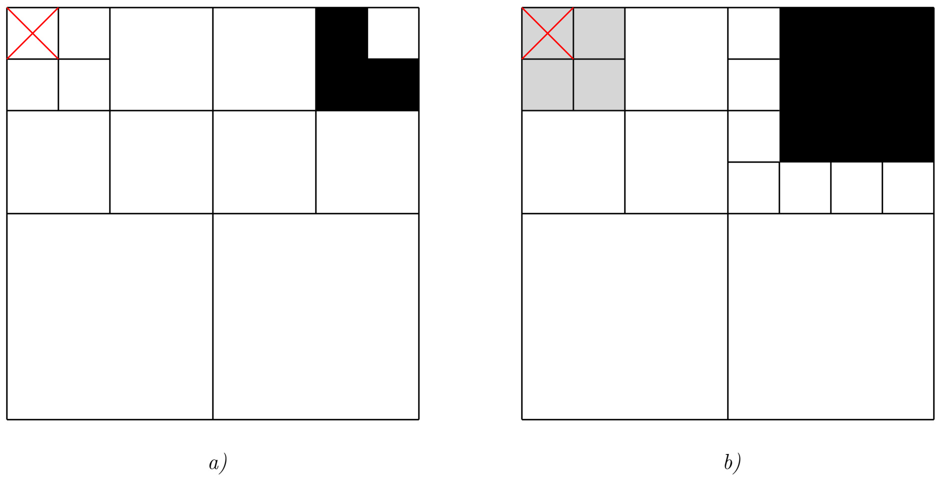


Figure 5.2.: *The start state is presented in a) while the first status before path planning for exploratory navigation starts is shown in b).*

Based on the UAV configuration a path has to be computed using the descent gradient $-\nabla$ of the harmonic potential field. Two unexplored leaves are at a equal distance to the UAV. These are the unexplored leaves on the right hand side and below the current UAV position in Figure 5.2 b). The UAV choses one of the tow alternatives as next to be explored dependent on the implementation and uses it for the computation of the first approximation. The descent gradients would be equal for both possible leaves if a perfect harmonic function could be computed. This is most times not the case as only a rough approximation is calculated by the UAVs and therefore, they favor the selected leaf. It is assumed that in this case $-\nabla$ leads the UAV to the leaf below its position. A path consisting of two route points is created to reach the leaf. Dependent on the direction the UAV flies it checks whether the neighboring leaf of the leaf at the end of the path is unexplored, too. This is true in this example and the leaf is also added to the path resulting in a new path length of three illustrated as red line in Figure 5.3 a).

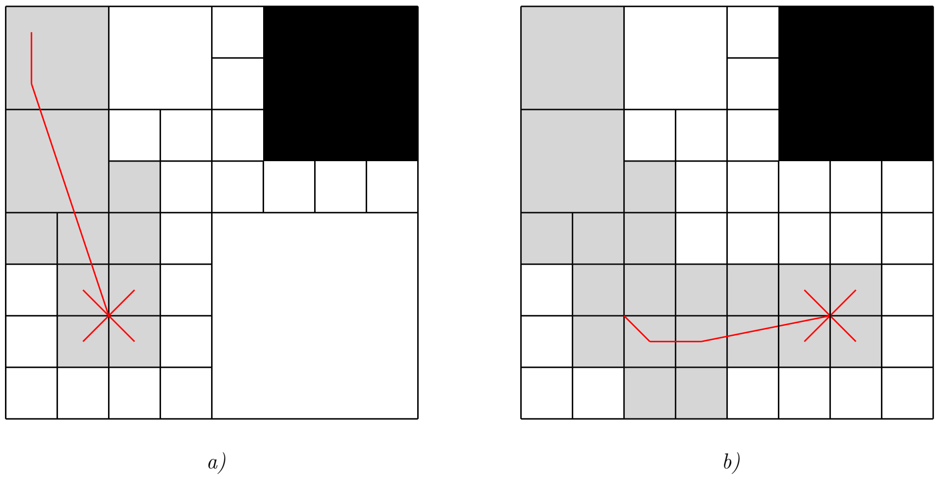


Figure 5.3.: *The environment after the first exploration target has been reached is illustrated in a) and the current state after the second exploration has been reached is depicted in b).*

The occupied area in the left part of the environment has no influence to the first route computation. It is surrounded by unexplored space which is assigned to the bound value of 0 and thus never changes. Therefore, only the unexplored areas, which are neighbors of the four explored leaves influence the unbound values.

The UAV flies from its start position the the center of the leaf which is at the end of the path. During flights it explores the area resulting in a new decomposition of the environment. It finally reaches the end point of the path marked as red cross in Figure 5.3 a). Due to cell decomposition the further leaf selected by the UAV became a node and has been subdivided into four child nodes.

Based on the configuration of the UAV in Figure 5.3 a) it is most likely that the UAV will move towards one of the two leaves below it. This is as these two leaves are mainly influenced by the unexplored space while the other two leaves are much stronger influenced by the leaves explored before during the relaxation. Thus, it is assumed that the UAV flies in the next step to the leaf which is in the lower right of its position. A path of length two is computed to finally reach the leaf by flying in right direction. Again, it checks if right from the node is another unexplored leaf, which is the case and adds this leaf also to its path. After path following the UAV is at the position marked by a red cross in Figure 5.3 b). The newly explored leaves, as well as the resulting decomposition of the environment is also shown in the figure.

This exploration behavior is repeated as long as leaves at the favorite altitude exist. Thereafter, the UAV changes its flight altitude as aforementioned and starts exploring the environment at the new height using the methodology described in this section. Selecting leaves for the first approximations favor these ones to be explored next as the resulting harmonic function used for path planning is still only an approximation. It does not necessary guarantee that the selected leaf is the next one to be explored. The assignment of unbound and bound leaves must be changed in order to fly to a specific leaf as it has to be performed if a given area of the environment must be reached. In such cases only the leaves to which the UAV has to fly is considered as target and assigned to the corresponding bound potential value of 0.

To adopt the system to real UAVs used in different applications, a verification of the presented approach is necessary. The following section shows that it is possible to validate a real world system using the presented approach.

5.5. Verification

This section shows that the mathematical potential field approach based on harmonic functions is sound (cf. Definition B.6) by showing that the system can lead to asymptotically stable control signals (cf. Section 3.6). An asymptotic stable system always relaxes into its equilibrium. So, if the system is asymptotically stable, the UAVs will always move to their desired target configurations, which are represented as globally stable equilibria.

An additional verification for real world UAVs is required as, e. g., the mass of the UAV must be taken into account. Showing that a harmonic potential field can be used to create Lyapunov stable systems has been done several times before, e. g., by Masoud and Masoud [211] from whom the following verification has been adopted. They also prove several properties of harmonic functions like the min-max principle (cf. Definition B.2) and the uniqueness principle (cf. Definition B.3). The stability proof for harmonic potential fields is briefly presented in this section for the sake of completeness.

5.5.1. Lyapunov Function Candidate

In order to show that a system is asymptotically stable, it must first be shown that the resulting potential field is a Lyapunov function candidate. Let Ω be the closed domain on which ϕ is given (cf. Section 3.3.1). The workspace of the UAVs is then $\Omega_{\mathcal{F}} = \Omega - \phi^h_{\mathcal{O}}$.

Lemma 5.1. *The harmonic potential field $V(\phi^h_{x,y,z})$ generated by the Dirichlet Boundary Condition is a Lyapunov function candidate (cf. Section 3.6.2).*

Proof. As by definition $V(\phi^h_{\mathcal{G}}) = 0$ it is only necessary to show that

$$\forall \phi^h_{x,y,z} \in \Omega_{\mathcal{F}} \setminus \phi^h_{\mathcal{G}} \colon V(\phi^h_{x,y,z}) > 0 \tag{5.19}$$

in order to show that $V(\phi^h_{x,y,z})$ is a Lyapunov function candidate. This follows directly from the min-max principle ([158], cf. Definition B.2). Thus, $V(\phi^h_{x,y,z})$ is a Lyapunov function candidate. □

5.5.2. Convergence

The second step is to show that each trajectory generated by the potential field converges globally. Assume that for the dynamic system $\dot{\phi}^h_{x,y,z} = f(\phi^h_{x,y,z})$ there exists a scalar function $V(\phi^h_{x,y,z})$ satisfying the LaSalle principle from Section 3.6.3.

Theorem 5.1. *The trajectory generated by the descent gradient will globally and asymptotically converge to $\phi^h_{\mathcal{G}}$, i. e.,*

$$\forall \phi^h_{x,y,z} \in \{\phi | \phi(0) \in \Omega_{\mathcal{F}}\} \colon \lim_{t \to \infty} \phi^h_{x,y,z}(t) = \phi^h_{\mathcal{G}} \tag{5.20}$$

Proof. From Lemma 5.1 it is obvious that $V(\phi^h_{x,y,z})$ is a Lyapunov function candidate. Let ∇ be the gradient operator.A harmonic potential field is constructed by forcing the divergence ∇° of the gradient $\nabla V(\phi^h_{x,y,z})$ of the potential field $V(\phi^h_{x,y,z})$, which is according to [211] analogous to the electric current in a resistive grid, to zero inside $\Omega_{\mathcal{F}}$

$$\nabla^\circ \nabla V(\phi^h_{x,y,z}) = \nabla^2 V(\phi^h_{x,y,z}) \equiv 0. \tag{5.21}$$

That condition guarantees the continuity of the current. This in turn guarantees the continuity of motion inside $\Omega_{\mathcal{F}}$. One result of this is the prevention of deadlocks while motion is steered towards the global equilibrium of $V(\phi^h_{x,y,z})$. Additionally, Masoud and Masoud [211] state that the resulting dynamic system can only generate solution trajectories that are bound to $\Omega_{\mathcal{F}}$. The zeros of the gradient $\nabla V(\phi^h_{x,y,z})$ are finally the zeros that determine convergence.

Let $\Omega'_{\mathcal{F}}$ be the directionally constraint subset of $\Omega_{\mathcal{F}}$. Further on let $\partial\Omega'$ be the boundary of $\Omega'_{\mathcal{F}}$ and let $\partial\Omega'_b$ be the subset on $\partial\Omega'$ that forms locally unstable equilibria. Masoud and Masoud [211] state that the value of $\nabla V(\phi^h_{x,y,z})$ is zero for the following three cases:

1. The gradient of the system has stable equilibrium points at $\phi^h_{\mathcal{G}}$ by design.
2. The gradient field of every scalar potential contains at least one zero-measure set π of unstable equilibria [212].
3. It is possible that an unstable equilibrium zone may form a subset $\partial\Omega'_b$ of the interface between the nonlinear anisotropic region and the rest of the workspace ($\partial\Omega'$).

Let S be the set of all points where $\dot{V}(\phi^h_{x,y,z}) = 0$. From this it follows that

$$S = \phi^h_{\mathcal{G}} \cup \pi \cup \partial\Omega'_b. \tag{5.22}$$

The largest invariant set M contained in S consists of the only stable equilibrium subset of $S(M = \phi^h_{\mathcal{G}})$ [211]. Therefore, the motion of the gradient system will converge to $\phi^h_{\mathcal{G}}$, which follows from the LaSalle theorem ([178], cf. Section 3.6.3). □

This shows that it is possible to create control signals for real UAVs based on trajectories obtained by harmonic functions that are asymptotically stable and thus always resulting in the expected behavior as, e. g., shown in [152] for a system with second order dynamics. Nevertheless, it does not proof the entire system presented in this thesis. To actually proof the system a guidance law considering the mass of the UAVs, as well as the computation of the velocity is needed as, e. g., presented in [213]. Additionally, a validation of the implementation including the influence of the termination conditions has to be performed to entirely proof a system.

Additionally, a proof regarding optimality of the trajectories generated by the gradient is given in [211] by constructing the Dirichlet integral. This integral is globally minimized if the potential field satisfies Laplace's equation, which is called the Dirichlet principle. The paths of the UAVs are derived from the gradient of the potential field which minimizes the corresponding energy function. Thus, the achieved paths are optimal.

5.6. Summary

This chapter describes the computation of a three-dimensional potential field represented by an approximation of a single harmonic function. Computing the potential field is performed by a three step procedure: First the Dirichlet Boundary Condition is created. Afterwards, a first approximation of the harmonic function is computed. Finally, this is iteratively approximated towards the resulting function, until given termination conditions are fulfilled.

The resulting potential fields guarantee obstacle avoidance. The planning approach is also sound (cf. Definition B.6) if the approximation is close enough to a harmonic function. Several approximations are introduced in order to decrease the computational effort. To reduce the needed effort even more and to solve arithmetical restrictions, a skillful implementation has been developed. Previous tests showed that the UAVs are able to compute paths and to react on environmental changes in a fast and efficient way [3], [5], [4], when using this approach.

The UAVs reach an implicit coordination if they combine the path planning approach with inter-UAV communication leading to less redundant explorations of the single subareas of the environments. Nevertheless, the UAVs spread out in different directions, like foraging ants if several UAVs start exploratory navigation using this approach.

Additional abilities have to be introduced to reach a higher degree of coordination during exploratory navigation. This would lead to a more efficient behavior in terms of the duration an exploration of the complete environment lasts. Formation flights can lead to such an increase in efficiency. The following chapter introduces a method for formation flights. These formations can be established and dissolved anytime. It is also possible to change the shape of a formation anytime.

Chapter 6

Behavioral Formation Flights

The UAVs can fly to specific positions in order to solve several tasks using the previously introduced methods. Entire exploration of different environments is also possible through consecutive selection of unexplored leaves to be explored next until all leaves are marked as explored. Increasing the number of UAVs decreases the duration needed for exploratory navigation (cf. Chapter 8) due to a reduction of redundant exploration by inter-UAV communication. A higher degree of cooperation beside the one obtained through information exchange should lead to an increase in efficiency not only for the exploration task. Additionally, UAVs working cooperatively can fulfill tasks single UAVs are not capable to. Therefore, a cooperative behavior based on formation flights is presented in this chapter.

It would be unfavorable if all UAVs form a single formation to explore an environment whilst preserving the formation. Simultaneous establishment of several different formations and their modification should be possible all times.

It has to be considered that flying in formation leads to a more complex obstacle avoidance. The single formations should adapt themselves autonomously to overcome obstacles. Additionally, new tasks, more important than the one currently processed, can appear anytime. From this it follows that the formation has to be dynamic in terms of the number of participating UAVs. Another requirement is that the resulting approach has to work in combination with the harmonic potential field introduced in Chapter 5. Last but not least, the approach has to be mathematically verifiable to show that the expected behavior will always be reached.

An obvious solution is the extension of the harmonic potential field in order to fulfill the aforementioned requirements. An additional potential field leading to formations is created and the resulting two potential fields are superpositioned. The additional potential field is based on bifurcation theory (cf. Section 3.4.2) and allows an online change of formation shapes during formation flights. Such changes can be brought about by changing single parameters of the equation resulting in different potential fields. The superposition of the two potential fields is conducted in a way that preserves the advantages of both approaches, e. g., obstacle avoidance, soundness (cf. Definition B.6), etc.

The chapter first describes the problem of formation establishment and thereafter presents state-of-the-art approaches using bifurcation theory for formation flights. The following section explains the method used to compute the bifurcating potential field, as well as the superpositioned potential field. The single potential fields and possible formation shapes are illustrated thereafter to provide a better impression of the approach. The problem of finding consensus to determine a position used for formation establishment is addressed in the further course of the chapter. Formation handling including movement and rotation is another topic presented in this chapter. Finally, a verification of the bifurcating potential field and the superpositioned potential field takes places. The chapter concludes with a summary.

6.1. Problem Statement

Given $n \in \mathbb{N}$ UAVs $\mathcal{U}_i, 1 \leq i \leq n$, at different positions $p_{\mathcal{U}_i}$. Each UAV has $n-1$ distances $\tau_{\mathcal{U}_{i,j}} = |p_{\mathcal{U}_i} - p_{\mathcal{U}_j}|, i \neq j$ to the other UAVs. The definition of a formation is now based on the definition of a rigid body given by Arnold [214].

Definition 6.1. Formation: *The $n > 3$ UAVs are called a UAV formation $\mathcal{P}$, iff*

$$\forall 1 \leq i \leq n \, \exists j \neq k \in \{1, \dots, n\} \setminus \{i\} \colon \tau_{\mathcal{U}_{i,j}} = \tau_{\mathcal{U}_{j,k}} \tag{6.1}$$

For $n = 3$ a formation is given iff

$$\exists \mathcal{U}_i \neq \mathcal{U}_j \neq \mathcal{U}_k \colon \tau_{\mathcal{U}_{i,j}} = \tau_{\mathcal{U}_{i,k}} \tag{6.2}$$

Definition 6.1 takes the distances between the single UAVs into account. For $n = 3$ it ensures that always one UAV exists, which has equal distances to the other two UAVs. To provide a better impression of the possible formations, two examples are illustrated for $n > 3$n Figure 6.1.

Various formations can be created using this definition like a V-formation illustrated by the UAVs $\mathcal{U}_1$ to $\mathcal{U}_5$ presented in Figure 6.1. The definition considers three UAVs $\mathcal{U}_j$,$\mathcal{U}_i$,$\mathcal{U}_k$. In case of the V-formation $\mathcal{U}_1$ can be assumed as $\mathcal{U}_j$ with equal distances to $\mathcal{U}_2$ and $\mathcal{U}_3$. Taking $\mathcal{U}_2$ as $\mathcal{U}_j$ leads to equal distances to $\mathcal{U}_1$ and $\mathcal{U}_4$ while $\mathcal{U}_3$ has equal distances to $\mathcal{U}_1$ and $\mathcal{U}_5$. Thus, the V-formation is valid using Definition 6.1.

The same holds true for the quadratic formation illustrated at the right hand side of Figure 6.1. $\mathcal{U}_6$, e. g., has equal distances to $\mathcal{U}_7$ and $\mathcal{U}_8$. This is analogously for the other UAVs and therefore, this formation is also valid using Definition 6.1.

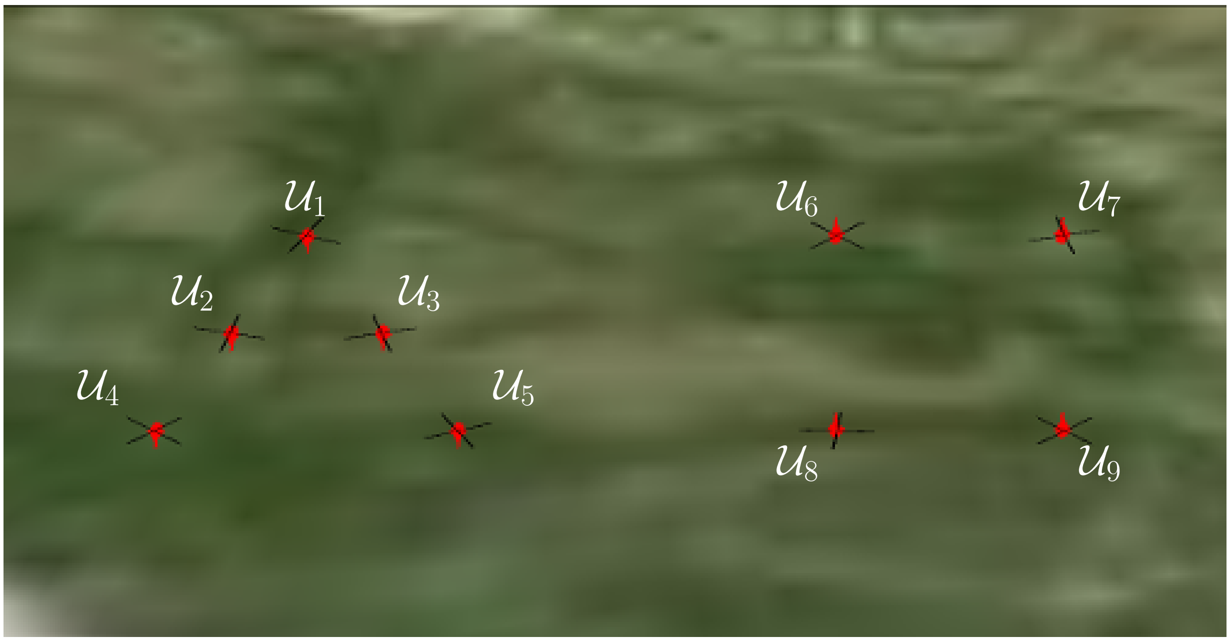


Figure 6.1.: *Two formations shapes using Definition 6.1.*

The formation approach described in the further course of this chapter leads always to arrangements of the UAVs that is a formation by Definition 6.1, as long as the number of UAVs fits the requirement for $n \geq 3$ as mentioned after the introduction of the potential field. The problem is now to preserve $\mathcal{P}$ during UAV flights as long as the formation is not dissolved and no environmental influences like obstacles cause a change of single positions $p_{\mathcal{U}_i}$.

Several very different approaches for the creation of formations exist. A selection of general state-of-the-art approaches is presented in Section 2.3.3. The use of artificial potential fields for the creation of formations is also performed nowadays. The objective of this chapter is to create potential fields based on a single equation that provides adequate equilibria for the creation of formations. Similar to potential field computation for path planning no standard method for the computation of potential fields utilizable to create and move formations exists. The next section presents a selection of current research in this field.

6.2. State-of-the-Art

First approaches of artificial potential fields for coordinated path planning of multiple robots have been introduced decades ago. Warren [215], e. g., presented such an approach in 1990. Thenceforth, a wide range of extensions and combinations of potential fields with other methodologies have been designed to solve the problem of coordinated robot movement.

Bentes and Saotome [216] use potential field theory in combination with the A^*-algorithm to control dynamic swarm formations in two-dimensional and three-dimensional environments. The A^*-algorithm is used for the computation of global and obstacle free paths. The potential field locally controls the swarm formation while following the planned trajectory. Bivariate functions (cf. Definition B.10) are used to compute a potential field for formations in the two-dimensional case and multivariate functions are used in the three-dimensional case. They are able to create ring like formations and move them by simultaneously avoiding obstacles.

Blazovics et al. [217] have designed a completely distributed algorithm to surround moving targets in formation. Therefore, they use two different kinds of trajectories. A superpositioned potential field based on three different potential fields is created for this purpose. They use the two common kinds of potential fields (cf. Section 3.3). The first attracts the entities of their swarm towards the target while the second simultaneously repels individuals from the target. Surrounding trajectories at a specific distance are reached through using these two potential fields. Additionally, a radial potential field is computed to move the swarm entities on a circular trajectory. They achieve a superpositioned field, which lets the swarm entities surround the target by defining an appropriate weight function for the superpositioning of the three single potential fields.

The work of Garcia et al. [218] tries to integrate UAVs into teams of both, manned and unmanned aerial vehicles. They use weighted potential fields to create the shape of an unmanned formation with values based on the state of the unmanned and manned assets including the desired formation, obstacles, task assignments, and perceived intentions. A sigmoid function weights the single potential fields. Navigation and stability is reached through the use of fuzzy logic controllers and a fuzzy reasoning engine predicts the intend of surrounding aircrafts. Formations are achieved by the selection of individual target positions of each UAV with respect to the formation based on the position of a manned aircraft and the required formation shape.

An approach for formation flights with potential fields based on bifurcation theory is presented by Bennet et al. [219]. They create potential fields by superpositioning attractive and repulsive potential fields for a sixth degree of freedom, fixed-wing UAV model. This potential field is transformed into a guidance law and a control law. The guidance law leads to a forward speed, as well as heading and pitch angles. The control law is responsible for the computation of longitudinal and lateral motion control. The approach leads to three-dimensional formation shapes. Line and double line formations with changing separation distances between the UAVs during flights have been achieved through their steering potential field.

The approach most similar to the one used by the UAV system presented in this thesis is the one designed by Bennet et al. [219]. In contrast to the UAV approach presented in this theses they focus on the computation of guidance laws for fixed-wing UAVs. These guidance laws result in reactive behavior and do not involve path planning. They also neglect the problem of local equilibria by assuming that through the dynamics of the system an equilibrium created at time t vanishes at time $t + k$.

The following section describes the potential field approach used by the UAVs of the resulting system. This potential field is combined with the harmonic potential field introduced in Section 5 to plan paths the UAVs can follow in formation.

6.3. Bifurcating Potential Field

The presented approach for the creation of formations results from a superpositioning of the former described harmonic potential field and a bifurcating potential field based on a pitchfork bifurcation [164]. Bifurcation theory allows to change the number of global equilibria leading to a change of the formation shape. It is also possible to influence the resulting potential field such that different types of increasing potential values ϕ are achieved. This allows the UAVs to position themselves in specific distances to each other in order to follow Definition 6.1.

The main idea is to create a potential field that forces the UAVs to arrange themselves in a specific formation. A line formation shape can, e. g., be created by an increase of the potential values, which is in one dimension lesser than in other dimensions.

Using a bifurcating potential field has several advantages. A change of the formation shape is, e. g., achieved by a single value change of the underlying equation. Another advantage is that the behavioral formation flights introduced in Section 3.4.1 are achievable. Additionally, the formations scale well with the number of participating UAVs.

The local stability properties of equilibria or other invariant sets change at a bifurcation (cf. Section 3.4.2). Based on the normal form of the Pitchfork bifurcation [164] an extension has been performed to compute a highly scalable potential field for the establishment of formations in three-dimensional environments. The normal form of the pitchfork bifurcation is

$$\frac{dx}{dt} = \mu x + \alpha x^3. \tag{6.3}$$

The extension of Equation 6.3 is described in detail in the following section leading to an equation for the computation of the bifurcating potential field.

6.3.1. Formation Potential Field

The bifurcating potential field works similar to the harmonic potential field by pulling the UAVs towards global equilibrium positions. In contrast to the harmonic approach it is relatively easy to compute several global equilibria forming, e. g., a line and thus leading to a line formation. The normal form from Equation 6.3 needs to be extended for the computation of a potential field suitable to establish different formations in three dimensions. In the sequel, the resulting Equation 6.6 will be developed in a stepwise manner.

To actually take three dimensions into account y - and z - dimensions have been added to Equation 6.3. This was conducted as shown in Equation 6.4. The resulting equation allows the creation of global equilibria in a three-dimensional space. The exponent has been increased to ensure that $x = y = z = 0$ and $\mu \geq 0$ results in the zero of the first partial derivative, which denotes the equilibrium position. Due to the properties of bifurcation theory the number of equilibria can be altered by just changing the sign of μ (cf. Section 3.4.2).

$$\phi^{b}_{x,y,z} = \mu\left(\bar{\tau}_x^2 + \bar{\tau}_y^2 + \bar{\tau}_z^2\right) + \alpha\left(\bar{\tau}_x^4 + \bar{\tau}_y^4 + \bar{\tau}_z^4\right) \tag{6.4}$$

The symbol b denotes the potential values to be values of the bifurcating potential field. Additionally, the normalized distances $0 \leq \bar{\tau}_i < 1$ are used instead of the coordinates x, y and z in order to be able to create equilibria at every position inside the environment. Normalizing the distances is a first step to achieve normalized potential values combinable with the harmonic potential values $\phi^{h}_{x,y,z}$. Such potential values should also be in a value range with the same size than the range of the harmonic potential values such that not one of the potential field dominates the other while superpositioning the two potential fields. Normalization is achieved through a division of the resulting distance in each direction by the length of the environment in the corresponding direction.

The value α is responsible for the amplitude of the resulting potential field. The UAVs work only with a subcritical bifurcation, i. e., $\alpha \in \mathbb{R}^{>0}$ (cf. Section 3.4.2). Changing the amplitude increases or decreases the distance between the equilibria if more than one equilibrium exist. Changing the sign of μ results in a change of the current formation shape. Equation 6.4 leads to a single equilibrium if $\mu \geq 0$. All UAVs will fly to this equilibrium and if they repel each other a cluster formation is created. Setting $\mu < 0$ results in eight stable and one unstable equilibrium positions guiding the UAVs into eight cluster formations.

The resulting formation is based on the position and the number of created equilibria. To create line or V-formations, it is necessary that the single dimensions can be weighted. Parameters $a, b, c \in \{0, 1\}$ have been introduced to be able to specify the influence of the dimensions as shown in Equation 6.5.

$$\phi^{b}_{x,y,z} = \mu\left(a\bar{\tau}_x^2 + b\bar{\tau}_y^2 + c\bar{\tau}_z^2\right) + \alpha\left(a\bar{\tau}_x^4 + b\bar{\tau}_y^4 + c\bar{\tau}_z^4\right) \tag{6.5}$$

A superpositioning of these values with the values from the harmonic potential field would lead to unbound values greater than one, which again leads to obstacle collisions. Therefore, the potential values are shifted by $v \in \mathbb{R}^{>0}$. This results in a potential value of $-v$ at the equilibrium position. Values greater than zero are still computed. They vanish by the superposition described in Section 6.3.2.

Finally, Equation 6.6 creates a steering potential field providing abilities for the creation of various formation shapes. The bifurcating potential fields are created through computing the single $\phi^b_{x,y,z}$ using this equation for each leaf marked as $\mathcal{C}_\mathcal{F}$. The repelling potentials needed for collisions avoidance are given by the harmonic potential field (cf. Section 5.3.3) and will be considered during superposition.

$$\phi^b_{x,y,z} = \frac{\mu(a\bar{\tau}_x^2 + b\bar{\tau}_y^2 + c\bar{\tau}_z^2) + \alpha(a\bar{\tau}_x^4 + b\bar{\tau}_y^4 + c\bar{\tau}_z^4)}{u} - v \tag{6.6}$$

The value $u = (|\mu| + \alpha) \cdot (a + b + c)$ ensures that $0 \leq \phi^b_{x,y,z} < 1$. A potential field containing two global stable equilibria fronts can be computed if, e. g., $\mu < 0, \alpha > 0, a = 1, b = c = 0$. These equilibria are created with some distance to the target configuration. In order to create, e. g., a double line formation with a specific distance between the single lines, it is necessary to know the positions of the equilibria. They are at the zeros of the first partial derivatives (cf. Section 3.4.2) shown in Equations 6.7 to 6.9.

$$\frac{\partial \phi^b_{x,y,z}}{\partial \bar{\tau}_x} = \frac{2\mu a\bar{\tau}_x + 4\alpha a\bar{\tau}_x^3}{u} \tag{6.7}$$

$$\frac{\partial \phi^b_{x,y,z}}{\partial \bar{\tau}_y} = \frac{2\mu b\bar{\tau}_y + 4\alpha b\bar{\tau}_y^3}{u} \tag{6.8}$$

$$\frac{\partial \phi^b_{x,y,z}}{\partial \bar{\tau}_z} = \frac{2\mu c\bar{\tau}_z + 4\alpha c\bar{\tau}_z^3}{u} \tag{6.9}$$

The zeros of the partial derivatives depend only on the values of α and μ. Only one stable equilibrium resulting from the zero of the first partial derivatives and shown in Equation 6.10 exists if $\mu \geq 0$. Given $\mu < 0$, Equation 6.11 presents the zeros of the first partial derivative which lead to stable equilibria. Equation 6.12 presents the zero leading to an unstable equilibrium, which is equal to the stable equilibrium if $\mu \geq 0$. The determination of the stable and unstable equilibria is described in Section 6.5.2. The zeros depend only on μ and α. So, the distances between, e. g., two lines can be changed by just changing α and μ.

$$\tau_i = 0, \qquad \mu \geq 0 \tag{6.10}$$

$$\tau_i = \pm\sqrt{\frac{-\mu}{2\alpha}}, \qquad \mu < 0 \tag{6.11}$$

$$\tau_i = 0, \qquad \mu < 0 \tag{6.12}$$

Considering Definition 6.1, a formation is always created if the number of UAVs is equal to or higher than three and one stable equilibrium exists as long as no environmental influences disturb the UAVs. This results from the symmetry of the potential field. All UAVs of a formation fly to the equilibrium position where they repel each other. Every repulsive force acts alike at equal distance to the UAV creating it. Thus, the UAVs arrange themselves at a given distance to each other. In case of multiple equilibria, it has to be ensured that always a number of UAVs unequal to two flies to each single equilibrium. Then, a formation is created at each single equilibrium resulting in a number of formations equal to the number of stable equilibria if Definition 6.1 is taken into account. This behavior is sufficient as it is insignificant for the approach whether one or multiple formations move consistent. Formations like the quadratic one illustrated in Figure 6.1 can also be established by the simple creation of four equilibria positions. These positions are all at the same distance to each other as Equation 6.11 shows. Thus, using Definition 6.1 results in a sufficient number of possible formations for this thesis.

The next step to reach a potential field combining the advantages of both approaches is to superposition the bifurcating potential field computed by the use of Equation 6.6 and the harmonic potential field introduced in Section 5.3 using Equations 5.5 and 5.18 thus that the advantages of both potential fields are still valid.

6.3.2. Superpositioned Potential Field

A superpositioning of the harmonic and the bifurcating potential field is performed to keep the advantages of both potential field approaches. As shown before, a harmonic potential field is globally asymptotically stable (cf. Section 5.5). Hence, this potential field should be used to lead the UAVs towards the desired formation area. At this area the bifurcating potential field should take over the control of the UAVs in order to create specific formation shapes.

The bifurcating potential field neglects the obstacle areas of the environment. These areas are bound fixed to a potential value of one due to the use of the Dirichlet Boundary Condition and are considered by the harmonic potential field. Therefore, the influence of the harmonic potential field should never vanish. This leads to the resulting superposition of the unbound leaves:

$$\phi_{x,y,z} = \begin{cases} \phi^b_{x,y,z} + w \cdot \phi^h_{x,y,z}, & \text{if } \phi^b_{x,y,z} \leq 0, \\ \phi^h_{x,y,z}, & \text{else.} \end{cases} \tag{6.13}$$

The value $0 \leq w \leq 1$ is responsible for the influence of the harmonic potential field close to the formation position. The final potential field $\phi_{x,y,z}$ used for flights in formation is achieved by the case differentiation shown in 6.13. The bifurcating potential field affects the superpositioned potential field only close by the equilibrium positions depending on the selected value v from Equation 6.6. This guarantees that the UAVs always fly towards the equilibrium positions due to the influence of the harmonic potential field. Superpositioning is only performed for the unbound potential values. The bound values stay fixed to the ones assigned for the computation of the harmonic potential field. The resulting behavior of the UAVs influenced by the bifurcating potential field is described in Section 6.5.

The superpositioned potential field $\phi_{x,y,z}$ is used by the UAVs to create formations and to fly in formation. The harmonic potential field still contains the repulsive potential values achieved from the Dirichlet Boundary Condition and ensures that the UAVs will not only fly to the global equilibria but are also repelled by each other and by each obstacle. Reducing the harmonic potentials results in higher influence of the bifurcating potential field responsible for the establishment of formations. The achieved behavior leads to the desired formation based on the selected values for the a, b, c parameters from Equation 6.6. The influence of the harmonic potential field at the formation position allows the formations to dynamically change their shapes in respect to obstacles arranged close by the trajectories.

Only potential values of the bifurcating potential field less or equal to zero are used for superposition. Thus, the resulting potential values are less or equal to the potential values of the harmonic potential field and the use of the negative gradient still ensures collision avoidance as no unbound potential value can become greater or equal to one. Example potential fields resulting from a harmonic function, a bifurcation, and the described superpositioning are illustrated in the following section in order to give a better impression of the approach.

6.3.3. Resulting Potential Fields

The UAVs compute potential fields to simultaneously explore complex three-dimensional terrains entirely in different formations using Equation 6.13. Formations can be established, changed, and dissolved by the UAVs anytime. This section illustrates the three different potential fields in order to visualize the approach.

To provide an easy and understandable visualization, the presented potential fields consider two-dimensional terrains using a grid as underlying data structure and an obstacle free environment. The size of the environment is 1000×1000 while the overall objective is to establish a line formation at the center of the environment considered as position $(0, 0)$. This leads to an environmental scaling from -500 up to 500 in x - and y - direction. Measurement units are neglected as the approach works independent of such units.

The first potential field computed by the UAVs is always based on harmonic function theory (cf. Figure 4.4). Figure 6.2 illustrates an example potential field whereby the single potential values are given by $\phi^h_{x,y} = \frac{\log(\tau_{x,y})}{\log(d)}$ with d denoting the diagonal of the environment and τ denoting the Euclidean distance of the single areas to the target position. The target configuration $q_{\mathcal{G}}$ contains the position $(0, 0)$ as target position leading to a global equilibrium at this position. The figure shows that all streamlines lead to $q_{\mathcal{G}}$ independent of the initial configuration $q_{\mathcal{I}}$. From this it follows that a UAV is always able to reach the target configuration using the descend gradient $-\nabla$.

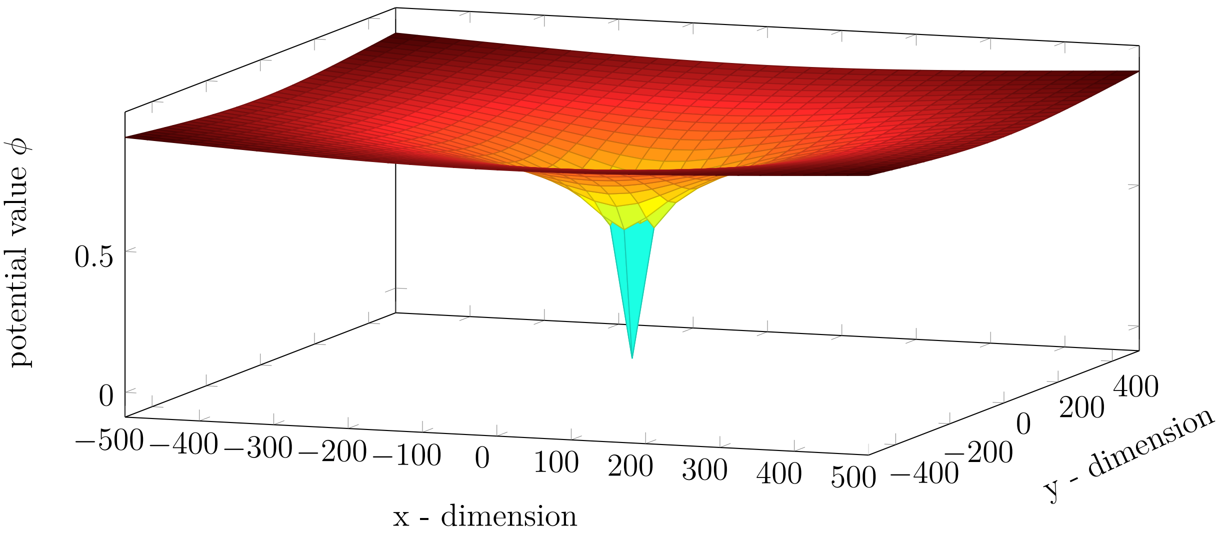


Figure 6.2.: *Potential field of an example harmonic function. This kind of potential fields is used for path planning of single UAVs.*

A bifurcating potential field computed using a two-dimensional version of Equation 6.6 is presented in Figure 6.3. The bifurcation parameter μ and the amplitude α of Equation 6.6 are assigned to the value one. It has been shifted by setting $v = 1$, which results in a complete superpositioning. The objective is to compute a potential field resulting in a line formation in y - direction at $x = 0$. This can be achieved by setting $b = 0$, in order to only take the distances in x - direction into account.

Figure 6.3 shows that the bifurcation function creates a potential field, wherein all streamlines lead to positions at $(0, \cdot)$. The use of the negative gradient $-\nabla$ always leads a UAV from its

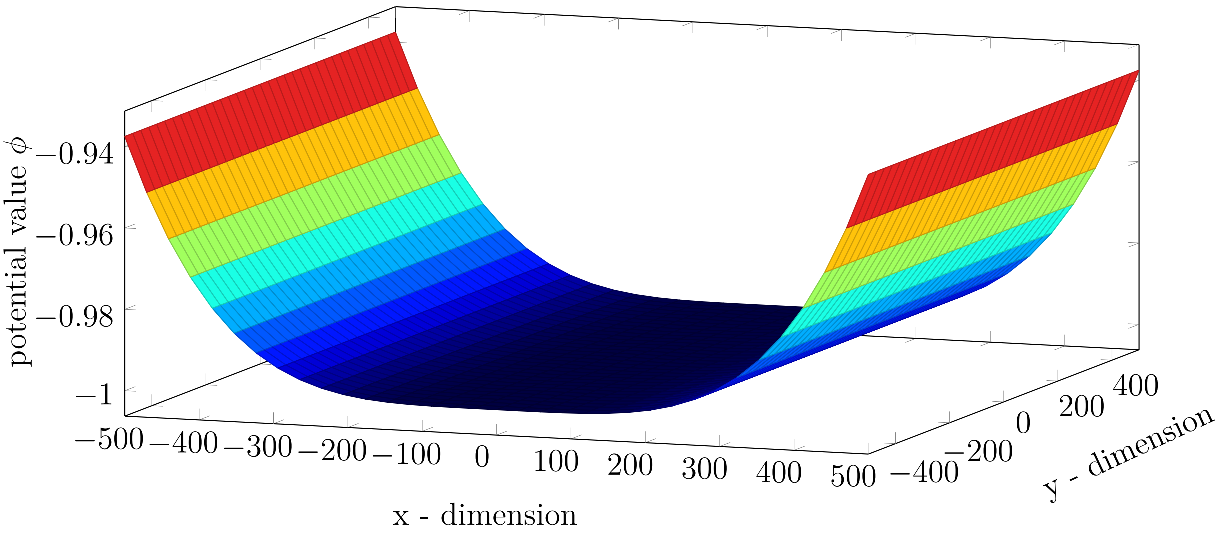


Figure 6.3.: *Potential field of an example bifurcation computed by a two-dimensional version of Equation 6.6. This kind of potential field can be used for formation establishment.*

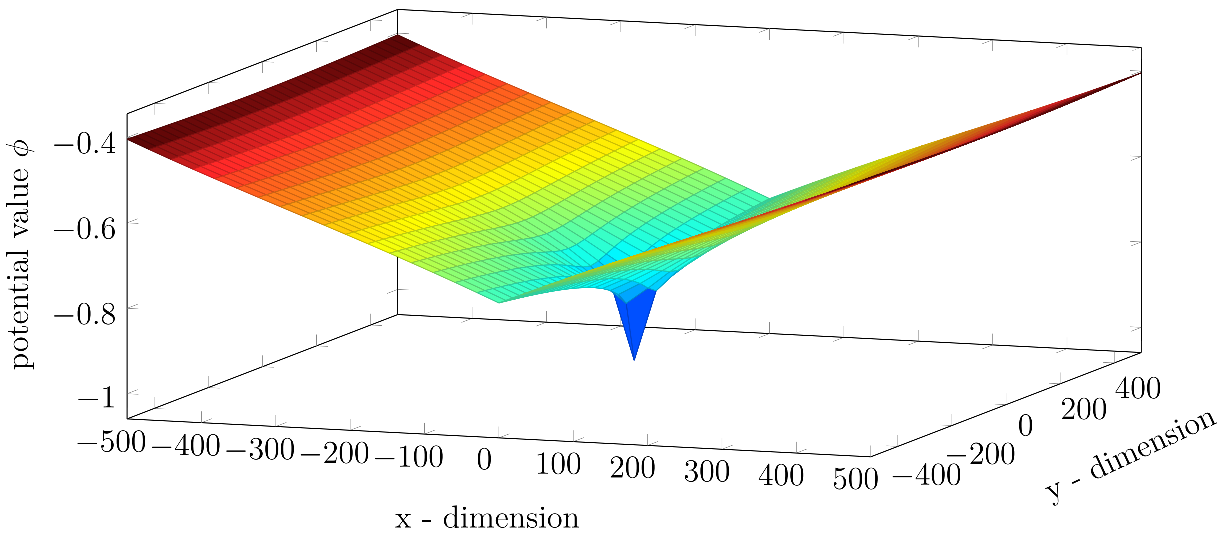


Figure 6.4.: *Potential field achieved by superpositioning the harmonic and the bifurcating potential fields using Equation 6.13. This kind of potential field is used for formation movement.*

initial position to a position at $(0, \cdot)$. The bifurcating potential field does not take repulsive potentials into account. Thus, it is impossible to guarantee collision avoidance using only this potential field.

The resulting superpositioned potential field is presented in Figure 6.4. The influence of the harmonic potential field has been set to $w = 0.3$ (cf. Equation 6.13) It shows that all streamlines lead from every $q_{\mathcal{I}}$ to $q_{\mathcal{G}}$, which is located at the center of the terrain. The single $\phi_{x,y}$ have a higher increase in x - direction than in y - direction. Considering consistent circular potential values, produced by additional UAVs ensures that all UAVs fly towards the center point of the environment and form a line in y - direction.

The UAVs create a line formation using the resulting potential field shown in Figure 6.4. As mentioned before, several different formation shapes are possible by changing the parameters of Equation 6.6. The next section shows a selection of possible formation shapes computed by the described approach.

6.3.4. Overview of Formation Shapes

Changing the shape of a formation is possible at anytime. Additionally, the formations are highly dynamic in terms of number of participating UAVs. Further shapes like a V-formation or a cross like shape are possible beside the mentioned line formation.

The presented formation shapes are based on the potential field calculated using Equation 6.6. Table 6.1 shows the parameter used in Equation 6.6 to achieve the shapes illustrated in Figures 6.5 and 6.6.

Figure	μ	α	a	b	c
6.5 a)	1	1	1	0	1
6.5 b)	-1	100	1	0	1
6.6 a)	1	1	0	0	1
6.6 b)	1	1	1	1	1

Table 6.1.: *Parameter values used to obtain the formation shapes shown in Figure 6.5 and Figure 6.6.*

Figure 6.5 a) illustrates a single line formation shape. Such a shape can easily be achieved by assigning one of the weighting parameters a, b, or c to zero. The parameter b has been set to zero as shown in the first row of Table 6.1 for the illustrated example causing the UAVs to arrange themselves side by side in the y - dimension.

Changing the bifurcation parameter μ to -1 can create a double line formation as shown in Figure 6.5 b). The amplitude of the bifurcating potential has been increased to 100 in order to keep the two lines close to each other. Such a formation shape can also be used to create a V-formation by rotating the two lines independently such that they cross at a precomputed position. The method used to rotate given formation shapes around three axes is described in detail in Section 6.4.3.

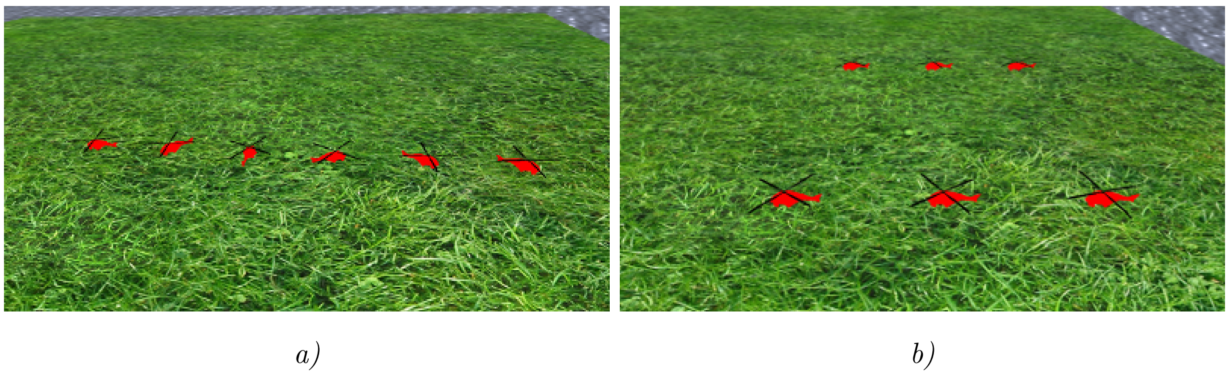


Figure 6.5.: *A line formation is shown in a) and a double line formation in b). The values presented in the first row of Table 6.1 lead to the line formation shape and the values presented in the second row result in a double line formation shape.*

Various other kinds of formation shapes are possible beside line based formation shapes. Figure 6.6 a) shows a two-dimensional cluster formation shape. It has been achieved by assigning the values presented in the third row of Table 6.1. Only the parameter c has been assigned to one affecting the resulting potential field in such a way that the UAVs arrange themselves around the target position. Changing the bifurcation parameter μ results in two different two-dimensional cluster formation shapes with a distance to each other related to the given amplitude α.

Not only two-dimensional formation shapes as presented so far are possible. Figure 6.6 b) shows a three-dimensional cluster formation shape. It has been achieved by assigning the

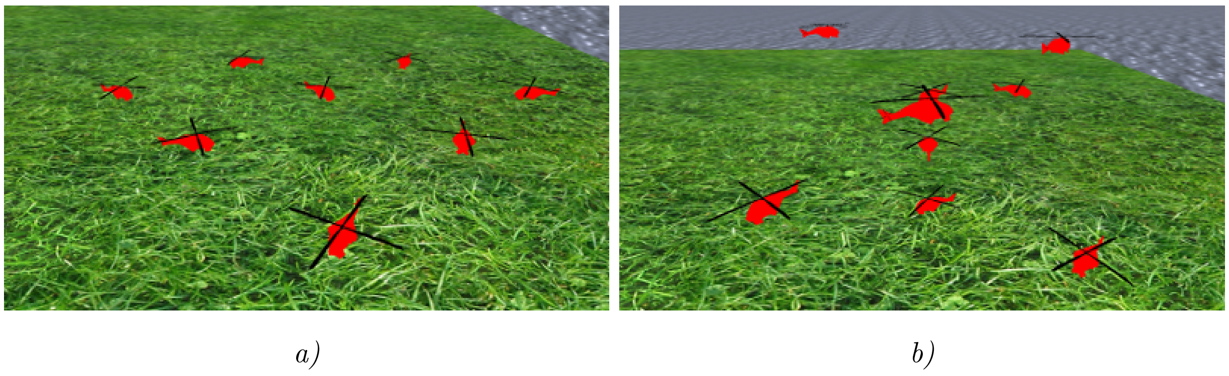


Figure 6.6.: *A 2D cluster formation is shown in a) and a 3D cluster formation is shown in b). The values presented in the third row of Table 6.1 lead to the 2D cluster formation shape and the values presented in the fourth row result in a 3D cluster formation shape.*

values as presented in the forth row of Table 6.1. The parameters a, b, and c are set to one. Based on this assignment the UAVs arrange themselves around the target position in all three dimensions. Changing the bifurcation parameter μ would also lead to two three-dimensional cluster formation shapes with a distance to each other based on the amplitude α.

This section presents a selection of formation shapes achievable by the used bifurcation approach. Several additional formation shapes can be created. Establishing and dissolving formations can be performed anytime. Changing a formation from one shape to another can also be conducted anytime, by just adapting the parameter values of Equation 6.6. Additional formations, such as a ring and a double ring formation can also be created using bifurcation theory shown, e. g., by Bennet et al. [219].

The previous sections present the approach used to establish various UAV formation shapes. Creating formations does not yet lead to any benefit. It must also be possible to handle the resulting formations. Handling means that requirements concerning the position where the formation has to be created and also about the participating UAVs have to be fulfilled. Additionally, it must be possible to move and rotate existing formations. The following section presents the methods used to handle these issues.

6.4. Formation Handling

The UAVs cannot just fly in formation following one single path preserving the given formation shape, due to several reasons. This would neglect the position changes of the UAVs during flights and resulting in collisions. Furthermore, the UAVs have to avoid obstacles in a skillful manner. Therefore, they need to change their positions inside the formation in order to still guarantee obstacle avoidance. Collisions would still occur if the other UAVs do not take such changes into account.

It is also necessary for the UAVs to be able to reach consensus on the position where a formation has to be created. A decision concerning the participants of a formation must also be made. All requirements considered so far are accomplished in a completely decentralized manner as described in the following section.

6.4.1. Requirements

Several objectives have to be considered in order to explore different environments using various formations simultaneously. A position for the creation of a formation has to be found at the beginning. Two methods are used to determine the position at which a formation should be established. The first one is given by the task allocation methodology presented in Chapter 7. The position is initially given by a task if the UAVs have to create a formation in order to solve a specific task given by a third party application.

The second kind of position determination is conducted if a UAV creates a task by itself or if a third party application forces the creation of a formation. In that case, the arithmetic mean of the positions of the UAVs participating in the formation is computed as a first possible consensus position. The center point of the leaf of the octree that is closest to this position and additionally part of the free space, is selected as consensus point if the computed position lies inside an obstacle.

The shape of a formation is always predetermined by the task to be fulfilled, by the UAV wanting to create a formation, or by a third party application forcing the establishment of a formation.

It is not beneficial to use only one single formation in which all UAVs participate. It must be possible to create various formations simultaneously using different UAVs. From this it follows that UAVs have to be selected to be participants of single formations. This is accomplished as described in Chapter 7.

The UAVs must not necessarily be able to communicate with all other UAVs in order to create formations. The currently possible inter-UAV communication can be mapped to a graph $G = (V, E)$ where $V = \{1, 2, \ldots, N\}$ is the set of vertices representing N UAVs. $E \subset V \times V$ is the set of edges of G denoted by (i, j) representing that UAV $\mathcal{U}_i$ can exchange information with UAV $\mathcal{U}_j$. Ni and Cheng [220] have shown that consensus between all UAVs represented by the graph can be obtained if a minimum spanning tree in the graph exists. This is important for an adaptation of the approach to real world applications where perfect communication is not guaranteed.

A formation can be created after the fulfillment of the mentioned requirements. UAVs forming a specific formation have to successively fly to several target configurations in order to explore some areas while simultaneously preserving the formation. Flying in formation differs from single UAV flights and is described in the following section.

6.4.2. Formation Flights

The next task after the fulfillment of the previously mentioned requirements is to actually establish a formation. The UAVs participating in a formation start flying to the position at which the formation has to be created. The paths of the single UAVs to this point can differ in length. Additionally, the flight speed of the UAVs can differ. Thus, in a completely decentralized system, it is not trivial to determine whether a formation has been established or if some UAVs are still on their way to the required position.

The fact that the harmonic potential field is local equilibria free is used to decide if a formation has been established. Each UAV checks whether it is able to fly to another position. It is assumed that the formation has been created if each UAV participating at the formation is unable to compute a path leading closer to the formation position.

After the creation of a formation, e. g., to explore areas of the terrain, one area to be explored is selected and one single formation path to this area is computed. The computation is conducted by the UAV with the lowest unique identifier (cf. Section 4.3) that participates in the formation. The path is created using only the harmonic potential field and is thereafter distributed to the other UAVs. It is not possible to just follow this path due to the aforementioned reasons. So, each UAV computes a second path to the next route point of the formation path using the superpositioned potential field. This paths lead only to a neighboring leaf. Thus, it may happen that more than one path has to be computed to reach the next formation route point. This is performed repeatedly until the target configuration has been reached. Using this kind of path-following ensures that each UAV of the formation avoids obstacles by changing the shape of the formation when necessary. It also ensures that no UAV collides with another.

Unpredictable behavior can occur when the sign of μ changes from positive to negative. The positions of the UAVs are then close to an unstable equilibrium with several stable equilibria at equal distances to the unstable equilibrium. If, e. g., the UAVs should change their formation shape from a single line to a double line, it may occur that all UAVs move to the same equilibrium. In this case the formation shape would not change and the UAVs would just form a line with some distance to the old one. Such behavior is avoided through the harmonic potential field. The UAVs distribute themselves to the single equilibria based on their unique identifier (cf. Section 4.3) and the identifier of the other UAVs participating in the formation. Therefore, they move the equilibrium position of the harmonic potential field to the selected equilibrium position of the bifurcating potential field.

A UAV formation should be able to fly in arbitrary directions while simultaneously preserving the formation. Rotating the flight direction of, e. g., a line formation wherein UAVs move side by side, by 90 degrees would result in a formation wherein the UAVs are arranged one after the other. An adaptation of the potential field must be performed online to avoid such formation changes. The next section presents the approach used for rotating formations.

6.4.3. Formation Rotation

In addition to formation flights, it should be possible to rotate the formations in three dimensions in order to respect changing flight directions.

A yaw angle $\beta_{\mathcal{P}}$, a pitch angle $\gamma_{\mathcal{P}}$, and a roll angle $\delta_{\mathcal{P}}$ have been introduced for each formation $\mathcal{P}$. Rotating the formations around three axes is based on these angles. They are similar to the UAV angles (cf. Section 4.3) and describe the flight direction of a UAV formation in the x, y, z plane.

The angles are based on the internal Cartesian coordinate system $\mathcal{F}_{\mathcal{W}}$. Figure 6.7 illustrates $\mathcal{F}_{\mathcal{W}}$ including the corresponding angles for the flight directions. An obstacle free environment has been chosen for an example terrain.

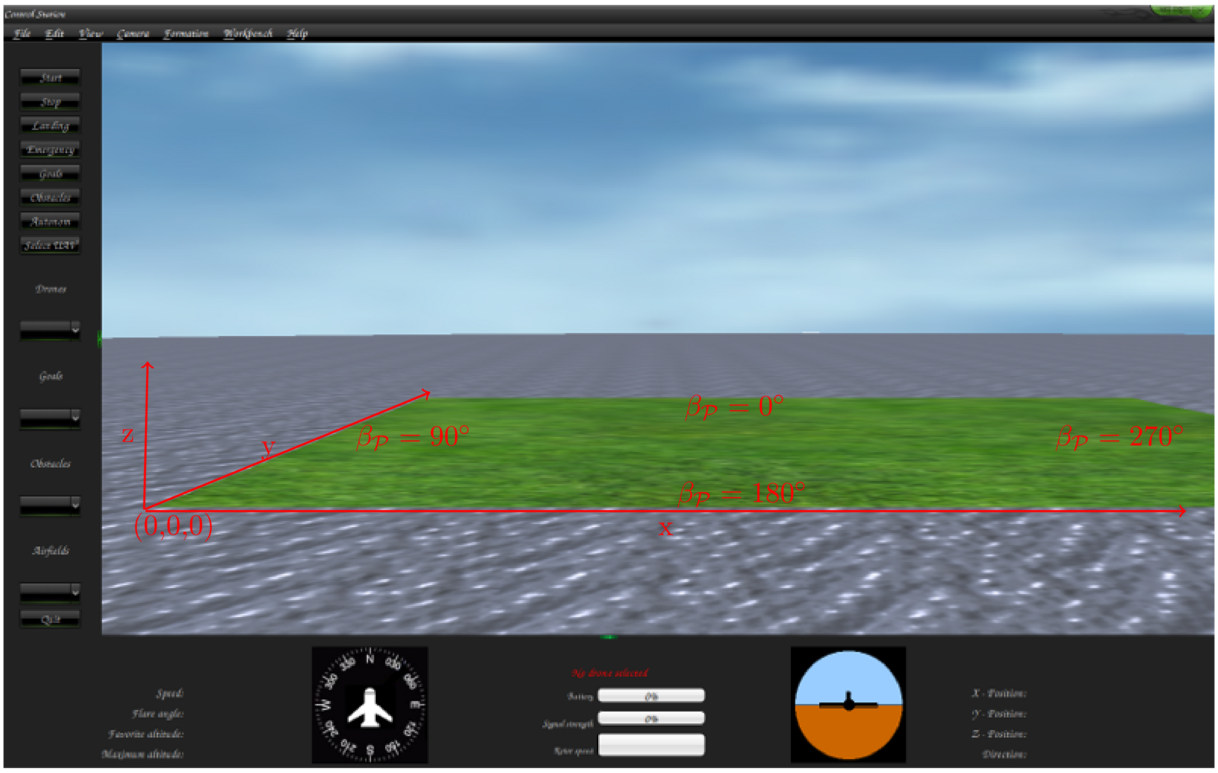


Figure 6.7.: *Coordinate System $\mathcal{F}_{\mathcal{W}}$ and the corresponding direction angles $\beta_{\mathcal{P}}$.*

The UAVs always fly from one configuration to the next. Additionally, all UAVs of a formation want to reach the same target configuration during formation flights. The relationship between the resulting flight direction of the formation and the corresponding angle needs to be defined. An illustration of the yaw angle is shown in Figure 6.7. It is based on the distances τ_x and τ_y between the x - and y - coordinates of the current configuration and the configuration given by the next route point. A case by case analysis shown in Equation 6.14 finally computes $\beta_{\mathcal{P}}$. The definitions and thus the equations are analogously for the pitch and roll angle.

$$\beta_{\mathcal{P}} = \begin{cases} 0, & \text{if } \tau_x = 0, \tau_y > 0, \\ 180, & \text{if } \tau_x = 0, \tau_y < 0, \\ \arctan\left(\frac{\tau_y}{\tau_x}\right) + 90, & \text{if } \tau_x < 0, \\ \arctan\left(\frac{\tau_y}{\tau_x}\right) + 270, & \text{if } \tau_x > 0 \end{cases} \tag{6.14}$$

Formation rotations are finally based on the resulting angles $\beta_{\mathcal{P}}, \gamma_{\mathcal{P}}$, and $\delta_{\mathcal{P}}$. The definition of the angles follows the international conventions in air vehicles published by the German Institute for Standardization (in German, *Deutsches Institut für Normung e. V.*) [221] as DIN 9300. The conventions have been adopted by the International Organization for Standardization (ISO) [222] as ISO 1151-2:1985. Taking this definition into account results in a standard 3×3 rotation matrix presented in Equation 6.15.

$$R = \begin{pmatrix} \cos(\beta_{\mathcal{P}})\cos(\gamma_{\mathcal{P}}) & \sin(\beta_{\mathcal{P}})\cos(\gamma_{\mathcal{P}}) & -\sin(\gamma_{\mathcal{P}}) \\ \cos(\beta_{\mathcal{P}})\sin(\gamma_{\mathcal{P}})\sin(\delta_{\mathcal{P}}) - \sin(\beta_{\mathcal{P}})\cos(\delta_{\mathcal{P}}) & \sin(\beta_{\mathcal{P}})\sin(\gamma_{\mathcal{P}})\sin(\delta_{\mathcal{P}}) + \cos(\beta_{\mathcal{P}})\cos(\delta_{\mathcal{P}}) & \cos(\gamma_{\mathcal{P}})\sin(\delta_{\mathcal{P}}) \\ \cos(\beta_{\mathcal{P}})\sin(\gamma_{\mathcal{P}})\cos(\delta_{\mathcal{P}}) + \sin(\beta_{\mathcal{P}})\sin(\delta_{\mathcal{P}}) & \sin(\beta_{\mathcal{P}})\sin(\gamma_{\mathcal{P}})\cos(\delta_{\mathcal{P}}) - \cos(\beta_{\mathcal{P}})\sin(\delta_{\mathcal{P}}) & \cos(\gamma_{\mathcal{P}})\cos(\delta_{\mathcal{P}}) \end{pmatrix} \tag{6.15}$$

Multiplying the distances $\tau_{x,y,z}$ with R as shown in Equation 6.16 leads to rotated center points for the single leaves rotated around the target position $(0, 0, 0)$. This causes a free rotation of the potential field in order to also rotate the resulting formation around all three axes by replacing $\tau_{x,y,z}$ through τ_R in Equation 6.6.

$$\tau_R = (\tau_x, \tau_y, \tau_z)^T \cdot R \tag{6.16}$$

The possible rotation angles of a formation are restricted due to the use of the underlying octree. Each UAV always moves from the center position of a leaf to the center position of a neighboring leaf (cf. Section 5.4). So the resulting angle between the center points of two neighboring leaves at maximum resolution is equal to the step size of the rotation.

The methods presented so far allow the UAVs to move to given target configurations and to explore environments in formation, as well as on their own. A further step towards the design of an autonomous UAV system is to state the global asymptotic stability of the potential field as done in the next section.

6.5. Verification

A proof for single UAV movement based on harmonic potential fields is presented in Section 5.5 in order to show that the flight behavior of the UAVs is verifiable. This section verifies the flight behavior of UAVs in formation based on the bifurcating potential field.

The objective of the bifurcating potential field, is to move the UAV to the desired equilibrium position that corresponds to the minimum potential value. Therefore, if Lyapunov's method can be used to show that the system is globally asymptotically stable, it will relax into the minimum energy state given by the minimum potential value. Potential fields created by extended Pitchfork bifurcations have been proved to be Lyapunov stable before, e. g., by Bennet and McInnes [167], [213]. Based on the main idea of the proofs from Bennet and McInnes, the bifurcating potential field created for the UAV system is proved as described in the further course of this section.

Occupied areas of the environment are not taken into account by the bifurcating potential field, but by the superpositioned potential field. Thus, the bifurcating potential field is considered to be obstacle free in order to avoid undefined areas. Bennet and McInnes [167], [213] argue that the repulsive potential values can be neglected if a proper scaling of attractive and repulsive potential fields is applied.

6.5.1. Lyapunov Function Candidate

First, it must be shown that the resulting potential field is a Lyapunov function candidate in order to show that the system is globally asymptotically stable.

The first requirement for a function to be a Lyapunov function candidate is that the equilibrium occurs at $x^* = 0$. The equilibrium created by Equation 6.6 is at $x^* = -v$ if $\mu \geq 0$. To achieve the first requirement the function is shifted back (cf. Section 3.6.1) and

$$V(\phi^b_{x,y,z}) = \mu(a\tau_x^2 + b\tau_y^2 + c\tau_z^2) + \alpha(a\tau_x^4 + b\tau_y^4 + c\tau_z^4) \tag{6.17}$$

is considered for the proof of stability. Again, let Ω be the closed domain on which ϕ is given. The workspace of the UAVs is then $\Omega_{\mathcal{F}} = \Omega - \phi^h_{\mathcal{O}}$ (cf. Section 5.5.1).

Lemma 6.1. *The bifurcating potential field $V(\phi^b_{x,y,z})$ is a Lyapunov function candidate.*

Proof. Taking Equation 6.17 into account $x^* = 0$ is given by definition. So, it is only necessary to show that

$$\forall \phi^b_{x,y,z} \in \Omega_{\mathcal{F}} \setminus \phi^h_{\mathcal{G}} \colon V(\phi^b_{x,y,z}) > 0 \tag{6.18}$$

in order to show that $V(\phi^b_{x,y,z})$ is a Lyapunov function candidate.

Considering the case $\mu \geq 0$. From the first partial derivatives shown in Equations 6.7 to 6.9 it can be seen that the function is strictly increasing with increasing distance to the stable equilibrium. Thus, all values except x^* are greater than zero.

Three zeros for each of the three partial derivatives exist if $\mu < 0$. These zeros are the extrema of the partial derivatives and are at the positions $-\sqrt{\frac{-\mu}{2\alpha}}, 0$, and $\sqrt{\frac{-\mu}{2\alpha}}$. These three positions lead to four intervals that have to be investigated separately in order to show that Equation 6.18 holds.

Let τ_i be the distances in the single directions and let i be the corresponding a, b, or c value. Equation 6.19 shows the sign of the first partial derivatives for the interval $]-\infty, -\sqrt{\frac{-\mu}{2\alpha}}[$ where the function becomes zero by definition. The sign is negative denoting that the function is strictly decreasing until it reaches zero at $-\sqrt{\frac{-\mu}{2\alpha}}$. Thus, all values of this area are greater than zero.

$$\forall \tau_i \in \left]-\infty, -\sqrt{\frac{-\mu}{2\alpha}}\right[: \frac{\partial \phi^b_{x,y,z}}{\partial \tau_i} = \frac{2\mu i \tau_i + 4\alpha i \tau_i^3}{u} < 0 \tag{6.19}$$

The second interval to be investigated is $]-\sqrt{\frac{-\mu}{2\alpha}}, 0[$. The sign of the first partial derivatives is positive denoting that the function is strictly increasing in this area as shown in Equation 6.20. Since the function is zero at $-\sqrt{\frac{-\mu}{2\alpha}}$, it follows that all values of this area are also greater than zero.

$$\forall \tau_i \in \left]-\sqrt{\frac{-\mu}{2\alpha}}, 0\right[: \frac{\partial \phi^b_{x,y,z}}{\partial \tau_i} = \frac{2\mu i \tau_i + 4\alpha i \tau_i^3}{u} > 0 \tag{6.20}$$

In the interval $]0, \sqrt{\frac{-\mu}{2\alpha}}[$ the function is decreasing again until zero is reached at $\sqrt{\frac{-\mu}{2\alpha}}$ as shown in Equation 6.21. So, the resulting potential values of the function are still greater than zero.

$$\forall \tau_i \in \left]0, \sqrt{\frac{-\mu}{2\alpha}}\right[: \frac{\partial \phi^b_{x,y,z}}{\partial \tau_i} = \frac{2\mu i \tau_i + 4\alpha i \tau_i^3}{u} < 0 \tag{6.21}$$

The last interval to be investigated is $]\sqrt{\frac{-\mu}{2\alpha}}, \infty[$. The function is zero at $\sqrt{\frac{-\mu}{2\alpha}}$ and the sign of the first partial derivatives is positive as shown in Equation 6.22 indicating that the function is strictly increasing. Thus, the function is positive within this area, too. From this it follows that the function is greater than zero except for the two stable equilibrium positions where it is zero by definition.

$$\forall \tau_i \in \left]\sqrt{\frac{-\mu}{2\alpha}}, \infty\right[: \frac{\partial \phi^b_{x,y,z}}{\partial \tau_i} = \frac{2\mu i \tau_i + 4\alpha i \tau_i^3}{u} > 0 \tag{6.22}$$

So, the requirement from Equation 6.18 is fulfilled. □

It must be shown that each stable equilibrium is globally asymptotically stable in order to show that the entire potential field is globally asymptotically stable. This can be achieved in two steps. First, local stability is proven for the single equilibrium positions. Thereafter, a global proof for the entire potential field is conducted. Therefore, the next section shows that the single equilibrium positions are locally asymptotically stable by the use of Lyapunov's indirect method (cf. Section 3.6.4).

6.5.2. Local Convergence

An eigenvalue analysis of the linearized equations of motion has been performed in order to show local convergence of the system. Motion is obtained from the paths planned by the UAVs using the descent gradient $-\nabla$ of the artificial potential function. Thus, $-\nabla$ describes a virtual force acting on each UAV to lead them towards the minimum energy state.

Lemma 6.2. *The equilibrium positions of the bifurcating potential field generated using Equation 6.6 are locally uniformly asymptotically stable.*

Proof. The calculation of the eigenvalue spectrum of the system is used to determine the local behavior. For this purpose, the equations of motion for the model are recast as follows:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} = \begin{pmatrix} -\frac{\partial V(\phi^b_{x_0,y_0,z_0})}{\partial \tau_x} \\ -\frac{\partial V(\phi^b_{x_0,y_0,z_0})}{\partial \tau_y} \\ -\frac{\partial V(\phi^b_{x_0,y_0,z_0})}{\partial \tau_z} \end{pmatrix} = \begin{pmatrix} f(x,y,z) \\ g(x,y,z) \\ h(x,y,z) \end{pmatrix}, \tag{6.23}$$

whereby x_0, y_0, and z_0 denote fixed points for which $\dot{x} = \dot{y} = \dot{z} = 0$ such that

$$f(x_0, y_0, z_0) = 0, \tag{6.24}$$

$$g(x_0, y_0, z_0) = 0, \tag{6.25}$$

$$h(x_0, y_0, z_0) = 0. \tag{6.26}$$

The negative gradient $-\nabla V(\phi^b_{x,y,z})$ then vanishes at the equilibrium.

Let $\rho_{\mathcal{R}}$ be the next route point. A single equilibrium is created at position $p_{\mathcal{R}}$ if $\mu \geq 0$ and $\alpha > 0$. Changing the number of equilibria can be done by changing μ such that $\mu < 0$. Up to eight different stable equilibria positions can then be generated simultaneously if $a = b = c = 1$ and $\mu < 0$. These equilibria are obtained at the following positions:

$$\begin{array}{ll} p_{\mathcal{R}}+(\sqrt{\frac{-\mu}{2\alpha}},\sqrt{\frac{-\mu}{2\alpha}},\sqrt{\frac{-\mu}{2\alpha}}), & p_{\mathcal{R}}-(\sqrt{\frac{-\mu}{2\alpha}},\sqrt{\frac{-\mu}{2\alpha}},\sqrt{\frac{-\mu}{2\alpha}}), \\ p_{\mathcal{R}}+(\sqrt{\frac{-\mu}{2\alpha}},-\sqrt{\frac{-\mu}{2\alpha}},\sqrt{\frac{-\mu}{2\alpha}}), & p_{\mathcal{R}}-(\sqrt{\frac{-\mu}{2\alpha}},-\sqrt{\frac{-\mu}{2\alpha}},\sqrt{\frac{-\mu}{2\alpha}}), \\ p_{\mathcal{R}}+(\sqrt{\frac{-\mu}{2\alpha}},\sqrt{\frac{-\mu}{2\alpha}},-\sqrt{\frac{-\mu}{2\alpha}}), & p_{\mathcal{R}}-(\sqrt{\frac{-\mu}{2\alpha}},\sqrt{\frac{-\mu}{2\alpha}},-\sqrt{\frac{-\mu}{2\alpha}}), \\ p_{\mathcal{R}}+(\sqrt{\frac{-\mu}{2\alpha}},-\sqrt{\frac{-\mu}{2\alpha}},-\sqrt{\frac{-\mu}{2\alpha}}), & p_{\mathcal{R}}-(\sqrt{\frac{-\mu}{2\alpha}},-\sqrt{\frac{-\mu}{2\alpha}},-\sqrt{\frac{-\mu}{2\alpha}}). \end{array} \tag{6.27}$$

Taking the distances $\tau = (\tau_x, \tau_y, \tau_z)$ from $p_{\mathcal{R}}$ into account and expanding them about the fixed points to linear order using Taylor Series expansion the resulting eigenvalues can be computed via the Jacobian matrix (cf. Section 3.6.4). Using the first order term of the Taylor Series expansion of a function around the point of interest results in the linearization of the function [177].

$$\begin{pmatrix} \dot{\tau_x} \\ \dot{\tau_y} \\ \dot{\tau_z} \end{pmatrix} = J \begin{pmatrix} \tau_x \\ \tau_y \\ \tau_z \end{pmatrix}. \tag{6.28}$$

Substituting the functions from Equations 6.24 to 6.26 into the Jacobian matrix results in

$$J = \left.\begin{pmatrix} \frac{\partial f(x,y,z)}{\partial \tau_x} & \frac{\partial f(x,y,z)}{\partial \tau_y} & \frac{\partial f(x,y,z)}{\partial \tau_z} \\ \frac{\partial g(x,y,z)}{\partial \tau_x} & \frac{\partial g(x,y,z)}{\partial \tau_y} & \frac{\partial g(x,y,z)}{\partial \tau_z} \\ \frac{\partial h(x,y,z)}{\partial \tau_x} & \frac{\partial h(x,y,z)}{\partial \tau_y} & \frac{\partial h(x,y,z)}{\partial \tau_z} \end{pmatrix}\right|_{x_0,y_0,z_0}. \tag{6.29}$$

The Jacobian matrix consists of the single derivatives of the functions $f(x, y, z), g(x, y, z)$ and $h(x, y, h)$. These derivatives are given by:

$$\frac{\partial^2 \phi(x,y,z)}{\partial \tau_x^2} = \frac{12\alpha a \tau_x^2 + 2a\mu}{u}, \tag{6.30}$$

$$\frac{\partial^2 \phi(x,y,z)}{\partial \tau_y^2} = \frac{12\alpha b \tau_y^2 + 2b\mu}{u}, \tag{6.31}$$

$$\frac{\partial^2 \phi(x,y,z)}{\partial \tau_z^2} = \frac{12\alpha c \tau_z^2 + 2c\mu}{u}, \tag{6.32}$$

$$\frac{\partial^2 \phi(x,y,z)}{\partial \tau_x \partial \tau_y} = \frac{\partial^2 \phi(x,y,z)}{\partial \tau_x \partial \tau_z} = 0, \tag{6.33}$$

$$\frac{\partial^2 \phi(x,y,z)}{\partial \tau_y \partial \tau_x} = \frac{\partial^2 \phi(x,y,z)}{\partial \tau_y \partial \tau_z} = 0, \tag{6.34}$$

$$\frac{\partial^2 \phi(x,y,z)}{\partial \tau_z \partial \tau_x} = \frac{\partial^2 \phi(x,y,z)}{\partial \tau_z \partial \tau_y} = 0. \tag{6.35}$$

As shown before, if $\mu \geq 0$ the equilibrium is at $\tau = (0, 0, 0)$. Substituting the matrix from Equation 6.23 into Equation 6.29 and evaluating the Jacobian matrix at the mentioned equilibrium position results in

$$J = \begin{pmatrix} \frac{-2a\mu}{u} & 0 & 0 \\ 0 & \frac{-2b\mu}{u} & 0 \\ 0 & 0 & \frac{-2c\mu}{u} \end{pmatrix}. \tag{6.36}$$

The eigenvalues λ_i of the matrix are given by the zeros of the determinant $\det(J - \lambda I) = 0$. It can be seen directly that

$$\lambda_1 = \tfrac{-2a\mu}{u}, \quad \lambda_2 = \tfrac{-2b\mu}{u}, \quad \lambda_3 = \tfrac{-2c\mu}{u} \tag{6.37}$$

are the corresponding eigenvalues. As $\mu, u > 0$ and $a, b, c \in \{0, 1\}$ the eigenvalues are negative real as long as $a = b = c = 1$. Hence, the equilibrium position can be considered as asymptotically stable (cf. Section 3.6.4). It can also be concluded that the equilibrium becomes unstable for $\mu < 0$.

If a, b, or c is zero, the eigenvalue for the corresponding dimension becomes zero, too. In this case no change of the values takes place in the corresponding dimension and all values in this dimension are also zero at the positions where the distances of the other dimensions become zero as shown, e. g., in Figure 6.3. Then, a line or area of equilibrium positions is formed and the system will not convert to a single position but to the resulting area of equally sized equilibrium values as desired.

As shown before, if $\mu < 0$ the further equilibria are at $\tau = (\pm\sqrt{\tfrac{-\mu}{2\alpha}}, \pm\sqrt{\tfrac{-\mu}{2\alpha}}, \pm\sqrt{\tfrac{-\mu}{2\alpha}})$. Substituting Equation 6.23 into Equation 6.29 and evaluating the Jacobian matrix at the equilibrium position $\tau = (-\sqrt{\tfrac{-\mu}{2\alpha}}, -\sqrt{\tfrac{-\mu}{2\alpha}}, -\sqrt{\tfrac{-\mu}{2\alpha}})$ results in

$$J = \begin{pmatrix} \frac{4a\mu}{u} & 0 & 0 \\ 0 & \frac{4b\mu}{u} & 0 \\ 0 & 0 & \frac{4c\mu}{u} \end{pmatrix}. \tag{6.38}$$

It can easily be seen that the eigenvalues of Matrix 6.38 are:

$$\lambda_1 = \tfrac{4a\mu}{u}, \quad \lambda_2 = \tfrac{4b\mu}{u}, \quad \lambda_3 = \tfrac{4c\mu}{u}. \tag{6.39}$$

The single eigenvalues are also negative real as long as $a = b = c = 1$ because $\mu < 0, u > 0$. Thus, these equilibrium positions are also asymptotically stable as described earlier. The same as before happens if a, b or c are set to zero and the equilibria will become areas. This can be used, e. g., to create double line formations by setting one of the weighting parameters to zero.

Finally, the equilibrium positions with only positive signs have been evaluated. Substituting Equation 6.23 into Equation 6.29 and evaluating the Jacobian matrix at the equilibrium position $\tau = (\sqrt{\frac{-\mu}{2\alpha}}, \sqrt{\frac{-\mu}{2\alpha}}, \sqrt{\frac{-\mu}{2\alpha}})$ results in

$$J = \begin{pmatrix} \frac{4a\mu}{u} & 0 & 0 \\ 0 & \frac{4b\mu}{u} & 0 \\ 0 & 0 & \frac{4c\mu}{u} \end{pmatrix}. \tag{6.40}$$

The eigenvalues of Matrix 6.40 can also easily be determined and are:

$$\lambda_1 = \frac{4a\mu}{u}, \quad \lambda_2 = \frac{4b\mu}{u}, \quad \lambda_3 = \frac{4c\mu}{u}. \tag{6.41}$$

These eigenvalues are also negative real as $\mu < 0, u > 0$. Changing the weighting parameter a, b, or c from one to zero leads to the same convergence behavior described before. It can also be seen that each combination of positive and negative signs, which leads to the remaining equilibrium positions always leads to $\lambda = \frac{4a\mu}{u}$. Thus, the eigenvalues of all equilibrium positions are negative real except for $\tau = (0, 0, 0)$ as shown before.

A global uniform asymptotic stability of the linearization implies local uniform asymptotic stability of the origin as shown by Theorem 3.3. Hence, the bifurcating potential field created using Equation 6.6 is locally asymptotically stable. □

The stability proof given so far shows local stability of the single equilibrium positions. This does not state that the system is also globally stable. Lyapunov's direct method has been applied using the idea of a proof presented in [167] in the next section in order to prove global stability.

6.5.3. Global Convergence

Lemma 6.3. *The trajectory generated by the descent gradient will globally and asymptotically converge to $\phi^h_{\mathcal{G}}$ (cf. Section 3.6.2), i. e.,*

$$\forall \phi^b_{x,y,z} \in \{\phi | \phi(0) \in \Omega_{\mathcal{F}}\} \colon \lim_{t\to\infty} \phi^b_{x,y,z}(t) = \phi^b_{\mathcal{G}}. \tag{6.42}$$

Proof. The Lyapunov function $V(\phi^b_{x,y,z})$ is defined as the total energy of the system, whereby $\phi^b_{x,y,z}$ is given by Equation 6.17. This is a simple shifting of the function as described in Section 3.6.1. Taking a unit mass into account, the Lyapunov function is given by the sum of the n potential values

$$V(\phi^b_{x,y,z}) = \sum_{i=0}^{n} \phi^b_{x,y,z}, \tag{6.43}$$

where $V(\phi^b_{x,y,z}) > 0$ holds at any position unequal to x^* where the system is $V(\phi^b_{x,y,z}) = 0$ by definition (cf. Section 6.5.1).

The rate of change of the Lyapunov function can be computed using the first partial derivatives due to the use of the descent gradient $-\nabla$. It is possible to express the change as the total sum of the potential derivatives [167]:

$$\dot{V} = \sum_{i=0}^{n} \frac{\partial V}{\partial \phi^b_{x,y,z}}. \tag{6.44}$$

Then, using Equation 6.23 in combination with Equation 6.44 to take the descent gradient $-\nabla$ instead of the gradient ∇ into account results in

$$\dot{V} = -\sum_{i=0}^{n} \nabla(\phi^b_{x,y,z}) \leq 0. \tag{6.45}$$

From Lyapunov's direct method (cf. Section 3.6.2) it can be concluded that if $V(\phi^b_{x,y,z})$ is a positive definite function and $\dot{V}(\phi^b_{x,y,z})$ is a negative definite function the system is a globally uniformly asymptotically stable. As shown before, each of the single equilibrium positions is also locally asymptotically stable.

A problem arises using superimposed artificial potential functions as $\dot{V}(\phi^b_{x,y,z}) \leq 0$. This implies that $\dot{V}(\phi^b_{x,y,z})$ could be zero at a position unequal to the equilibrium position. In this case the system might contain local equilibria. The LaSalle theorem can be used (cf. Section 3.6.3) to ensure that the system is asymptotically stable at the desired equilibrium position. It extends the mentioned constraints to show that if $V(0) = \dot{V}(0) = 0$ and the set $\{\phi^b_{x,y,z} | \dot{V} = 0\}$ only occurs if $\phi^b_{x,y,z} = x^*$ then the equilibrium is asymptotically stable. Thus, the LaSalle theorem holds true for the potential field. As it is a smooth and well defined symmetric potential field equilibrium positions occur at the target states and the system relaxes into the desired equilibrium positions. $\square$

The bifurcating potential field is therefore globally uniformly asymptotically stable. Due to the superposition conducted by the system presented in this thesis, it is not necessary to show global stability of the bifurcating potential field as it is only considered for a small region of the entire environment at which the formation is located. Nevertheless, for the sake of completeness and to show that bifurcating potential fields can be used for influencing the entire environment global stability has been stated.

The stability of the single potential fields has been stated so far. Showing that the single parts of a system are stable does not prove that the potential field achieved by superposition is also stable. The following section considers the stability of the resulting superpositioned potential field.

6.5.4. Conclusion

Both, the harmonic and the bifurcating potential field are asymptotically stable. Nevertheless, a linear superposition of these potential fields is not necessarily asymptotically stable. The bifurcating potential field is strictly increasing with the Euclidean distance to the equilibrium positions while the harmonic potential field is strictly increasing with the path length which is not necessarily the same. Thus, close to occupied areas, it happens that the harmonic potential field is decreasing with the distance to the equilibrium position while the bifurcating potential field is increasing.

The superposition performed by the UAV system presented in this thesis takes this into account. The UAVs use the harmonic potential field to fly close to the equilibrium positions. Only a sphere around the equilibrium position is given by the superpositioned potential field. A UAV will then reach the equilibrium position as long as no obstacle between the UAV and the equilibrium position exists. Obstacles lead to rearrangement of the single UAVs participating in a formation, which leads to an adaption of the formation due to the obstacle as desired.

A problem may occur as the emergence of local equilibria close to obstacles is possible. It is then possible that single UAVs get stuck, while the rest of a formation's UAVs fly towards the next route point. This leads to new equilibrium positions and after a while the position at which a UAV $\mathcal{U}_i$ got stuck is outside of the superpositioned area. Then, only the harmonic potential field is considered and $\mathcal{U}_i$ is able to leave the local equilibrium position and follows the formation. Finally, $\mathcal{U}_i$ has to speed up while the other UAVs must decrease their flight speed, so that $\mathcal{U}_i$ is able to catch up with the rest of the formation.

One disadvantage of harmonic functions mentioned in Section 3.3.1 occurs if a superpositioning of several harmonic functions takes place. It has been stated that obstacle avoidance is not guaranteed in this case [34]. This disadvantage does not hold for the presented superimposed potential field. The potential value of obstacles is still firmly set to one. This is due to the min-max principle (cf. Definition B.2), the highest harmonic potential value. The bifurcating potential field only computes values for the free space and the values taken into account by superposition are always less or equal to zero. Thus, the resulting values obtained through an addition of the two potential fields are at any point of the resulting potential field less than one and obstacle avoidance is still guaranteed.

Another point not considered by the proofs presented earlier are the dynamics of the resulting system. The harmonic potential field is globally asymptotically stable. This means that a path from each initial configuration to each target configuration can be computed using the descent gradient if such a path exists. But it does not consider the simultaneous movement of other UAVs, which can make a replanning necessary to avoid collisions with other UAVs. Such behavior can foil target reaching as the UAVs might compute paths which result in never ending disturbances and replannings. For a real world UAV system it must be stated that multiple UAVs do not influence each other in a way, which renders it impossible for them to fulfill their duties and to reach their goals. One possible solution is the creation of a task allocation systems that takes this issues into account and distributes the tasks by considering the UAV flights necessary to reach the targets. This is neglected for the system presented in this thesis as the evaluation of the system presented in Chapter 8 shows that the UAVs were always able to reach their objectives.

6.6. Summary

This chapter addresses formation flights. First, a problem statement is given, then state-of-the-art approaches using potential fields for formation flights are presented. The artificial potential field used by the UAVs for formation flights is introduced thereafter. It is based on bifurcation theory and the resulting potential field is superpositioned with the harmonic potential field in order to keep the advantages of both approaches.

Examples for possible potential fields are illustrated to give a better impression of the approach. It is possible to create different formation shapes and to create, dissolve and change formations anytime. The resulting formations are dynamic in terms of newly added and departing UAVs and scale well with the number of participants.

Formation handling is described in detail in the further course of the chapter. The presented approach works completely decentralized and the UAVs have to fulfill various requirements in order to create formations. A methodology for formation flights is also presented in this chapter. The UAVs fly in formation using the behavioral approach (cf. Section 3.4.1) allowing them to avoid obstacles by changing their position within the formation. It is possible to rotate the formations around three axes as described in Section 6.4.3. Finally, the underlying mathematics is stated to provide a first step for the use in real world applications.

As mentioned before, using a single formation containing all UAVs is adverse in many applications. Some sort of cooperative behavior is necessary to actually enable the UAVs to create different formations and to choose whether they want to participate in any formation. The UAVs should only create formations resulting in higher benefit and should dissolve formations on their own when benefits vanish. The next chapter presents a cooperative behavior that takes such deliberations into account.

Chapter 7

Cooperative Behavior

The UAVs are able to plan paths in order to fly in complex and dynamically changing three-dimensional environments based on the previously presented potential fields. They also have abilities to create formations and to fly in formation. An extension has been conducted. It enables the UAVs to autonomously plan i discrete paths ρ_{d_i} consecutively for thorough explorations of complex and dynamically changing three-dimensional environments. An implicit coordination has been reached by the use of information exchange concerning the UAV configurations, as well as the areas they currently explore. A high degree of cooperation has been achieved by formation flights.

In contrast to other approaches as, e. g., Subsumption [61] the UAVs do not work in a reactive way. Instead, the UAVs always plan their paths using the descent gradient of an artificial potential field based on environmental information. Additionally, an internal world model of the environment is created during flights if no information about the terrain exists a priori.

Exploration of different environments is only one of the UAVs' duties. A task creation and allocation system is presented in this chapter such that various tasks can be processed simultaneously using the previously described UAV capabilities to achieve the overall goal of supporting ground units. Such a system can be beneficial for various applications like, e. g., site clearing to ensure that ground units reach specific areas, monitoring of specific areas, delivery of goods, etc.

Different UAVs must be able to process several tasks simultaneously. From this it follows that a higher degree of cooperation to process all tasks in an efficient way is needed. An overall UAV behavior based on a highly dynamic task creation and allocation system and enabling the UAVs to process tasks themselves, as well as in cooperation has been designed. The UAVs have to create tasks themselves if this is beneficial or avoids damage. They should also process tasks given by third party applications like ground units or control center personal. These requirements lead to the multi-robot task allocation (MRTA) problem [223].

The UAVs are considered to be heterogeneous in terms of their equipment $\eta_{\mathcal{U}}$ making them more or less suitable to solve specific tasks. A decentralized marketplace (cf. Section 2.2.2), where each UAV bids for the tasks it is willing to allocate, has been established. Task processing is treated as being beneficial to the UAVs, which ensures that they actually allocate tasks and make offers respectively. This chapter presents the task handling system used by the UAVs to distribute various tasks via the single UAVs, as well as the constraints given by such a system.

The aim of this chapter is to design a flexible task allocation system. It should not be limited by the number of different tasks and equipments the UAVs can carry. It should also easily be adaptable in order to reach further UAV behaviors if necessary. Thus, the introduced equipments, tasks, task properties, etc. are only examples and can be replaced or modified in order to design another system or an extension of the presented one.

The chapter first gives a problem statement on task creation, as well as task allocation. Afterwards, state-of-the-art approaches for multi-robot multi-task allocation based on the market place approach are presented. The following section deals with task properties, the role system of the UAVs, and constraints to be considered. The technique for task creation and task allocation is described in detail in the further course of the chapter. Finally, a summary is given.

7.1. Problem Statement

The presented control architecture for multiple UAVs implies three main basics for decision making: task creation, task allocation and coordinated task achievement. Two objectives are mainly considered in this chapter for the design of a cooperative system where the UAVs are able to decide autonomously how to perform all challenges in order to achieve the overall objectives introduced in Section 1.2. The first objective addressed in this chapter is task creation, the second objective is task allocation. Task achievement is performed using the methods presented in Chapter 5 and Chapter 6.

Various tasks must be created and fulfilled to efficiently support ground units. They have different requirements and might not be processable by all UAVs. Additionally, tasks can require multiple UAVs to be processed. They are created dynamically during run time and it takes different time spans to perform them. This makes it impossible to calculate a task allocation that optimizes one objective, like minimizing the costs of the worst UAV, once at the beginning.

One of the most important parts of the presented behavior is the creation of tasks by the UAVs themselves. The next section introduces the situations leading to the creation of such tasks.

7.1.1. Autonomous Task Creation

The single UAVs create tasks by themselves for two purposes. The first one is damage prevention in case of low fuel or connection loss and the second purpose is the design of an efficient multi-UAV behavior. Task creation for damage avoidance is based on the current UAV configuration $q_{\mathcal{U}}$ (cf. Section 4.3).

Configurations forcing a task creation concern the current fuel level of a UAV $\mathcal{U}$. The entry q_9 of $q_{\mathcal{U}}$ is the percentage of remaining fuel. Under all circumstances it must be averted that

$$q_9 < f_s \tag{7.1}$$

with $f_s > 0$ describing a safety fuel level necessary to keep the UAV airborne. Thus, the UAV has to land before Condition 7.1 becomes true.

Configurations treating the communication status q_{10} of $q_{\mathcal{U}}$ are responsible for safe UAV flights. A UAV can only avoid collisions with other UAVs if it knows all other UAVs' current positions. This is normally given by inter-UAV communication, used by all UAVs to send their current configuration to other UAVs. A UAV should fly as safe as possible to the closest landing area if the communication links fail in order to land as soon as possible.

Increasing efficiency is another purpose for task creation beside damage avoidance. Most tasks are designed in a way that they need either only one or multiple UAVs to be processed. Compared to that, exploratory navigation can be performed more efficiently through formation flights in some situations. Let $\mathcal{E}_{\mathcal{U}}$ be the exploration rate achieved by exploratory navigation through single UAVs and let $\mathcal{E}_{\mathcal{P}}$ be the exploration rate using formations. If

$$\mathcal{E}_{\mathcal{P}} > \mathcal{E}_{\mathcal{U}} \tag{7.2}$$

is true, a formation should be created and exploratory navigation should then be performed in formation until Inequality 7.2 does not hold any longer.

Beside the creation of tasks by the UAVs themselves, tasks given by ground units should also be processed. The simple creation of tasks does not automatically lead to their execution. Task execution should be based on an efficient task allocation and has to take various constraints into account. The problem of multi-UAV multi-task allocation as it is considered in this thesis is described in the following section.

7.1.2. Task Allocation

Given n UAVs $\mathcal{U}_i$, $1 \leq i \leq n$ and m tasks $\mathcal{T}_j$, $1 \leq j \leq m$, whereby m, as well as n can change at anytime. Additionally, a benefit value $\mathcal{B}_{i,j}$ of UAV $\mathcal{U}_i$ for task $\mathcal{T}_j$ is given. It is assumed that each UAV $\mathcal{U}_i$ is capable to process only one task $\mathcal{T}_j$ simultaneously and that each task $\mathcal{T}_j$ requires at least one UAV $\mathcal{U}_i$ to be executed. A mapping

$$\forall i, j \colon \mathcal{B}_{i,j} \mapsto \mathcal{T}_j^{\mathcal{U}_i} \tag{7.3}$$

has then to be performed where the benefit values or bids $\mathcal{B}_{i,j}$ of each UAV for each task lead to a successive assignment $\mathcal{T}_j^{\mathcal{U}_i}$ of each task $\mathcal{T}_j$ to at least one UAV $\mathcal{U}_i$.

Several different approaches for multi-robot multi-task allocation exist. A selection of general state-of-the-art approaches is presented in Section 2.3.4. In addition to these approaches a selection of today's task allocation systems based on market places or auctions is presented in the next section.

7.2. State-of-the-Art

A wide range of task allocation methods for multi-agent systems based on auctions exists. Various objectives for task allocation have been identified and dependent on the kind of robots or UAVs and on the properties of the tasks, as well as different requirements be fulfilled. Thus, no method directly applicable to most allocation problems has been designed yet. This section presents a selection of decentralized state-of-the-art algorithms solving the problem of multi-robot multi-task allocation.

The first methodology presented in this section has been developed by Simon et al. [136]. They have designed a group-dependent multi-UAV multi-task allocation system by taking different equipments into account. The equipment is different sets of sensor requirements, whereby each task needs a specific set to be executable. An extension of the consensus-based bundle algorithm has been developed to solve this problem. The algorithm converges to a conflict free and feasible solution. One difference to the approach presented in this thesis is that they allow the simultaneous assignment of UAVs to multiple tasks. Each UAV considered by Simon et al. stores a $m \times n$ matrix with m tasks and n UAVs to achieve the mentioned requirements. In a first step, a bid value is computed by each UAV for each task it assigned and stored in the matrix. The bid value is based on the reward a UAV earns for processing a task and the distance to that task. Thereafter, the single matrices are sent to neighboring UAVs. The points in time at which bids have been received from other UAVs are also stored as it is impossible to guarantee that each UAV always has the relevant up-to-date data. Consensus about the task processing order for each UAV is finally achieved in decentralized manner by the UAVs themselves.

Another approach has been designed by Delle et al. [224]. They presented a task allocation system for the coordination of teams and evaluated it using real UAVs. The aim of their approach is similar to the one presented in this thesis and consists mainly of environmental exploration after disasters using a camera system. Different tasks are of different importance and the performance of the UAVs has to be maximized. In contrast to the UAV approach presented in this thesis the importance of single tasks can change over time and a task is always to send images of a specific area. Delle et al. have also created a completely decentralized task allocation system wherein task allocation is modeled as an optimization problem and solved fully asynchronously and in a decentralized manner using the max-sum algorithm. A utility value similar to a bid value is computed by each UAV for each task based on the distance to the task, the remaining battery capacity, and the properties of the task. Thereafter, an adaptation of the max-sum algorithm solves the task assignment problem.

A further kind of task allocation is considered by Lujak et al. [225]. They assume two groups of mobile agents, so called attackers and targets. The attackers try to capture the targets while the targets try to move away from the attackers. Each attacker allocates one target as its next task and has only a limited communication range and also no insights into the other agents' decision making processes. Lujak et al. use this scenario to investigate the problem that not every agent has up-to-date information, which is also a realistic assumption for real world UAV systems. Every agent stores the most recent knowledge of the actual bid value for each target and the assignments made so far. The algorithm designed by Lujak et al. ensures that each agent exchanges information with all other agents within a given communication range. Thereafter, each agent updates its stored values and checks whether another target is closer than the currently assigned one. In that case it selects the closer target and computes a new bid value for this target.

Similar to the approaches presented in this section the presented UAV system uses tasks to reach an efficient behavior, too. These tasks have different properties and various additional constraints. The UAVs perform task allocation considering given constraints and try to allocate the most important tasks. The properties of the tasks, the different UAV properties used to allocate single tasks, and the constraints of the task allocation system are described in detail in the following section.

7.3. Task Handling Background

Several decisions have to be made in order to design a multi UAV task creation and task allocation system. First, the tasks themselves and their properties have to be defined. The presented system also considers the single UAVs differently based on their equipment and the skills resulting from the equipment. This distinction leads to an assignment of different roles to the single UAVs. Finally, several constraints for task allocation have to be identified and regarded.

The first step in the design of a task allocation system is the definition of the single tasks the UAVs should process with their associated properties as presented in the following section.

7.3.1. Task Properties and Definition

The UAVs have to allocate tasks dependent on various requirements. Each task consists of several properties relevant for task allocation. Some of them change over the course of time while others are static. Other properties are UAV dependent and thus different for each UAV storing the task and the corresponding values. Table 7.1 shows the single properties each task consists of, as well as a brief description.

Property	Description
$I_\mathcal{T}$	Importance of the task
$\bar{\tau}_\mathcal{T}$	Normalized distance between the UAV and the task
$A_\mathcal{T}$	Age of the task
$\eta_\mathcal{T}$	Equipment, needed to process the task
$N_\mathcal{T}$	Number of known UAVs, able to process the task
$Mi_\mathcal{T}$	Minimum number of UAVs, needed to perform the task
$Ma_\mathcal{T}$	Maximum number of UAVs, able to perform the task simultaneously
$O_\mathcal{T}$	Current number of UAVs processing the task simultaneously

Table 7.1.: *The single properties of each task stored by the single UAVs.*

It has to be ensured that some tasks like refueling are always processed before others. This is achieved by assigning different importance values to each kind of task, which has to be taken into account by the resulting task allocation system. A task can only be processed at a given position $p \in \mathcal{W}$ or an area. The position leads to a distance $0 \leq \bar{\tau}_\mathcal{T} < 1$ between the task and the UAV. The center position of an area is considered for the computation of the distance if the task includes an area instead of a position. The UAVs normally fly from one position to another. Thus, the distance between a UAV and a task changes over time.

A UAV is not able to process a task and as a consequence should not try to allocate a task if it is not equipped with appropriate equipment $\eta_\mathcal{T}$. Additionally, aging of tasks is represented by age $A_\mathcal{T}$ in order to avoid starvation. Tasks need a minimum number of UAVs $Mi_\mathcal{T}$ to be performed, while a maximum number of UAVs $Ma_\mathcal{T}$ able to process the task at any given point in time exists. Finally, the number $O_\mathcal{T}$ of UAVs currently processing the task is also a property.

Based on the mentioned requirements the objective of this chapter is the design of a task allocation system were the optimal task to be allocated next $\mathcal{T}_{\text{opt}}$ depends on the properties of the currently existing n tasks $\mathcal{T}_i$, $1 \leq i \leq n$ as follows:

$$\mathcal{T}_{\text{opt}} = f(\{I_\mathcal{T}, \bar{\tau}_\mathcal{T}, A_\mathcal{T}, \eta_\mathcal{T}, Mi_\mathcal{T} \colon 1 \leq i \leq n\}). \tag{7.4}$$

Related to Equation 7.4, the optimal task to be processed is the one with the highest importance, the lowest distance to the UAV, the best fitting requirements, and the oldest one that needs the most UAVs to be processed. It is unlikely that all of this is true for one task. Thus, a tradeoff has to be found to ensure that each UAV is able to determine the task that seems to be most beneficial to itself.

The first step of enabling the UAVs to allocate a task is to use the task properties for a subdivision into different task types:

- Single UAV tasks.
  - UAV dependent tasks (SUDT).
  - UAV independent tasks (SUIT).
- Multi-UAV tasks (MUT).

UAV dependent tasks are the reconnection and the refueling task mentioned before. These tasks cannot be performed by any other UAV than the one that created them. All other tasks are independent from the UAV beside the necessary equipment and most times, different UAVs able to process these tasks exist. Tasks that can only be processed by multiple UAVs are always UAV independent tasks.

Table 7.2 shows an overview of all tasks, which are considered by the presented UAV system. As mentioned before, the tasks differ in several terms and it is possible to extend the task allocation system in order to support additional tasks in an efficient manner.

Task	Description	Type
Reconnect	Landing the UAV due to loss of network connection	SUDT
Refueling	Return of the UAV to the start position for refueling	SUDT
Single exploration	Exploration of the environment using one single UAV	SUIT
Single monitoring	Monitoring of a subarea by one UAV	SUIT
Road spotting	Road spotting for ground units	SUIT
Single deliverance	Deliverance of small goods by one UAV	SUIT
Multiple exploration	Exploration of the environment in formation	MUT
Multiple monitoring	Monitoring of a subarea by several UAVs	MUT
Escort	Escorting ground units	MUT
Multiple deliverance	Deliverance of big goods by several UAVs	MUT
Site clearing	Movement of several obstacles in a certain order	MUT

Table 7.2.: *Tasks considered for the presented UAV system.*

The site clearing task is a complex task where multiple obstacles have to be moved to clear the way for ground units. The task contains the movement of obstacles in a specific order as obstacles may lie beneath others. It is a simplified version of the site preparation task identified by the NASA for Mars missions [226] with the need of subdividing high level tasks into sub-tasks. Deliverable goods can be, e. g., pharmaceutical, water, food, etc., needed by people, as well as the transportation of water used, e. g., for fire fighting.

The presented tasks are only a small selection of tasks a multi-UAV system can fulfill. Thus, the resulting task allocation system should be designed in a way that facilitates the possibility to easily extend the number of supported tasks.

An example scenario with several tasks and various UAVs is presented in Figure 7.1. The figure provides a visualization of the tasks using a cutout from the *Control Station* (cf. Section 4.1.1) and gives an impression of the information ground units obtain from the task handling. The UAVs must distribute the tasks by themselves while these tasks can appear and disappear anytime. Tasks are always visualized the same way while the *Control Station* provides a specific task description in the task list on the left hand side of the graphical user interface.

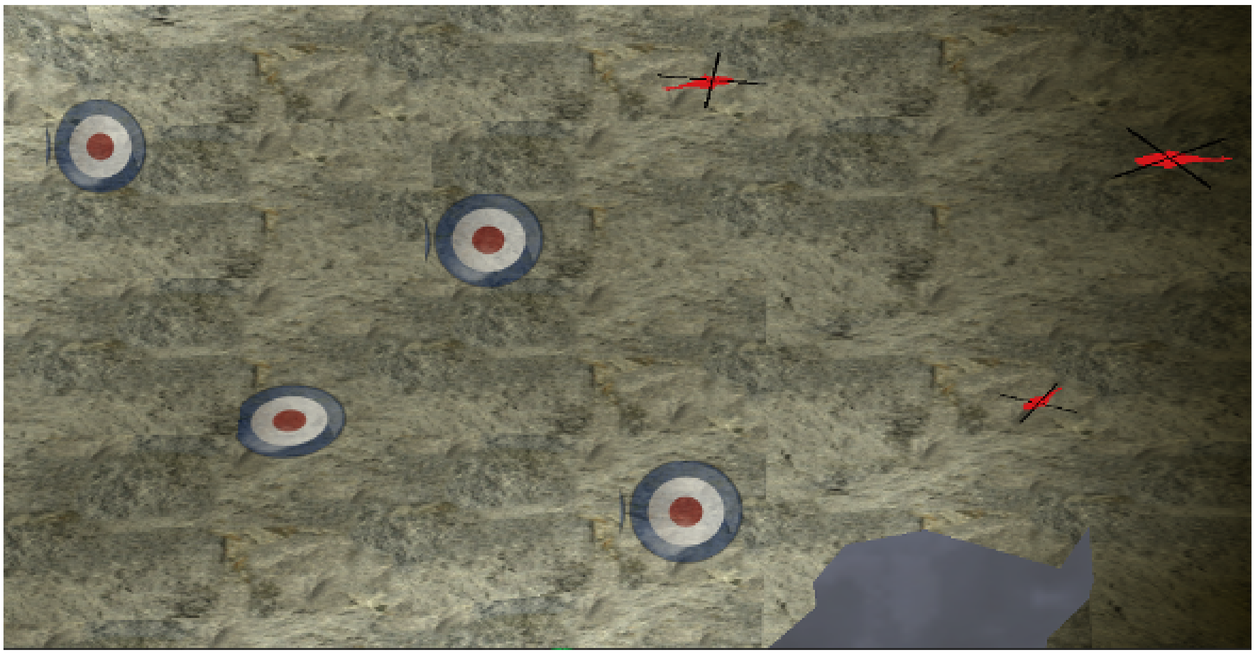


Figure 7.1.: *An example task allocation problem is shown. Three UAVs—located in the upper right—have to allocate four different tasks—illustrated as ring shaped objects—beside exploratory navigation.*

As mentioned before, the UAVs are considered to be heterogeneous in terms of their equipment. The equipment of a UAV is one objective to be taken into account for task allocation. Thus, the division of labor approach (cf. Section 2.1.4) is used to distinguish single UAVs as described in the following section.

7.3.2. Division of Labor

The UAVs need equipment in order to process the single tasks introduced in the previous section. Each UAV has only a limited load capacity. It is therefore impossible to endow each UAV with the entire equipment needed to process every type of task. This leads to different sets of equipment whereby each UAV is able to carry only one specific set. Dependent on the set the UAVs are equipped with, they are more or less suitable for the execution of specific tasks.

In order to model the suitability of the single UAVs, subdivision into different roles based on the nature's division of labor approach (cf. Section 2.1.4) is conducted. These roles are related to the single equipment the UAVs carry and lead to specific priorities for the single tasks as described later in Section 7.4.2. Four roles shown in Table 7.3 have been introduced for the presented system. The roles and the corresponding equipment are only examples. In general UAVs can carry various additional equipment leading to an increase in the number of roles. This should also be considered in the design of the resulting task allocation system.

Role	Equipment
Collector	A gripper to transport and to move objects and a medium-resolution camera
Explorer	A high-resolution wide-angle camera
Prospector	A high-resolution heat camera
Surveillant	A medium-resolution heat camera and a medium-resolution camera

Table 7.3.: *Considered UAV roles and their different equipment*

Collector UAVs are equipped with a gripper and are therefore predestined for the movement of objects necessary for delivery and site clearing tasks. The explorer UAVs carry the best camera equipment making them the optimal choice for exploratory navigation, while the prospector UAVs are equipped with a high-resolution heat camera in order to find, e. g., missing people. Finally, surveillant UAVs carry two cameras. A medium-resolution heat camera and a medium-resolution camera. They are mainly used to monitor areas in order to detect changes.

The roles are used to compute bid values for the different kinds of tasks as described later in order to achieve a task allocation where a UAV that fits best to a single task will allocate it. The roles lead to different priority values used to calculate the bid values in order to achieve an efficient allocation. As mentioned before, UAV dependent tasks exist. The roles should have no influence on the computation of the bid values for these tasks.

The equipment and therefore the role of a UAV cannot change during a flight period. Beside the constraints given by the different equipment several additional requirements must be considered to ensure a feasible task allocation without starvation, deadlocks, and redundant task processing. The resulting constraints considered by the UAV system presented in this thesis are introduced in the following section.

7.3.3. Task Allocation Constraints

A multi-UAV multi-task system faces various challenges introduced in this section. The resulting task allocation system must ensure that several requirements are always met in order to overcome possible issues like starvation. Additionally, task execution should be as efficient as possible.

Exploratory navigation is the main task processed as long as no task more beneficial to the UAVs exists. This task can be processed by each UAV on its own, as well as in formation. The UAVs exchange information about the area they want to explore next to reduce redundant exploration and to increase the exploration rate. Areas selected by a UAV or a formation to be explored next are not selected by other UAVs or formations simultaneously.

Table 7.2 shows several additional multi-UAV tasks with the need of multiple UAVs to be processed beside exploratory navigation. These tasks are always executed by a UAV formation $\mathcal{P}$, whereby

$$\mathcal{P} = \bigcup_{i=0}^{n} \mathcal{U}_i, \tag{7.5}$$

with n describing the number of UAVs $\mathcal{U}_i$ participating at formation $\mathcal{P}$. It is possible that several multi-UAV tasks are processed simultaneously. In this case it has to be guaranteed that no UAV $\mathcal{U}_i \in \mathcal{P}_j$ tries to participate in another formation ($\mathcal{U}_i \in \mathcal{P}_k$) at the same time, i. e.,

$$\forall i \exists k \forall j \neq k \colon \mathcal{U}_i \in \mathcal{P}_j \Rightarrow \mathcal{U}_i \notin \mathcal{P}_k. \tag{7.6}$$

Another issue to be handled is information storage. UAVs can emerge and dissolve during task execution. Thus, it is error-prone if one single UAV is treated as central information holder as all information would be lost if the UAV vanishes. Therefore, the single tasks and the corresponding task properties should be stored redundantly to avoid information loss regarding the currently existing tasks. The easiest way to achieve redundancy is storing all tasks by each UAV. Such a highly redundant storage also has the benefit of each UAV always having up-to-date information about all tasks, allowing for an efficient and completely decentralized task allocation given an appropriate information exchange. Additionally, cases where idle UAVs exist while other UAVs have to process several tasks consecutively are reduced leading to a more efficient behavior in terms of time needed to process all tasks in this case.

Another constraint is given by the importance of the single kinds of tasks. As shown by Constraint 7.7 tasks with much higher importance than other exist. These tasks must be processed first to avoid damage. An example task is the refueling task created when a UAV runs out of fuel. Such a task must be processed before all other tasks and the task currently executed by the UAV has to be interrupted. From this it follows that task execution must be preemptive to obtain such behavior.

$$\exists I_{\mathcal{T}_i} \gg I_{\mathcal{T}_j} \tag{7.7}$$

Additionally, a reassignment of tasks to other UAVs as described by Mapping 7.8 must be possible in order to reach efficient task procession. UAV $\mathcal{U}_i$ should stop the execution of task $\mathcal{T}_j$ such that UAV $\mathcal{U}_k$ can start processing $\mathcal{T}_j$ if this is beneficial.

$$\mathcal{T}_j^{\mathcal{U}_i} \to \mathcal{T}_j^{\mathcal{U}_k} \tag{7.8}$$

UAVs can give up on tasks for several reasons like failure of UAVs. Another reason would be that a UAV leaves a formation to process a newly created, more important task like refueling or re-establishing connection. This has to be taken into account and a UAV that gives up on a task should inform the other UAVs about this decision if possible. Tasks a UAV gave up need to be processed by other UAVs even when the UAV is unable to inform the other UAVs, e. g., because it lost connection. The task then has to be continued by another UAV after a given time span resulting from the point in time the task has been created and the current time. Therefore, the age $A_{\mathcal{T}}$ has been introduced before.

Each task needs some equipment $\eta_{\mathcal{T}_j}$ to be fulfilled. A UAV $\mathcal{U}_i$ must have $\eta_{\mathcal{U}_i} \supseteq \eta_{\mathcal{T}_j}$ to be able to process the task as shown by Constraint 7.9. Thus, the resulting bid system in combination with the division of labor approach presented previously has to ensure that only UAVs carrying the equipment needed to process a task are able to allocate it.

$$\forall 1 \leq i \leq n, 1 \leq j \leq m \colon \eta_{\mathcal{T}_j} \subseteq \eta_{\mathcal{U}_i} \tag{7.9}$$

The refueling task and the reconnection task are special tasks and have to be processed before all other tasks. A connection loss to the other UAVs makes safe flights impossible as the UAV lacks information about the positions of the other UAVs. It then has to fly to the next landing zone while it simultaneously tries to reconnect in order to minimize the collision probability.

A highly dynamic environment wherein tasks can appear and disappear anytime is considered. It makes no sense for each UAV to calculate and offer bids for every single task it is informed about. Thus, each UAV only makes offers for the task it wants to allocate next. The UAVs store the $0 < o \leq \mathcal{O}_{\mathcal{T}}$ highest bid values made by other UAVs for each task currently processed. If a task processed by another UAV $\mathcal{U}_j$ is the most beneficial to UAV $\mathcal{U}_i$, it compares its own bid value $\mathcal{B}_i$ with the previously calculated bid value $\mathcal{B}_j$ of UAV $\mathcal{U}_j$. In this case two possibilities exist:

$$\mathcal{B}_i > \mathcal{B}_j, \tag{7.10}$$
$$\mathcal{B}_i \leq \mathcal{B}_j. \tag{7.11}$$

If Condition 7.10 holds, the UAV is treated to take over the task. If Condition 7.11 holds true, the UAV tries to offer the bid value for the task for which it has computed the next lower bid value. It then offers bid values for tasks with the next highest bid value repeatedly until an allocatable task has been found. Conditions 7.10 and 7.11 are only considered if the number of UAVs currently processing the related task $O_{\mathcal{T}}$ is equal to the maximum number $Ma_{\mathcal{T}}$ of UAVs able to process the task. The UAVs always process the task for which they have computed the highest bid value if $O_{\mathcal{T}} < Ma_{\mathcal{T}}$ holds.

To ensure that tasks created by the UAVs themselves, as well as tasks created by ground units or control center personal are processed, it must be ensured that these tasks are or are going to become more beneficial to the UAVs than exploratory navigation. Additionally, tasks must become more beneficial over time to avoid starvation. Otherwise it is possible that new tasks more beneficial than old ones are created consistently and starvation of tasks becomes possible. Exceptions are continuously performed tasks like monitoring. They should only become more beneficial as long as no UAV has allocated these tasks.

This section describes the task properties, the role system of the UAVs, and constraints that have to be considered by the task allocation system. Based on the information presented so far in this chapter a task handling can be designed. Thus, the following section describes the resulting task creation and task allocation system used to design an efficient multi-UAV behavior that considers all relevant constraints.

7.4. Task Handling

The resulting task handling approach is divided into two subareas. The first part of this section describes task creation by the UAVs themselves. It takes place if it leads to an increase in efficiency or if it is required for damage avoidance. The second part of the section presents the resulting task allocation system. Two situations at which the UAVs create tasks on their own are considered. The first situation is the avoidance of damage while the second situation is the increase of the system's efficiency. The following section describes the resulting situations forcing the UAVs to create new tasks by themselves.

7.4.1. Autonomous Task Creation

Let $\mathcal{U}$ be a UAV and let $\mathcal{T}$ be a task. Three different states for the creation of a task directly by a UAV have been identified for the resulting system. The two states responsible for the avoidance of damage are directly related to the configuration $q_{\mathcal{U}}$ of a UAV $\mathcal{U}$ presented in Table 4.3 and depend on the entries q_9 of $q_{\mathcal{U}}$ and q_{10} of $q_{\mathcal{U}}$.

Configurations $q_{\mathcal{U}}$ forcing a UAV $\mathcal{U}$ to create a task by itself depend on the fuel level and the UAV's speed. Each UAV $\mathcal{U}$ has a fuel level $0 \leq q_9 \leq 100$, whereby q_9 decreases by one every time span t_s. Additionally, $\mathcal{U}$ has a maximum speed q_2, which is part of $q_{\mathcal{U}}$ and the environment a diagonal d. To preserve some safety level, it is assumed that $\mathcal{U}$ needs at the longest time

$$t_m = \frac{d \cdot 1.5}{q_2} \tag{7.12}$$

to reach the fuel station. Combining Equation 7.12 with the fuel level leads to the creation of a refueling task if Condition 7.13 is true.

$$t_s \cdot q_9 \leq t_m \tag{7.13}$$

Further configurations $q_{\mathcal{U}}$ forcing the creation of a task are based on the communication status represented by entry q_{10} of $q_{\mathcal{U}}$. Every UAV exchanges information with all other UAVs. It creates a reconnect task if it does not receive messages from other UAVs and the *Control Station* (cf. Section 4.1) within a certain time span. Messages received by the *Control Station* are considered such that a single UAV is also able to work in the environment.

The tasks based on the previously described two configurations are responsible for avoiding damage to the UAVs. They cannot be performed by UAVs other than the one that created the tasks. The third situation considered for task creation should lead to an increase in efficiency. The task created for this purpose can be performed by any UAV carrying the required equipment.

A UAV $\mathcal{U}$ processing the single exploration task selects unexplored nodes successively at its favorite altitude represented by entry q_4 of $q_{\mathcal{U}}$ to be explored next (cf. Section 5.4.1). Let w_e be the width of the terrain a single UAV camera picture shows and let l_e be the length of the terrain a single UAV camera picture shows. Both values depend on the height of the UAV and the camera's flare angle. Based on these values the UAV is able to explore an area $A_e = w_e \cdot l_e$ at the ground of the environment mainly based on its current altitude and the flare angle θ_d of the bird's eye camera it carries. Inclines of the underlying environment are neglected for task creation.

After the selection of an unexplored leaf l_i by a UAV the maximum coherent and convex area of unexplored leaves $A_{ux} = x_{ux} \cdot y_{ux}$ in the x, y plane containing l_i is computed. If for this area

$$n, m > 2\colon x_{ux} \geq n \cdot w_e \ \wedge \ y_{ux} \geq m \cdot l_e \tag{7.14}$$

is true, $\mathcal{U}$ creates a multi-exploration task with $Mi_{\mathcal{T}} = 3$ and $Ma_{\mathcal{T}} = \max\{n, m\}$. Such a task creation to reach an increase in efficiency is an example far from optimal. One possibility for a more beneficial creation system might be the use of learning strategies, enabling the UAVs to learn when the creation of tasks leads to better results. Nevertheless, the chosen approach should give a first impression of the benefit reachable by the use of formations for exploratory navigation.

Besides for the creation of tasks, task allocation has to take place in order to actually process the single tasks. As described in the previous section each task has several properties and the single UAVs are treated to be heterogeneous in terms of their equipment. Additionally, some constraints must be considered. This leads to the resulting task allocation system presented in the following section.

7.4.2. Task Allocation

The last design step of the presented multi-UAV system is the task allocation approach. Based on this approach the UAVs decide on their own, which tasks they fulfill at which time. The main task is exploratory navigation by single UAVs. This task is replaced if other tasks become more beneficial. Exploratory navigation itself can be conducted by single UAVs and in formation, whereby formations can lead to a more efficient exploration. The UAVs create formations in the cases described in Section 7.4.1. Exploration in formation is evaluated in Section 8.4 and 8.5.

The number of UAVs participating at a formation is restricted by the number of existing UAVs able to process the multi-UAV task, the size of the area to be explored, and tasks of higher importance. Single UAVs can leave formations anytime in order to conduct other tasks if tasks more beneficial than the current formation task come into existence.

As mentioned before, several different tasks exist in addition to exploratory navigation and the UAVs use a decentralized market place (cf. Section 2.2.2) to perform task allocation. A bid system has been designed taking the equipment of the UAVs into account by simultaneously respecting the constraints introduced before.

Priorities have been introduced for single tasks (cf. Section 7.3.1) in order to determine tasks more beneficial to a UAV than others, as well as tasks that need to be processed before all other. The priorities of the tasks depend on the equipment of the UAV, leading to a specific role (cf. Section 7.3.2). Table 7.4 shows the priority values for the single tasks the UAVs of the system presented in this thesis use. These priorities are only a selection and different values should be used by other systems if this seems to be beneficial.

Each UAV uses the role-dependent priority values from Table 7.4 to compute the resulting bid values based on its role. The priorities of the tasks reconnect and refueling are independent of the role as these tasks must always be processed first in order to avoid damage.

Task \ Role	Collector	Explorer	Prospector	Surveillant
Single exploration	0.1	0.5	0.3	0.3
Multiple exploration	0.1	0.7	0.5	0.5
Reconnect	1.1	1.1	1.1	1.1
Refueling	1.2	1.2	1.2	1.2
Single monitoring	0.1	0.3	0.4	0.5
Multiple monitoring	0.1	0.5	0.6	0.7
Escort	0.2	0.4	0.3	0.2
Road spotting	0.3	0.4	0.5	0.6
Single deliverance	0.5	0.0	0.0	0.0
Multiple deliverance	0.7	0.0	0.0	0.0
Site clearing	0.6	0.0	0.0	0.0

Table 7.4.: *The single task priorities dependent on the UAV role*

Tasks the UAVs cannot process are assigned to a priority value of zero. This must lead to bid values $\mathcal{B}_{\mathcal{T}}$ of zero. Additionally, tasks that have to be performed before all other tasks are assigned to priorities greater than one in order to compute higher bid values for these tasks than for the remaining. Thus, bid values $0 \leq \mathcal{B}_{\mathcal{T}} < 1$ are computed for the remaining tasks. The resulting equation for the computation of bid values is developed step-wise in the further course of this section.

First, the single priorities and the distance between the UAV and the target position of the task is taken into account to achieve a bid function. This results in a bid value $\mathcal{B}_{\mathcal{T}}$ for each single task $\mathcal{T}$ as follows:

$$\mathcal{B}_{\mathcal{T}} = I_{\mathcal{T}}^{c+\bar{\tau}_{\mathcal{T}}}, \tag{7.15}$$

whereby the priority of a task is denoted by the value $\mathcal{I}_{\mathcal{T}}$, which is taken up to the power of $c \in \{0, 1\}$ plus the normalized distance $\bar{\tau}_{\mathcal{T}}$. The priorities are based on the role of a UAV and are presented in Table 7.4. The normalized distance $0 \leq \bar{\tau}_{\mathcal{T}} < 1$ between a UAV and a task is strictly increasing with the distance to the center of corresponding task's target area.

The value $c \in \{0, 1\}$ is set to zero for each task except the refueling task. For refueling, it is set to one in order to increase the resulting bid value. This ensures that the bid value is $1 \leq \mathcal{B}_{\mathcal{T}} < 1.1$ for the reconnecting task and $1.2 \leq \mathcal{B}_{\mathcal{T}} < 1.44$ for the refueling task. Due to the other tasks' priorities, which are less than one, the resulting bid values are also less or equal to one.

Equation 7.15 assures that the refueling task is executed before all other tasks followed by the reconnection task. This avoids an alternating connection fail and connection establishment to cause the UAV to switch between the refueling and the reconnection task until it runs out of fuel.

The next task property taken into account for the resulting bid values of uncritical tasks is the age of the task $0 < A_\mathcal{T} \leq 1$ as shown in Equation 7.16 to avoid starvation.

$$\mathcal{B}_\mathcal{T} = I_\mathcal{T}^{c+\bar{\tau}_\mathcal{T}} \cdot A_\mathcal{T} \tag{7.16}$$

$A_\mathcal{T}$ is always set to one for the two critical tasks reconnecting and refueling. For all other tasks $A_\mathcal{T}$ is assigned to an initial value when a task is created and increases over time until it becomes one. Exploratory navigation is an exception. This task is created once at the beginning of the simulation. It is reset to the initial value each time the UAV pauses exploratory navigation to start another task execution in order to avoid that the aging value of this task is always one.

A task becomes more beneficial the higher its aging value is in order to avoid starvation. To increase the bid values of multi-UAV tasks dependent on the number of UAVs needed to process the task, the minimum number of UAVs needed $(Mi_\mathcal{T})$ and the number of UAVs able to process the task $(N_\mathcal{T})$ has also been taken into account for the computation of a multi-UAV value $\mathcal{V}$ presented in Equation 7.17. The number of necessary UAVs is always known to the UAV as it is always sent to them with the task itself. The number of UAVs able to process the task might not be known if communication problems occur and single UAVs are unable to exchange information with each other.

$$\mathcal{V} = \begin{cases} 0, & \text{if } N_\mathcal{T} < Mi_\mathcal{T}, \\ \frac{Mi_\mathcal{T}}{N_\mathcal{T}}, & \text{else.} \end{cases} \tag{7.17}$$

The first case of Equation 7.17 multiplied with the bid value ensures that the bid values for tasks needing more UAVs than exist are zero, whereby $N_\mathcal{T}$ denotes the number of UAVs able to process the task. Such cases can occur as UAVs can leave the application anytime in order to refuel, when they lost connection, or if they are ordered back. The second case multiplied with the bid value decreases the resulting bid values for tasks needing several UAVs to be processed less than for tasks needing only one UAV. It also ensures that the values for the refueling and reconnecting tasks are always one. This is due to the fact that only the UAV that created the tasks is able to process them, i. e., $N_\mathcal{T} = M_\mathcal{T} = 1$ holds for these tasks. Finally, it is ensured that $0 \leq \mathcal{V} \leq 1$.

The number of currently participating UAVs $O_\mathcal{T}$ is not considered. This allows UAVs to take over tasks more easily if they are more suitable for the execution of the task than other UAVs currently performing it. Finally, Equation 7.18 is used by the UAVs to calculate the bid values $\mathcal{B}_\mathcal{T}$ for the single tasks.

$$\mathcal{B}_\mathcal{T} = I_\mathcal{T}^{c+\bar{\tau}_\mathcal{T}} \cdot A_\mathcal{T} \cdot \mathcal{V} \tag{7.18}$$

A UAV computes a bid value $\mathcal{B}_{\mathcal{T}}$ for each single task if it needs to allocate a new task. Afterwards, an offer is made for the task with the highest bid value if Condition 7.10 holds true or if $O_{\mathcal{T}} < Ma_{\mathcal{T}}$. Each UAV also recalculates after a minimum time span the bid values. It returns the current task into its task list and starts processing a new one if the bid values indicate that a new task allocation is more beneficial. Recalculations of the bid values are also performed every time a UAV completes a task and every time the UAV receives a new task. A task can become more beneficial as, e. g., the distance to tasks, as well as the age changes over time. The UAVs can also create new tasks, which can then be most beneficial to them. Additionally, tasks created by other UAVs or by third parties can be received.

In classical market places a time span, within bids can be offered for each task, exists. All bids for a specific task are then offered during this time span. In contrast to that approach, the UAVs can make offers for tasks as long as these tasks are not fulfilled. This makes it possible that UAVs more suitable for tasks than the UAV currently processing them can take over, after they fulfilled the one currently the most beneficial to them, in order to increase the multi-UAV behavior's efficiency.

Two different ways to create multi-UAV tasks exist. UAVs create tasks for exploratory navigation in formation if this results in a more beneficial exploration of single areas. The other possibility is that third party applications like ground units create such tasks. This can be, e. g., monitoring of special areas.

If a multi-UAV task for which $Mi_{\mathcal{T}}$ UAVs are needed has been created, all UAVs compute an offer for this task. A UAV makes an offer if such a task is most beneficial to it. The $Mi_{\mathcal{T}} \leq i \leq Ma_{\mathcal{T}}$ UAVs with the highest offers are finally selected for task execution.

7.5. Summary

The behavior presented in this section is mainly based on two methodologies: task creation and task allocation. The environment changes dynamically and tasks can appear and disappear anytime. Thus, not all tasks are known a priori. This makes it impossible to compute an optimal processing sequence for all tasks once at the beginning. Amongst others, task creation is used by the UAVs to reach a more beneficial exploration in terms of decreasing the time needed to explore entire environments.

Task allocation is not only based on one objective to be optimized like the overall time needed to process all tasks. It takes several properties, like different priorities, various requirements, the age of the task, and also the distance of the tasks to the UAVs into account. Each UAV computes a bid value to determine the task most beneficial to itself based on the properties. Afterwards, it tries to allocate and thereafter to fulfill the task. These properties allow changes of the objectives to be optimized by, e. g., changing the priority values of the single tasks.

The UAV behavior is completely decentralized and autonomous. A central instance assigning the single tasks to the UAVs and also creating tasks as, e. g., exploratory navigation in formation would reduce the complexity of the approach. However, this would lead to a bottleneck and to a single point of failure. Additionally, the approach would not be as robust, flexible, and scalable as a decentralized one. These are some of the main characteristics which should be supported by an autonomous multi-UAV system as mentioned in Chapter 1.

Several different kinds of tasks with different requirements like necessary equipment and minimum number of executing UAVs are considered. An equation to calculate a bid value that forms the basis for task allocation is presented in this chapter. The equation can also be used for further types of tasks and ensures that tasks with unique requirements, e. g., if they need to be processed first are executed in a way that all requirements are fulfilled. Starvation of tasks is also avoided.

The methods presented in Chapter 4 to Chapter 7 describe the entire multi-UAV system in detail. Validation of the mathematical approaches is also conducted in the course of the relevant chapters and the UAVs have all necessary capabilities to reach the objectives introduced in Section 1.2.

The task handling approach presented in this chapter is less formal than the potential field approaches. Therefore, no proof of desired properties like stability is given in this chapter. An evaluation of the resulting system is presented in the following chapter including the task handling system using the example tasks and priorities. The empirical evaluation shows that an implementation of the approach works well and that all approximations described in the preceding chapters result in a computable and efficient system. Additionally, the efficiency of the approach concerning exploratory navigation using different numbers of UAVs is evaluated in the following chapter.

Chapter 8

Evaluation

A detailed description of the system presented in this thesis is given throughout the course of Chapters 4 to 7. Several approximations have been introduced for the resulting approach. They decrease the computational effort needed to compute paths from the initial configurations $q_{\mathcal{I}}$ of the UAVs to the target configurations $q_{\mathcal{G}}$. The basics of the potential field approaches have been verified by showing that they are globally asymptotically stable as described in Section 5.5 and Section 6.5. The verification takes a continuous mathematical space into account.

This chapter presents the evaluation of the entire system. It shows that the approximations made to reduce the computational complexity do not break the system, that the single parts of the system work well with each other, and that the approach is efficient by running various tests using the implementation of the approach. It is also shown that the objectives for a cooperative UAV system, described in Section 1.2, have been reached.

A simulation environment of the resulting system has been implemented for several reasons. As mentioned before, a simulation should be created for safety reasons (cf. Section 1.2). It is also used to examine whether the methods work using discrete arithmetic provided by a CPU. It enfolded a *Control Station* (cf. Section 4.1.1) responsible for the creation and visualization of different environments in three dimensions. Additionally, UAVs are simulated as described in Section 4.1.2. They exchange data with the *Control Station* using TCP/IP in order to receive information regarding new and deleted targets and obstacles, as well as about the environment. They also exchange data with each other using UDP/IP, to always plan their actions based on up-to-date information. All parts of the system have been implemented using GNU g++-4.7 [179].

Several different tests on various environments have been run and published before [3], [4], [5], [6], [8]. They show that the UAVs are always able to fulfill their tasks. The presented evaluation results contain the computation times needed to compute the potential fields, the reduction of nodes achieved by octree use (cf. Section 4.2), and show the increase in efficiency gained by raising the number of participating UAVs for exploratory navigation.

The single test runs are presented during the following sections including exploratory navigation with and without the possibility to create formations. The environment is in some tests known a priori while the UAVs have to explore a completely unknown environment in other test runs. All UAVs used for the test cases have been simulated through a single PC simultaneously. Additional tests regarding obstacle avoidance behavior for both, single UAVs and UAV formations, as well as further test runs regarding the task allocation system described in Chapter 7 are also presented.

The following section describes the assumptions made for the evaluation. These assumptions are partly unrealistic for a real world system. Thereafter, the test environment is introduced. The measured result values and the parameter settings, as well as the objectives to be evaluated are motivated in the further course of the chapter. The following section presents test results for exploratory navigation without formation flights based on the potential field introduced in Chapter 5. Afterwards, test runs evaluating formation flights using the superpositioned potential field presented in Chapter 6 are performed and the results are compared with the previous test results. The last test section considers the UAV behavior with focus on the methodology described in Chapter 7. Finally, a conclusion considering the achieved results and a summary of the chapter is given.

8.1. Assumptions

The tests performed in this chapter are based on assumptions that might not hold for real world environments. No artificial disturbances regarding the network communication have been introduced. Thus, as a single PC simulates all UAVs a nearly perfect network communication is given. In contrast to this thesis, only wireless solutions are suitable for UAV flights in real world environments. These solutions cannot always guarantee perfect communication, e. g., due to network range if techniques like W-Lan are used. Communication can become more robust by the use of a combination of, e. g., LTE, UMTS and dedicated satellite connections. Additionally, interferences, e. g., due to objects being between the UAVs and the counterpart station can always occur. This can cause communication loss, as well as long delays. Section 4.4 briefly introduces a few possible solutions for the establishment of robust wireless communication networks in order to overcome the mentioned issues.

Another challenge real world systems have to face is positioning. The UAVs used for this evaluation can rely on a perfect positioning. The use of currently available positioning systems like GPS [227] results in inaccuracies. Such a system has various deviations and it is not possible to always receive a perfect positioning the UAVs can rely on. The deviations have to be taken into account for real world applications in order to avoid obstacles and other UAVs. The future use of Galileo [228] might reduce the deviations but it will still not be possible to receive a perfect positioning at all. Thus, state-of-the-art UAV systems that fly in formation rely on a tracking system based on bird's eye cameras as, e. g., presented in [132] to compute nearly perfect positions. But such systems cannot be installed in every area UAVs might work in future applications.

A possibility to reduce the positioning inaccuracies is the use of sensors to detect obstacles and other UAVs close by. Such sensor information can be merged with the received position in order to compute a more precise one. Another way to overcome the positioning problem in real world applications is the use of a relative positioning system as it is performed, e. g., in SLAM approaches (cf. Section 2.2.3). The positions computed by the localization part can also be merged with positions received from a positioning system to reach higher accuracy.

The UAVs used for the evaluation are able to fly from one route point to another in a straight line. In real world applications especially in environments consisting of various obstacles wind influences can make this impossible. While the wind influence in an obstacle free environment is most times relatively steady and thus the UAVs might be able to compensate it, strong wind changes are often caused due to obstacles. Strong wind that often changes its direction and strengths after obstacles must be compensated or at least taken into account by obstacle growing in order to avoid collisions using real world UAVs.

The evaluation is based on the assumptions described in this section. A test environment has been created. It is based on real world data in order to achieve meaningful results. The tests consider the potential fields presented before, as well as the task handling approach. They do not consider parts necessary for a real world UAV system and therefore neglected in this thesis. Thus, for the evaluation of the presented UAV system the assumptions can be made and the potential fields are still valid. The following section introduces the environment.

8.2. Experimental Setup

The environment used for the evaluation is based on a heightmap obtained from SRTM data [229] in order to create a realistic environment. SRTM stands for the shuttle radar topography mission of the NASA. It was flown on the space shuttle Endeavour that launched for this mission on 11. February 2000 and took eleven days to measure the world. Digital elevation data of the earth's land mass between 60 degrees north latitude and 54 degrees south latitude have been produced. The achieved data are freely available with a horizontal resolution of one arcsecond (30 m at the equator) for the United States territory and three arcseconds (90 m at the equator) for the rest of the world. The vertical resolution is approximately 6 m.

SRTM data require sensor processing if they are used for autonomous real world flights, since the resolution is too coarse to only rely on this data. In future, more detailed data should be available with a resolution of up to one meter through the German TerraSAR-X satellite [230], [231]. Nevertheless, sensor inputs will then still be necessary as the terrain data provided by satellites does not include buildings, trees, power supply lines, etc.

The test terrain is a heightmap representing a part of the European mountain range Alps and has been scaled to a size of $4,000\ m \times 4,000\ m$. It includes the Lago di Cavazzo lake in the Province of Udine, Friuli-Venezia Giulia, Italy. The terrain has been scaled to a maximum height of $1,200\ m$. The resulting heightmap heights vary between $31.77\ m$ and $650.26\ m$. The simulated UAVs flew at a favorite altitude of $150\ m$ and a maximum altitude of $300\ m$ with a speed of $80\ km/h$ and a climbing speed of $2.22\ m/s$. This results in a free space $\mathcal{C}_{\mathcal{F}}$ of approximately 1.9 billion m^3 the UAVs have to explore.

The UAVs have been equipped with a high resolution bird's eye camera pointing directly towards the ground in order to explore the underlying terrain. Thus, the single UAVs are assigned to the role explorer (cf. Section 7.3.2). The camera has a flare angle of 64° and a camera ratio of $16 : 9$. This leads to a maximum area of approximately $228,500\ m^3$, a single UAV can explore at a time, with this camera at its favorite altitude. The UAV equipment also includes a front camera for all test runs to reach comparability of the single test series. Such a camera is necessary for exploratory navigation in unknown environments to detect obstacles. It has the same flare angle as the bird's eye camera and it is assumed that the UAVs are able to detect obstacles within a range of $100\ m$. Thus, this camera is able to explore an area of up to $67,700\ m^3$ if no obstacles reduce the visibility range. Using a camera is kind of unrealistic to detect obstacles in a distance of $100\ m$. But techniques like lidar (light detection and ranging) or radar (radio detection and ranging) can be used by real world UAVs to reach such detection ranges.

The exploration rate and the necessary time to entirely explore the environment is directly related to the UAV's speed and to the area the cameras can explore at a moment. Changing these values will result in different exploration durations.

The UAVs are homogeneous regarding their dimensions, which are treated to be $8\ m \times 8\ m \times 4\ m$ to ensure a safe distance between them. The dimensions of the UAVs and the dimensions of the environment result in a maximum octree level of eight leading to a maximum number of $8^8 = 16,777,216$ leaves. The environment created at last is visualized by the *Control Station* shown in Figure 8.1.

An extension of the terrain by four models created with 3ds max [182] has been performed. The UAVs have to fly through these models in order to entirely explore the environment. These three-dimensional models are a Buddhist temple [232] shown in Figure 8.2 a), a bridge [233] shown in Figure 8.2 b), a triumph arch [234] shown in Figure 8.2 c), and finally a model of the Eiffel Tower [235] shown in Figure 8.2 d). The model of the Eiffel tower has been scaled to its original height of $324\ m$. The other three models have been scaled such that they are big enough to be entered by the UAVs. All models are distributed over the environment in a way that the UAVs can reach and enter them. They are freely available on the internet and demonstrate that the approach works not only with models explicitly created for the system but also with further three-dimensional objects such as real buildings.

It is assumed that real UAVs are equipped with dedicated CPUs for path computation. The multi-core CPU used for the test runs distorts some of the computation time results presented

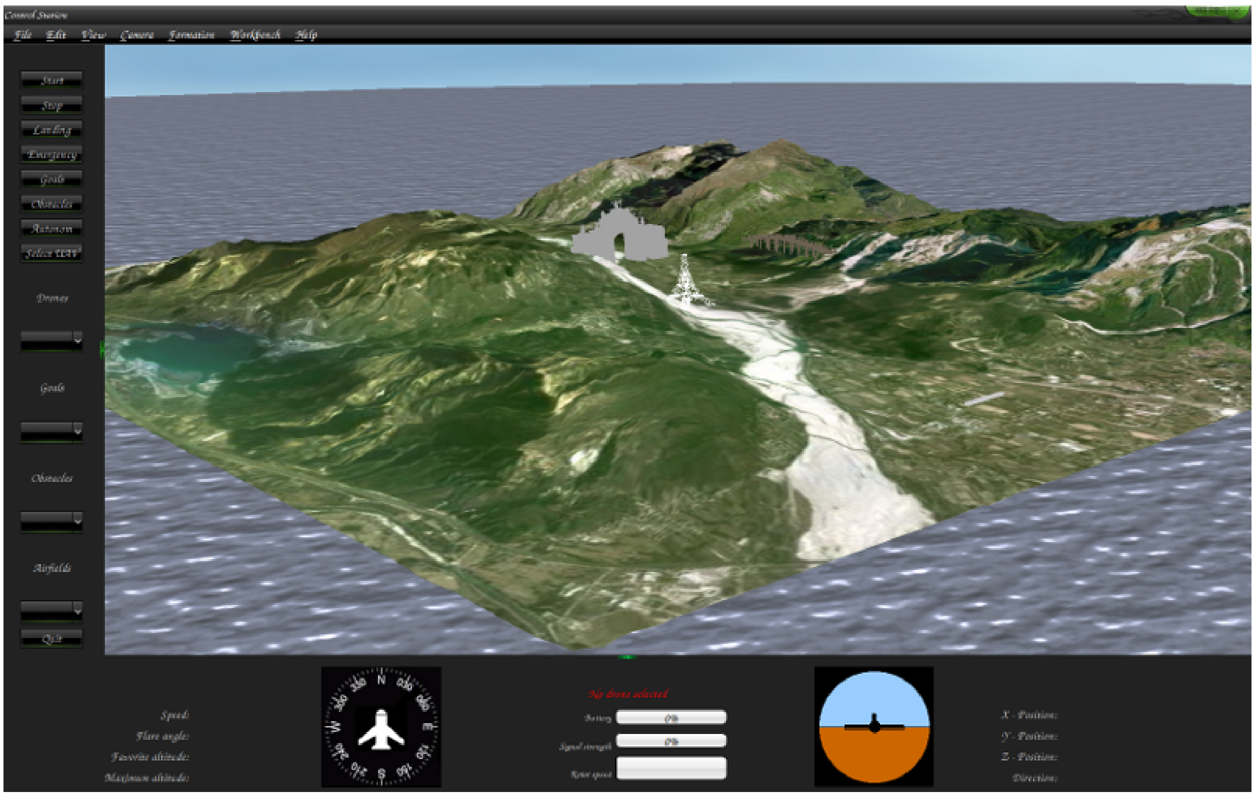


Figure 8.1.: *Visualization of the test environment using the* Control Station.

in the following sections. This is due to the simple fact that path computation for n UAVs using $m < n$ cores leads to simultaneous calculations on the same cores. Nevertheless, the tests have been run on an AMD Phenom II X6 1100T processor with 6×3.3 GHz as one objective is to show that the approach is computationally efficient using up-to-date hardware.

The selected environment is based on real world data and thus represents a possible environment, through which real UAVs might fly in the future. It has been extended with three-dimensional models to respect that real environments also contain three-dimensional structures to be explored, e. g., in case of disasters or in order to find missing people. It is still partly a fictive environment based on real data, which makes it a representative environment leading to relevant results. A countrified environment has been chosen to evaluate the formation approach. Formation flights seem not be very beneficial in a city with many skyscrapers and small street canyons. For such environments only the harmonic potential field approach presented in Chapter 5 might be more beneficial if the main task is exploratory navigation.

The following section introduces, explains, and motivates the measured values used to evaluate the approach. It additionally presents the parameter settings of the single test series used during exploratory navigation.

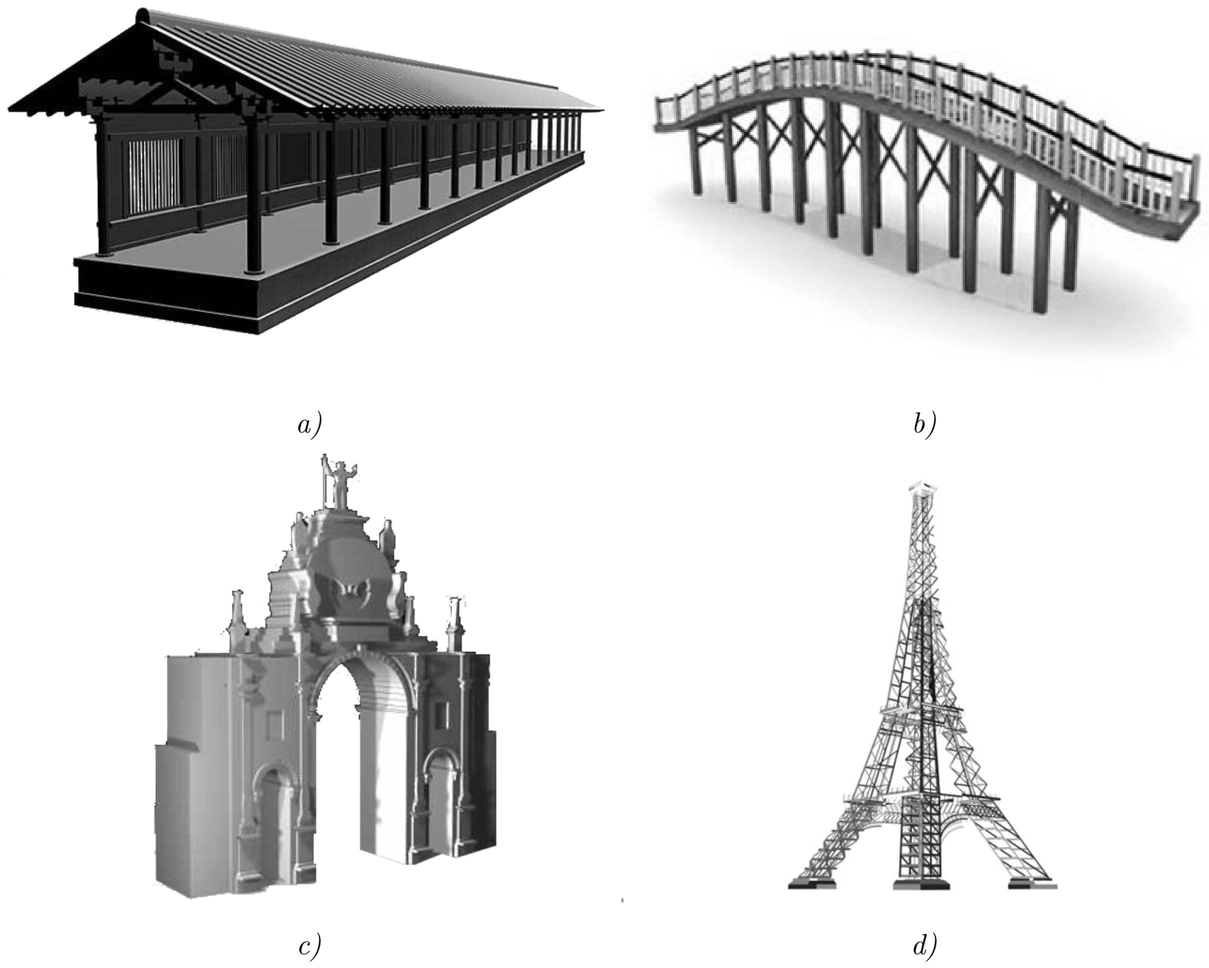


Figure 8.2.: *The four models included in the test environment: a Buddhist temple [232] shown in a), a bridge [233] shown in b), a triumph arch [234] shown in c), and the Eiffel Tower [235] shown in d).*

8.2.1. Measured Values and Parameter Settings

A wide range of results is presented in the following sections. These results often show specific values like the time needed to compute single potential fields. The gained values are only partly reproducible as they depend on various factors, like the start positions of the UAVs, which UAV moves first, the scheduler of the operating system, the used C++ compiler, as well as the code optimization options used for compiling and so on. Nevertheless, these values are presented and used to formulate statements regarding the objectives to be evaluated because it makes, e. g., no big difference if potential field computation takes in average 110 milliseconds or 112 milliseconds. It would be considered as efficient in both cases. The evaluation has not been conducted to just show some specific values but to investigate if the objectives of Section 1.2 have been reached. Therefore, it has been evaluated

- if the approach is efficient using up-to-date hardware.
- what benefit has been achieved by the use of the octree.
- if an increase in the number of UAVs influences the results positively.
- if the consecutive computation of the potential fields leads to the expected behavior
  - during non-formation flights.
  - during formation flights.
  - in static environments represented by a priori known terrains.
  - in dynamic environments represented by a priori unknown terrains.
- if a combination of non-formation flights and formation flights can lead to an increase in efficiency.
- how single UAVs perform obstacle avoidance.
- how UAV formations change in order to avoid obstacles.
- how the task allocation approach works.

Various tests performed using the implementation of the approach in order to give answers to these questions are presented in the further course of this chapter. One focus of this thesis lies on efficient exploration of three-dimensional environments. Thus, the presented environment has been explored in four test series under differing environmental conditions and varying UAV capabilities, regarding the formation's abilities to show that the designed system is able to achieve entire explorations of the environment. The conditions of the single test series are in the following:

1. The environment is known a priori and the UAVs do not fly in formation.
2. The environment is unknown a priori and the UAVs do not fly in formation.
3. The environment is known a priori and the UAVs always fly in formation.
4. The environment is unknown a priori and the UAVs always fly in formation.

Each of the test series contains four test runs while the single runs differ in number of UAVs. Runs using three, six, nine, and twelve UAVs have been performed. Various results have been obtained from each test run. Some of them are average values whereby in this context the term average denotes the arithmetic mean.

One UAV from each test run has been selected to show main results graphically, using diagrams. The most important results consider efficiency in terms of durations the single explorations took and efficiency in terms of low time spans single potential field computations last. The percentage of explored area in the course of time is presented for each single test run to show that the UAVs are able to entirely explore the environment, as well as to highlight effects caused by varying the number of UAVs.

The second value, which is graphically illustrated, is the number of unbound nodes $\#l_{un}$ taken into account for potential field computations. The most costly part of potential field computation is the relaxation of the harmonic potential field (cf. Section 5.3.3). Many researchers try to relax until they reach a given precision, which results in a number of n_i iterations needed to stabilize the potential field. Approximately $n_i \simeq \frac{1}{4}s \cdot n_l$ iterations are necessary to reach a predefined precision of $\epsilon < 10^{-s}, s \in \mathbb{R}^{>0}$,[188], whereby n_l denotes the number of octree leaves. Such a precision is not computed by the UAVs present in this thesis. They create potential fields "good enough" for path planning. So, the number of unbound nodes is presented and compared to the single computation durations t_i, the iteration depth $\varnothing_{depth}$ of the relaxation resulting from the termination conditions (cf. Section 5.3.4), and the average duration $t_\varnothing$ needed to compute the single potential fields. These durations are also illustrated graphically.

For each test run the minimum number of unbound leaves $\#_{min}(l_{un})$, the maximum number of unbound leaves $\#_{max}(l_{un})$, and the average number of unbound leaves $\#_\varnothing(l_{un})$ used for potential field computation by taking all UAVs of each test run into account are presented. These leaves influence the computation times directly.

Additionally, the minimum number of leaves $\#_{min}(l_i)$, the maximum number of leaves $\#_{max}(l_i)$, and the average number of leaves $\#_\varnothing(l_i)$ taken into account for potential field computations are presented. These numbers show the significant decrease in the number of necessary leaves through the use of an octree compared to grid-like structures.

The number of potential field computations $\#_{computations}$, is also presented. This value in combination with the durations the single explorations took, leads to the average time difference between two potential field computations $\varnothing t_{diff}$. If the UAVs are able to compute the potential fields several times within this time difference, they are able to compute a new path before they reach their target configuration. They will fly smoothly without waiting for newly computed paths in this case.

Beside the graphical representation of the computation durations for a selected UAV per test run, the minimum computation duration t_{min}, the maximum computation duration t_{max}, the average computation duration $t_\varnothing$, and the complete computation duration $t_{complete}$ considering all UAVs of each test run are presented.

The presented durations do not denote the time the computation threads spend at the CPU. They are real durations from the start to end of the computation. These times are considered to be more relevant, as real UAVs would also have additional tasks to solve, such as requests from the operating system or stabilizing the UAV in midair, delaying the resulting potential field. This makes it nearly impossible to achieve exactly the same specific values running the following tests twice. Nevertheless, the main statements can be repeated by repeating the tests and only differences of a few negligible milliseconds are achieved by repeating the test runs. Thus, the aforementioned evaluation objectives can be audited.

The iteration of the relaxation has terminated during the following test series if (1) no reachable local equilibrium was left, if (2) the mean potential value change is less than 0.001, or if (3) the iteration depth is equal to the number of considered nodes multiplied by 0.01 plus ten additional iterations (cf. Section 5.3.4). The constant ten has been chosen to ensure that even by a low number of nodes taken into account multiple iterations can be performed to compute out possible existing local equilibria.

These settings have direct influence on the durations needed to compute the single potential fields, as well as on the UAVs exploration capabilities. Using these thresholds such that only a low number of iterations is possible, lowers the times needed to compute the potential fields. This can result in remaining local equilibria, making it impossible for the UAVs to thoroughly explore the environment.

The three termination conditions are mainly responsible for the number of iterations conducted by the UAVs during potential field computations. The number of iterations also affects the computation duration. Hence, the average iteration depth $\varnothing_{depth}$ taking all UAVs of the single test runs into account is presented.

Another part of cost reduction regards the number of nodes taken into account for potential field computations. Leaves, further away from the UAV than 150 m are combined to larger leaves, which are then considered during the following tests.

All UAVs start their flights within a radius of 200 m. This small start area results in disturbances at the beginning of the single test runs. The disturbances increase when the number of UAVs raises. If they are too high, several possibilities to overcome them exist. One possible solution is the distribution of the UAVs over a larger area, which is unlikely for most applications. Another solution would be to start the UAVs one after another with an appropriate time interval between the starts.

The next two sections present the exploration capabilities of the designed system in detail, whereby the next section evaluates the approach without formation flights. The tests described in the succeeding section are run, using formation flights only.

8.3. Decentralized Exploration

The results presented in this section have been obtained while the UAVs never create formations and plan their trajectories based on the potential field presented in Chapter 5. One focus lies on the exploration rate over time. Additionally, the number of nodes used for the potential field computations and the durations needed to compute the single potential fields are graphically presented using the results obtained by a single UAV per test run.

The tests have been run in order to show that several of the objectives necessary to create a robust, flexible and scalable system (cf. Section 1.2) are reached by the designed system. It is investigated whether the UAVs are able to entirely explore the environment in both states, the terrain information is known a priori and the terrain information is unknown a priori. Additionally, it is shown that the approach is computationally efficient and that the behavior is completely autonomous and distributed as claimed in Section 1.2. Further on, the efficiency of the potential field computation and the benefit achieved by using an octree as underlying data structure has been investigated.

All exploratory navigation tests presented in this thesis are conducted as follows. The UAVs first connect themselves to the *Control Station* (cf. Section 4.1.1) in order to receive environmental information. They do not receive any other helpful data or commands from the *Control Station* beside information regarding the terrain. Thus, the UAV behavior is completely autonomous and distributed.

As described in Section 5.4.1, an entire exploration is done stepwise at different altitudes. The UAVs first select leaves at their favorite altitude as part of their next target configuration due to quality requirements of the camera equipment. If all leaves at their favorite altitude have been reached, they start selecting leaves at their favorite altitude multiplied by two and so on until the maximum altitude has been reached. Finally, the remaining unexplored areas of the environment, like rooms in buildings, are explored.

The test environment is described in the previous section and test cases with different numbers of UAVs have been run. The results show that the exploration rate could be increased by raising the number of UAVs due to inter-UAV communication (cf. Section 4.4). Four test runs have been conducted for each parameter setting using three, six, nine, and twelve UAVs.

The presented calculation durations for the computation of the single potential values $\phi^h_{x,y,z}$ are achieved by the algorithm presented in Appendix A.5 and include the following six tasks:

1. Creation of the Dirichlet Boundary Condition (cf. Section 5.3.1).
2. Creation of a list of all unbound leaves (cf. Section 5.3.5).
3. Conduction of the pre-calculations (cf. Section 5.3.5).
4. Computation of the first approximation (cf. Section 5.3.2).
5. Relaxation of the first approximation (cf. Section 5.3.3).
6. Check for termination conditions (cf. Section 5.3.4).

Path computation itself is not included in the durations as it is based on the negative gradient $-\nabla$. The computation of a first route point consists only of comparing the potential value of the leaf enclosing the UAV with the potential values of the neighboring leaves. These comparisons are done in less than one millisecond and thus have no noticeable influence on the results. The computation durations show the efficiency of the approach. These durations, as well as additional results are presented in the next sections obtained by exploratory navigation in an a priori known environment through three, six, nine, and twelve UAVs without formation flights.

8.3.1. Known Environment

The environment presented in Section 8.2 has been known to the UAVs a priori and has been explored four times using different numbers of UAVs in the first test series. Additionally, the UAVs do not create formations in these test series. Figure 8.3 shows the percentage of explored space over time. The x - axis represents the time line. It starts at time zero, which marks the start of the simulation and illustrates the duration of the exploration in minutes. The y - axis shows the percentage of explored free space $\mathcal{C}_{\mathcal{F}}$ at each point in time.

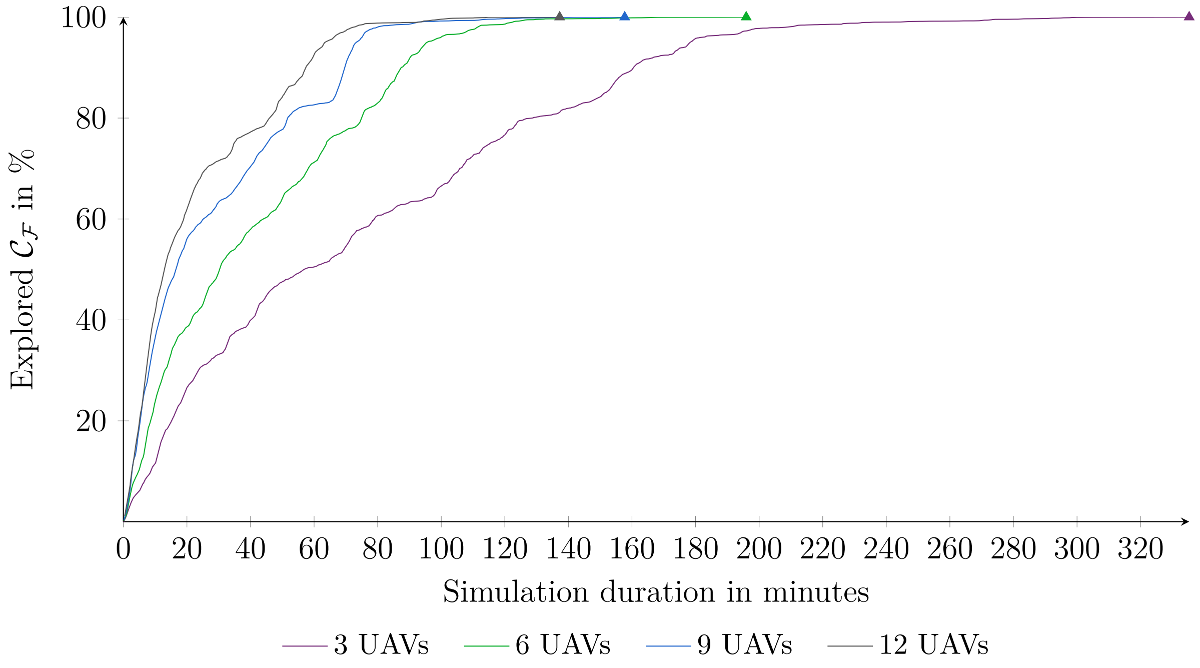


Figure 8.3.: *Percentage of explored free space at each point in time for an environment known a priori without formation abilities by 3, 6, 9, and 12 UAVs performing exploratory navigation.*

It can be seen that an exploration of the entire free space has been achieved in all four test runs. The time needed to explore the environment decreases significantly by increasing the number of UAVs from three to six. Three UAVs needed 335.28 minutes to explore the environment while six UAV performed the task in 195.93 minutes. A further reduction of the exploration duration down to 157.71 minutes has been achieved by using nine UAVs while twelve UAVs were able to explore the entire environment in 137.25 minutes.

These durations show that raising the number of UAVs leads to a benefit regarding the time needed to entirely explore the environment. Nevertheless, while an increase from three to six

UAVs results in a benefit of 139.35 minutes a further raise of three UAVs to nine reduced the time needed only by about 38.22 minutes. The last increase of the number of UAVs to twelve reduced the time needed only by about 20.46 minutes. This shows that the exploration duration does not correlate linearly with the number of UAVs used for exploratory navigation. Instead, the benefit is lowered the more UAVs are used.

Using real UAVs leads to costs per flight hour. Taking the complete flight times into account results in 1,005.84 minutes using three UAVs, 1,175.22 minutes using six UAVs, 1,419.39 minutes using nine UAVs, and 1,647 minutes using twelve UAVs to entirely explore the environment. So, considering the costs per flight hour, it becomes more beneficial to only use a relatively low number of UAVs for exploratory navigation.

Figure 8.3 also shows the exploration rate to decrease, the more free space has been explored. Three UAVs, e. g., needed only 57 minutes to explore 50% of the environment while it took them two hours to explore 77.5%. To reach an exploration of further 17.5% to 95% took them about an additional hour and they had to spend another two and a half hours to explore the remaining 5% of the environment. This is mainly caused by the relatively uncoordinated exploration at the beginning. The UAVs start like foraging ants (cf. Section 2.1.1) by spreading out in various directions to explore the environment and thus several small areas are missed. These areas are distributed over the environment and have to be explored one after another at the end of the exploration

Figure 8.4 shows the number of unbound nodes taken into account for the single potential field computations by one selected UAV per test run. The potential field computations are numbered from 1 to number of potential field calculations $\#_{computations}$ needed by the selected UAVs until the entire free space has been explored. They are pictured on the x - axis. The y - axis shows the number of explored nodes used to compute the potential field number $i = 1, \ldots, \#_{computations}$. The unbound nodes have been selected as they influence the necessary time to compute the potential fields directly, while the bound nodes only need to be assigned to a fixed value (cf. Section 5.3). It can be seen that the number of explored nodes is relatively steady independent of the number of UAVs used for the single test runs. A maximum number of 8,324 explored nodes has been taken into account for potential field computations by the presented UAVs. The differences in the number of nodes results from the position of each UAV during potential field computations. The closer the UAV is to the center of the environment the higher is the number of leaves. This is due to the fact that if the UAV is, e. g., more than 150 m away from the center of the terrain, $\frac{7}{8}$ of the environmental nodes can be combined to seven nodes.

Table 8.1 shows several statistical data regarding the nodes considered for potential field computations. The complete number of nodes $\#(l_i)$ taken into account for potential field calculations lies within the range from 575 to 12,852 for all UAVs, while the number of explored nodes $\#(l_{un})$ is between 67 and 12,131. These values are barely influenced by the number of UAVs used for single test runs.

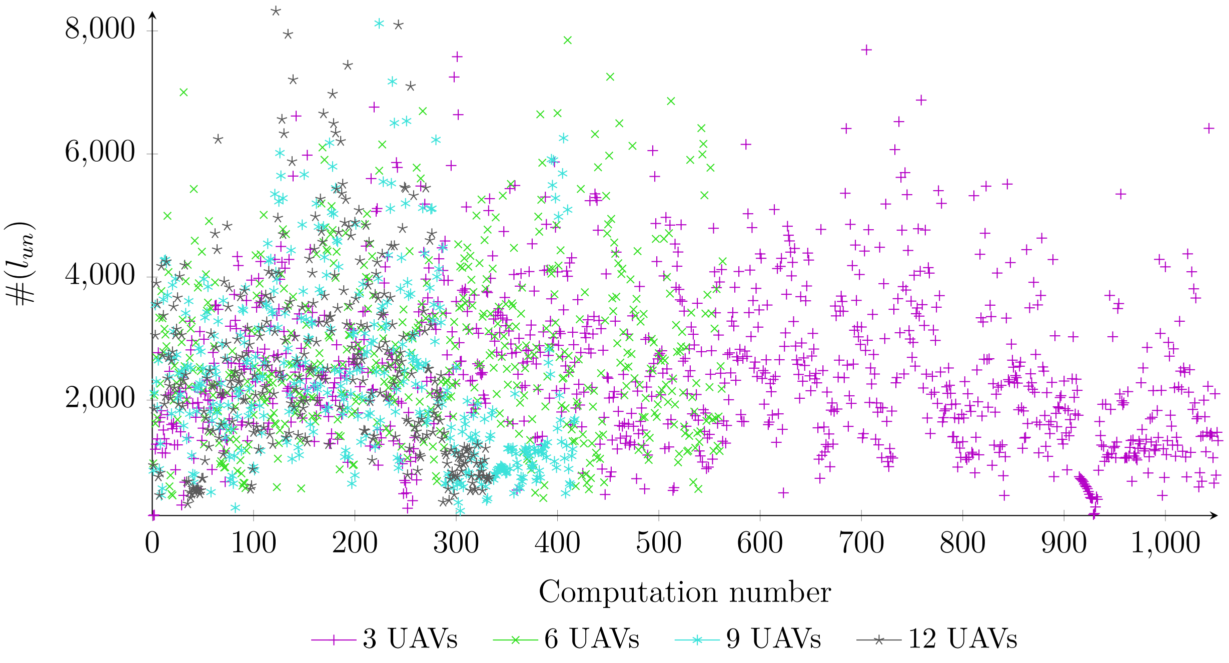


Figure 8.4.: *Nodes considered for the single potential field computations using an environment known a priori without formation abilities by the use of 3, 6, 9, and 12 UAVs for exploratory navigation.*

The highest average number of nodes has been considered by six UAVs, which needed 4,054.13 nodes in average. The maximum average number of unbound nodes has also been taken into account by six UAVs, which needed 2,769.92 unbound nodes in average. This shows that the number of UAVs does not correlate with the number of octree nodes needed to compute the potential fields.

The maximum number of 12,468 nodes used for potential field computations shows the great advantage of a tree-based data structure compared to a grid-like structure, which would have resulted in approximately 16.777 million leaves (cf. Section 8.2) to reach an equal discretization.

Additionally, it can be seen from Table 8.1 that raising the number of UAVs will result in an increase of the number of necessary computations $\#_{computations}$ to entirely explore the environment. The number of computations is an indicator for distortion of UAVs amongst others. As they always get too close to each other, avoidance paths are calculated.

Figure 8.5 shows the single durations needed to calculate the potential fields, as well as the average durations. Always one UAV has been selected from each test run for the representation of this data. The x - axis shows the computation number while the y - axis illustrates the durations needed to compute the single potential fields, as well as the average computation durations in milliseconds.

	3 UAVs	6 UAVs	9 UAVs	12 UAVs
$\#_{min}(l_{un})$	67.00	67.00	67.00	67.00
$\#_{max}(l_{un})$	8,476.00	11,464.00	11,261.00	12,131.00
$\#_{\varnothing}(l_{un})$	2,381.64	2,769.92	2,688.79	2,641.78
$\#_{min}(l_i)$	603.00	659.00	575.00	575.00
$\#_{max}(l_i)$	9,479.00	12,314.00	11,852.00	12,468.00
$\#_{\varnothing}(l_i)$	3,608.05	4,054.13	3,921.35	3,898.93
$\#_{computations}$	2,862.00	3,135.00	3,346.00	3,750.00

Table 8.1.: *Minimum $\#_{min}(l_{un})$, maximum $\#_{max}(l_{un})$, and average $\#_{\varnothing}(l_{un})$ number of unbound nodes, as well as minimum $\#_{min}(l_i)$, maximum $\#_{max}(l_i)$, and average $\#_{\varnothing}(l_i)$ number of nodes used for potential field computation. The number of potential field computations $\#_{computations}$ is presented, too.*

The average duration increases significantly at the beginning and decreases in the further course of exploratory navigation. This is independent from the number of UAVs used for the test runs. It results, on the one hand, from the fragmentation of the environment at the beginning. The UAVs start to fly in different directions, which results in small explored areas distributed all over the environment.

On the other hand, the UAVs start in short distance to each other. This causes a higher complexity of the area considered to be at maximum discretization level and leads to a higher number of necessary iterations to compute out all local equilibria created by the small obstacle areas, which are close to each other and are caused by the other UAVs. These two effects are reduced over time in this test series, leading to a decrease of the durations needed to compute the single potential fields.

It can also be seen that the duration needed for potential field computations increases when the number of UAVs is raised. This is not surprising as more UAVs lead to a higher complexity of the environment. The average computation durations $t_{\varnothing}$ using the selected UAVs from the test runs with three and six UAVs are 75.77 milliseconds and 102.38 milliseconds, which does not differ as much as the computation durations between six, nine (134.36 milliseconds) and twelve (226.48 milliseconds) UAVs. This can partially be explained by the number of cores and the number of simulated UAVs. As mentioned before using more than six UAVs on a CPU with only six cores leads to simultaneous potential field computations on the same core. This is an external influence and increases the durations needed to calculate single potential fields. Nevertheless, it can be seen in Figure 8.5 that the single computation durations t_i for the selected UAVs are above one second, six times only, which only occurs, using twelve UAVs. Thus, it can be assumed that the approach is really efficient.

Table 8.2 shows some statistical data regarding the calculation durations needed to compute the potential fields by taking all UAVs of the single test runs into account. The maximum computation duration t_{max} for potential field calculations are often peaks, e. g., using six UAVs t_{max} took 639 milliseconds, while it only occurred five times during the entire test

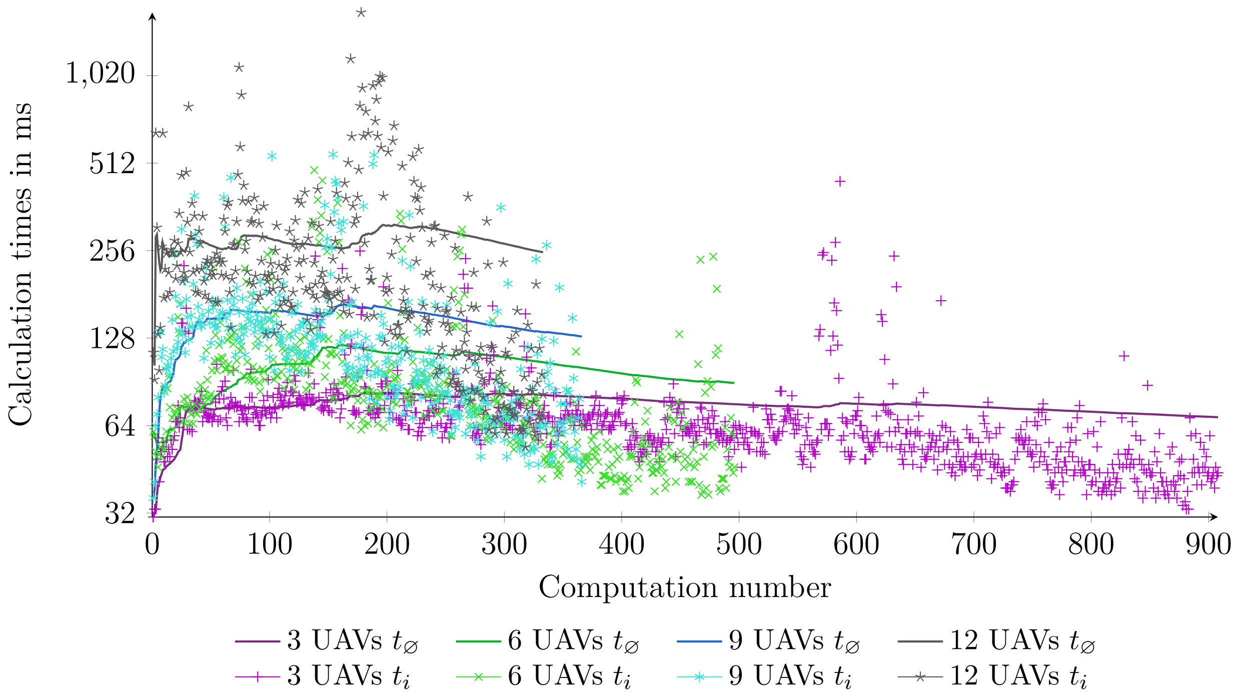


Figure 8.5.: *Single computation durations t_i, as well as the average computation durations $t_\varnothing$ needed to compute the potential fields in an environment known a priori without formation abilities by the use of 3, 6, 9, and 12 UAVs for exploratory navigation.*

run, that more than 500 milliseconds were necessary to compute the potential field. But in average it took only 102.38 milliseconds to compute the potential fields.

The results presented in Table 8.2 also show that the statement "raising the number of UAVs also increases the average computation durations $t_\varnothing$" holds true for all UAVs. Doubling the number of UAVs from three to six increased the computation durations by approximately 35% to 102.38 milliseconds. Tripling the number of UAVs from three to nine leads to higher computation durations of approximately 77% to 134.36 milliseconds, while the use of twelve UAVs resulted in durations of approximately 299% higher than in case of three UAVs up to 226.48 milliseconds. Nevertheless, a maximum average duration of 226.48 milliseconds by the use of twelve UAVs enabling them to entirely explore such a large environment denotes a really fast path planning approach.

The average iteration depth $\varnothing_{depth}$ needed to reach the termination conditions introduced before, shows that this depth is not directly related to the number of UAVs as it increases by raising the UAVs from three to six while the lowest value has been reached using twelve UAVs. It can also be seen that a higher number of UAVs also leads to an increase of the entire computation durations $t_{complete}$ from 216.86 seconds using three UAVs up to 849.31 seconds using twelve UAVs.

Finally, Table 8.2 shows the average time elapsed between two potential field computations $\varnothing t_{diff}$ in seconds. These values are between 21.09 seconds and 26.35 seconds. Comparing

		3 UAVs	6 UAVs	9 UAVs	12 UAVs
t_{min}	*ms.*	31.00	32.00	32.00	33.00
t_{max}	*ms.*	443.00	639.00	1,102.00	1,687.00
$t_{\varnothing}$	*ms.*	75.77	102.38	134.36	226.48
$t_{complete}$	*sec.*	216.86	320.95	449.57	849.31
$\varnothing_{depth}$		8.37	11.05	9.67	7.77
$\varnothing t_{diff}$	*sec.*	21.09	22.50	25.45	26.35
$\#_{computations}$		2,862.00	3,135.00	3,346.00	3,750.00

Table 8.2.: *Minimum t_{min}, maximum t_{max}, and average $t_{\varnothing}$ durations needed for potential field computations. Additionally, the complete duration $t_{complete}$ needed for the computations, the average iteration depth $\varnothing_{depht}$, the average duration difference between two potential field computations $\varnothing t_{diff}$ and the complete number of necessary computations $\#_{computations}$ by taking all UAVs of the single test runs into account.*

them with the durations needed to compute a potential field–the average durations $t_{\varnothing}$, as well as the maximum durations t_{max}–shows that the UAVs are able to compute various paths on their way from the initial configuration to the target configuration. It is not always desirable to reach a long time span between two potential field computations as this also indicates that the UAVs must plan long paths from one unexplored area to another. Such cases result in higher durations necessary for the entire exploration of the environment.

This section presents in detail the abilities of the UAV system described in this thesis without formation flights in an a priori known environment. In addition to these capabilities, the UAVs are able to entirely explore unknown environments, for which the results are described in the following section.

8.3.2. Unknown Environment

The test series described in the previous section has been repeated, while the terrain was unknown a priori to the UAVs. Unknown environments lead to different results as the UAVs are then not only able to compute paths to unexplored areas, but also to occupied areas they detect during path following. In such cases a recalculation of the path is performed.

The detection of formerly unknown obstacles makes the environment more dynamic. UAVs working in real world environments must be able to react properly to new situations like newly detected obstacles intersecting their paths. The section presents results obtained in such a dynamic scenario and shows that the system is able to face this challenge.

The intention of this test series is to show that the objectives of Section 1.2 have also been reached for exploratory navigation in unknown environments. This is important for many applications, such as the exploration after big disasters, which caused significant changes to the affected terrain.

Figure 8.6 shows the exploration rate over the course of time. The x - axis represents the time line while the y - axis shows the percentage of explored free space at any point in time. It can be seen that environmental exploration took longer in each test run than in the corresponding test runs described within the previous section. This shows that the approach becomes more efficient, the more information the UAVs have on the environment.

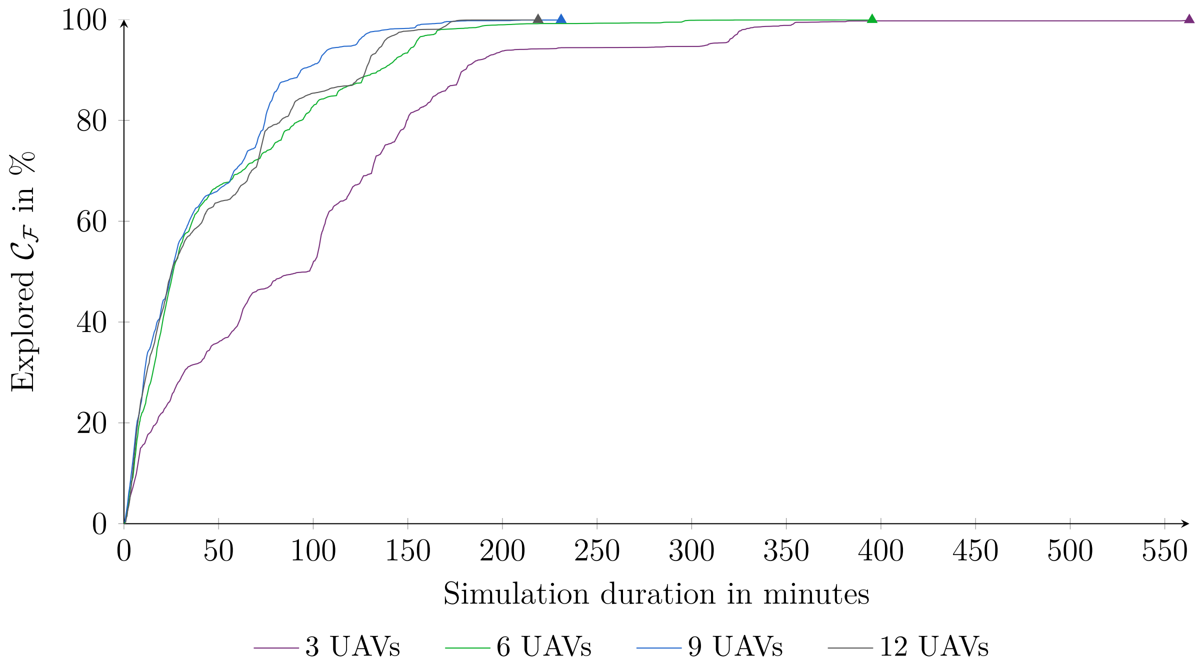


Figure 8.6.: *Percentage of explored free space at each point in time for an environment unknown a priori without formation abilities by 3, 6, 9, and 12 UAVs performing exploratory navigation.*

It can also be seen that the use of twelve UAVs results in a small benefit compared to the test run with nine UAVs in this test series. An increase of the number of UAVs from three to six decreased the time needed to entirely explore the environment from 583.75 minutes to 395.36 minutes and a further raise of the UAVs to nine additionally reduced the time needed to 231.01 minutes. Exploratory navigation of the entire environment took 218.8 minutes using twelve UAVs, which is not a big benefit compared to the result using nine UAVs. Taking the costs per flight hour into account, leads to the same result, which was obtained in the previous section. The lesser UAVs are used, the lesser number of flight hours is necessary to entirely explore the environment. This effect would only be reversed if the duration needed to explore the entire environment could be reduced significantly, e. g., about more than half if twice the number of UAVs are used.

A higher duration is needed in all four test runs to entirely explore the environment, compared to the results obtained when the environment was known a priori. This is caused by newly detected obstacles as these obstacles often make the current path obsolete and a new path has to be computed. The result also means that the path computed before is less optimal than it would be if information on the obstacles exists before path planning starts. Thus, the better knowledge the UAVs have regarding the environment, the more efficient the paths are, which they compute.

The exploration rate of the free space is similar for the first 20% in all test runs. Noticeable is that nine UAVs were able to explore 90% of the environment faster than twelve UAVs. Only at the end, when the UAVs had to visit the small remaining unexplored areas the distribution of twelve UAVs allowed them to explore the entire environment faster than nine UAVs.

Exploration of unknown environments leads to a high number of environmental changes making the environment highly dynamic. The results show that the UAVs are able to entirely explore such environments. This means that they behave self-adaptive to environmental changes, which fulfills one objective from Section 1.2.

Figure 8.7 shows the number of unbound nodes used to compute the single potential fields by one chosen UAV per test run. The computations are numbered from 1 to $\#_{computations}$. The number of nodes is pictured on the y - axis while the number of the potential field computations is shown on the x - axis.

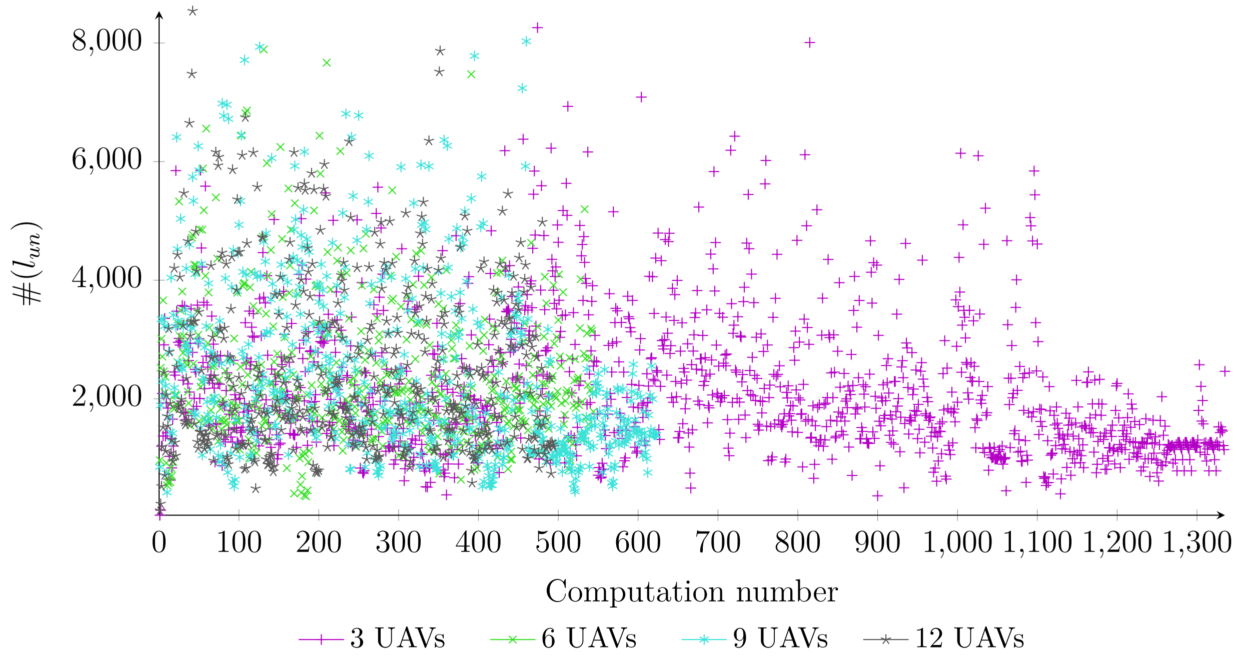


Figure 8.7.: *Nodes considered for the single potential field computations using an environment unknown a priori without formation abilities by the use of 3, 6, 9, and 12 UAVs for exploratory navigation.*

Similar to the results obtained in an a priori known environment, the number of nodes is relatively steady independent from the number of used UAVs for the single test runs. It always lies below 10,000, which is much less than the 16.777 million leaves a grid-like structure would have produced. This again shows the benefit achieved through the octree. Reducing the number of nodes is necessary to guarantee fast potential field computations (cf.

Section 5.3.3). The octree fulfills this objective without breaking the approach as still entire explorations of the environment under varying conditions have been achieved.

Table 8.3 shows several statistical data regarding the nodes taken into account for potential field computation. Compared to exploratory navigation in an a priori known environment, the average number of nodes $\#_{\varnothing}(l_i)$ taken into account is higher. This results from the distribution of the occupied space. In a known environment all occupied leaves are known at the beginning and the UAVs are able to combine them to bigger nodes and thus reduce the overall number of nodes. This is not the case in unknown environments and results in a higher number of nodes as only small areas are marked as occupied in the beginning.

Another noticeable aspect shown in Table 8.3 is the number of potential field computations. Compared to the results obtained in an a priori known environment more potential field computations were necessary in all four test runs. This is caused by newly detected obstacles. These obstacles can on the one hand intersect the current path of the UAVs, which forces a new path computation. On the other hand, newly detected obstacles can lead to areas unreachable for the UAVs, but set as reachable before obstacle detection. A new path computation must occur if these unreachable areas have been selected as next target area earlier.

	3 UAVs	6 UAVs	9 UAVs	12 UAVs
$\#_{min}(l_{un})$	37.00	67.00	37.00	37.00
$\#_{max}(l_{un})$	8,594.00	8,466.00	10,602.00	10,119.00
$\#_{\varnothing}(l_{un})$	2,044.23	2,196.71	2,351.04	2,361.30
$\#_{min}(l_i)$	505.00	603.00	617.00	400.00
$\#_{max}(l_i)$	12,580.00	13,448.00	13,434.00	14,169.00
$\#_{\varnothing}(l_i)$	4,466.40	4,652.04	4,826.21	4,891.05
$\#_{computations}$	3,982.00	4,994.00	4,652.00	5,056.00

Table 8.3.: *Minimum $\#_{min}(l_{un})$, maximum $\#_{max}(l_{un})$, and average $\#_{\varnothing}(l_{un})$ number of unbound nodes, as well as minimum $\#_{min}(l_i)$, maximum $\#_{max}(l_i)$, and average $\#_{\varnothing}(l_i)$ number of nodes used for potential field computation. The number of potential field computations $\#_{computations}$ is presented, too.*

Figure 8.8 shows the durations needed to calculate the single potential fields, as well as the average durations. Similar to the results shown in the other diagrams, one single UAV has been selected from each test run to graphically represent the obtained data. The x - axis shows the number of potential field computations $\#_{computations}$ while the y - axis shows the durations needed to compute the potential fields.

It can be seen that the results regarding the computation times are higher than the ones achieved in the test series with an a priori known environment. This means that the higher complexity a dynamic environment comes with, directly influences the efficiency of potential field computations.

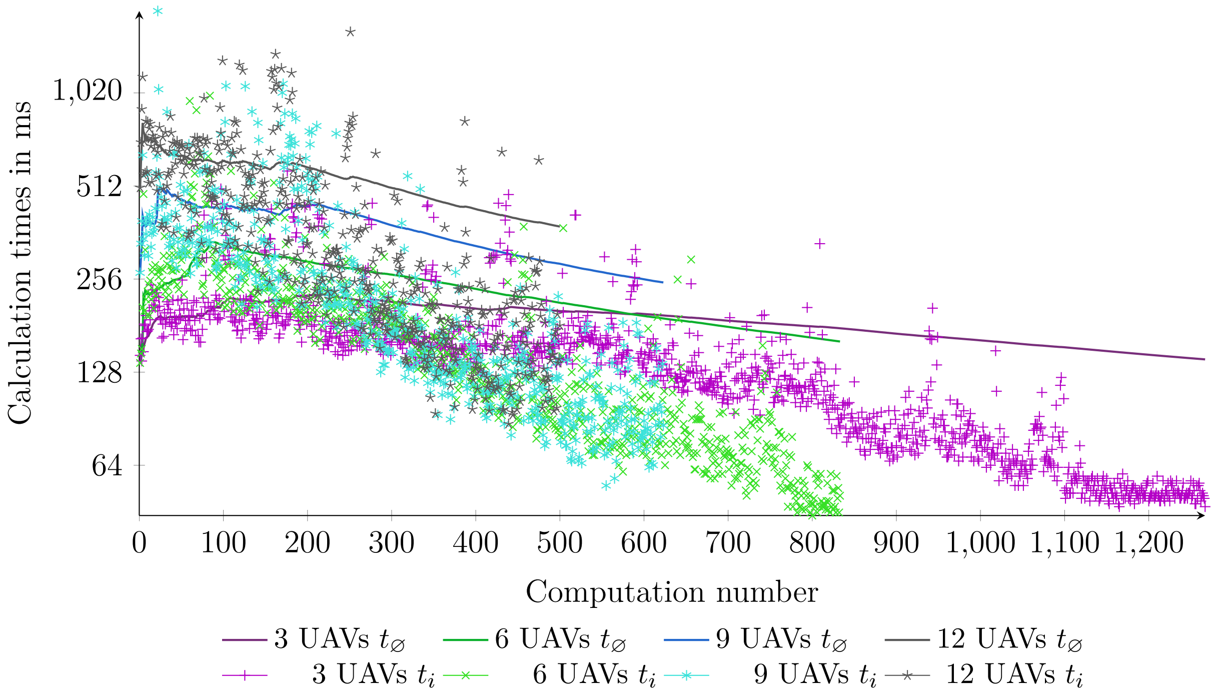


Figure 8.8.: *Single computation durations t_i, as well as the average computation durations $t_\varnothing$ needed to compute the potential fields in an environment unknown a priori without formation abilities by the use of 3, 6, 9, and 12 UAVs for exploratory navigation.*

The computation times are relatively high at the beginning of exploratory navigation and diminish over the course of time similar to the results achieved by the use of an a priori known environment. The peaks are also similar to the ones in an a priori known environment and thus the approach is also efficient for dynamic environments with many environmental changes. So, the objective from Section 1.2 that the approach has to be computationally efficient can be reached using harmonic potential fields in both applications, static and highly dynamic environments.

Other statements from the previous section also hold true for unknown environments. The durations needed to compute potential fields rise with an increasing number of UAVs. This is caused by the higher complexity, more UAVs lead to. The durations do not correlate linearly with the number of UAVs. The UAV representing three UAVs in Figure 8.8 needed 140.70 milliseconds in average, while the UAV selected to represent six UAVs needed 160.74 milliseconds. This is only a small increase. A much higher raise of the durations has been achieved by nine UAVs where the presented UAV needed 249.48 milliseconds in average. But even the average duration of 378.36 milliseconds necessary to compute the potential fields using twelve UAVs is still a good result as the selected UAV was able to compute nearly three potential fields in average per second. With these durations the UAVs are able to quickly react to newly detected obstacles and to compute feasible paths avoiding obstacles.

Table 8.4 shows some statistical data regarding the computation durations needed to calculate the single potential values by taking all UAVs of the test runs into account. It can be seen that similar to the results obtained in the test runs with an a priori known environment raising the number of UAVs increases the durations necessary for potential field computations. Increasing the number of UAVs from six to nine results in much higher durations than in an a priori known environment. This is partially caused by the higher computational effort an unknown environment leads to. Determination of unreachable areas, e. g., has to be done several times for unknown environments. As the simulations are conducted using a six core CPU such computations have a higher effect on simultaneous potential field computations if they have been performed on the same core.

		3 UAVs	6 UAVs	9 UAVs	12 UAVs
t_{min}	*ms.*	36.00	44.00	43.00	74.00
t_{max}	*ms.*	577.00	1,002.00	1,883.00	2,581.00
$t_{\varnothing}$	*ms.*	140.46	178.98	357.78	463.00
$t_{complete}$	*sec.*	559.31	893.82	1,270.12	2,193.72
$\varnothing_{depth}$		6.66	8.17	8.08	7.73
$\varnothing t_{diff}$	*sec.*	25.45	28.50	26.81	31.16
$\#_{computations}$		3,982.00	4,994.00	4,652.00	5,056.00

Table 8.4.: *Minimum t_{min}, maximum t_{max}, and average $t_{\varnothing}$ durations needed for potential field computations. Additionally, the complete duration $t_{complete}$ needed for the computations, the average iteration depth $\varnothing_{depht}$, the average duration difference between two potential field computations $\varnothing t_{diff}$ and the complete number of necessary computations $\#_{computations}$ by taking all UAVs of the single test runs into account.*

The duration of more than two and a half seconds the potential field computation took at maximum is not a good result. But this value is more than five times higher than the average duration needed and only single peaks are that high. Such values can be avoided by introducing different termination conditions for the potential field relaxation (cf. Section 5.3.4). Instead of or additionally to a maximum iteration depth used by the UAVs presented in this thesis, a maximum time span after which potential field computation stops can be introduced. Such a termination condition might be useful if real time constrains must be met, but can also lead to suboptimal trajectories.

Using three and six UAVs results in average durations of less than 200 milliseconds for potential field computations. Even 463 milliseconds using twelve UAVs is not a bad result. Potential field computations are only conducted in order to compute new paths. These new paths often lead to directional changes the UAVs must perform. Physics like inertia of physical bodies—especially if they move with high speed—often cause much longer time spans for direction changes than 178 milliseconds, which are, e. g., in average needed by six UAVs for recalculations. Thus, using real world UAVs a higher time to react to environmental changes than in simulation is indispensable and has to be considered during the design of such a system.

The average number of iterations needed to compute the potential fields are between 6.66 and 8.17. They are less than the average iterations needed in an a priori known environment.

This is mainly based on the fact that the environment is obstacle-free at the beginning of the exploration and thus the first approximation often leads directly to an equilibrium-free potential field.

Finally, Table 8.4 shows the time elapsed between two potential field computations. The average durations lie between 25.45 seconds and 31.16 seconds. From this it follows that it is in average possible to compute several potential fields before recalculation must be performed. The time differences are higher than in an a priori known environment. This is caused by the path recalculations necessary to avoid newly detected obstacles. Such obstacle avoidance often cause the UAVs to rotate if new paths result in directional changes. Rotation takes longer for the simulated UAVs than straight forward flights and therefore causes longer durations needed to follow such paths. But rotating the UAVs is necessary. Backwards or sidewards flight in unknown environments with obstacle detection capabilities only built into the front of the UAVs will result in collisions.

This section presents in detail the abilities of the UAV system described in this thesis without formation abilities in an unknown environment. In addition to these capabilities, the UAVs are able to entirely explore environments in formation as shown in the following section.

8.4. Exploration in Formation

An evaluation of the system using the superpositioned potential field described in Chapter 6 is presented in this section. In contrast to the previous section, the potential field allows the creation of formations for the presented test runs. The conditions for the test runs are equal to the ones in the test series described before and the focus also lies on the same results. The tests have been run in order to show that the objectives reached by non-formation flights are also achieved through the use of formation flights. This is necessary as if the UAVs were, e. g., unable to entirely explore the environment the formation approach would only be usable for a limited number of applications.

Sections 5.5 and 6.5 showed that the single potential fields computed by the UAVs are uniformly globally asymptotically stable. Still, this verification says nothing about the superpositioned potential field and the dynamics of the system achieved by consecutively computed potential fields influenced by flying UAVs. To show that the computed superpositioned potential field is also stable such that entire explorations are possible, the test runs from the previous sections have been repeated while the UAVs always fly in formation.

The presented calculation durations for the computation of the potential fields are achieved by the consecutive run of the algorithms presented in Appendix A.5 and Appendix A.6. Calculating the superpositioned potential field includes two additional tasks beside the five tasks needed to compute the harmonic potential field as used in the test runs, described in the previous section:

1. Computation of the bifurcating potential field (cf. Section 6.3.1).
2. Superpositioning of the two potential fields (cf. Section 6.3.2).

During formation flights one selected UAV of the formation computes a formation path all UAVs have to follow. Therefore, the UAVs continuously recalculate the potential field in order to follow the formation path without colliding with each other (cf. Section 6.4.2). One intention for presenting the computation times is to show the efficiency of the continuous potential field updates, needed for collision avoidance. Thus, the UAVs selected to graphically show the results are UAVs not responsible for the computation of formation paths (cf. Section 6.4.2). For continuous path recalculations during formation path following settings that differ from the ones used to compute the formation paths towards unexplored areas have been used. The maximum iteration depth of the harmonic potential field relaxation has been set to number of used nodes multiplied by 0.01 and the distance after which leaves are combined has been set to 50 meters (cf. Section 5.3.4) for formation path following. This has been conducted to reach a further reduction of the computation times as these paths consist only of one route point (cf. Section 6.4.2) such that they do not cover long distances, which makes it unnecessary to consider a long distance for potential field computation. Additionally, short distances lead to frequent path computation. Furthermore, the same conditions as in the previous test series for potential field computations are used to compute formation paths.

The formations used to achieve the following results are static, single line formation shapes. Always three UAVs form one formation and entirely explore the environment by keeping the formation. As mentioned before, it does not seem to be very beneficial using only one single formation for exploratory navigation. Nevertheless, a first step to design a behavior, which uses formation flight for exploratory navigation such that an increase in efficiency is obtained, is to investigate the exploration behavior in formation. Thus, this section evaluates how the system behaves if exploratory navigation is conducted using formation flights only.

The next section shows the achieved results using formations to explore the a priori known environment presented in Section 8.2. Single test runs using three, six, nine and twelve UAVs have been performed.

8.4.1. Known Environment

The test runs presented in Section 8.3.1 have been repeated with the difference that the UAVs now have to fly in fixed formations. Based on the number of used UAVs between one and four formations have been created for the single test runs in order to entirely explore the environment. Figure 8.9 shows the percentage of explored space over time. The x - axis represents the time line while the y - axis shows the percentage of explored free space at any point in time.

It can bee seen that the UAVs were always able to entirely explore the a priori known environment in formation. This shows that the superpositioned potential field leads to the expected behavior. The duration needed by three UAVs forming one single formation to explore the entire environment was 492.92 minutes. This duration can be decreased to 408.84 minutes using six UAVs, forming two formations. A further reduction of the necessary time down to 203.25 minutes has been achieved by using nine UAVs, forming three formations, while twelve UAVs needed only 196.46 minutes, flying in four formations.

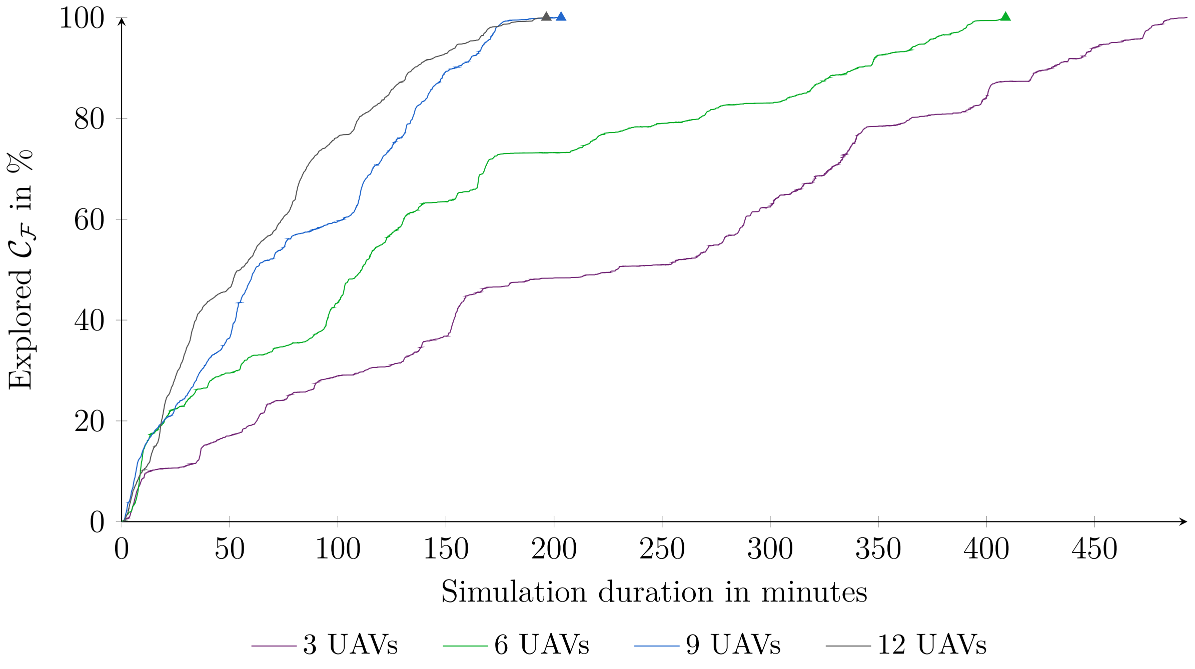


Figure 8.9.: *Percentage of explored free space at each point in time for an environment known a priori with formation abilities by 3, 6, 9, and 12 UAVs performing exploratory navigation.*

From these results it can be seen that raising the number of UAVs from six to nine leads to a significant increase of efficiency while a further enhancement from nine to twelve UAVs barely results in a noticeable benefit in this test scenario. Comparing the results with the durations needed by the UAVs without formation flights shows that always using formations decreases the efficiency of the approach significantly. From this it follows, that the use of formation flights for exploratory navigation is not automatically beneficial and it has to be considered whether the creation of a formation really makes sense. Taking the costs per flight hour into account as described in the previous sections, leads to the same result described before. Reducing the number of used UAVs decreases the costs to entirely explore the environment.

Figure 8.10 shows the number of nodes taken into account for the single potential field computations. Therefore, the results of one UAV per test run has been used to illustrate them graphically. Formation flights are conducted as follows. First, one of the UAVs computes a formation path the complete formation has to follow. Then each UAV computes its own path to the next route point of the formation path while these paths only lead to a neighboring leaf. This ensures collision avoidance (cf. Section 6.4.2). The UAVs selected to represent the results graphically are UAVs not responsible for computing formation paths. The x - axis of the diagram presented in Figure 8.10 shows the number of potential field calculations $\#_{computations}$ while the y - axis shows the single unbound leaves used for the single computations.

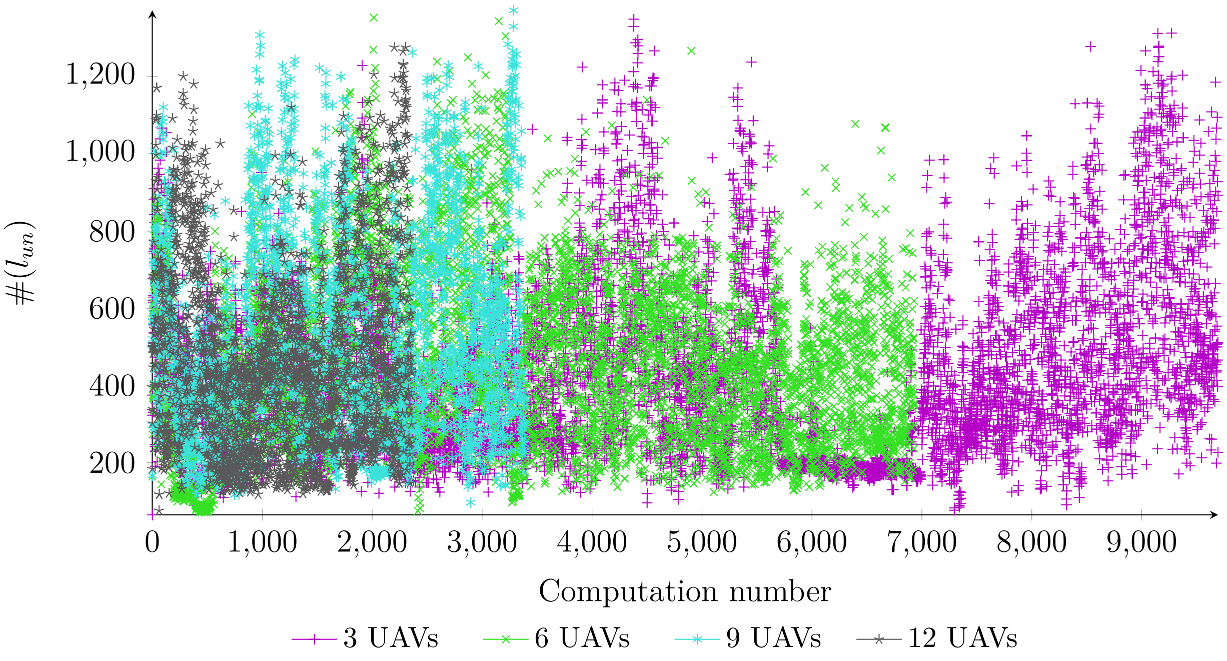


Figure 8.10.: *Nodes considered for the single potential field computations using an environment known a priori with formation abilities by the use of 3, 6, 9, and 12 UAVs for exploratory navigation.*

It can be seen that much more potential field calculations are necessary to safely fly in formation, than in the test runs using non-formation flights. In case of three UAVs more than 9.000 calculations have been performed, while three UAVs not flying in formation only need approximately 1.000 potential field calculations to entirely explore the environment as presented in Figure 8.4. Furthermore, reducing the distance, in which leaves at maximum level are considered from 150 meters to 50 meters reduces the number of used nodes from approximately 8.000 to 1.200. This should lead to lower durations needed to compute the single potential fields and enables more calculations in the same time span.

Table 8.5 shows several statistical data considering the octree created for this test series. The number of nodes $\#(l_i)$ taken into account for potential field computations lies within the range from 92 to 12,293 for all UAVs, while the number of explored nodes $\#(l_{un})$ lies between 30 and 10,804 nodes. Similar to the results obtained by non-formation flights, the number of used UAVs has no noticeable influence on the number of nodes.

In contrast to the results obtained by non-formation flights, the average number of used nodes has been decreased significantly. This was assumed as the distance, at which nodes of maximum level are taken into account for potential field computations, has been reduced from 150 meters to 50 meters. The highest average number of nodes has been taken into account by nine UAVs, which averagely used 768.25 nodes for potential field calculations.

The highest average number of unbound nodes has also been achieved by nine UAVs, which needed 632.30 nodes in average.

Again, the average numbers show that due to the use of an octree, a dramatic reduction of nodes, necessary for potential field computations, can be achieved compared to grid-like structures, which would need approximately 16.777 million nodes to equally represent the environment. It also indicates that formation flights are computationally efficient.

A noticeable result shown in Table 8.5 regards the number of potential field computations $\#_{computations}$. This number increases significantly from 30,702 by the use of three UAVs to 48,404 using six UAVs. A further raise of the number of UAVs to nine and twelve leads to a reduction of $\#_{computations}$ needed for the entire exploration of the environment. This result shows that no statement about the number of potential field computations can be made based on the number of UAVs.

	3 UAVs	6 UAVs	9 UAVs	12 UAVs
$\#_{min}(l_{un})$	42.00	36.00	30.00	37.00
$\#_{max}(l_{un})$	10,101.00	10,239.00	10,804.00	10,039.00
$\#_{\varnothing}(l_{un})$	507.28	525.69	632.30	616.79
$\#_{min}(l_i)$	113.00	113.00	113.00	92.00
$\#_{max}(l_i)$	11,796.00	11,649.00	12,293.00	10,956.00
$\#_{\varnothing}(l_i)$	647.85	692.00	768.25	749.04
$\#_{computations}$	30,702.00	48,404.00	34,586.00	33,666.00

Table 8.5.: *Minimum $\#_{min}(l_{un})$, maximum $\#_{max}(l_{un})$, and average $\#_{\varnothing}(l_{un})$ number of unbound nodes, as well as minimum $\#_{min}(l_i)$, maximum $\#_{max}(l_i)$, and average $\#_{\varnothing}(l_i)$ number of nodes used for potential field computation. The number of potential field computations $\#_{computations}$ is presented, too.*

Figure 8.11 shows the durations needed to calculate the single potential fields. The UAVs used to represent the data are the same as the ones used earlier to present the results in Figure 8.10. The x - axis shows the computation number while the y - axis shows the duration needed to compute the single potential fields, as well as the average computation duration. In contrast to the representation used in the previous sections, two diagrams are used to illustrate the results. The first one shows the single durations, while the second one presents the average duration. Using two diagrams is due to visibility reasons.

The average duration increases significantly at the beginning while they become relatively steady afterwards compared to the test runs without formation capabilities. This behavior is independent of the number of UAVs. This indicates that the single formations were close to each other during the entire exploration phase, causing a consistent computational effort as the results are not repeated in the following test series. Additionally, it shows that the disturbances caused by the other formations have a noticeable effect on the computation durations.

The durations still increase if the number of UAVs is raised due to the higher complexity resulting from more UAVs. It can be seen that no noticeable spikes occur using the selected settings for potential field computation. The single durations needed by the selected UAVs are always within a small range. Additionally, only a few computations using twelve UAVs took more than 512 milliseconds. This shows that formation flights are also computationally efficient as claimed in Section 1.2.

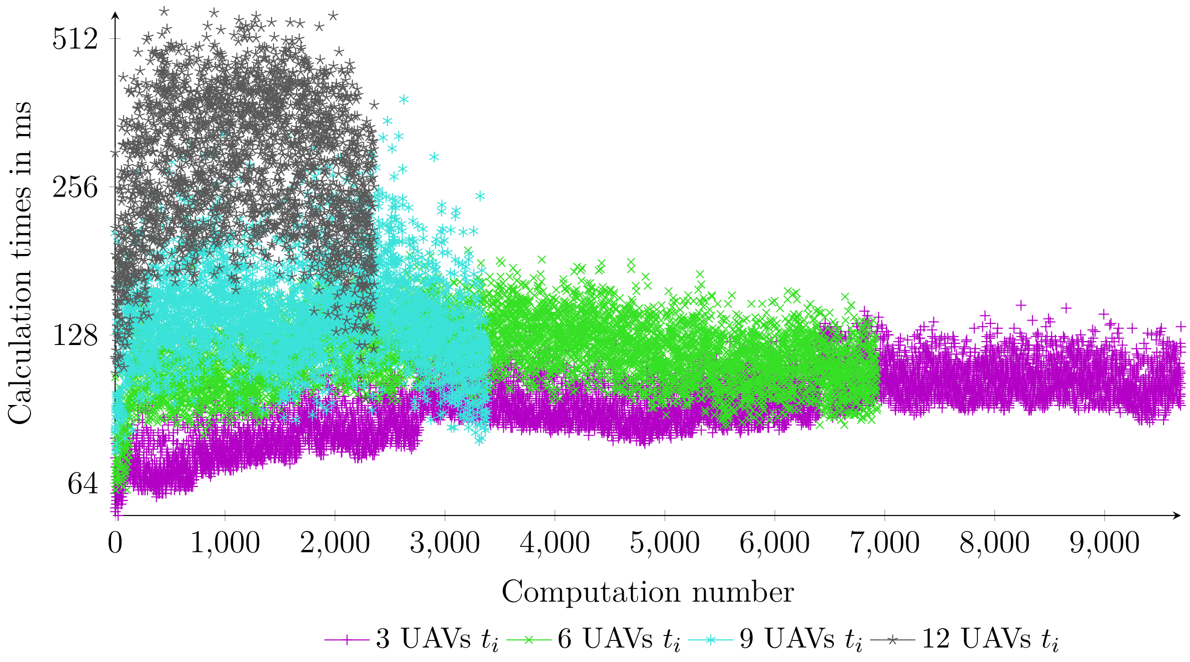
The average computation durations $t_\varnothing$ of the selected UAVs from the test runs with three and six UAVs are 92.72 milliseconds and 117.98 milliseconds. These times do not differ as much as the durations achieved using nine (143.37 milliseconds) and twelve UAVs (293.06 milliseconds). Nevertheless, even an average computation time of less than 300 milliseconds is considered to be really efficient.

Table 8.6 shows some statistical data regarding the computation durations needed to compute the potential fields by taking all UAVs of each single test run into account. In contrast to Figure 8.11, these data include formation path computations. Thus, the maximum calculation durations for potential field computations are peaks, which is similar to the results obtained in the previous sections. This effect is obvious using twelve UAVs. The maximum duration necessary to compute a potential field was 1,902 milliseconds while only 293.89 milliseconds were needed in average.

		3 UAVs	6 UAVs	9 UAVs	12 UAVs
t_{min}	*ms.*	50.00	59.00	65.00	79.00
t_{max}	*ms.*	482.00	697.00	971.00	1,902.00
$t_\varnothing$	*ms.*	90.47	115.91	145.86	293.89
$t_{complete}$	*sec.*	2,777.82	5,749.70	5,044.57	9,894.20
$\varnothing_{depth}$		1.03	1.05	1.04	1.04
$\varnothing t_{diff}$	*sec.*	2.89	3.04	3.17	4.20
$\#_{computations}$		30,702.00	48,404.00	34,586.00	33,666.00

Table 8.6.: *Minimum t_{min}, maximum t_{max}, and average $t_\varnothing$ durations needed for potential field computations. Additionally, the complete duration $t_{complete}$ needed for the computations, the average iteration depth $\varnothing_{depht}$, the average duration difference between two potential field computations $\varnothing t_{diff}$ and the complete number of necessary computations $\#_{computations}$ by taking all UAVs of the single test runs into account.*

The previously made statement, that raising the number of UAVs increases the computation times, holds true for all UAVs during formation flights. Doubling the number of UAVs from three to six increases the computation durations by approximately 28% from 90.47 milliseconds to 115.91 milliseconds. Tripling the number of UAVs from three to nine raises the durations by approximately 61% to 145.86 milliseconds, while another increase to twelve UAVs leads to 293.89 milliseconds, which is approximately 225% higher than the result obtained by three UAVs. Nevertheless, the average duration of 293.89 milliseconds using twelve UAVs enabling them to entirely explore such a complex environment, denotes a highly efficient path planning approach.



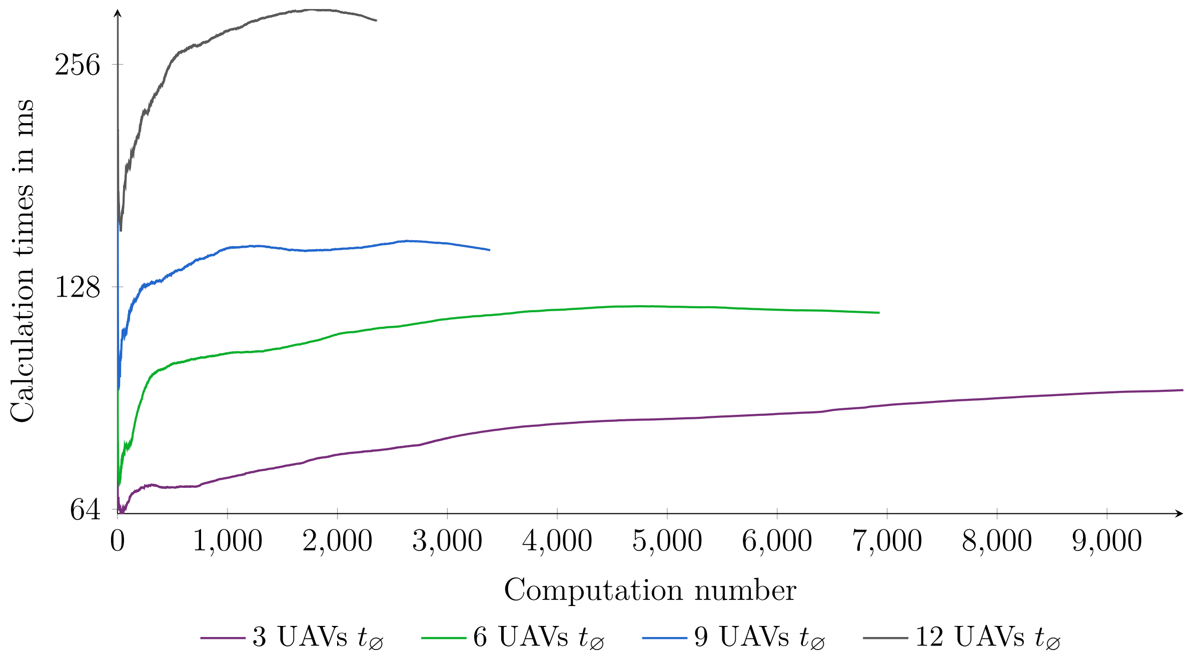


Figure 8.11.: *Single computation durations t_i, as well as the average computation durations $t_\varnothing$ needed to compute the potential fields in an environment known a priori with formation abilities by the use of 3, 6, 9, and 12 UAVs for exploratory navigation.*

The time needed to compute all potential fields $t_{complete}$ is also presented in Table 8.6. It can be seen that it has been raised about more than fifty percent by doubling the number of UAVs from three to six. A further increase to nine UAVs reduces the computational effort, while twelve UAVs lead to a further increase of the computational effort. Thus, no valid statement regarding an effect of the number of UAVs on the overall computational effort can be made for formation flights in a priori known environments. What can be stated is that formation flights cause a much higher computational effort than non-formation flights when the results shown in this table are compared to Tables 8.2 and 8.4.

The average iteration depth $\varnothing_{depth}$ is similar for the single test runs. It is mainly influenced by the termination condition, which terminates the iterations of the relaxation if the depth is greater or equal to the number of nodes multiplied by 0.01 during formation path following. As only a low number of nodes has been used to compute potential fields for formation path following, the average iteration depth is close to one.

Finally, Table 8.6 shows the average time elapsed between two potential field computations $\varnothing t_{diff}$ in seconds. This duration lies between 2.89 seconds by the use of three UAVs and 4.20 seconds using twelve UAVs. It is mainly influenced by path following computations. As only paths to neighboring nodes are computed if the UAVs follow a formation path fast recalculations often have to be performed. A further extension of the presented system could be to just update the potential field in this case, instead of conducting complete recalculations as it is performed by the system presented in this thesis. Nevertheless, the average time spans between two potential field computations are below the maximum durations needed to compute a potential field, enabling the UAVs to fly smoothly without needing to wait for potential field computations to be finished after they have reached their target configurations.

This section describes in detail the formation abilities of the multi-UAV system presented in this thesis for exploratory navigation in an a priori known environment. In addition it has to be investigated if the UAVs are also able to entirely explore unknown environment in formation in order to evaluate whether the approach is also stable in highly dynamic environments, which is necessary in order to use formation flights in a wide range of applications. The results obtained by these test runs are presented in the following section.

8.4.2. Unknown Environment

The test runs described in the previous section have been repeated, whereby this time the environment has been set to be unknown to the UAVs a priori. Figure 8.12 shows the percentage of explored free space over time. The x - axis represents the time line while the y - axis shows the percentage of explored free space. It can be seen that the UAVs were still able to entirely explore the environment. One effect, well-marked for this test series, is that the UAVs do not need such long time spans to explore the last percentage of the environment, as in the test runs without formations. This indicates that it should be possible to reach a benefit by combining formation flights and non-formation flights.

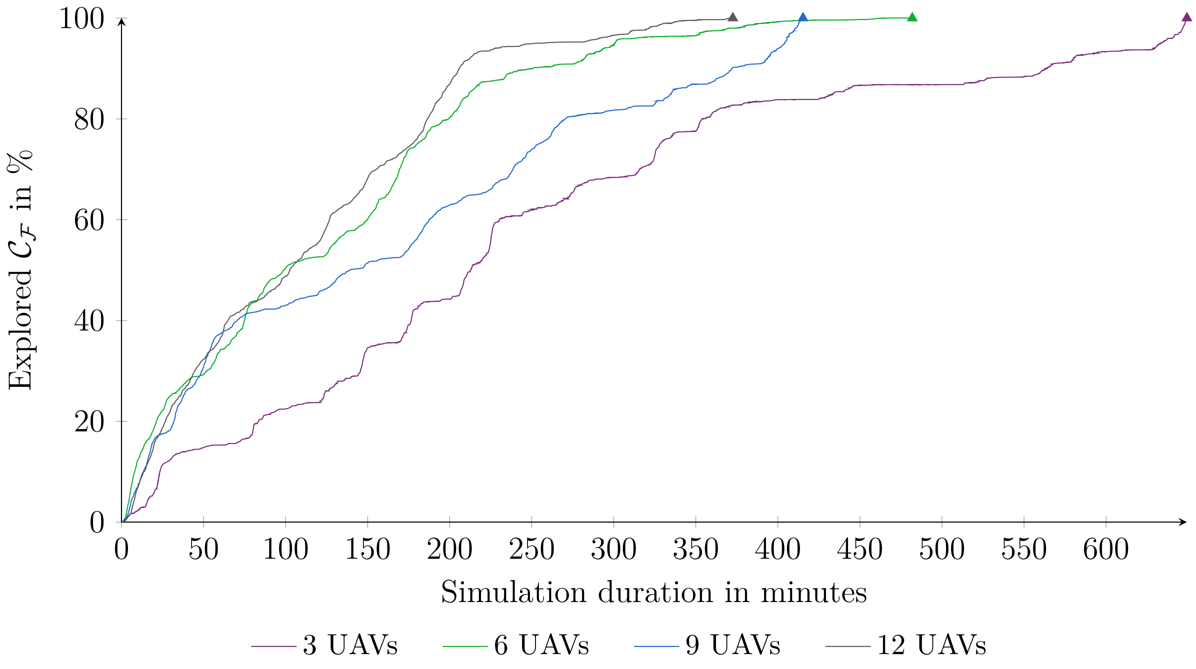


Figure 8.12.: *Percentage of explored free space at each point in time for an environment unknown a priori with formation abilities by 3, 6, 9, and 12 UAVs performing exploratory navigation.*

Similar to the results obtained earlier, a raise of the number of UAVs reduces the time needed to entirely explore the environment. Three UAVs needed 649.25 minutes while six UAVs were able to explore the environment in 481.9 minutes. A further increase to nine UAVs leads to a total time of 415.32 minutes, while twelve UAVs were able to perform exploratory navigation of the entire environment in 372.55 minutes. Comparing these results with the corresponding results achieved by non-formation flights, indicates that in unknown environments, it is not beneficial either to always fly in formation.

The results also show that the necessary durations do not correlate linearly with the number of UAVs as in the earlier test runs. The first increase to six UAVs reduces the time needed by about 167.35 minutes. A further increase from six to nine UAVs results in a benefit of 66.58 minutes, while twelve UAVs were able to save additional 42.77 minutes. Thus, using formation flights, the benefit reached by a higher number of UAVs decreases as more UAVs are used.

Figure 8.13 shows the number of unbound nodes taken into account for the single potential field computations. The results are achieved by one selected UAV per test run, not responsible for the computation of the formation path. The number of potential field computations $\#_{computations}$ is pictured on the x - axis while the y - axis shows the number of unbound nodes used to compute the single potential fields.

It can be seen that the number of nodes does not differ much from the results obtained for the a priori known environment. Always less than 1,400 nodes have been taken into account, which still indicates that the potential field computation should be performed really efficient. The number of nodes does not change noticeably with a changing number of UAVs used for the single test runs, as well as the resulting number of formations. Thus, in this test series the octree leads to a high reduction of nodes necessary to compute the single potential fields, too.

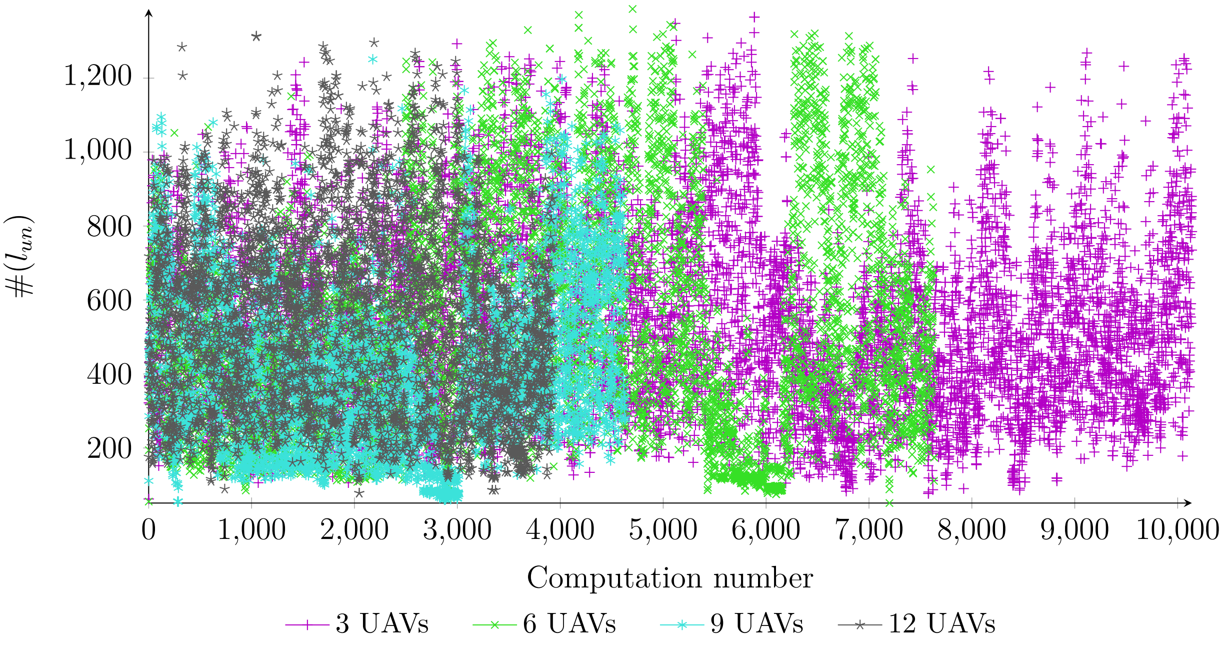


Figure 8.13.: *Nodes considered for the single potential field computations using an environment unknown a priori with formation abilities by the use of 3, 6, 9, and 12 UAVs for exploratory navigation.*

Table 8.7 shows several statistical data regarding the octree nodes, necessary for potential field computations. The entire number of nodes $\#(l_i)$ taken into account for potential field computations is between 106 and 13,371 for all UAVs. The number of unbound nodes $\#(l_{un})$ lies thereby in the range from 36 to 10,593. Like in the previous test runs these values are not noticeably influenced by the number of used UAVs.

The highest average number of nodes has been taken into account by nine UAVs, which needed 834.36 nodes in average. The maximum average number of unbound nodes $\#_{\varnothing}(l_{un})$ has also been used by nine UAVs. They needed 616.8 unbound nodes on average to compute the single potential fields. Thus, in this test case no valid statement regarding a correlation between the number of UAVs and the number of used nodes can be given.

Regarding the number of potential field computations necessary to entirely explore the environment, shows the same effect as in the test runs using an a priori known environment. The number increases significantly while raising the number of UAVs from three to six. Further increases of the number of UAVs lead to a reduction of the necessary computations. This substantiates the statement made in the previous section, that the number of UAVs has no direct influence on the needed number of potential field calculations.

	3 UAVs	6 UAVs	9 UAVs	12 UAVs
$\#_{min}(l_{un})$	36.00	39.00	48.00	42.00
$\#_{max}(l_{un})$	9,706.00	10,593.00	10,544.00	10,102.00
$\#_{\varnothing}(l_{un})$	595.10	575.77	616.80	576.13
$\#_{min}(l_i)$	106.00	120.00	106.00	127.00
$\#_{max}(l_i)$	11,649.00	13,371.00	11,985.00	12,237.00
$\#_{\varnothing}(l_i)$	784.72	801.80	834.36	797.42
$\#_{computations}$	33,633.00	51,339.00	49,179.00	42,547.00

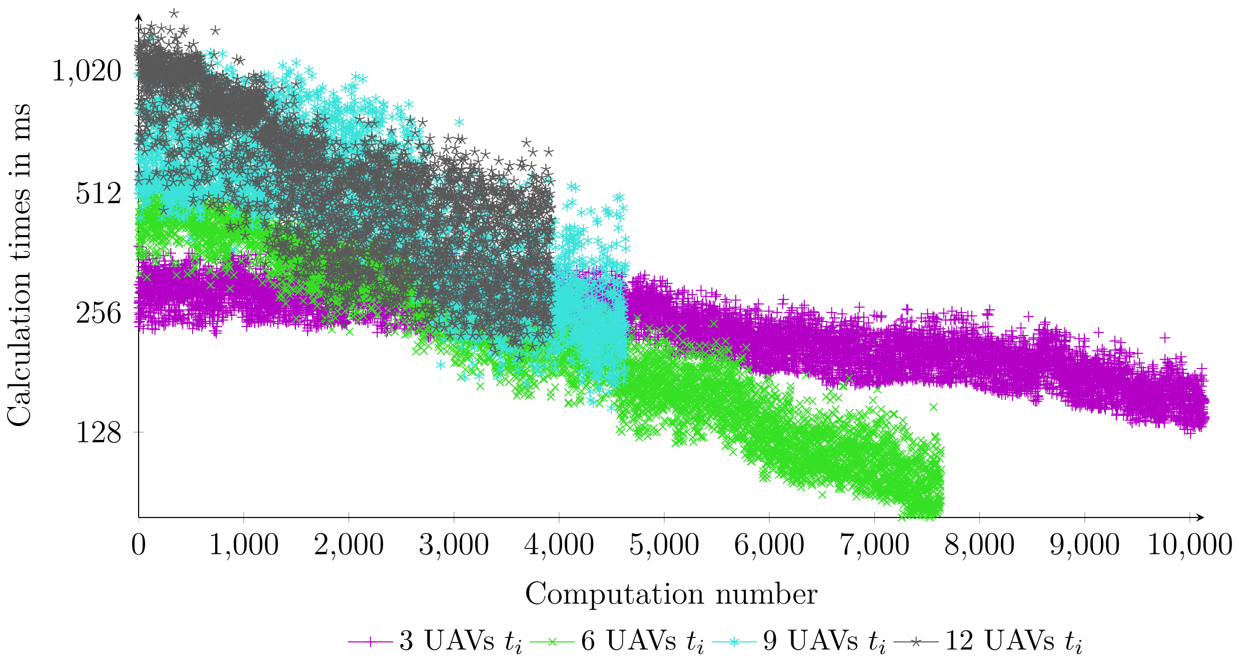
Table 8.7.: *Minimum $\#_{min}(l_{un})$, maximum $\#_{max}(l_{un})$ and average $\#_{\varnothing}(l_{un})$ unbound nodes, as well as minimum $\#_{min}(l_i)$, maximum $\#_{max}(l_i)$ and average $\#_{\varnothing}(l_i)$ nodes used for potential field computation. The number of potential field computations $\#_{computations}$ is presented, too.*

Figure 8.14 shows the durations needed to calculate the single potential fields, as well as the average durations in milliseconds. The same UAVs selected to represent the results in Figure 8.13 have been used to illustrate these results. The x - axis shows the computation number while the y - axis shows the duration needed to compute the single potential fields and the average computation durations reached by the selected UAVs in milliseconds. Two diagrams are used to illustrate the results. The upper diagram shows the durations needed to calculate the single potential fields and the lower diagram illustrates the average computation times.

It can be seen that the durations are much higher than for the test runs using an a priori known environment. But they decrease over time in contrast to the results obtained in an a priori known environment. This indicates that the single formations are not that close to each other during the entire exploration than in the previous test series because UAVs close to each other lead to a higher complexity as then the nodes they occupy are considered at maximum level for potential field computations. One interesting aspect is, that the durations' decrease is higher over time the more UAVs are used. This can be seen by the average durations using six UAVs. After approximately 7,000 computations, six UAVs averagely needed less time to compute the potential fields than three UAVs. This effect is true for all explorations thus at the end the UAV representing three UAVs averagely needed less time for potential field computation than the selected UAV of the test run with six UAVs.

The increase of the durations, resulting during potential field computation, compared to exploration of an a priori known environment is much more pronounced in case of formation flights. This holds true for all test runs. Such an increase can lead to problems if it results in computation durations higher than the time spans between two potential field computations. As aforementioned, several possible solutions exist to overcome this issue.

Table 8.8 shows some statistical data regarding the computation durations needed to compute the potential fields by taking all UAVs for each of the single test runs into account. Similar to the results obtained before, the maximum computation durations for potential field calculations are single peaks. The worst value has been achieved by the use of nine UAVs



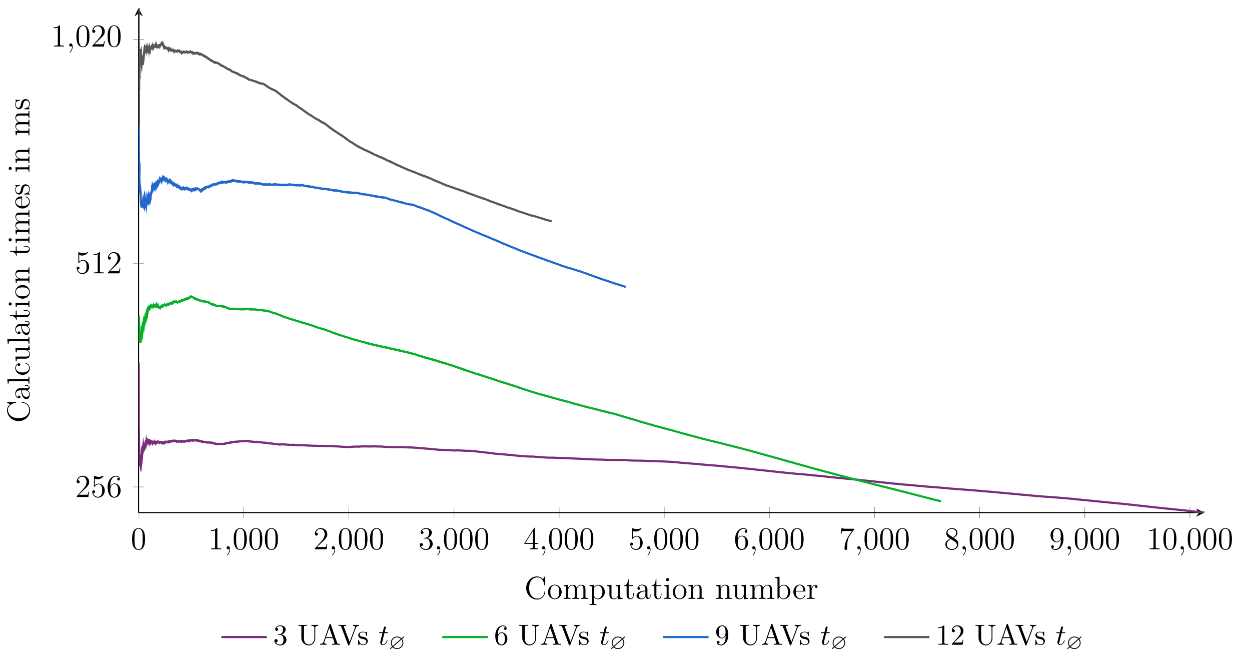


Figure 8.14.: *Single computation durations t_i, as well as the average computation durations $t_\varnothing$ needed to compute the potential fields in an environment unknown a priori with formation abilities by the use of 3, 6, 9, and 12 UAVs for exploratory navigation.*

whereby one UAV once needed 4,133 milliseconds for the computation of a single potential field. In contrast to this value the UAVs needed 468.46 milliseconds on average, which shows the big gap between the single durations needed to compute the potential fields.

This test results again show that the average time $t_{\varnothing}$ needed to compute the potential fields increases with an increasing number of UAVs. The complete time $t_{complete}$ to compute all potential fields, shows that the test runs described in this section needed by far the highest computational effort.

		3 UAVs	6 UAVs	9 UAVs	12 UAVs
t_{min}	*ms.*	127.00	76.00	124.00	152.00
t_{max}	*ms.*	1,244.00	1,956.00	4,133.00	3,993.00
$t_{\varnothing}$	*ms.*	239.14	248.70	468.46	624.82
$t_{complete}$	*sec.*	8,042.83	12,768.16	23,038.43	26,584.12
$\varnothing_{depth}$		1.06	1.06	1.08	1.09
$\varnothing t_{diff}$	*sec.*	3.47	3.38	4.56	6.30
$\#_{computations}$		33,633.00	51,339.00	49,179.00	42,547.00

Table 8.8.: *Minimum t_{min}, maximum t_{max}, and average $t_{\varnothing}$ durations needed for potential field computations. Additionally, the complete duration $t_{complete}$ needed for the computations, the average iteration depth $\varnothing_{depht}$, the average duration difference between two potential field computations $\varnothing t_{diff}$ and the complete number of necessary computations $\#_{computations}$ by taking all UAVs of the single test runs into account.*

Comparing the average durations for the computations of the potential fields with the average time span $\varnothing t_{diff}$ between the single calculations, shows that the UAVs were mostly still able to compute new potential fields before they reach their target position. The high peaks on the other hand show that this might not be true for all path following activities. Thus, one has to take this into account for the design of a real world UAV system.

It is indicated that in case of exploratory navigation, only flying in formation is not beneficial compared to non-formation flights. But one advantage of formation flights in case of exploratory navigation, regarding the time span needed to explore the last percentage of the environment, has been noticed in this section. Thus, a combination of both approaches to increase efficiency should be possible. The results obtainable by a combination of formation flights and non-formation flights are amongst others presented in the following section.

8.5. Cooperative Behavior

This section presents several tests regarding the UAV behavior and the task allocation system described in Chapter 7. First, a test run combining non-formation flights and formation flights to entirely explore the test environment is presented. The test shows an obtainable benefit reached by combining both approaches in contrast to using non-formation flights or formation flights only. Thereafter, obstacle avoidance behavior for both, single UAVs and UAV formations, is evaluated. It illustrates some of the self-adaptation capabilities of single UAVs and UAV formations, regarding the environment and environmental changes. The final tests presented in this thesis concern the task allocation system and describe possible task distribution over a single UAV, as well as over multiple UAVs.

The results used to describe the avoidance and task allocation behavior are presented using screenshots of the *Control Station's* cutouts instead of diagrams used in the previous sections. The cutouts show the initial configuration, the final configuration, and additional configurations if necessary.

8.5.1. Behavioral Exploration

Four test series regarding exploratory navigation have been conducted and the results are presented in detail in the previous sections. The tests considered non-formation flights and formation flights in a priori known, as well as in unknown environments. They showed that the behavior resulting from the consecutive computation of the single potential fields is stable such that the UAVs are always able to entirely explore the environment. It is also shown that potential field computation is efficient using the octree described in Section 4.2.

Another statement made in the previous chapters is that a combination of formation flights and non-formation flights can increase efficiency even in case of exploratory navigation if higher efficiency is defined as lower durations needed, to entirely explore the environment. The previous tests show that this is not the case when the UAVs always fly in formation. Nevertheless, combining both approaches can increase efficiency. Therefore, a test run combining non-formation flights with formation flights is presented in this section. The formation shape the UAVs create is a line formation equal to the formation shape used to obtain the results in the previous section. The combined approach is compared to the results achieved with the same settings in the previous sections.

One of the main objectives for the resulting multi-UAV system mentioned in Section 1.2 is to create and dissolve formations dynamically. The test run presented in this section shows that such a behavior has been reached.

The environment was known a priori in this test scenario and six UAVs must explore the entire terrain. Figure 8.15 shows the exploration rate over time. The x - axis shows the time elapsed since the simulation has been started in minutes while the y - axis shows the percentage of explored free space. The result from Section 8.3.1 where six UAVs explored the environment using non-formation flights is presented by the red line. Additionally, the result from Section 8.4.1 where six UAVs explored the terrain in formation is illustrated by the green line. It can be seen that the UAVs need a noticeable higher duration to entirely explore the environment if they always fly in formation. These results are compared to the use of a combination of the approaches, represented by blue line. It can also be seen that the time needed by six UAVs without formation capabilities, which was 195.93 minutes to entirely explore the environment could be reduced to 172.37 minutes using a combination of formation flights and non-formation flights. So, for this test scenario, a benefit of more than ten percent has been reached, which shows that even the non-optimal strategy presented in Section 7.4.1 can result in a noticeable increase of efficiency.

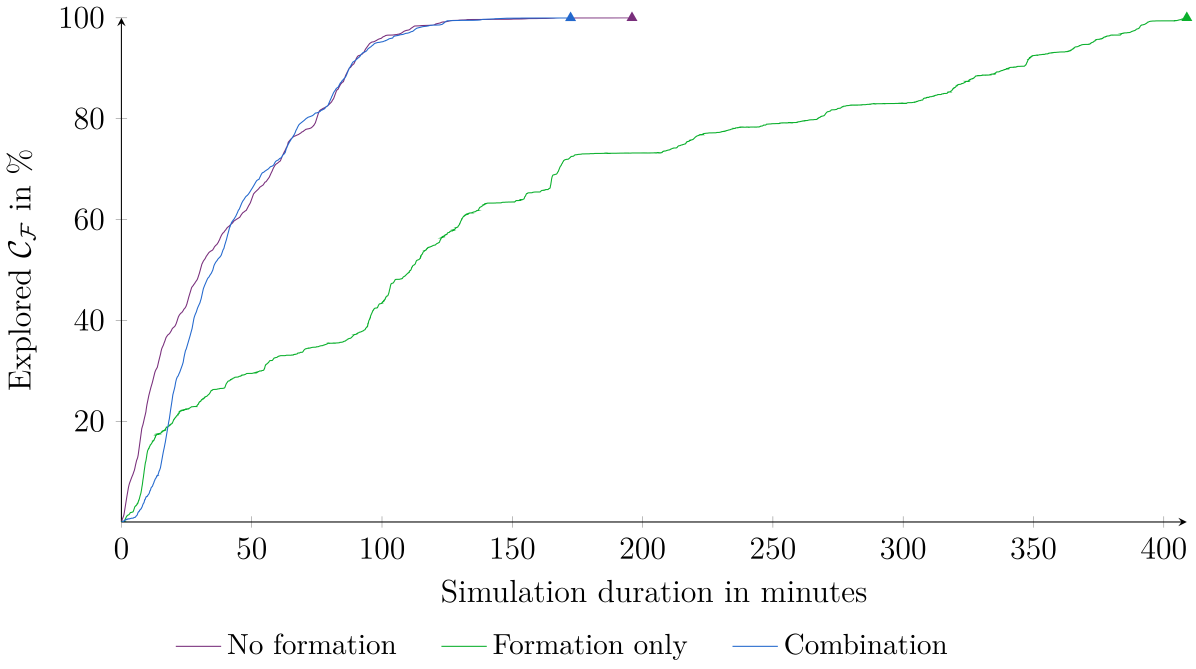


Figure 8.15.: *Percentage of explored free space at given points in time comparing non-formation flights, formation flights and a combination by six UAVs in an a priori known environment.*

Using both flight types as described in Section 7.4.1 the UAVs first create one big formation, consisting of six UAVs until approximately ten percent of the environment has been explored. This can also be seen in Figure 8.15. The exploration rate using the combined approach is worse at the beginning compared to the other ones. This is due to the time the UAVs need to create the formation. The more UAVs participate in a formation the longer it takes to establish the formation at a given point in space. Thus, the result is even worse than in the formation only approach where two formations consisting of three UAVs have been created at the beginning.

The exploration rate using the combined approach is the highest of the three presented test runs, after the exploration of the first ten percent of the free space. Additionally, the UAVs were able to explore the last remaining percent of unexplored terrain faster than the UAVs using the other approaches. This mainly results from the fact that they miss less small areas at the beginning of exploratory navigation than the UAVs using one of the other approaches. Thus, less small areas, which are distributed over the environment must be reached at the end.

The UAVs did not form more formations in this test run. Several times, single UAVs created multiple exploration tasks (cf. Section 7.4.1), but the UAVs were widely scattered over the terrain and thus the multiple exploration tasks do not seem be very beneficial to the other UAVs.

The results presented in Figure 8.15 show that a combination of formation flights and non-formation flights can lead to an increase in efficiency for exploratory navigation. Nevertheless, only one single run is presented and the results might not hold for all possible environments and numbers of used UAVs. Further interesting research would be to investigate, in which cases formation flights always reduce the duration needed to explore both, entire environments and parts of environments, in order to design a highly beneficial approach combining non-formation flights and formation flights.

Results concerning the number of used nodes and the computation times needed to compute the single potential fields are not presented for this test run. They are similar to the results obtained before, which are described in detail in the previous sections.

Another important issue to be addressed for UAV flights is that obstacle avoidance must be guaranteed and that the UAVs themselves, as well as formations they create, are able to adapt to the environment. Thus, the following sections first show the avoidance behavior of single UAVs and thereafter the avoidance behavior of a UAV formation.

8.5.2. Single UAV Obstacle Avoidance

One of the main objectives mentioned in Section 1.2 is to reach an intelligent UAV behavior for self-adaptation to environmental changes. Therefore, it should be demonstrated that such a behavior has been reached. The test run presented in this section shows an example path change caused by a newly detected obstacle. This is a simple test but it shows the resulting behavior without leading to hardly understandable results. The behavior shown in this section is essential for UAV flights in order to avoid damage. The UAV starts at position $p_{\mathcal{U}} = (2050; 500; 140)$ in the previously described environment and has to move to a target position $p_{\mathcal{G}} = (2048; 650; 100)$.

Figure 8.16 shows the initial path to reach the target position, computed by the UAV. The red UAV in the upper right of the figure is able to compute a smooth and short path depicted as red line towards the target, which is illustrated in the lower left part of the figure as a ring shaped object. Target areas are always represented on the terrain ground for better visibility and to avoid confusion due to various hovering textures representing target areas. Using the descent gradient, the UAV was able to plan a path consisting of 13 route points to reach the target area.

The UAVs' previously computed path has been intersected by an obstacle, shown in Figure 8.17. Such an occupied area could arise, e. g., by detecting a hangar unknown before or by a newly introduced no-fly area. The obstacle is illustrated as yellow-black-striped object and stretches from position $(2000; 525; 100)$ to $(2100; 575; 140)$. The resulting intersection forces a new path planning in order to avoid the obstacle. Thus, a new path is computed and presented

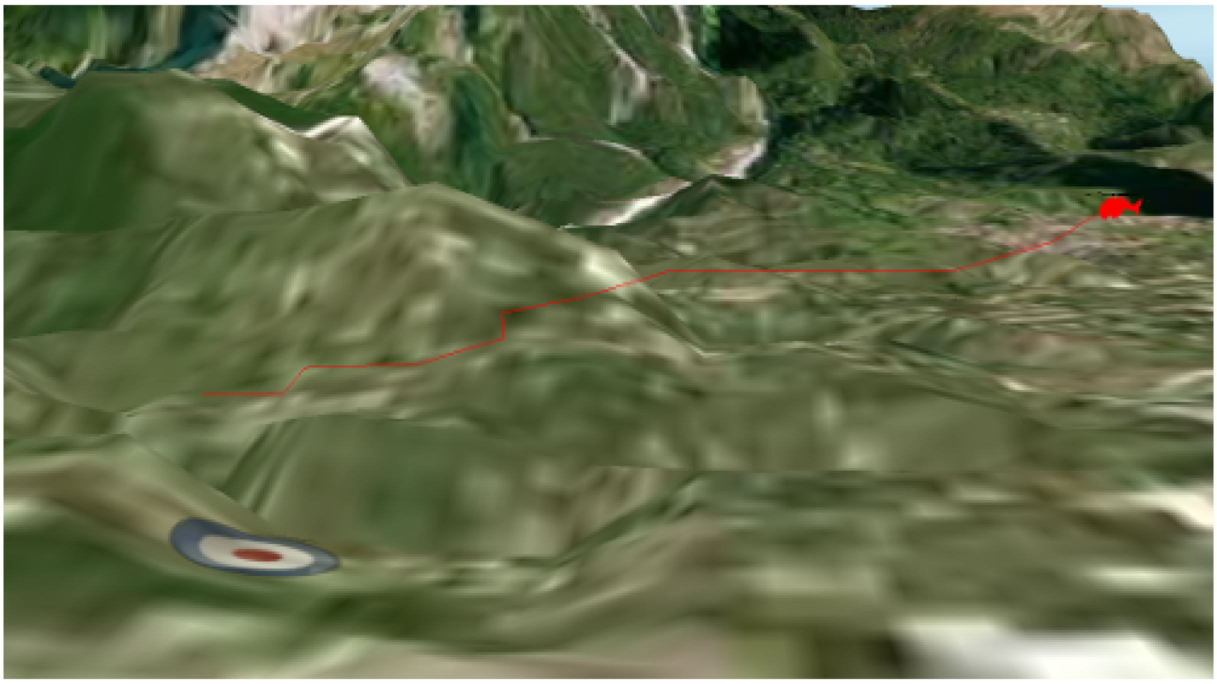


Figure 8.16.: *The trajectory planned by the red UAV from its position to the target position illustrated as ring shaped object. No obstacle has to be avoided.*

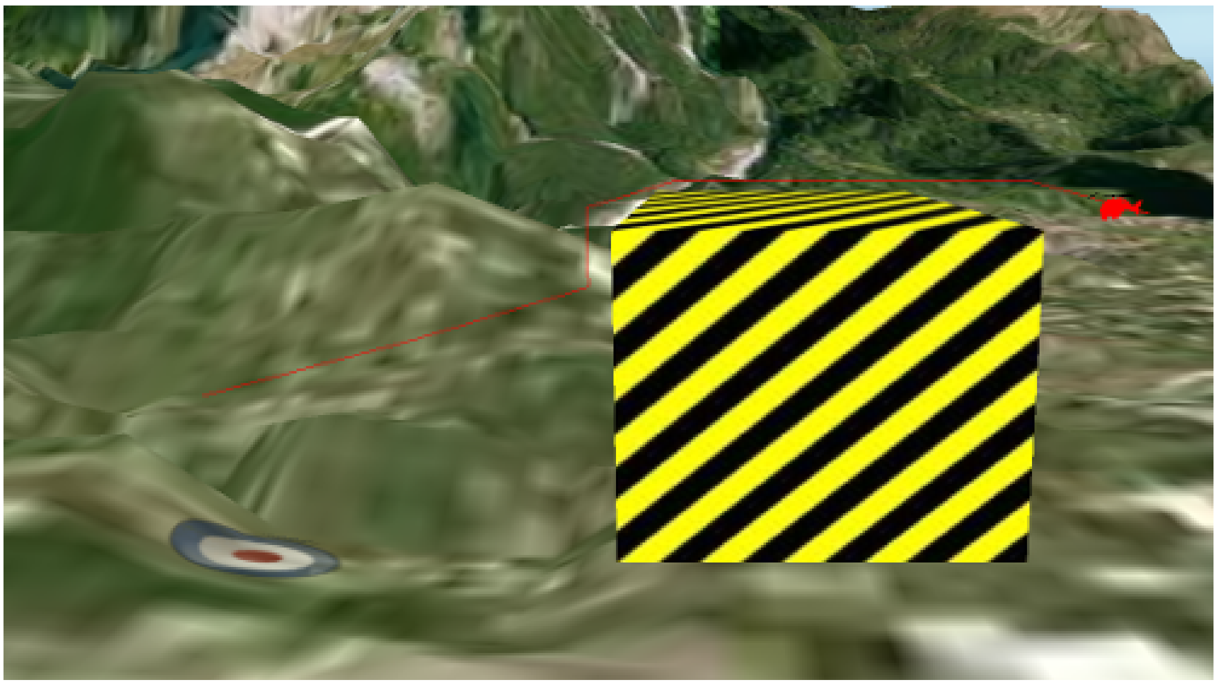


Figure 8.17.: *Newly computed trajectory after an obstacle intersected the previously computed UAV path.*

as a red line in Figure 8.17. It can be seen that the UAV is able to avoid the obstacle by simultaneously reaching the target position using an extended path of 15 route points.

The test presented in this section gives an impression of the obstacle avoidance behavior of single UAVs. A more challenging obstacle avoidance methodology is necessary to avoid obstacles in formation. This can often be reached, only by changing the formation shape as presented in the following section.

8.5.3. Obstacle Avoidance in Formation

Another considerable issue is obstacle avoidance in formation. The UAVs have to change the formation shape in order to be able to overcome an obstacle in the test run presented in this section. Formation flights can not efficiently be performed in complex and dynamically changing environments if the formations are unable to adapt themselves to the environment. The test run presented in this section shows the UAV behavior for obstacle avoidance resulting from the superpositioned potential field described in Section 6.3.2. It is shown that the UAV formation is able to overcome an obstacle by changing the shape of the formation. The triumph arch presented in Figure 8.2 has been chosen as obstacle. Additionally, the environment has been extended by a wall, to ensure that the formation not just flies around the triumph arch. The wall is illustrated as a yellow-black-striped object in Figures 8.18 to 8.20. Finally, the maximum altitudes of the UAVs have been set, such that they are unable to fly above the triumph arch. This is necessary as the UAVs tend to avoid areas surrounded by obstacles like the small passage provided by the triumph arch due to the approximations and termination conditions performed to efficiently compute the harmonic potential field.

To complete this test run, four UAVs start at differing positions as shown in Figure 8.18. The triumph arch is depicted in the center of the figure, while the yellow-black-striped object represents the mentioned wall. The UAVs have to fly to the two target positions illustrated as ring shaped objects. The first target position is the one before the triumph arch. It is used as consensus point where the UAVs have to establish the formation. The second target position lies behind the triumph arch such that the UAVs have to move through the passage, which is too small for the formation to fly through, without changing its shape.

Based on the start positions, they first compute a consensus point at which they create the formation shape. A line formation has to be created as presented in Figure 8.19. It can be seen that the UAVs were able to create the mentioned formation shape.

The first target area is deleted after the creation of the formation shape, as the corresponding task is fulfilled. This is graphically shown in Figure 8.20. The following task is to reach the remaining target area in formation. Between the UAV formation and the target area is now the triumph arch, which offers only a small passage the UAVs have to fly through. They are unable to fly through this passage by simultaneously keeping the formation shape. Thus, they adapted the formation shape as presented in Figure 8.20.

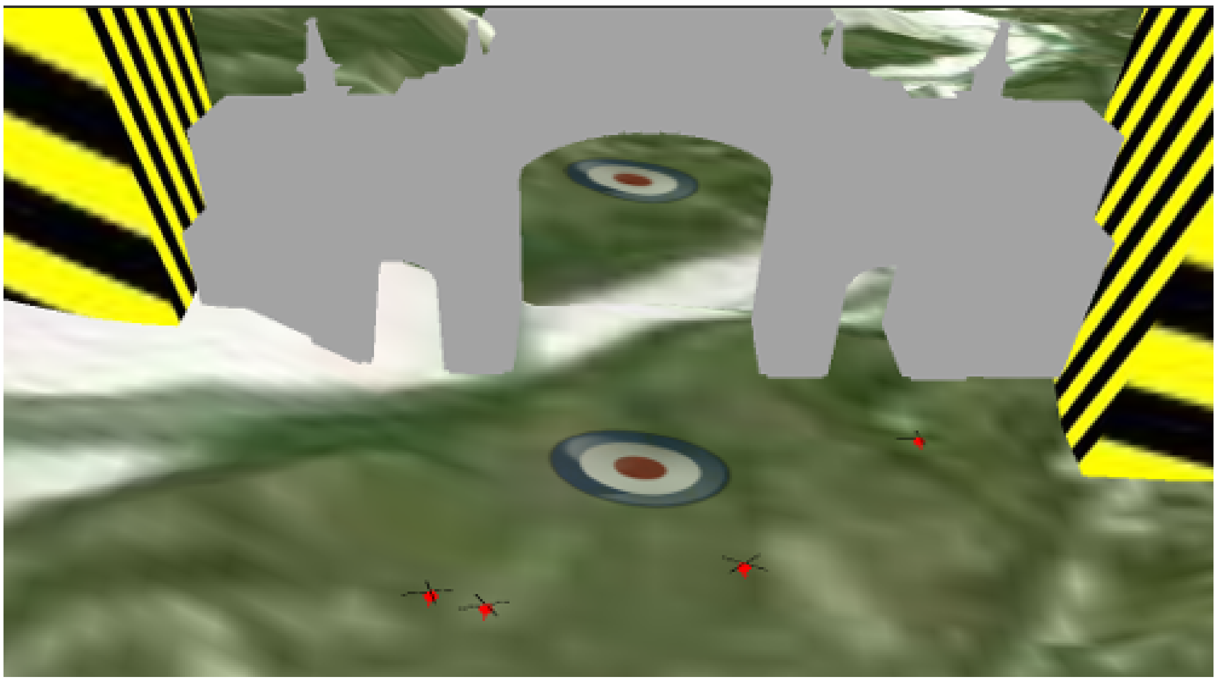


Figure 8.18.: *The initial configuration used to show the obstacle avoidance behavior of a UAV formation. The red UAVs are distributed over the environment and have to create a line formation at the target position in front of the triumph arch represented by a ring shaped object.*

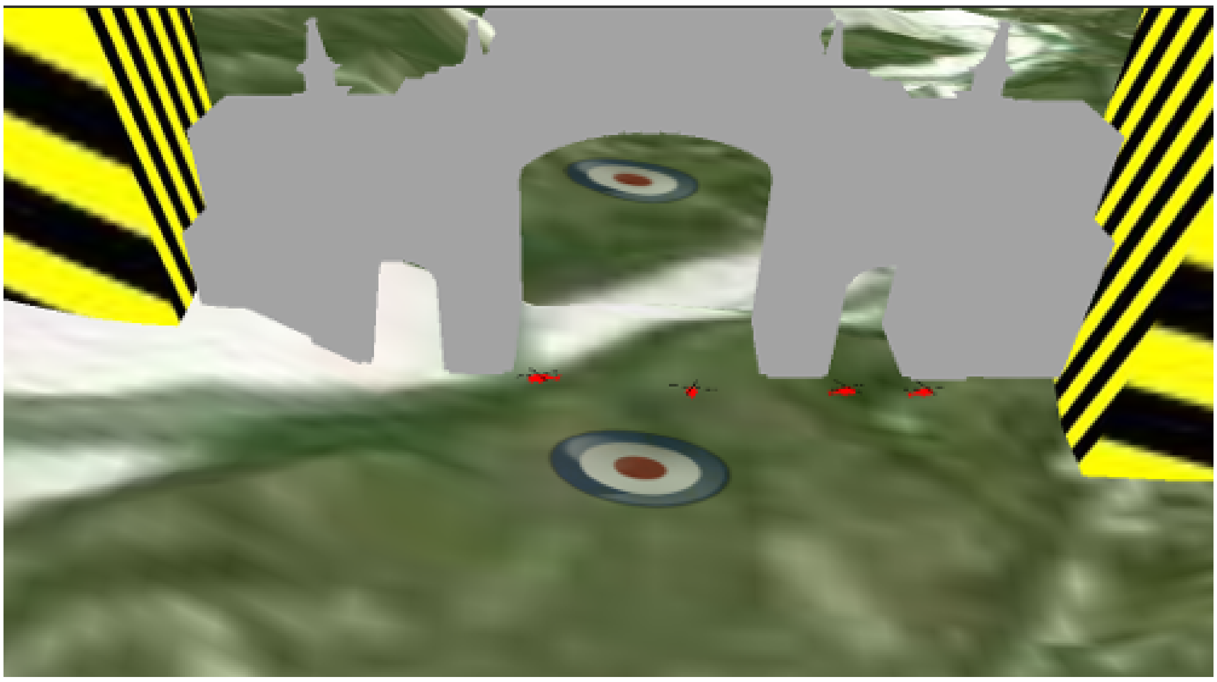


Figure 8.19.: *Cutout of the* Control Station *after the UAVs created a line formation in which they have to fly to the second target position behind the triumph arch represented by a ring shaped object.*

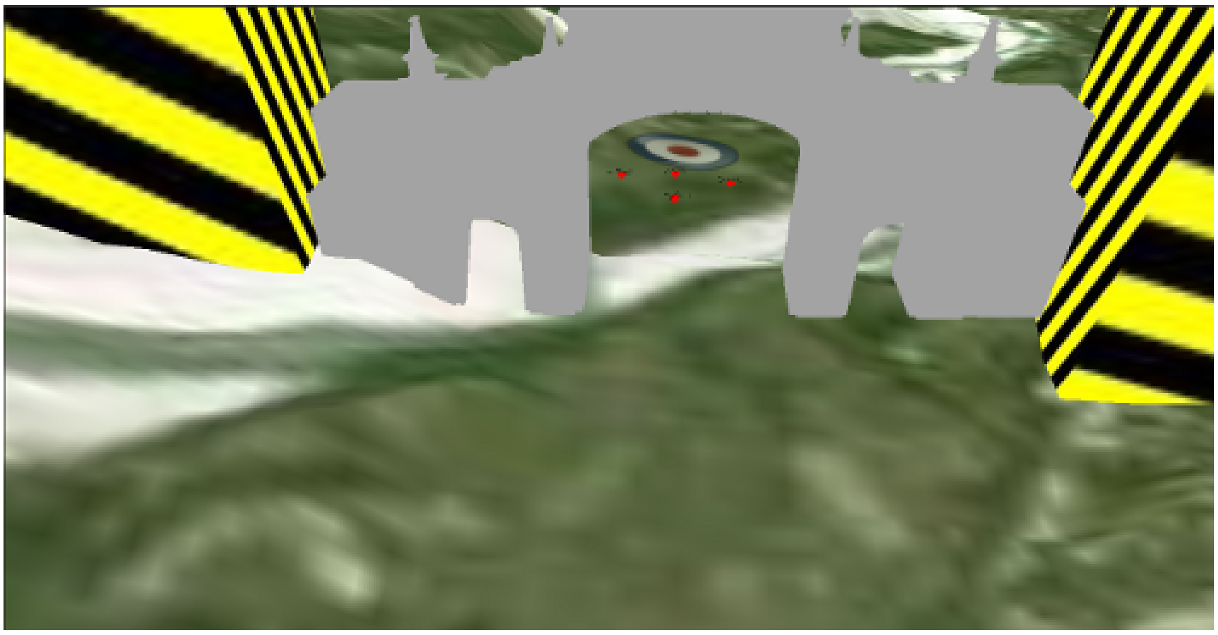


Figure 8.20.: *Cutout of an screenshot from the* Control Station *showing the change of the formation shape conducted by the UAVs to adapt the formation to the environment in order to fly through the small passage given by the triumph arch.*

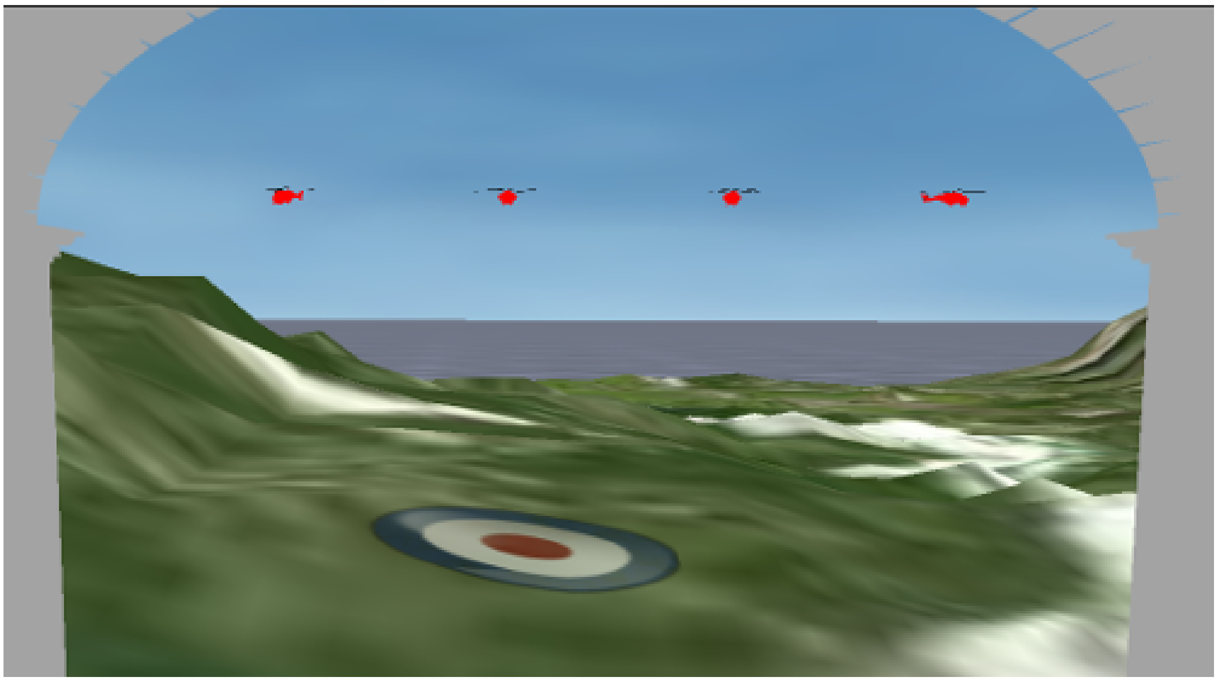


Figure 8.21.: *The final configuration after the test run. The red UAVs were able to restore the original line formation shape and fly towards the target configuration.*

After they passed the small passage, the original line formation shape is restored and the UAVs fly to the target position, as shown in Figure 8.21. They were able to perform this task, always using the same equation with the same parameters responsible for the creation of the formation shape (cf. Section 6.3.1). Thus, it is not necessary to detect and to directly consider obstacles as they are taken into account during potential field computation independent of the current formation parameters.

The position of the scene camera has been moved such that the camera is inside the arch for the creation of Figure 8.21. This has been done to provide an image of the entire formation at the final target area. The single paths of the UAVs are not shown for this test run, as they would cause too much confusion.

The test runs presented in this and the previous section give a short impression of the UAVs' self-adaptation behavior, resulting from the potential field approach. The results show that the corresponding objective from Section 1.2 has been reached. Beside path planning and path following behavior, a task allocation system has been designed. It is briefly evaluated during the following section for a single UAV and a multi-UAV scenario.

8.5.4. Single UAV Task Allocation

The test run performed in this section considers the self-organizing behavior of a single UAV, which is another objective mentioned in Section 1.2. The presented test for evaluating the task allocation system, described in detail in Section 7.4.2, regards the allocation behavior of a single UAV. Therefore, a UAV has to fulfill three different tasks. The test run shows that the task allocation behavior of single UAVs works as expected. The initial position of the UAV and the positions of the single tasks are presented in Table 8.9. Additionally, the role of the UAV and the task types are also shown in the table.

Name	x - position	y - position	z - position	role / type
$\mathcal{U}_1$	2050	500	140	Prospector UAV
$\mathcal{T}_1$	2060	406	100	Single deliverance
$\mathcal{T}_2$	1988	363	100	Single monitoring
$\mathcal{T}_3$	2015	451	100	Road spotting

Table 8.9.: *Initial position of the UAV and its role, as well as positions of the tasks and task types.*

Normally, the visualization of tasks vanishes as soon as the tasks are fulfilled as shown in the previous section. This visualization behavior has been turned off as illustrated in Figure 8.23 for the tests regarding task allocation in order to be able to directly compare the trajectories the UAVs used to complete the test runs with the positions of the single tasks.

Specific task priorities result from the task types in combination with the UAV role (cf. Table 7.4). These priorities are as follows: 0.5 for road spotting, 0 for single deliverance, 0.4

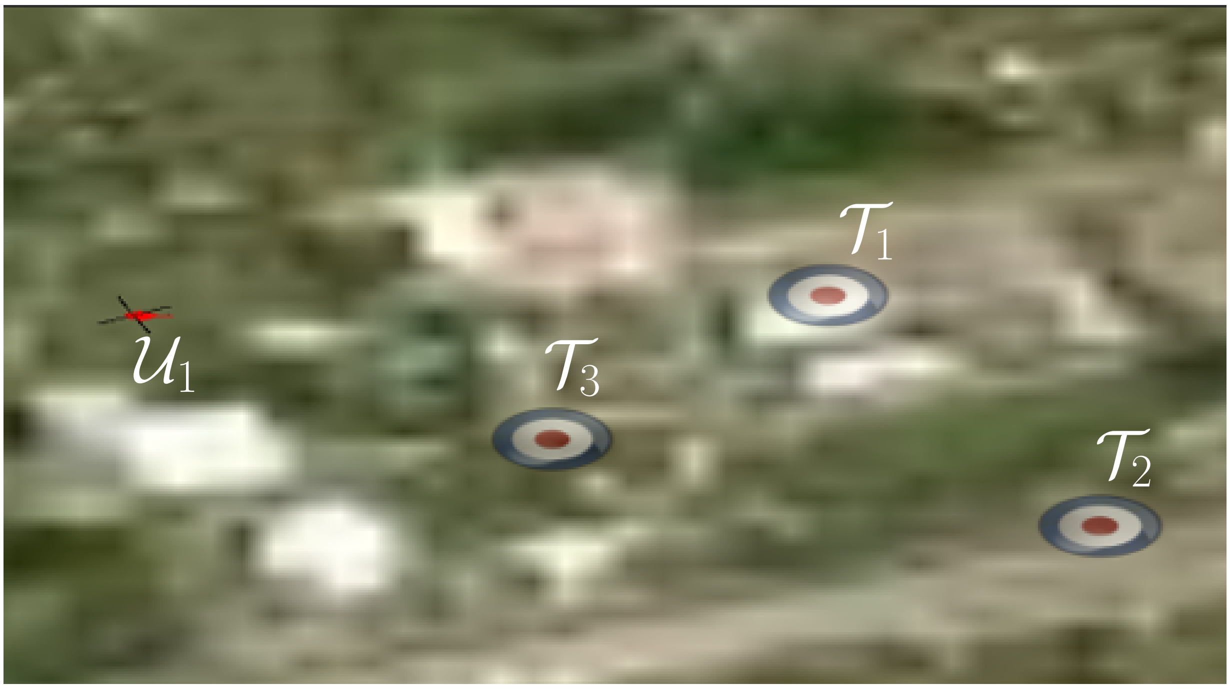


Figure 8.22.: *Illustration of the initial state before task allocation. The tasks $\mathcal{T}_1, \mathcal{T}_2$ and $\mathcal{T}_3$ are illustrated as ring shaped objects while the red UAV is in the left part of the image.*

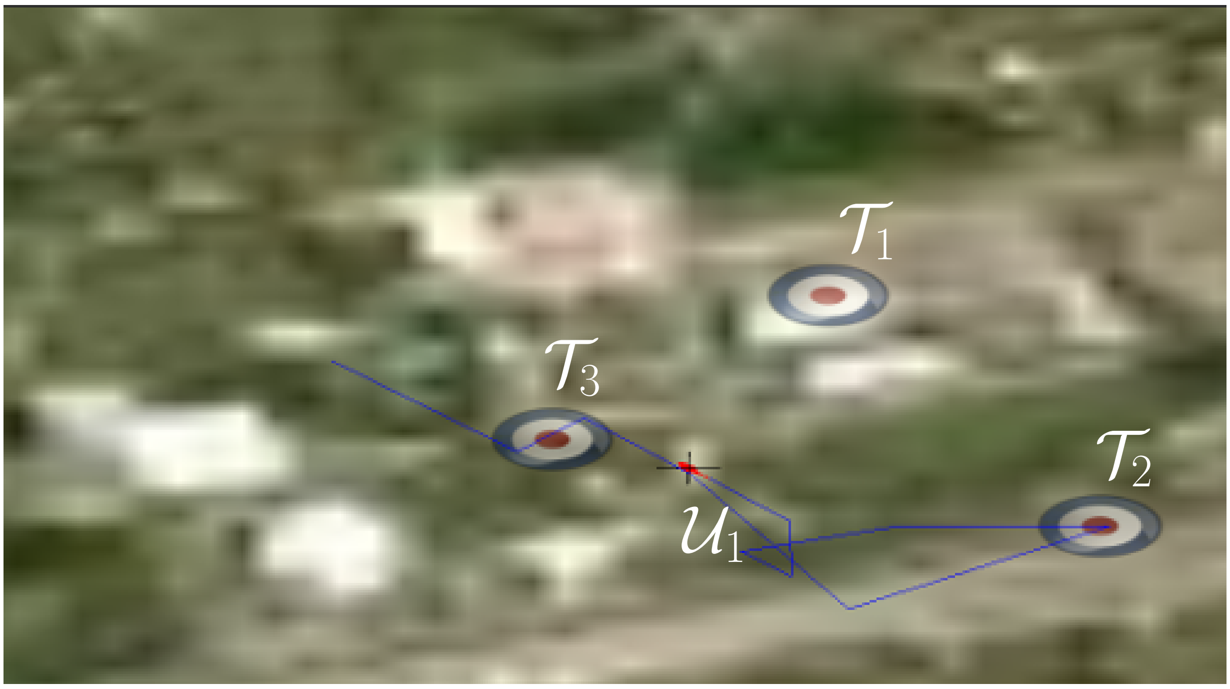


Figure 8.23.: *Illustration of the UAV path, which is marked as a blue line. It shows that the UAV moves to the third task $\mathcal{T}_3$ followed by the second $\mathcal{T}_2$ before it starts exploratory navigation.*

for single monitoring, and 0.3 for single exploration. As mentioned before, single exploration is always a task, even if it is not created explicitly.

The test run considers task allocation behavior only and neglects task execution. Meaning that, e. g., the UAV only has to fly to the initial position of the road spotting task and must not actually spot a road in order to fulfill it. This behavior has been chosen, because only the task allocation system is evaluated in this chapter.

Figure 8.22 shows the environment with the red UAV in the upper left and the tasks—illustrated as ring shaped objects—before task allocation occurs. Additionally, the UAV $\mathcal{U}_1$ and also the tasks $\mathcal{T}_1, \mathcal{T}_2$ and $\mathcal{T}_3$ are labeled to provide discriminability. The UAV then starts to fulfill the single tasks. It first moves to the road spotting tasks $\mathcal{T}_3$, which is the one resulting in the highest bid value due to the distance to the UAV and the priority of the task. The second task fulfilled by the UAV is the single monitoring task $\mathcal{T}_2$. Subsequently to the fulfillment of these two tasks, the UAV starts to explore the environment as the single deliverance task $\mathcal{T}_1$ is not processable by the UAV, due to its current equipment. This is represented by the priority value zero leading to a bid value of zero using Equation 7.18.

Figure 8.23 shows the paths of the UAV illustrated as coherent blue line. It can be seen that the UAV flies in the prescribed order, from one task to another, until it starts exploratory navigation. It can also be seen that the paths towards the task's target areas are not the shortest ones. This is a result of the approximations performed to efficiently compute the harmonic potential field (cf. Section 5.3.4). Using a higher iteration depth for the relaxation (cf. Section 5.3.3) and combining a lower number of nodes to bigger nodes increases the optimality of the resulting paths. Nevertheless, the allocation order shows that the task allocation system works as expected for this test run and that the UAV is able to execute self-organized task allocation.

Various additional tests regarding single UAV task allocation could be conducted to investigate the influence of different ages, as well as the tradeoff between the distance to the task and the priorities (cf. Equation 7.18). But the values selected for the presented system are only example values and can be changed, such that the expected behavior is triggered. The test run performed in this section gives a first impression of the results obtainable using the task allocation system, described in detail in Chapter 7. Beside single UAV task allocation, the allocation behavior of several UAVs is briefly evaluated in the following section.

8.5.5. Multi-UAV Task Allocation

The evaluation of the multi-UAV task allocation behavior beside single UAV task allocation behavior is necessary, too. It must be ensured that different UAVs do not fulfill single tasks redundantly, while other tasks starve in order to arrive at an efficient multi-UAV system. The test run presented in this section evaluates this. Additionally, it is shown, that two of the main objectives from Section 1.2, stating that the UAVs must have a completely autonomous and distributed behavior and that simultaneous task allocation has to be performed by the UAVs, have been reached.

To demonstrate this, three UAVs with differing roles have to allocate five tasks during this test run. It is shown that task allocation works as expected and that it is efficient. The start positions of the UAVs and their roles are presented in Table 8.10. Additionally, the table also shows the positions of the tasks and the single task types.

Name	x - position	y - position	z - position	role / type
$\mathcal{U}_1$	2140	575	140	Explorer UAV
$\mathcal{U}_2$	1910	540	140	Surveillant UAV
$\mathcal{U}_3$	2010	410	140	Collector UAV
$\mathcal{T}_1$	2100	530	100	Multiple monitoring
$\mathcal{T}_2$	1940	490	100	Single deliverance
$\mathcal{T}_3$	2050	370	100	Single monitoring
$\mathcal{T}_4$	1960	360	100	Road spotting
$\mathcal{T}_5$	2090	450	100	Escort

Table 8.10.: *Initial positions of the UAVs, positions of the tasks, and task types*

Table 8.11 shows the resulting priorities for the single UAVs. Again, executing the single tasks has been neglected and the UAVs only have to fly to the task positions, except for the multiple monitoring task. This behavior is selected to show an issue to be handled by such a task allocation system. The role prospector UAV is not considered for this test run as the UAV used for the test presented in the previous section has been assigned to this role and thus the tests obtained in that test run give an impression of the task allocation behavior reached by using prospector UAVs.

Task \ UAV	$\mathcal{U}_1$	$\mathcal{U}_2$	$\mathcal{U}_3$
Multiple monitoring	0.5	0.7	0.1
Single deliverance	0.0	0.0	0.5
Single monitoring	0.3	0.5	0.1
Road spotting	0.4	0.6	0.3
Escort	0.4	0.2	0.2
Single exploration	0.5	0.3	0.1

Table 8.11.: *The single task priorities dependent on the UAV role*

Figure 8.24 shows the initial setup of the test run. The red UAVs are illustrated, as well as the target positions of the single tasks. As before, the target positions of the single tasks are depicted as ring shaped objects. Additionally, the UAVs $\mathcal{U}_1$,$\mathcal{U}_2$ and $\mathcal{U}_3$ and also the tasks $\mathcal{T}_1, \ldots, \mathcal{T}_5$ are labeled to provide discriminability.

Figure 8.25 shows the paths of the single UAVs computed to reach the target positions of the tasks, as well as the end positions of the UAVs. The paths are colored based on the corresponding UAV. $\mathcal{U}_1$ creates red paths and the paths of $\mathcal{U}_2$ are green while the paths of $\mathcal{U}_3$ are illustrated by a blue line.

As aforementioned, the tasks are considered to be fulfilled as soon as a UAV reaches the corresponding target position except for the multiple monitoring task $\mathcal{T}_1$ in order to show an issue to be handled if multi-UAV tasks are supported. The multiple monitoring task is considered to be fulfilled as soon as a minimum of three UAVs reach the target area of this task. As can be seen in Figure 8.25 UAV $\mathcal{U}_1$ starts exploring the environment as the exploration path is, due to the close unexplored space, most beneficial. After some time it reaches a state in which it is close to the multiple monitoring task, such that it takes over this task. Thereafter, it flies to the task and waits for further UAVs in order to fulfill it. During this test run, no further UAVs selected $\mathcal{T}_1$ and thus $\mathcal{U}_1$ never moved on. Such a behavior is not only inefficient as one UAV does not do much more than to wait, it can also lead to starvation of other tasks if no further UAV is able to fulfill them. The UAVs should not start processing multi-UAV tasks as long as the minimum number of UAVs $Mi_{\mathcal{T}}$ needed to perform this task (cf. Section 7.3.1) select it as next to be executed, in order to overcome such behavior. In the mean time, the UAVs selecting the multi-UAV task as next to be executed, should put the task aside and perform other tasks.

The second UAV $\mathcal{U}_2$ started close to the single deliverance task $\mathcal{T}_2$. The priority of this task is zero to $\mathcal{U}_2$ and thus the UAV first starts to explore the environment as all other tasks are much further away than the closest unexplored space. It never stopped exploratory navigation during this test, as at the points in time it got close to one task's target position, such that the task would become more beneficial to $\mathcal{U}_2$ than single exploration the corresponding task has already been fulfilled by $\mathcal{U}_3$.

The UAV mainly responsible for task allocation and fulfillment during this test run was $\mathcal{U}_3$. Its priority for single exploration is only 0.1 and thus most other tasks are more beneficial to this UAV. It started close to the road spotting task $\mathcal{T}_4$ and allocated this one directly as next to be executed. During its flight towards the target area of $\mathcal{T}_4$ it got close to $\mathcal{T}_2$ and due to the higher priority (0.5 instead of 0.3) it would be possible that $\mathcal{U}_3$ changes its task to be processed next. This was not the case during this test run since checks for more beneficial tasks are not continuously performed, but rather from time to time, in order to avoid cases in which a UAV ends by only changing the next task. For this test run the result is that the UAV did not recognize that $\mathcal{T}_2$ was the most beneficial for a short time period and $\mathcal{U}_3$ fulfilled $\mathcal{T}_4$ first. After fulfilling $\mathcal{T}_4$ task $\mathcal{T}_2$ becomes most beneficial to $\mathcal{U}_3$ followed by $\mathcal{T}_5$ and $\mathcal{T}_3$. After fulfilling $\mathcal{T}_3$ only task $\mathcal{T}_1$ was left with $\mathcal{U}_1$ waiting for further UAVs to solve the task, too. This task has the same priority to $\mathcal{U}_3$ as exploratory navigation and since the next unexplored area is closer than $\mathcal{T}_1$, $\mathcal{U}_3$ started to perform the single exploration task.

The test runs presented so far evaluate several parts of the system presented in this thesis. Based on the achieved results, a conclusion is given in the following section.

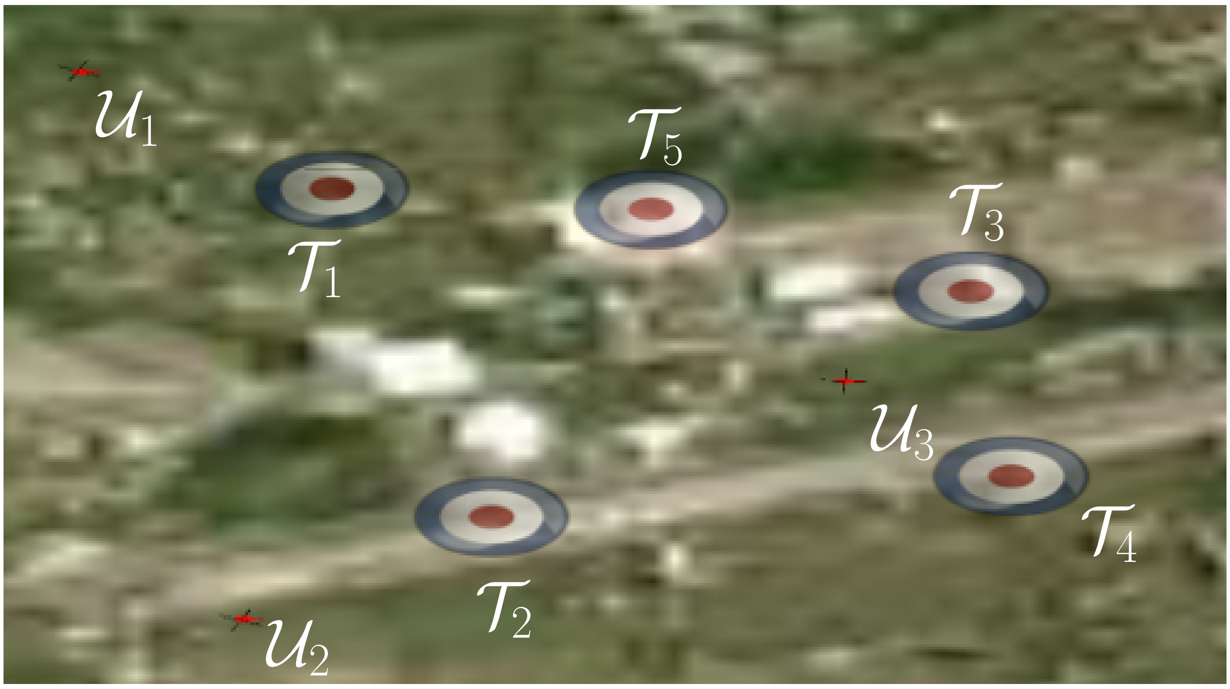


Figure 8.24.: *Illustration of the initial state before task allocation. The single tasks $\mathcal{T}_1, \ldots, \mathcal{T}_5$ are illustrated as ring shaped objects while the red UAVs $\mathcal{U}_1$,$\mathcal{U}_2$ and $\mathcal{U}_3$ are also distributed over the environment.*

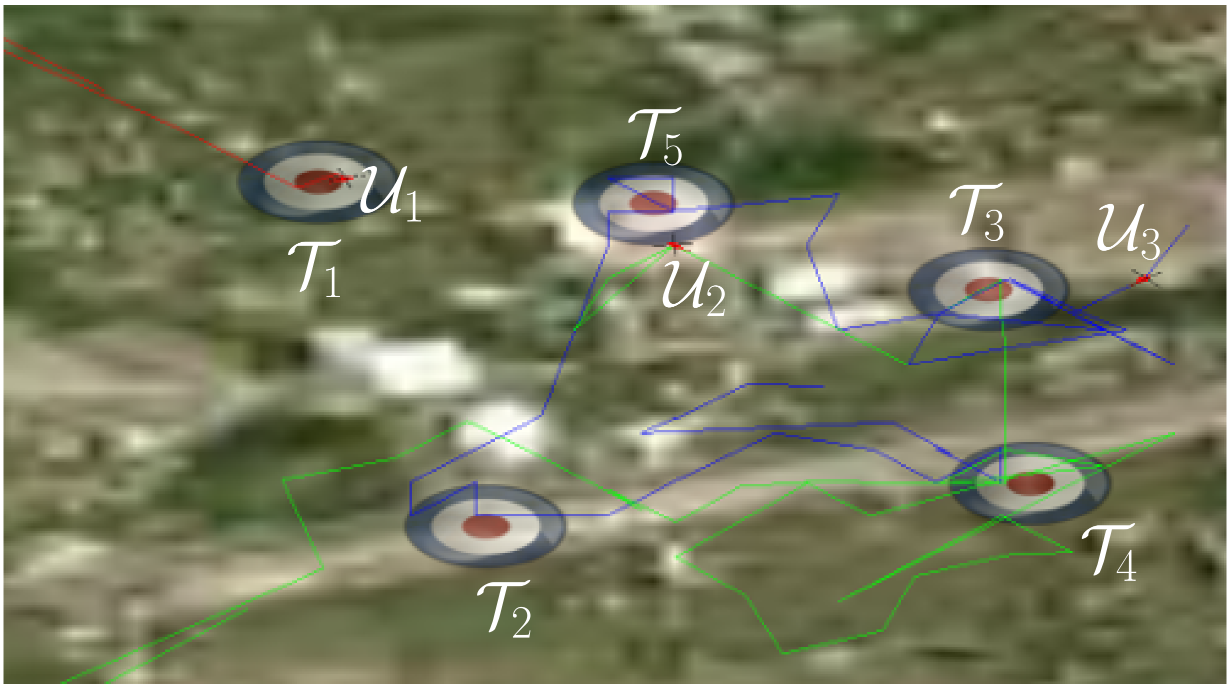


Figure 8.25.: *The paths, as well as the final positions of the UAVs after task allocation and flights to these tasks. The paths are colored whereby different colors have been used to represent the path of the single UAVs*

8.6. Conclusion

The results of the test runs described in this chapter, show that it is possible to create a dynamic potential field system leading to entire explorations of static, as well as highly dynamic environments based on real world terrain information. An example application where dynamically created and dissolved formations lead to a faster exploration of the environment has also been presented. The UAVs were always able to entirely explore the environment. Even the interaction of the UAVs, which consecutively compute differing potential fields does not break the system.

Based on the described results, it can be stated that the approach is computationally efficient. This is mainly due to the highly dynamic world model, provided by an octree (cf. Section 4.2). Noticeable is that the necessary computational effort rises with a higher complexity of the exploration task. The lowest computational effort was needed to explore the a priori known environment without formation flights, while the highest one has been reached by exploratory navigation of the unknown environment flying in formation.

Further tests beside the ones regarding UAV behavior for exploratory navigation have been conducted. It can be seen from the achieved results that the designed system leads to an intelligent behavior for self-adapation to environmental changes and that the UAVs are able to simultaneously create and allocate tasks. Thus, the evaluation has shown that the system presented in this thesis reached all objectives from Section 1.2.

The test series showed that the simple use of formation flights is far from beneficial in case of exploratory navigation. Comparing the durations needed by differing numbers of UAVs in unknown and a priori known environments showed in all test runs that never flying in formation is more beneficial than always flying in formation. Nevertheless, it has also been shown that a skillful use of additional formation flights in combination with non-formation flights can decrease the time needed to entirely explore the environment. Thus, formation flights are not always worse than non-formation flights but they must be used cautiously.

Due to the termination conditions (cf. Section 5.3.4), the discretization errors, and approximations an entire exploration of any kind of environment is not necessarily guaranteed. The discretization, as well as the termination conditions must be such that the UAVs do not create and fly into local equilibria. It has been shown in Section 5.5 that a harmonic potential field is globally asymptotically stable. Thus, UAVs create and fly only in local equilibria if the termination conditions are not sufficient. In this cases, a higher iteration depth and a higher distance at which nodes are combined vanishes the local equilibria and the UAVs are able to reach their target areas.

Further exploratory navigation tests using differing environments, as well as other numbers of UAVs and formations and also differing formation sizes would also be interesting. The results obtained from such further tests could be compared with the presented results in order to find more similarities and to detect contingencies, which occur only in the presented test scenarios. Testing additional parameter settings for the termination conditions and the distance in which leaves at maximum level are considered, could lead to a more efficient exploration behavior.

The test runs regarding the UAV behavior show that single UAVs are able to avoid obstacles, as well as that they can adapt formation shapes by themselves in order to overcome large obstacles. It has also been briefly shown that the task allocation system works as expected, using single UAV task allocation and multi-UAV task allocation. Not any part of the system has been evaluated in this chapter as this would exceed the thesis. To entirely evaluate a complex multi-UAV system, various additional tests have to be performed.

A part of the UAV system that has been neglected in this chapter is task processing. Additional tests with tasks to be processed, which have additional constraints would be interesting. A multiple deliverance task, e. g., has most times the constraint that the UAVs are not allowed to change their position inside the formation. Further checks with such constraints to investigate whether the UAVs are able to overcome obstacles for which they have to change the formation shape would also be worth performing.

Test runs using varying numbers of UAVs to show the interactions of the UAVs with each other should be performed, too. It would also be interesting to show, how changing numbers of UAVs affect the efficiency of the system and to deeply investigate the properties robustness, flexibility, and scalability mentioned in Chapter 1, as the three properties are essential for the survival of animal societies and also beneficial for complex multi-UAV systems like the one presented in this thesis.

Finally, jumping to wrong conclusions should be avoided. The evaluation states that the system works and that it is efficient in theory. A wide range of additional tests must be run in order to port it to real UAVs. In that case it has to be investigated how the system behaves on faulty sensor inputs and with an inaccurate positioning system, not only of the UAV itself but also of all other UAVs flying simultaneously in the environment. This can, e. g., be simulated using white noise. The influence of environmental conditions like changing wind directions, rain and so on must also be investigated in order to adopt the system to real world UAVs, as well as several other conditions not mentioned in this section.

8.7. Summary

This chapter presents the evaluation of the multi-UAV system. To evaluate the system, a simulation environment has been used instead of real UAVs. This is necessary for first tests regarding the UAV behavior due to several reasons explained in Section 1.2. Additionally, multiple assumptions that might not hold for real world applications have been made. Thereafter, an introduction of the environment used to evaluate the system takes place. It is based on real world data, in order to achieve meaningful results and has been extended by three-dimensional structures, to show that the approach works in three-dimensional environments.

Based on the environment, various parameters have been set and differing values were measured to provide results regarding the objectives of the evaluation. They are presented after the environment's description. The single objectives of the evaluation are also described in detail.

After this introduction, test series regarding exploratory navigation have been performed. First, the UAVs had to explore the environment without formation capabilities. The explorations took place in a priori known environments, as well as in unknown environments and were repeated using differing number of UAVs.

Thereafter, the test series have been run again while this time the UAVs always fly in formation. A comparison of the obtained results to the results achieved before takes place and based on these results, a last test run regarding exploratory navigation has been conducted. For the last test run, six UAVs combined non-formation flights with formation flights in order to achieve a benefit, concerning the duration needed to entirely explore the a priori known environment.

In addition to the test series regarding exploratory navigation, a brief evaluation of the UAV behavior is presented. Therefore, test runs regarding obstacle avoidance behavior of single UAVs and UAVs in formation are described. Finally, the task allocation system is briefly evaluated using a single UAV and several UAVs, which had to allocate various tasks.

During the last chapters, the entire system is described in detail. It is also evaluated in this chapter and therefore completely described. The next chapter gives a summary of the thesis and an outlook on future work.

Chapter 9

Summary and Outlook

Autonomous robot movement received increasing attention by researchers in the beginning of the 1980s through the introduction of several popular methodologies. Henceforth, countless methods to autonomously move single robots right up to coordination approaches for heterogeneous robot groups have been designed. Contemporarily, a new research area called swarm robotics arose.

A wide range of approaches based on different research areas have been designed and a few of them have become popular. Numerousness research conducted by several researchers over the course of the last decades has been extended and combined in this thesis for the development of a cooperative UAV system usable to support ground units in various applications. A summary of the resulting system presented in this thesis is given in the following section.

9.1. Summary

A unique multi-UAV path planning system extended by methods for exploratory navigation and for the establishment of formations based on harmonic function theory is presented. It goes far beyond state-of-the-art systems. The UAVs are able to entirely explore three-dimensional environments consisting of complex structures like buildings, tunnels, etc. on their own, as well as in formation.

Inter-UAV communication is one basis for cooperative UAV behavior. Additionally, the UAVs have abilities to dynamically create formations that support a changing number of participants by the use of bifurcation theory. Establishing and dissolving a formation, as well as changing the formation shape can be conducted anytime.

Formation flights are based on a behavioral approach enabling the UAVs to change their positions related to the formation in order to avoid obstacles and other UAVs autonomously.

This is the first system combining harmonic function theory and bifurcation theory to create dynamic formations used for exploratory navigation in complex and dynamically changing three-dimensional environments. The resulting UAV behavior is verifiable, e. g., by showing that the created potential fields are globally asymptotically stable. A verification is important to apply the system to real world applications. Further properties of the system are:

- Self-adaptation related to environmental changes.
- Self-optimization due to the designed task creation and task allocation approach.
- Verifiable due to the use of Lyapunov stability theory.
- Cooperative as the UAVs exchange information to increase efficiency.
- Able to explore complex and dynamically changing three-dimensional environments.

The three objectives robustness, flexibility, and scalability observed by Sahin et al. [17] (cf. Chapter 1) are also covered by the resulting system. The completely autonomous and decentralized architecture ensures that failures of single UAVs do not cause the entire system to break down. Such faults are compensated by other UAVs. This makes the system robust. The absence of a centralized coordination instance avoids possible bottlenecks and makes the system highly scalable. Artificial potential fields provide the single UAVs with abilities for self-coordination and the market place approach provides necessary capabilities for simultaneous task procession leading to a highly flexible system.

9.2. Contributions

A survey of agent coordination and robotic movement for single agents and for heterogeneous robot groups is provided in Chapter 2. Swarm approaches from nature adaptable for robot control are also presented in that chapter. Chapter 3 introduces the basic methodologies used to design the resulting system. The remainder delivers the following contributions:

- **An architecture for terrain representation as path planning basis (Chapter 4).** The presented architecture is mainly based on an octree as highly dynamic data structure leading to fast potential field computations and feasible paths. The UAVs receive the environmental information as triangle meshes to make the system universally applicable.
- **A potential field represented by a harmonic function for path computations (Chapter 5).** The combination of artificial potential field theory and harmonic functions is used and adapted to work with a dynamically changing octree in three dimensions. This provides the capabilities needed to entirely explore dynamic three-dimensional environments including complex structures like buildings, tunnels, etc. The

method mathematically guarantees obstacle avoidance and can lead to a sound path planning application.

- **A potential field based on bifurcation theory for formation flights (Chapter 6).** The normal form of a Pitchfork bifurcation has been extended for potential field computations allowing the UAVs to establish different formation shapes in three dimensions. The octree is still the basis of the internal world model. The potential field is superpositioned with the harmonic potential field in a way that preserves the advantages of both methodologies.

- **A behavioral approach for task creation and task scheduling to create a cooperative UAV behavior (Chapter 7).** The UAVs establish formations autonomously if a formation is beneficial. They also create tasks on their own if this is necessary to avoid damage. A task scheduling approach based on a decentralized market place has also been designed for the support of ground units in various applications.

- **A detailed evaluation of the single parts, as well as of the overall system using a large and highly complex terrain (Chapter 8).** An implementation of the system has been conducted and a large and complex environment based on real world data has been created to evaluate the approach. The evaluation shows that the system works using discrete arithmetic and that it is highly efficient.

The mentioned contributions have been published in the following journal, book, and conference proceedings:

- **ICAS:** 6th International Conference on Autonomic and Autonomous Systems, 2010 (best paper award, invitation to write a journal article for ThinkMind) [3],

- **ICARCV:** 11th International Conference on Control Automation Robotics Vision, 2010 [6],

- **ThinkMind:** International Journal On Advances in Software, 2011, pages 351–370 [5],

- **ICARA:** The 5th International Conference on Automation, Robotics and Applications, 2011 (invitation to write a book chapter in Recent Advances in Robotics and Automation) [4],

- **Studies in Computational Intelligence:** Recent Advances in Robotics and Automation, 2013 [8],

9.3. Outlook

The presented system covers a wide range of applications. It is a step towards a completely autonomous real world UAV system for the support of ground units without the need of human interactions. Still, several challenges for the design of a real world system exist. Until now, the approach has been tested only simulative. An interesting and also challenging task would be an adoption to real UAVs. The UAVs should be equipped with sensors like cameras, laser, radar, etc. to adjust inaccuracies of a positioning system like GPS. Such sensors are usable to detect the surrounding environment and to map it to the world model of the UAVs. Three-dimensional maps of terrains might then be created. Such a system can be used, e. g., in disaster areas to quickly create an overall map of the area. Several approaches for the creation of maps from single pictures exist. One of these approaches is, e. g., presented in [7], [9] and [10].

Positioning of the UAVs must also be verifiable at least in terms of inaccuracies to adopt the approach for the use in real environments. Known inaccuracies can be compared, e. g., by obstacle growing. Inter-UAV communication is also an issue for real world scenarios. Only a limited communication range exists if technologies like wireless LAN are used. Routing protocols can be an interesting point to make the communication task more robust.

Another interesting point of future work would be the extension of the approach to work in a more heterogeneous robotic environment. Fixed wing UAVs, as well as different unmanned ground vehicles and unmanned underwater vehicles could be introduced to make the system suitable for additional applications. One application could be the erection of skyscrapers, in which UAVs transport material to the upper levels of the building.

The evaluation of the approach shows that using formations to entirely explore environments is not necessarily beneficial. A skillful behavior able to use formations only when it is beneficial is definitely worth to be designed. A few first thoughts about how such a behavior can be created are presented in Section 7.4.1. A further idea would be the use of expert knowledge given by control station personal. Nevertheless, formation capabilities allow the execution of several tasks single UAVs are unable to process.

The introduction of learning algorithms to support UAV behavior, as well as for target selection during exploratory navigation could lead to an increase in efficiency regarding the system's overall performance. The designed system can be made more robust through approaches from the area of swarm intelligence, which might also be used to supersede the global positioning system.

The mentioned aspects are still interesting and relevant for the design of a completely autonomous and decentralized real world UAV system. Based on the behavior resulting from the methods presented in this thesis the UAVs are able to coordinate themselves, to plan paths to each reachable region of various three-dimensional environments, to create formations, to fly in formation, and to distribute tasks on their own. Thus, this thesis provides a first step towards a decentralized multi-UAV system.

Appendix A

Algorithms

The most important algorithms used for the design of the system presented in this thesis are described in detail in this appendix. They are all based on the octree introduced in Section 4.2. The octree is the basic data structure for terrain representation, as well as for path planning and is used as world model.

Functional descriptions of the algorithms are provided in the single chapters of the thesis. The appendix gives more detailed insights to the implementation of single algorithms and presents them using pseudo code. A brief overview of the algorithms and their duties is provided in the following.

- The neighbor finding algorithm presented in Appendix A.1 is the most frequently used algorithm within this thesis. It takes a node and a search direction as input and returns a list of neighboring nodes. Several other algorithms rely on the output of the neighbor finding algorithm.
- It is possible that the terrain consists of unreachable areas. To avoid the selection of such areas by the UAVs as next to be explored, which would lead to path computation failures, nodes enclosing such areas are marked as occupied. An algorithm based on the flood fill method has been designed for this purpose. It is presented in Appendix A.2.
- The UAVs fly in dynamically changing complex three-dimensional environments. An update algorithm has been designed in order to reach an efficient mapping of environmental changes into the internal world model. It is described in Appendix A.3.
- The update algorithm increases the number of octree nodes. Such an increase slows down the entire system's performance. A node combination algorithm is used to overcome this problem. It is presented in Appendix A.4.
- UAV flights are based on artificial potential field theory. Two potential field algorithms have been designed for the resulting system. The harmonic potential field algorithm

described in Appendix A.5 is used for path planning and the bifurcating potential field algorithm presented in Appendix A.6 is used for formation flights.

- Finally, a highly complex UAV system with purposes to simultaneously process several different tasks autonomously and to efficiently perform exploratory navigation needs a task creation algorithm presented in Appendix A.7 and a task allocation algorithm described in Appendix A.8 for task distribution.

The single algorithms are described in detail in the following sections starting with the neighbor finding approach. References to the corresponding sections that describe the functionality and provide the mathematical equations are also included in the next sections.

A.1. Neighbor Finding

The neighbor finding algorithm is based on ray tracing used in the computer graphics domain. The functionality and advantages of the algorithm are briefly described in Section 4.2.4. The algorithm takes a node l_i at level n and a search direction as input. The output is a list of the neighboring nodes in the search direction.

First, offsets are computed a priori as presented in Listing A.1. These offsets are based on the dimensions of the single leaves at each level of the octree. The computation can easily be done using the Pythagorean theorem. Twenty-six offsets representing the twenty-six possible search directions are computed for each of the octree's levels. A single offset is half of the dimension of the node at the given level in the search direction plus half of the dimension that a node at maximum level has in the given direction.

```
CalculateOffsets(terrain_dimensions, octree_levels)
  node_dimensions = terrain_dimensions

  for(i ← 9; i ≤ octree_levels; i++)
    offsets[i] = CalculateOffsets(node_dimensions)
    node_dimensions /= 2
```

Listing A.1: *Method to compute offsets a priori.*

The algorithm determines neighboring nodes by finding areas that enclose specific positions. To compute such positions, the relevant offset is added to the center position of the start node l_i at level n as shown in Listing A.2. This ensures that the computed position is enclosed by the corresponding neighboring leaf.

It is necessary to determine all neighboring leaves of a leaf for potential field computation. Leaves at the environment border do not have neighboring nodes in every direction. Thus, checking the existence of neighboring leaves is necessary. The method *FindEqualLevelNeighbor*

```
FindOcTreeNeighbors(node, direction, root_node)
  neighbors ← root_node

  neighbor_position ← node.center + offset[node.level][direction]
  neighbors ← NULL

  if(neighbor_position is outside the terrain)
    return no_neighbors_exist

  FindEqualLevelNeighbor(neighbor_position, neighbors, direction)
  return neighbors_found
```

Listing A.2: *Entry method for the neighbor finding algorithm. The methods main responsibility is the computation of a specific point enclosed by the neighboring leaf.*

in Listing A.3 is called if a neighboring node exists. It determines a single neighboring node in the search direction at level $r \leq n$ based on the previously computed position.

Each node is either a leaf or is subdivided by eight child nodes. The single children always encapsulate the same parts of the parent node. These parts are illustrated in Figure 4.7 and are in particular at the lower left front, the lower left back, the lower right front, the lower

```
FindEqualLevelNeighbor(neighbor_position, neighbors, direction)
  for (i ← 0; i < node.level; i++)
    if (neighbors is leaf)
      neighbors_found ← neighbors
      return
    child_number ← 0
    if (neighbor_position.x > neighbors.center.x)
      child_number += 1
    if (neighbor_position.y > neighbors.center.y)
      child_number += 2
    if (neighbor_position.z > neighbors.center.z)
      child_number += 4

    neighbors ← neighbors.child[child_number]

  num_childs_in_dir ← GetNumChilds(direction)
  switch(num_childs_in_dir)
    case 1:
      FindNeighborAlongVertex()
    case 2:
      FindNeighborsAlongEdge(neighbors)
    case 4:
      FindNeighborsAlongFace(neighbors)
```

Listing A.3: *The method determines a neighbor node with a level lower or equal to the level of the start node. If that node is not a leaf, additional methods are called to determine the single neighboring leaves.*

right back, the upper left front, the upper left back, the upper right front and the upper right back of the parent node. The for loop in Listing A.3 is used to iteratively run from the current node to the child node that encloses the computed position. The loop

terminates if the current node is a leaf or if the node level is equal to the node level of l_i. The corresponding neighbor leaf has been found in these cases. The computation of the resulting child number is based on the distribution of the single children presented in Figure 4.7.

Only one neighboring leaf exists if the node determined by method *FindEqualLevelNeighbor* is a leaf. It is then stored in the list of found neighbors and the algorithm terminates.

A distinction based on the search direction takes place if the determined node is not a leaf. Only one neighboring leaf exists if the search direction leads to neighboring leaves along a vertex. In this case, the method *FindNeighborAlongVertex* shown in Listing A.4 is called. Two child nodes are neighbors of l_i if the search direction leads to neighboring nodes along an edge. In that case the method *FindNeighborsAlongEdge* shown in Listing A.5 is called. The last case is that the search direction leads to neighboring nodes along a face. Then four child nodes of the currently determined nodes are neighbors and the method *FindNeighborsAlongFace* shown in Listing A.6 is called.

Only one single neighbor node exists if a neighboring leaf along a vertex has to be determined. Method *FindNeighborAlongVertex* is called if this leaf is additionally at a higher level than l_i. The method moves iteratively starting with the previously determined node to one of the child nodes until a leaf node has been reached. The child node to which the method moves is the one that encapsulates the area of the parent node. This node lies in the opposite direction of the search direction and leads to the correct neighbor leaf.

```
FindNeighborAlongVertex()
  for (i ← node.level; i ≤ tree_depth; i++)
    if (neighbors is not leaf)
      neighbors ← neighbors.child[search_direction[0]]

  neighbors_found ← neighbors
```

Listing A.4: *Method to find a single neighbor leaf along a vertex.*

Method *FindNeighborsAlongEdge* presented in Listing A.5 is also used to find neighbors of higher level than n. The methodology is similar to the one used by method *FindNeighborAlongVertex*. Based on the previously determined neighbor node two neighboring child nodes are recursively selected. The selected child nodes are the one that encapsulate the area of the parent node that lies in the opposite direction of the search direction. The recursions stop when all reached nodes are leaves. Each determined leaf is added to the found neighbor list. Finally, the found neighbor list contains all neighboring leaves of l_i.

Neighboring leaves can also be adjacent to l_i along a face. Method *FindNeighborAlongFace* presented in Listing A.6 is used to find these neighboring leaves. It works similar to the methods *FindNeighborAlongVertex* and *FindNeighborAlongVertex* and also starts with the previously determined node. The method recursively selects four child nodes encapsulating the areas in the opposite direction of the search direction. The recursions stop if all reached nodes are leaves. Each leaf that is determined this way is added to the found neighbor list. Finally, the found neighbor list contains all neighboring leaves of l_i in the given search direction.

```
FindNeighborsAlongEdge(node)
  if (node is leaf)
    neighbors_found.push_back(node)
  else
    FindNeighborsAlongEdge(node.child[search_direction[0]])
    FindNeighborsAlongEdge(node.child[search_direction[1]])
```

Listing A.5: *Method to find neighboring leaves along an edge.*

```
FindNeighborsAlongFace(node)
  if (node is leaf)
    neighbors_found.push_back(node)
  else
    for (i ← 0; i < 4; i++)
      FindNeighborsAlongFace(node.child[search_direction[i]])
```

Listing A.6: *Method to find neighboring leaves along a face.*

The algorithm is subdivided into six methods and is able to determine neighboring leaves of given nodes in twenty-six different search directions. It is computationally fast as it works mainly on pointer arithmetic. Only simple comparisons to decide whether a node is a leaf and simple addition operations must be performed by the algorithm.

A.2. Flood Fill

The UAVs fly in highly complex three-dimensional environments. In such terrains areas unreachable for the UAV can exist, e. g., rooms in buildings the UAV cannot enter. This has to be taken into account for a completely autonomous UAV system, in order to ensure that a UAV does not select such an area as its next target. Otherwise, it would be unable to compute a feasible path.

An algorithm based on the flood fill technique of paint programs has been adapted to overcome the problem. The algorithm is also briefly described in Section 4.2.2 and works as follows.

First, the algorithm uses the methodology presented in Listing A.3 to determine the leaf that currently encloses the UAV. The method *FindReachableTerrain* presented in Listing A.8 takes this leaf as input to detect all areas of the environment the UAV can reach. Afterwards, the method *SetOccupied* presented in Listing A.9 assigns all unreachable areas to $\mathcal{C}_{\mathcal{O}}$.

The method *FindReachableTerrain* presented in Listing A.8 is used to detect all reachable nodes. The leaf enclosing the UAV is the entry point for the search. First, it is checked whether the leaf to be examined is occupied or has been checked before. No further actions are conducted for the leaf in these cases.

If no examination of the leaf has been conducted before, the leaf is set to be reachable. Afterwards, the neighbor finding algorithm described in Appendix A.1 is used to determine all neighboring leaves of the current leaf. The method *FindReachableTerrain* is then called again for each neighboring node. It terminates when no further leaf passes the initial check. After the termination of the method all leaves $l_i \in \mathcal{C}_\mathcal{F}$ the UAV is able to reach are marked as reachable.

```
SetUnreachableTerrainOccupied(root_node, uav_position)
  uav_node ← root_node

  for (i ← 0; i < tree_depth; i++)
    if (uav_node is leaf)
      break
    child_number ← 0
    if (uav_position.x > uav_node.center.x)
      child_number += 1
    if (uav_position.y > uav_node.center.y)
      child_number += 2
    if (uav_position.z > uav_node.center.z)
      child_number += 4

      uav_node ← uav_node.child[child_number]

  FindReachableTerrain(uav_node)
  SetOccupied(root_node)
```

Listing A.7: *Entry method of the algorithm used to assign unreachable leaves to $\mathcal{C}_\mathcal{O}$.*

```
FindReachableTerrain(uav_node)
  if (uav_node ∈ $\mathcal{C}_\mathcal{O}$ or uav_node.is_reachable)
    return

  uav_node.is_reachable ← true

  for (i ← 0; i < 26; i++)
    neighbors ← FindOcTreeNeighbors(uav_node, i)
      foreach (neighbor in neighbors)
        FindReachableTerrain(neighbor)
```

Listing A.8: *Method to determine nodes reachable by the UAV.*

The method *SetOccupied* presented in Listing A.9 is called after method *FindReachableTerrain* terminated. The method takes the root node as input and recursively moves through the entire octree. It assigns each leaf that is in $\mathcal{C}_\mathcal{F}$ and not marked as reachable to $\mathcal{C}_\mathcal{O}$.

The algorithm is executed every time the UAV receives new terrain objects and maps them into the octree. It is also executed if the environment is unknown a priori and newly explored occupied leaves intersect the current path of the UAV. This ensures that a UAV will never select an unreachable area as its next target area. It also decreases the computational effort for the potential field computations as leaves, irrelevant for the computations, are no longer treated as unbound leaves and may also be combined with occupied leaves to even larger leaves.

```
SetOccupied(node)
  if (node is not leaf)
    foreach (child)
      SetOccupied(child)
    return

  if (node is not reachable)
    node.space_type ← $\mathcal{C}_\mathcal{O}$
    node.potential ← 1.f

node.reachable ← false
```

Listing A.9: *Method to assign unreachable nodes to $\mathcal{C}_\mathcal{O}$.*

A.3. Update Octree

The internal world model of the UAVs is based on an octree (cf. Section 4.2). This octree is dynamically changeable (cf. Section 4.2.1) in order to internally represent environmental changes like newly explored terrain (cf. Section 4.3.1). An adaption of classical ray tracing algorithms has been designed and is used to continuously update the octree.

The main purpose of the update octree algorithm is the mapping of newly explored terrain into the octree. The entry method is *Explore* presented in Listing A.10. First, values to help computing the maximum environment explorable by the camera are calculated. These values are four two-dimensional boundaries on the terrain ground.

Based on the previously computed boundaries, method *DetermineExploredNodes* presented in Listing A.11 is used to actually determine ground positions that lie within the area the camera equipment of the UAV can explore. Finally, method *SendRay* presented in Listing A.12 is used to compute discretized rays that determine the newly explored nodes.

```
Explore(UAV)
  cam_boundaries ← CalcCamViewBoundaries(position, angles, CameraRatio,
      CameraFlareAngle)
  DetermineExploredNodes(position)
```

Listing A.10: *Entry method to map environmental changes into the octree.*

First, it is assumed that the environment below the UAV is obstacle free. This enables the computation of the terrain ground's boundary positions, enclosing a two-dimensional area on the terrain ground the camera can explore. Equations A.1 to A.8 compute eight coordinates—four in x - dimension and four in y - dimension—leading to four two-dimensional boundary points. These points are illustrated in Figure A.1 and are in particular (cb_{x0}, cb_{y0}), (cb_{x1}, cb_{y1}), (cb_{x2}, cb_{y2}) and (cb_{x3}, cb_{y3}).

The UAV $\mathcal{U}$ is always at a specific position $p_{\mathcal{U}}$ with $p_{\mathcal{U}}(i)$ denoting the x, y, and z dimension of the position. Additionally, the UAVs can rotate around a yaw $\beta_{\mathcal{U}}$, a pitch $\gamma_{\mathcal{U}}$, and a roll $\delta_{\mathcal{U}}$ angle. These angles must also be taken into account to compute a proper area explored by the UAV. Further on, the camera of the UAV has a ratio r and a flare angle θ and is mounted at a camera angle λ. The ratio leads to a flare angle for the width θ_w and a flare angle for the height θ_h of the camera. These angles are also taken into account for the computation of the boundary points.

$$cb_{x0} = \tan\left(\delta_{\mathcal{U}} - \frac{\theta_w}{2} + \lambda\right) \cdot p_{\mathcal{U}}(z) \cdot \cos(\beta_{\mathcal{U}}) - \tan\left(\gamma_{\mathcal{U}} + \frac{\theta_h}{2} + \lambda\right) \cdot p_{\mathcal{U}}(z) \cdot \sin(\beta_{\mathcal{U}}) + p_{\mathcal{U}}(x) \quad \text{(A.1)}$$

$$cb_{y0} = \tan\left(\delta_{\mathcal{U}} - \frac{\theta_w}{2} + \lambda\right) \cdot p_{\mathcal{U}}(z) \cdot \sin(\beta_{\mathcal{U}}) + \tan\left(\gamma_{\mathcal{U}} + \frac{\theta_h}{2} + \lambda\right) \cdot p_{\mathcal{U}}(z) \cdot \cos(\beta_{\mathcal{U}}) + p_{\mathcal{U}}(y) \quad \text{(A.2)}$$

$$cb_{x1} = \tan\left(\delta_{\mathcal{U}} + \frac{\theta_w}{2} + \lambda\right) \cdot p_{\mathcal{U}}(z) \cdot \cos(\beta_{\mathcal{U}}) - \tan\left(\gamma_{\mathcal{U}} + \frac{\theta_h}{2} + \lambda\right) \cdot p_{\mathcal{U}}(z) \cdot \sin(\beta_{\mathcal{U}}) + p_{\mathcal{U}}(x) \quad \text{(A.3)}$$

$$cb_{y1} = \tan\left(\delta_{\mathcal{U}} + \frac{\theta_w}{2} + \lambda\right) \cdot p_{\mathcal{U}}(z) \cdot \sin(\beta_{\mathcal{U}}) + \tan\left(\gamma_{\mathcal{U}} + \frac{\theta_h}{2} + \lambda\right) \cdot p_{\mathcal{U}}(z) \cdot \cos(\beta_{\mathcal{U}}) + p_{\mathcal{U}}(y) \quad \text{(A.4)}$$

$$cb_{x2} = \tan\left(\delta_{\mathcal{U}} + \frac{\theta_w}{2} + \lambda\right) \cdot p_{\mathcal{U}}(z) \cdot \cos(\beta_{\mathcal{U}}) - \tan\left(\gamma_{\mathcal{U}} - \frac{\theta_h}{2} + \lambda\right) \cdot p_{\mathcal{U}}(z) \cdot \sin(\beta_{\mathcal{U}}) + p_{\mathcal{U}}(x) \quad \text{(A.5)}$$

$$cb_{y2} = \tan\left(\delta_{\mathcal{U}} + \frac{\theta_w}{2} + \lambda\right) \cdot p_{\mathcal{U}}(z) \cdot \sin(\beta_{\mathcal{U}}) + \tan\left(\gamma_{\mathcal{U}} - \frac{\theta_h}{2} + \lambda\right) \cdot p_{\mathcal{U}}(z) \cdot \cos(\beta_{\mathcal{U}}) + p_{\mathcal{U}}(y) \quad \text{(A.6)}$$

$$cb_{x3} = \tan\left(\delta_{\mathcal{U}} - \frac{\theta_w}{2} + \lambda\right) \cdot p_{\mathcal{U}}(z) \cdot \cos(\beta_{\mathcal{U}}) - \tan\left(\gamma_{\mathcal{U}} - \frac{\theta_h}{2} + \lambda\right) \cdot p_{\mathcal{U}}(z) \cdot \sin(\beta_{\mathcal{U}}) + p_{\mathcal{U}}(x) \quad \text{(A.7)}$$

$$cb_{y3} = \tan\left(\delta_{\mathcal{U}} - \frac{\theta_w}{2} + \lambda\right) \cdot p_{\mathcal{U}}(z) \cdot \sin(\beta_{\mathcal{U}}) + \tan\left(\gamma_{\mathcal{U}} - \frac{\theta_h}{2} + \lambda\right) \cdot p_{\mathcal{U}}(z) \cdot \cos(\beta_{\mathcal{U}}) + p_{\mathcal{U}}(y) \quad \text{(A.8)}$$

The UAVs' rotation possibilities around three axes often avoid that the ground area of the camera view is like a rectangular shape but similar to a trapezoid as shown in Figure A.1.

Based on the four boundary points, method *DetermineExploredNodes* shown in Listing A.11 calculates ground positions to which rays have to be send. The method first sets the space type of the node enclosing the UAV as free space. The system supports aging of data and therefore the exploration time is stored at the node, too. The next step is to sort the boundary

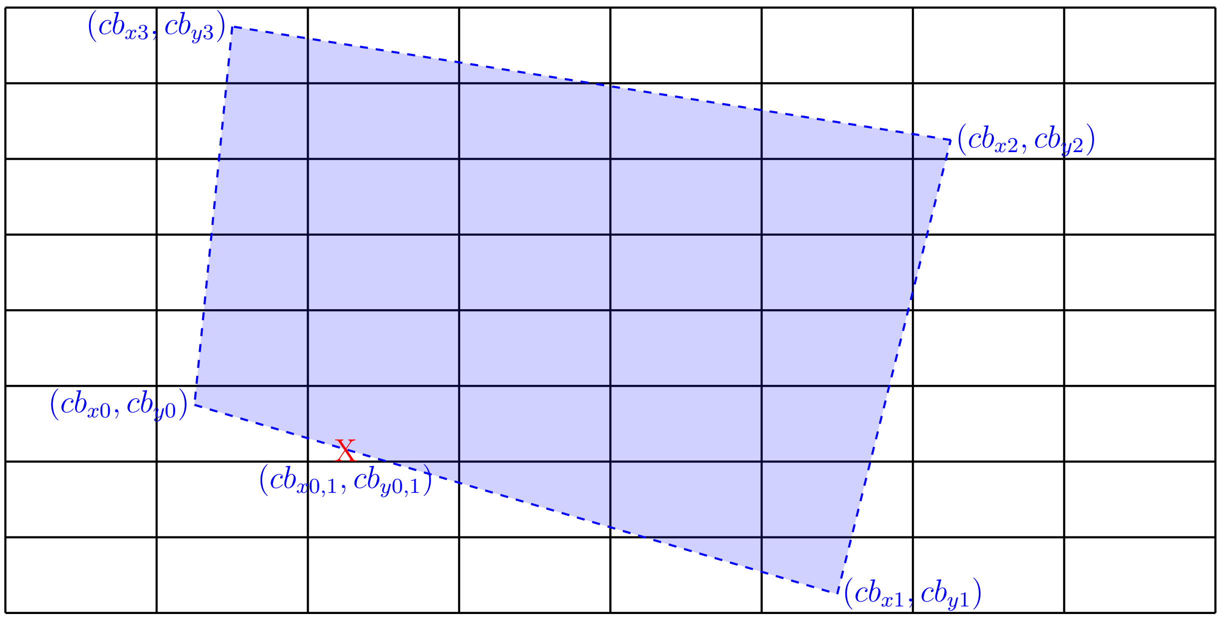


Figure A.1.: *Schematic illustration of an area at the terrain ground that might be explorable.*

points (cb_{xi}, cb_{yj}) in a specific way to ensure that the arrangement of the nodes is always the same independent of the yaw angle of the UAV.

The end position for the first ray is the first boundary point (cb_{x0}, cb_{y0}). This point is due to the sorting mentioned before always in the lower left related to $\mathcal{F}_{\mathcal{W}}$ as shown in Figure A.1. The second boundary point (cb_{x1}, cb_{y1}) is in the lower right of the terrain. The third boundary point (cb_{x2}, cb_{y2}) is the one in the upper right while the fourth boundary point (cb_{x3}, cb_{y3}) is in the upper left.

The method now moves in spiral form from one boundary point to another until the leaf at the center of the boundary positions is reached. A single movement step requires a movement in x - direction, as well as in y - direction and is computed as follows. To move, e. g., starting at (cb_{x0}, cb_{y0}) towards (cb_{x1}, cb_{y1}) requires a movement in x - direction equal to the x - dimension of a leaf at maximum resolution. Let $x(l_i)$ be the size of a leaf in x - direction at maximum resolution. Then

$$n = \frac{cb_{x1} - cb_{x0}}{x(l_i)} \tag{A.9}$$

movement steps are necessary. Based on the n steps computed using Equation A.9 the movement in y - direction y_m per step to compute new positions is computed by

$$y_m = \frac{cb_{y0} - cb_{y1}}{n}. \tag{A.10}$$

Computing the movement between the other boundary points is performed analogously. The next computed position $(cb_{x0,1}, cb_{y0,1})$—marked by a red X in Figure A.1—is finally given by

$$cb_{x0,1} = cb_{x0} + x(l_i) \tag{A.11}$$

$$cb_{y0,1} = cb_{y0} + y_m. \tag{A.12}$$

```
DetermineExploredNodes(position)
  uav_node.space_type ← $\mathcal{C}_{\mathcal{F}}$
  uav_node.exp_time ← now
  sortet_bounds ← SortCamBoundaries(cam_boundaries)

  ray_end ← sorted_bounds.x
  number_leaves ← distances / min_node_size

  step_distances ← distances / number_leaves
  exploring ← true

  while (exploring)
    exploring ← false
    for (i ← 0; i < number_leaves.x1; i++)
      SendRay()
      ray_end.x += min_node_size.x
      ray_end.y += step_distances.y1
      exploring = true

    number_leaves.x1 -= 2
    for (i ← 0; i < number_leaves.y1; i++)
      SendRay()
      ray_end.x += step_distances.x1
      ray_end.y -= min_node_size.y
      exploring = true

    number_leaves.y1 -= 2
    for (i ← 0; i < number_leaves.x2; i++)
      SendRay()
      ray_end.x -= min_node_size.x
      ray_end.y += step_distances.y2
      exploring = true

    number_leaves.x2 -= 2
    for (i ← 0; i < number_leaves.y2; i++)
      SendRay()
      ray_end.x += step_distances.x2
      ray_end.y += min_node_size.y
      exploring = true

    number_leaves.y2 -= 2
    SendRay()
```

Listing A.11: *Method for the determination of end positions for rays.*

Each computed position is used as an end position of a ray. Method *SendRay* presented in Listing A.12 is called to finally determine the newly explored nodes. It is similar to classical ray tracing approaches. Compared to classical ray tracing not every polygon of the environment is checked whether it is intersected by a ray. The rays intersect a single leaf at every position of the environment. Thus, it is only necessary to discretize the ray in a way that each intersected leaf can be determined.

First, the three-dimensional vector *new_position* from Listing A.12 is set to the UAV position. The ray always starts at the UAV position. Thereafter, a so called *node_diff* is computed. This is also a three-dimensional vector and it is initialized with the distance of the UAV to the end position of the ray. The end position has previously been computed in method *DetermineExploredNodes*. The vector *node_diff* is then divided by a step size, which is also a three-dimensional vector, on the one hand taking the z dimension of a leaf at maximum level. On the other hand, taking the corresponding x and y values one has to move within these dimensions in order to compute a point on the ray if the z - dimension is subtracted from the previous point. In general, the discretization of the rays works analogously to the method introduced before for the determination of the end positions of the rays.

The following while loop subtracts the *node_diff* vector continuously from the vector *new_-position*, to compute positions that lie on the ray until the ray exits the environment. Each leaf at maximum level enclosing a position computed this way, is determined to use the approach presented in method *FindEqualLevelNeighbor* from Listing A.3 to compute leaves at maximum level based on positions. The leaves are finally set to be newly explored until a node assigned to an occupied space is reached. Using the two termination conditions ensures that the *while (true)* loop terminates.

```
SendRay()
  new_position ← position
  node_diff ← position - ray_end
  node_diff /= step_size

  while (true)
    new_position -= node_diff
      if (new_position is outside the terrain)
        return

    new_explored ← GetMaxLevelLeafFromPosition(new_position)

    if (p_new_explored ∈ $\mathcal{C}_\mathcal{O}$)
      break

    new_explored.space_type ← $\mathcal{C}_\mathcal{F}$
    new_explored.last_visit ← now
```

Listing A.12: *Method to determine the newly explored leaves.*

The discretization of the ray using the *node_diff* vector is an approximation. Not every node intersected by a ray is computed this way. This is desired as only nodes for which the main volume lies inside the exploration area should be assigned to $\mathcal{C}_{\mathcal{E}}$. Basing the value on the z axis leads to an insufficient exploration when the flare angle of the camera is set to a value greater than 120°. In that case the *node_diff* vector should be based on the x and the y dimension. The UAVs considered in this thesis have a much lesser flare angle so this computation is sufficient for the presented work.

A.4. Combine Octree

The UAVs are able to successively explore the environment using the update octree algorithm presented in Appendix A.3. This algorithm creates new leaves at maximum level leading to an increasing number of unbound leaves. The unbound leaves are taken into account by the algorithms for potential field calculations presented in Appendix A.5 and Appendix A.6. An increasing number of unbound leaves results in increasing computation times for these algorithms. A combination of leaves is conducted by the algorithm *CombineTree* presented in Listing A.13 in order to cushion this effect.

The algorithm takes the root node as input and moves recursively through the entire octree. Leaves are combined and the recursion allows a combination of all possible leaves in one single run.

The algorithm is called recursively with each of the eight child nodes of the current node as input if the check fails. First, the leaf property of the node is checked again. The children of a node could have been combined such that the status of a non-leaf might has changed due to the recursion. Leaves containing a UAV are not combined. Another check considering the possibility to combine leaves is only performed for one leaf of each node. The seventh child node (cf. Figure 4.7) is chosen. Only one leaf is examined in order to avoid redundant checks. The seventh child node is used for this purpose as the recursion is executed for all single child nodes before the examination of the first leaf starts. Using another child node would cause the method to examine nodes combined before and thus non-existent at the time they are checked. Additionally, only leaves not encapsulating a UAV are combinable. This is checked, too.

The parent node is determined after the selection of a leaf to be a possible candidate for a combination. The siblings of the leaf are then also investigated using the parent node to easily access them. Only nodes assigned to the same space type can be combined. From this it follows, that all siblings of a node must be assigned to the same space type as the original node.

A further check is required as the system supports aging of data. Leaves assigned to $\mathcal{C}_{\mathcal{F}}$ can only be combined if the difference of the points in time they were explored (cf. Section 4.2.3) is below a given threshold. This threshold depends on the time, after which a newly explored leaf is considered unexplored again.

Child leaves of a node are combined if they passed the previously described checks. An approximation of the points in time the single leaves have been recently explored is conducted during a leaf combination. The parent node of the leaves to be combined is set to the arithmetic mean of the single points in time the leaves have been explored lastly.

```
CombineTree(node, child_number)
  if (node is not leaf)
    foreach (child)
      CombineTree(child, child_number)

  if (node is not leaf or child_number $\neq$ 7 or node encloses uav)
    return

  node $\leftarrow$ node.parent

  for (i $\leftarrow$ 0; i < 7; i++)
    if(node.child[i].type $\neq$ node.child[7].type or node.child[i])
      return

  if (node.child[7].type $\in \mathcal{C}_\mathcal{F}$ and aging is active)
    for (i $\leftarrow$ 0; i < 7; i++)
      if (node.child[7].exp_time - node.child[i].exp_time > threshold)
  return

    node.exp_time $\leftarrow$ 0
    for (i $\leftarrow$ 0; i < 8; i++)
      node.exp_time += node.child[i].exp_time
    node.exp_time /= 8

  node.ground_height = 0.f
  for (i $\leftarrow$ 0; i < 8; i++)
    if (node.child[i].ground_height > node.ground_height)
      node.ground_height $\leftarrow$ node.child[i].ground_height

  node.space_type $\leftarrow$ node.child[7].space_type
  if (node.space_type$\in \mathcal{C}_\mathcal{O}$)
    node.potential $\leftarrow$ 1.f

  for (i $\leftarrow$ 0; i < 8; i++)
    delete node.child[i]

  node.leaf $\leftarrow$ true
```

Listing A.13: *Algorithm for node combination to reduce the number of octree nodes.*

Each leaf stores its distance to the closest underlying environment. This is used to easily determine the nodes that enclose the favorite height of the UAV (cf. Section 4.3). The new leaf obtains the maximum distance of the child nodes. This ensures that the UAV selecting a leaf at its favorite altitude as next to be explored does not reduce its altitude in order to achieve a larger exploration area.

The new leaf is assigned to the space type of its child nodes and its potential value is bound to one if it contains occupied space in order to fulfill the Dirichlet Boundary Condition (cf. Section 5.3.1). Finally, the children of the node are deleted and the node becomes a leaf.

The algorithm presented in this appendix ensures that the system uses some of the advantages offered by an octree. It decreases the number of nodes by combining leaves with similar properties.

A.5. Harmonic Potential Field Calculations

The UAVs main duty is to fly from initial configurations $q_{\mathcal{I}}$ to target configurations $q_{\mathcal{G}}$ in order to process tasks and to explore environments. UAV flights are due to several reasons based on previously planned trajectories. Artificial potential field theory combined with harmonic function theory is used for the computation of such trajectories.

The algorithm presented in this appendix computes an approximation of a three-dimensional harmonic function. This harmonic function represents an artificial potential field. Trajectories can then be computed, using the descent gradient (cf. Chapter 5).

The algorithm performs a three step procedure as shown in the main method *CalculatePotentialField* presented in Listing A.14. First, some computations including the creation of the Dirichlet Boundary Condition (cf. Section 5.3.1) are conducted by the method *Precalculations* presented in Listing A.15. Afterwards, a first approximation of a harmonic function is computed by the method *FirstApproximation* presented in Listing A.16. This approximation is finally relaxed towards a harmonic function as presented in Listing A.17. The resulting potential field guides the UAVs to the target configurations $q_{\mathcal{G}}$ while it simultaneously ensures obstacle avoidance.

```
CalculatePotentialField(root_node, target)
  Precalculations(root_node, target)
  FirstApproximation(target)
  while(termination_conditions not fulfilled)
    Relaxation()
```

Listing A.14: *Main potential field calculation method.*

First, some preparations presented by method *Precalculations* from Listing A.15 are conducted. The method takes the root node as input and runs recursively through the entire octree. Only leaves or nodes further away than distance τ are relevant for potential field computations (cf. Section 5.3.4) and therefore all other nodes remain unconsidered.

A distinction of leaves takes place in order to create the Dirichlet Boundary Condition. Every leaf assigned to $\mathcal{C}_{\mathcal{O}}$ obtains a potential value of one and every leaf assigned to $\mathcal{C}_{\mathcal{G}}$ obtains a potential value of zero. These leaves are the so called bound leaves (cf. Section 5.3.1). Leaves enclosing other UAVs are also assigned to $\mathcal{C}_{\mathcal{O}}$.

Potential values need to be calculated for the remaining unbound leaves. The algorithm utilizes the fact that the octree is static during potential field computations. The most costly part of the relaxation is the consideration of the different leaf dimensions (cf. Section 5.3.3). Thus, another approximation performed by the algorithm is to take only the average distance of the neighboring leaves in each direction into account. First, the neighbor nodes in each of the six relevant directions are computed for each unbound leaf. Afterwards, the arithmetic mean of the distances to the neighboring nodes is computed for each direction. Thereafter, the same is conducted for the opposite directions needed by the relaxation equation. The relaxation equation includes a division by a combination of the distances as shown in Equation 5.18. All calculated values and the corresponding unbound leaves are finally stored in a list.

```
Precalculations(node, target)
  dist ← GetDistanceToTarget(node, target)

  if (node not leaf and dist < $\tau$)
    foreach (child of node)
      Precalculations(child, target)
    return

  switch(node.space_typ)
    case $\mathcal{C}_{\mathcal{O}}$:
      node.potential ← 1
    case $\mathcal{C}_{\mathcal{G}}$
      node.potential ← 0
    default:
      leaf_values.leaf ← leaf
      for (direction ← 0; direction < 6; direction++)
      leaf.neighbors ← FindNeighbors(direction)
        leaf_values.avg_dist[dir] ← CalcAgvDist(dir)
        leaf_values.opp_dist[dir] ← CalcOppDist(dir)

      leaf_values.div ← CalculateDivisor()
      leaf_list.push_back(leaf_values)
```

Listing A.15: *Method performing precalculations including Dirichlet Boundary Condition.*

After the computation of all necessary values and the introduction of the Dirichlet Boundary Condition, a list containing all relevant data has been created. The method *FirstApproximation* presented in Listing A.16 is then called and runs through the entire list. It computes a first approximation of the resulting harmonic function by computing single potential values. The bound values set by the method *Precalculations* are fixed during the complete computation of the potential field. Only the unbound values are stored in the list and thus only the potential values of these leaves are computed. The resulting potential field is the combination of the bound and unbound leaves.

```
FirstApproximation(target)
  foreach (leaf in leaf_list)
    dist ← GetDistanceToTarget(leaf,target)
    leaf.potential ← 1-(1/dist)
```

Listing A.16: *First approximation of the harmonic potential field.*

The last step of the potential field computation is the relaxation of the approximated values towards a harmonic function. For this purpose, method *Relaxation* presented in Listing A.17 is called in a while loop.

The method also uses the list created by method *Precalculations*. It runs through *leaf_list* and assigns each potential value to the arithmetic mean of the surrounding potential values. An approximation takes place and only the arithmetic mean of the potential values in each direction is considered. The method starts by summing up the single potential values in each direction.

The cumulative potential values are divided by the number of neighbors in the corresponding directions that reach the arithmetic mean value if neighbors in the current direction exist. The values are then multiplied by the average distance of the neighbors in the corresponding direction. The resulting potential values are multiplied with the relevant distances given by the current direction to compute the numerator of Equation 5.18.

```
Relaxation()
  foreach (leaf_values in leaf_list)
    leaf_values.leaf.potential ← 0.0

    for (direction ← 0; direction < 6; direction++)
      pre_potential[dir] ← 0.0
      foreach (neighbor in leaf_values.neighbors[direction])
        first_potential += neighbor.potential

      if(leaf_values.leaf.neighbors[dir].number > 0
        first_potential /= leaf_values.neighbors[direction].number
        pre_potential[dir] ← first_potential * leaf_values.avg_dist[dir]

    for (dir ← 0; direction < 6; direction += 2)
      leaf_values.leaf.potential += leaf_values.opp_dist[dir/2] * (
          pre_potential[dir] + pre_potential[dir+1])

    leaf_values.leaf.potential ← leaf_values.potential / leav_values.div
```

Listing A.17: *Relaxation of approximated values towards a harmonic potential field.*

The distance must be subtract out and thus the resulting potential values are divided by the denominator of Equation 5.18 to achieve the arithmetic mean. Finally, the resulting potential value must be assigned to the corresponding leaf. This way each leaf gets the arithmetic mean of the surrounding potential values, independent from the dimensions of its neighbors. This method approximates the potential field closer to a harmonic function with each call.

As mentioned before, it is not necessary to compute a perfect harmonic function to be able to plan feasible paths. Thus, method *CalculatePotentialField* presented in Listing A.14 implements termination conditions introduced in Section 5.3.4. The algorithm terminates and the descent gradient is used to compute a feasible path from $q_{\mathcal{I}}$ to $q_{\mathcal{G}}$ if one of these termination conditions is fulfilled.

A.6. Bifurcating Potential Field Calculations

Formation flights are possible beside single UAV flights from initial configurations $q_{\mathcal{I}}$ to target configurations $q_{\mathcal{G}}$ in order to explore the environment and to solve various tasks. They are achieved by an extension of the harmonic potential field for path planning computed by the algorithm presented in Appendix A.5.

The algorithm presented in this section computes a potential field for formation flights based on bifurcation theory and is superpositioned with the previously computed harmonic potential field.

The entry method *CalculateBifurcatingPotentialField* presented in Listing A.18 takes the position of the UAV, its next path point, the roll angle, and the pitch angle of the formation as input.

```
CalculateBifurcatingPotentialField(position, next position, $\gamma_{\mathcal{P}}$, $\delta_{\mathcal{P}}$, root)
  $\beta_{\mathcal{P}}$ ← CalculateYawAngle(position,next position)

  CalculatePotentialValues(root)
```

Listing A.18: *Entry method for the calculation of a bifurcating potential field.*

First, the yaw angle $\beta_{\mathcal{P}}$ of the resulting formation is computed using the current route point and the next route point of the formation. The angle is used to rotate the potential field in order to rotate the formation such that the formation shape is prepared (cf. Section 6.4.3). Afterwards, method *CalculatePotentialValues* presented in Listing A.19 is called with the root node of the octree, as well as parameters v (cf. Equation 6.6) and w (cf. Equation 6.13)and the target position as input for the computation of the resulting potential values.

The method *CalculatePotentialValues* from Listing A.19 runs recursively through the entire octree. Only leaves or nodes further away than distance τ are taken into account for potential field computations (cf. Section 5.3.4). Additionally, the Dirichlet Boundary Condition created by method *Precalculations* presented in Listing A.15 for the computation of the path planning potential field is respected by neglecting all leaves assigned to $\mathcal{C}_{\mathcal{O}}$. First, the Euclidean distance τ from the node to the target node is computed for the remaining unbound leaves.

To be able to rotate a formation when necessary, a rotation of the distance around the target position is also conducted.

```
CalculatePotentialValues(node, v, w, target)
  dist ← GetDistanceToTarget(node, target)

  if (node not leaf and dist < $\tau$)
    foreach (child)
      CalculatePotentialValues(child)
    return

  if (node space type $\in \mathcal{C}_{\mathcal{O}}$)
    return

  $\tau$ ← ComputeDistances()
  if (rotation is necessary)
    $\tau$ ← RotateDistances()

  $\bar{\tau}$ ← GetNormalizedDistances($\tau$)

  $u \leftarrow (|\mu| + \alpha) \cdot (a + b + c)$

  bifurcation_potential ← $\mu(a\bar{\tau}_x^2 + b\bar{\tau}_y^2 + c\bar{\tau}_z^2)$
  bifurcation_potential ← bifurcating_potential + $\alpha(a\bar{\tau}_x^4 + b\bar{\tau}_y^4 + c\bar{\tau}_z^4)$
  bifurcation_potential ← bifurcating_potential / u
  bifurcation_potential ← bifurcating_potential - v

  if (bifurcation_potential < 0)
    node.potential ← node.potential · 0.5 + bifurcation_potential
```

Listing A.19: *Method for the calculation of the single bifurcating potential values.*

A potential value for each leaf is now computed using Equation 6.6 as shown in Listing A.19. Finally, the bifurcation value is superpositioned with the previously computed harmonic potential value using the case differentiation as described in Section 6.3.2. This leads to the resulting superpositioned potential field for formation flights.

A.7. Task Creation

Task creation is performed when at least one of the three situations described in Section 7.4.1 occurs. The algorithm mainly checks whether at least one of the “if queries”, presented in Listing A.20 holds. A refueling task is created and afterwards allocated when the fuel level of the UAV is below the threshold defined in Equation 7.13.

A UAV tries to reestablish communication and simultaneously flies to the next landing area if it does not receive messages for a given time span, neither from the *Control Station* nor from other UAVs it creates a reconnect task.

The last check considers the creation of formations. A UAV creates a new task requiring several UAVs to explore the area in formation if it processes the single UAV exploration task and if it selects a large area, as defined in Equation 7.14, as next to be explored.

```
CreateNewTask()
  if(fuel_level < fuel_threshold)
    CreateRefuelingTask()
    AllocateTask()
  if(time_since_last_message < time_threshold)
    CreateReconnectTask()
    AllocateTask()
  if(current_task == SingleExploration)
    area_size $\leftarrow$ ComputeSizeOfAreaToBeExploredNext()
    if(area_size $\geq$ CameraArea$\cdot 4n$)
      CreateMultiExplorationTask()
      AllocateTask()
```

Listing A.20: *Algorithm for task creation in order to increase efficiency and to avoid damage.*

A.8. Task Allocation

The last algorithm presented in this appendix is responsible for task allocation and shown in Listing A.21. It is called continuously in order to perform a reallocation if the fulfillment of tasks by other UAVs is delayed or if a new task more beneficial than the current one has been received.

```
AllocateTask()
  foreach (task in task_list)
    ComputeBidValue()

  while(search_for_task)
    next_task $\leftarrow$ GetTaskWithHighestBidValue()
    if (next_task.type == SingleExploration)
      search_for_task $\leftarrow$ false
    else if (next_task.own_bid_value $\leq$ next_task.foreign_bid_value)
      next_task.own_bid_value $\leftarrow$ 0
    else
      search_for_task $\leftarrow$ false

  If (next_task.type $\neq$ Refueling $\neq$ Reconnect $\neq$ SingleExploration)
    SendBidValueToOtherUAVs()
```

Listing A.21: *Task allocation algorithm.*

The algorithm first runs through the entire task list and calculates a bid value for each task based on the current configuration of the UAV as shown in Equation 7.18. The following while loop then selects the task with the highest bid value. The UAV starts to explore the environment on its own if the most beneficial task is exploratory navigation. A check is performed whether the task to be allocated is not exploratory navigation and whether it is currently processed by another UAV. In this case, a bid value from the UAV processing the task has been received before. If this bid value is higher than the one the UAV computed, it sets its own bid value to zero and selects the task which has then the highest bid value. A UAV takes over a task and informs the other UAVs if the bid value the UAV computed is higher than the one of the UAV currently processing the task.

Refueling and reconnection tasks will always be executed first and the bid values computed for these tasks are not send to other UAVs.

Appendix B

Definitions

Definition B.1. Mean - Value property: *Let Ω be a domain in Euclidean space possessing finite measure. Let the complement of Ω possess non-empty interior. Assume now that a point $p \in \Omega$ exists such that for every harmonic function ϕ in Ω and integrable over Ω the mean value of ϕ over Ω equals $\phi(p)$. Then Ω is a sphere with center at p [157].*

Definition B.2. Min - Max Principle: *A harmonic function $f(x, y)$ defined on a closed domain Ω takes its minimum and maximum values on the boundaries [158].*

Definition B.3. Uniques Principle: *If $f(x, y)$ and $g(x, y)$ are harmonic functions defined on a closed domain Ω such that $f(x, y) = g(x, y)$ for every bound point, then $\forall x, y\colon f(x, y) = g(x, y)$ [34].*

Definition B.4. Analytic: *An analytic function is an infinitely differentiable function such that the Taylor series at any point x_0 in its domain Ω*

$$T(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$$

converges to $f(x)$ for x in a neighborhood of x_0.

Definition B.5. Completeness: *If a path ρ_d from $q_{\mathcal{I}}$ to $q_{\mathcal{G}}$ exists and the configuration space contains a finite number of configurations, then a discrete path ρ_d is always computed by a complete planner. If such a path does not exist, the planner returns an error.*

Definition B.6. Sound: *A path planner is sound if it always computes a discrete path ρ_d, leading to the target configuration $q_{\mathcal{G}}$ if such a path exists. If not, the planner returns an error.*

Definition B.7. Dirichlet Boundary Condition: *When imposed on an ordinary or a partial differential equation, it specifies the values a solution needs to take on the boundary of the domain. The question of finding solutions to such equations is known as the Dirichlet Problem.*

Definition B.8. Dirichlet problem: *It is the problem of finding a function, which solves a specified partial differential equation in the interior of a given region that takes prescribed values on the boundary of the region.*

Definition B.9. Equilibrium: *Let l_i be a leaf with potential value ϕ_{l_i}. The leaf l_i is an equilibrium or minimum if no neighboring leaf $l_j \colon (l_i, l_j) \in \mathcal{N}, i \neq j$ with a potential value $\phi_{l_j} < \phi_{l_i}$ exists.*

Definition B.10. Bivariate function: *A function is called bivarite if it contains two variables, e. g., $f(x, y) = x + y^2$*

Definition B.11. Finite difference approximation: *Following the definition from Smith [207] applying Taylor's theorem to any smooth function f results in:*

$$\begin{aligned} f(x+h) &= f(x) + f'(x)h + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(x) + O(h^4), \\ f(x-h) &= f(x) - f'(x)h + \frac{1}{2}h^2 f''(x) - \frac{1}{6}h^3 f'''(x) + O(h^4). \end{aligned} \tag{B.1}$$

It is deduced that

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h). \tag{B.2}$$

Hence, $\frac{f(x+h)-f(x)}{h}$ is a first order forward finite difference approximation to $f'(x)$. It is called first order as the error term known as truncation error is $O(h)$. Subtracting the two Taylor expansions results in:

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2). \tag{B.3}$$

This is the second order finite difference.

Analogous, approximations for higher derivatives can be found. A second order central finite difference for the second derivative is:

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^4). \tag{B.4}$$

Nomenclature

$-\nabla$	Descent gradient, page 34
$\#_{\varnothing}(l_i)$	Average number of leaves, page 154
$\#_{\varnothing}(l_{un})$	Average number of unbound leaves, page 154
$\#_{computations}$	Number of computations, page 154
$\#_{max}(l_i)$	Maximum number of leaves, page 154
$\#_{max}(l_{un})$	Maximum number of unbound leaves, page 154
$\#_{min}(l_i)$	Minimum number of leaves, page 154
$\#_{min}(l_{un})$	Minimum number of unbound leaves, page 154
$\#l_{un}$	Number of unbound leaves, page 154
α	Amplitude of a bifurcation function, page 45
$\bar{\tau}$	Normalized Euclidean distance $0 \leq \bar{\tau} < 1$, page 134
$\bar{\tau}_{\mathcal{T}}$	Normalized Euclidean distance between a UAV and a task, page 134
$\bar{\tau}_x$	Normalized distance $0 \leq \bar{\tau}_x < 1$ in x - direction, see equation (6.4), page 108
$\bar{\tau}_y$	Normalized distance $0 \leq \bar{\tau}_y < 1$ in y - direction, see equation (6.4), page 108
$\bar{\tau}_z$	Normalized distance $0 \leq \bar{\tau}_z < 1$ in z - direction, see equation (6.4), page 108

$\beta_{\mathcal{P}}$	Yaw angle of a formation, page 118
$\beta_{\mathcal{U}}$	Yaw angle of a UAV, page 77
$\delta_{\mathcal{P}}$	Roll angle of a formation, page 118
$\delta_{\mathcal{U}}$	Roll angle of a UAV, page 77
$\eta_{\mathcal{T}}$	Equipment needed to process a task, page 134
$\eta_{\mathcal{U}}$	Equipment of a UAV, page 130
$\gamma_{\mathcal{P}}$	Pitch angle of a formation, page 118
$\gamma_{\mathcal{U}}$	Pitch angle of a UAV, page 77
λ	Camera angle of the bird eye camera, page 78
$\mathcal{B}_{\mathcal{T}}$	The bid value for a task $\mathcal{T}_i$, see equation (7.18), page 144
$\mathcal{C}_{\mathcal{E}}$	Explored space, page 65
$\mathcal{C}_{\mathcal{F}}$	Free space, page 31
$\mathcal{C}_{\mathcal{G}}$	Target space, page 65
$\mathcal{C}_{\mathcal{O}}$	Occupied space, see equation (3.3), page 30
$\mathcal{C}_{\mathcal{UK}}$	Unknown space, page 65
$\mathcal{C}_{\mathcal{UX}}$	Unexplored space, page 65
$\mathcal{C}$	Configuration space, see equation (3.1), page 29
$\mathcal{D}$	Neighboring direction related to the octree, page 33
$\mathcal{E}_{\mathcal{P}}$	Current exploration rate using UAV formations, see equation (7.2), page 131
$\mathcal{E}_{\mathcal{U}}$	Current exploration rate using single UAVs, see equation (7.2), page 131
$\mathcal{F}_{\mathcal{W}}$	Cartesian coordinate system, page 30
$\mathcal{N}$	The set of all pairs of neighboring nodes l_i and l_j, page 33

$\mathcal{O}_j$	A three-dimensional obstacle, page 29
$\mathcal{P}$	A UAV formation, page 104
$\mathcal{R}_{\mathcal{U}}$	Role of a UAV, page 29
$\mathcal{T}$	A single task, page 131
$\mathcal{T}_{\mathcal{U}}$	The task currently executed by a UAV, page 29
$\mathcal{U}$	A single UAV, page 29
$\mathcal{W}$	A terrain, see equation (3.5), page 33
e	Euler number, page 37
μ	Bifurcation parameter, page 45
∇	Gradient, page 88
∇^2	Laplace operator, see equation (3.10), page 37
Ω	Closed domain on which the ϕ^h are given, page 37
ω	Limit set, page 54
Ω_c	Compact set, page 54
ϕ	A potential value, page 28
$\phi^h_{\mathcal{G}}$	Bound potential value of target areas, page 39
$\phi^h_{\mathcal{O}}$	Bound potential value of occupied areas, page 39
$\phi_{x,y,z}$	Superpositioned potential field, see equation (6.13), page 110
$\phi^b_{x,y,z}$	Three-dimensional bifurcation potential value, see equation (6.6), page 109
$\phi^h_{x,y,z}$	Three-dimensional harmonic potential value, see equation (5.5), page 88
ρ	A path, page 31
ρ_d	A discrete path, see equation (3.6), page 34

$\rho_{\mathcal{U}_i}$	Path of a specific UAV, page 42
τ_i	Euclidean distance in direction i, page 92
$\tau_{x,y,z}$	Three-dimensional Euclidean distance, see equation (5.3), page 88
θ_d	Flare angle of the bird's eye camera, page 78
θ_f	Flare angle of the front camera, page 78
$\varnothing t_{diff}$	Time difference between two potential field computations, page 154
$\varnothing_{depth}$	Average iteration depth, page 155
a	Weight value for the bifurcating x - dimension, see equation (6.5), page 108
$A_{\mathcal{T}}$	Age of a task, page 134
b	Weight value for the bifurcating y - dimension, see equation (6.5), page 108
c	Weight value for the bifurcating z - dimension, see equation (6.5), page 108
d	Diagonal of the environment, page 78
$I_{\mathcal{T}}$	Importance of a task, page 134
J	Jacobian matrix, see equation (3.25), page 55
l_i	Single leaf of the octree, page 32
M	Positively invariant set, see equation (3.22), page 54
$Ma_{\mathcal{T}}$	Maximum number of UAVs able to perform a task simultaneously, page 134
$Mi_{\mathcal{T}}$	Minimum number of UAVs needed to perform a task, page 134
$N_{\mathcal{T}}$	Number of UAVs able to process a task, page 134
$O_{\mathcal{T}}$	Current number of UAVs processing a task simultaneously, page 134
p	Point in $\mathcal{F}_{\mathcal{W}}$, page 29
$p_{\mathcal{G}}$	Target position of a UAV in $\mathcal{F}_{\mathcal{W}}$, page 29

$p_{\mathcal{I}}$	Initial position of a UAV in $\mathcal{F}_{\mathcal{W}}$, page 29
$p_{\mathcal{U}}$	Current position of a UAV in $\mathcal{F}_{\mathcal{W}}$, page 29
$q_{\mathcal{O}}$	An obstacle configuration, page 39
$q_{\mathcal{U}}$	Current configuration of a UAV, page 56
q_i	A single configuration $q \in \mathcal{C}$, page 29
$q_{\mathcal{G}}$	A target configuration $q_{\mathcal{G}} \in \mathcal{C}_{\mathcal{G}}$, page 29
$q_{\mathcal{I}}$	Initial configuration of a UAV $q_{\mathcal{I}} \in \mathcal{C}$, page 29
$q_{\mathcal{U}}$	UAV configuration $q_{\mathcal{U}} \in \mathcal{C}$, page 29
S	Set, see equation (3.23), page 54
$t_{\varnothing}$	Average computation duration, page 154
t_i	Single computation duration, page 154
$t_{complete}$	Complete computation duration, page 154
t_{max}	Maximum computation duration, page 154
t_{min}	Minimum computation duration, page 154
u	Divisor for the bifurcating potential values, see equation (6.6), page 109
V	A continuous function, see equation (3.17), page 53
v	Shifting value for the bifurcating potential values, see equation (6.6), page 109
x^*	Equilibrium position, page 51

List of Figures

List of Tables

Own Publications Related to the Thesis

[1] N. Prüßner, W. Richert, C. Rasche, and A. Rettberg, "Action selection for multi-smdp based robots with dynamically priotized goals," in *Proceedings of the IADIS International Conference Intelligent Systems and Agents 2010 (part of MCCSIS 2010)*, Freiburg, Germany, 29.–31. July 2010, pp. 75–82, [Online Available; Condition from 03. June 2013]. http://www.iadisportal.org/digital-library/action-selection-for-multi-smdp-based-robots-with-dynamically-prioritized-goals.

[2] C. Rasche, A. Jungmann, T. Schierbaum, B. Werdehausen, and B. Kleinjohann, "Towards hierarchical self-optimization in autonomous groups of mobile robots," in *Proceedings of the 10th IEEE International Conference on Industrial Informatics*, ser. INDIN 2012, IEEE. IEEE Xplore, 25.–27. July 2012, pp. 1098–1103, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1109/INDIN.2012.6301137.

[3] C. Rasche, C. Stern, W. Richert, L. Kleinjohann, and B. Kleinjohann, "Combining autonomous exploration, goal-oriented coordination and task allocation in multi-uav scenarios," in *Proceedings of the 2010 Sixth International Conference on Autonomic and Autonomous Systems (ICAS 2010)*, ser. ICAS '10, IARIA. Cancun, Mexico: IEEE Computer Society, 7.–13. March 2010, pp. 52–57, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1109/ICAS.2010.16.

[4] C. Rasche, C. Stern, L. Kleinjohann, and B. Kleinjohann, "A distributed multi-uav path planning approach for 3d environments," in *Proceedings of the 5th International Conference on Automation, Robotics and Applications (ICARA 2011)*. Wellington, New Zealand: IEEE Xplore, 6.–8. December 2011, pp. 7–12, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1109/ICARA.2011.6144847.

[5] C. Rasche, C. Stern, L. Kleinjohann, and B. Kleinjohann, "Coordinated exploration and goal-oriented path planning using multiple uavs," *International Journal On Advances in Software*, vol. 3, no. 3&4, pp. 351–370, April 2011, [Online Available; Condition from 03. June 2013]. http://www.thinkmind.org/index.php?view=article&articleid=soft_v3_n34_2010_3.

[6] C. Rasche, C. Stern, L. Kleinjohann, and B. Kleinjohann, “Role-based path planning and task allocation with exploration tradeoff for uavs,” in *Proceedings of the 11th International Conference on Control Automation Robotics Vision (ICARCV 2010).* Singapore: IEEE Xplore, 7.–10. December 2010, pp. 417–422, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1109/ICARCV.2010.5707317.

[7] C. Stern, C. Rasche, L. Kleinjohann, and B. Kleinjohann, “Efficient alignment of aerial images based on virtual forces,” in *Proceedings of the Eighth International Conference on Autonomic and Autonomous Systems*, ser. ICAS 2012, IARIA. St. Maarten, Netherlands Antilles: IEEE Xplore, 25.–30. March 2012, pp. 93–98, [Online Available; Condition from 03. June 2013]. http://www.thinkmind.org/index.php?view=article&articleid=icas_2012_4_40_20110.

[8] C. Rasche, C. Stern, L. Kleinjohann, and B. Kleinjohann, “A 3d path planning approach extended by bifurcation theory for formation flights,” in *Recent Advances in Robotics and Automation*, ser. Studies in Computational Intelligence, G. Sen Gupta, D. Bailey, S. Demidenko, and D. Carnegie, Eds. Springer Berlin Heidelberg, 2013, vol. 480, pp. 103–113, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1007/978-3-642-37387-9_8.

[9] C. Stern, C. Rasche, L. Kleinjohann, and B. Kleinjohann, “Towards using virtual forces for image registration,” in *Proceedings of the 5th International Conference on Automation, Robotics and Applications*, ser. ICARA 2011, Wellington, New Zealand, 6.–8. December 2011, pp. 447 –452, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1109/ICARA.2011.6144925.

[10] C. Stern, C. Rasche, L. Kleinjohann, and B. Kleinjohann, “Evaluating quality of online image registration of aerial images using virtual forces,” in *Proceedings of the 16th International Conference on Image Processing, Computer Vision, & Pattern Recognition*, ser. IPCV 2012, Las Vegas, Nevada, USA, 16.–19. July 2012.

Bibliography

[11] M. Dorigo, V. Maniezzo, A. Colorni, and V. Maniezzo, “Positive feedback as a search strategy,” Politecnico di Milano, Tech. Rep., 1991, [Online Available; Condition from 03. June 2013]. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.52.6342.

[12] M. Dorigo, V. Maniezzo, and A. Colorni, “Ant system: optimization by a colony of cooperating agents,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 1, pp. 29–41, February 1996, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1109/3477.484436.

[13] D. Karaboga and B. Akay, “A survey: algorithms simulating bee swarm intelligence,” *Artificial Intelligence Review*, vol. 31, no. 1, pp. 61–85, 2009, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1007/s10462-009-9127-4.

[14] N. Xiong, J. He, Y. Yang, Y. He, T. Kim, and C. Lin, “A survey on decentralized flocking schemes for a set of autonomous mobile robots,” *Journal of Communications*, vol. 5, no. 1, 2010, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.4304/jcm.5.1.31-38.

[15] K. Kaneko, F. Kanehiro, M. Morisawa, K. Akachi, G. Miyamori, A. Hayashi, and N. Kanehira, “Humanoid robot hrp-4 - humanoid robotics platform with lightweight and slim body,” in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, September 2011, pp. 4400–4407, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/IROS.2011.6094465.

[16] N. J. Mlot, C. A. Tovey, and D. L. Hu, “Fire ants self-assemble into waterproof rafts to survive floods,” *Proceedings of the National Academy of Science*, vol. 108, pp. 7669–7673, May 2011, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1073/pnas.1016658108.

[17] E. Sahin, S. Girgin, L. Bayindir, and A. E. Turgut, "Swarm robotics," in *Swarm Intelligence: Introduction and Applications.* Springer, 2008, pp. 87–100, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1007/978-3-540-74089-6_3.

[18] S. J. Park, T. K. Yeu, S. M. Yoon, S. Hong, and K. Y. Sung, "A study of sweeping coverage path planning method for deep-sea manganese nodule mining robot," in *OCEANS 2011*, September 2011, pp. 1–5, [Online Available; Condition from 03. June 2012]. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6107109.

[19] K. Alam, T. Ray, and S. Anavatti, "A new robust design optimization approach for unmanned underwater vehicle design," *Journal of Engineering for the Maritime Environment*, February 2012, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1177/1475090211435450.

[20] M. Lewis, M. Bunting, B. Salemi, and H. Hoffmann, "Toward ultra high speed locomotors: Design and test of a cheetah robot hind limb," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, May 2011, pp. 1990–1996, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/ICRA.2011.5979812.

[21] J. de Best, R. van de Molengraft, and M. Steinbuch, "A novel ball handling mechanism for the robocup middle size league," *Mechatronics*, vol. 21, no. 2, pp. 469–478, 2011, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1016/j.mechatronics.2010.04.001.

[22] M. Dorigo, D. Floreano, L. M. Gambardella, F. Mondada, S. Nolfi, T. Baaboura, M. Birattari, M. Bonani, M. Brambilla, A. Brutschy, D. Burnier, A. Campo, A. L. Christensen, A. Decugníere, G. A. D. Caro, F. Ducatelle, E. Ferrante, A. F̈oster, J. M. Gonzales, J. Guzzi, V. Longchamp, S. Magnenat, N. Mathews, M. M. de Oca, R. O'Grady, C. Pinciroli, G. Pini, P. Ŕetornaz, J. Roberts, V. Sperati, T. Stirling, A. Stranieri, T. S̈tutzle, V. Trianni, E. Tuci, A. E. Turgut, and F. Vaussard, "Swarmanoid: a novel concept for the study of heterogeneous robotic swarms," Brussels Belgium Tech., Tech. Rep., July 2011, [Online Available; Condition from 03. June 2012]. http://www.idsia.ch/~gianni/Papers/Swarmanoid-techrep.pdf.

[23] Parrot S. A., "Parrot ar drone," 2013, [Online Available; Condition from 03. June 2012]. http://www.parrot.com.

[24] Draganfly Innovations Inc., "Draganfly innovations inc." 2013, [Online Available; Condition from 03. June 2012]. http://www.draganfly.com/.

[25] Microdrones GmbH, "Microdrones uav," 2013, [Online Available; Condition from 03. June 2012]. http://www.microdrones.com.

[26] Parrot S. A., “Parrot ar drone,” 2012, [Online Available; Condition from 03. June 2012]. http://ardrone.parrot.com/press-photos.

[27] Gizmodo, “Draganflyer x8: The dreamboat uav,” 2012, [Online Available; Condition from 03. June 2012]. http://gizmodo.com/5619680/draganflyer-x8-the-dreamiest-uav.

[28] ACRE - Surveying Solutions, “Microdrones en kenya,” 2012, [Online Available; Condition from 03. June 2012]. http://www.grupoacre.com/uavs/noticias/ver/microdrones-en-kenia.

[29] M. Pitzke, “Interview with defense expert p. w. singer,” 2010, [Online Available; Condition from 03. June 2012]. http://www.spiegel.de/international/world/interview-with-defense-expert-p-w-singer-the-soldiers-call-it-war-porn-a-682852.html.

[30] E. Winsberg, “Drones, predators and reapers: The ethics of unmanned armed aircraft,” September 2009, in The Philosopher’s Eye [Online Available; Condition from 03. June 2012]. http://thephilosopherseye.com/2011/04/22/drones-predators-and-reapers-the-ethics-of-unmanned-armed-aircraft/.

[31] J. V. Gardner and A. A. Armstrong, “The mariana trench: A new view based on multibeam echosounding,” *American Geophysical Union*, 2011.

[32] R. L. Taylor, M. T. Zuber, D. H. Lehman, and T. L. Hoffman, “Managing grail: Getting to launch on cost, on schedule, and on spec,” in *Proceedings of the IEEE Aerospace Conference*, March 2012, pp. 1–10, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/AERO.2012.6187421.

[33] M. Trincavelli, M. Reggente, S. Coradeschi, A. Loutfi, H. Ishida, and A. J. Lilienthal, “Towards environmental monitoring with mobile robots,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, September 2008, pp. 2210–2215, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/IROS.2008.4650755.

[34] C. Connolly, J. B. Burns, and R. Weiss, “Path planning using laplace’s equation,” in *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, vol. 3, May 1990, pp. 2102–2106, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1109/ROBOT.1990.126315.

[35] H. Kielhöfer, *Bifurcation Theory: An Introduction with Applications to Partial Differential Equations*, 2nd ed. Springer New York, 2011.

[36] J. L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels, “The self-organizing exploratory pattern of the argentine ant,” *Journal of Insect Behavior*, vol. 3, no. 2, pp. 159–168, 1990, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1007/BF01417909.

[37] F. Heppner and U. Grenander, “A stochastic nonlinear model for coordinated bird flocks,” in *The Ubiquity of Chaos*, S. Krasner, Ed. AAAS Publications, 1990, pp. 233–238.

[38] E. Bonabeau, A. Sobkowski, G. Theraulaz, and J. L. Deneubourg, “Adaptive task allocation inspired by a model of division of labor in social insects,” Santa Fe Institute, Working Papers, 1998, [Online Available; Condition from 03. June 2012]. http://EconPapers.repec.org/RePEc:wop:safiwp:98-01-004.

[39] S. Camazine, J. L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraula, and E. Bonabeau, *Self-Organization in Biological Systems.* Princeton University Press, August 2003.

[40] T. H. Labella, M. Dorigo, and J. L. Deneubourg, “Division of labor in a group of robots inspired by ants’ foraging behavior,” *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 1, no. 1, pp. 4–25, September 2006, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1145/1152934.1152936.

[41] M. Dorigo and C. Blum, “Ant colony optimization theory: A survey,” *Theoretical Computer Science*, vol. 344, no. 2–3, pp. 243–278, 2005, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1016/j.tcs.2005.05.020.

[42] Vehicle Passion, “Schools of fish may hold secret to solve traffic woes,” 2011, [Online Available; Condition from 03. June 2012]. http://www.vehiclepassion.com/2011/11/10/schools-of-fish-may-hold-secret-to-solve-traffic-woes/.

[43] Cracked Camera, “Flock of birds, san diego,” 2011, [Online Available; Condition from 03. June 2012]. http://www.crackedcamera.com/flock-of-birds-san-diego-ca/.

[44] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, November/December 1995, pp. 1942–1948, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1109/ICNN.1995.488968.

[45] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems.* Oxford University Press, October 1999.

[46] W. Richert, “Learning and imitation in heterogeneous robot groups,” *PhD thesis*, December 2009, [Online Available; Condition from 03. June 2013]. http://d-nb.info/1000897915/34.

[47] A. G. Barto, R. S. Sutton, and P. S. Brouwer, “Associative search network: A reinforcement learning associative memory,” *Biological cybernetics*, vol. 40, no. 3, pp. 201–211, 1981, [Online Available; Condition from 19. September 2012]. http://dx.doi.org/10.1007/BF00453370.

[48] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[49] E. Maehle, W. Brockmann, K. E. Grosspietsch, A. El Sayed Auf, B. Jakimovski, S. Krannich, M. Litza, R. Maas, and A. Al-Homsy, “Application of the organic robot control architecture orca to the six-legged walking robot oscar,” in *Organic Computing – A Paradigm Shift for Complex Systems*, ser. Autonomic Systems, C. Müller-Schloer, H. Schmeck, and T. Ungerer, Eds. University of Lübeck, Lübeck, Germany: Springer Basel, 2011, vol. 1, pp. 517–530, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/978-3-0348-0130-0_34.

[50] J. Randløv and P. Alstrøm, “Learning to drive a bicycle using reinforcement learning and shaping,” 1998, [Online Available; Condition from 03. June 2012]. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.52.3038.

[51] G. Tesauro, “Practical issues in temporal difference learning,” in *Machine Learning*. Springer Netherlands, 1992, vol. 8, pp. 257–277, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/BF00992697.

[52] G. Tesauro, “Temporal difference learning and td-gammon,” *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, March 1995, [Online Available; Condition from 03. June 2012]. http://doi.acm.org/10.1145/203330.203343.

[53] R. H. Crites, “Large-scale dynamic optimization using teams of reinforcement learning agents,” Ph.D. dissertation, Graduate School of the University of Massachusetts Amherest, September 1996.

[54] R. H. Crites and A. Barto, “Improving elevator performance using reinforcement learning,” in *Advances in Neural Information Processing Systems 8*. MIT Press, 1996, pp. 1017–1023, [Online Available; Condition from 03. June 2012]. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.5519.

[55] D. Ferrucci, “Build watson: an overview of deepqa for the jeopardy! challenge,” in *Proceedings of the 19th international conference on Parallel architectures and compilation techniques*, ser. PACT ’10. New York, NY, USA: ACM, 2010, pp. 1–2, [Online Available; Condition from 03. June 2012]. http://doi.acm.org/10.1145/1854273.1854275.

[56] F. H. Hsu, *Behind Deep Blue: Building the Computer that Defeated the World Chess Champion*. Princeton University Press, September 2002.

[57] M. Newborn, *Deep Blue*, 1st ed. Springer, December 2002.

[58] C. W. Reynolds, “Flocks, herds and schools: A distributed behavioral model,” in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH ’87. New York, NY, USA: ACM, August 1987, pp. 25–34, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1145/37401.37406.

[59] W. J. Crowther, “Flocking of autonomous unmanned air vehicles,” *The Aeronautical Journal*, vol. 107, no. 1068, pp. 99–110, 2003.

[60] W. J. Crowther, "Rule-based guidance for flight vehicle flocking," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 218, no. 2, pp. 111–124, 2004, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1243/0954410041322005.

[61] R. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. 2, no. 1, pp. 14–23, March 1986, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1109/JRA.1986.1087032.

[62] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *IJRR*, vol. 5, no. 1, pp. 90–98, 1986, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1177/027836498600500106.

[63] R. Volpe and P. Khosla, "Manipulator control with superquadric artificial potential functions: theory and experiments," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 20, no. 6, pp. 1423–1436, November/December 1990, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1109/21.61211.

[64] S. S. Ge and Y. J. Cui, "Dynamic motion planning for mobile robots using potential field method," *Autonomous Robots*, vol. 13, no. 3, pp. 207–222, 2002, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1023/A:1020564024509.

[65] B. Nguyen, Y.-L. C., D. Tung, H. Chung, J. Zhipu, S. Ling, D. Marthaler, A. Bertozzi, and R. M. Murray, "Virtual attractive-repulsive potentials for cooperative control of second order dynamic vehicles on the caltech mvwt," in *Proceedings of the 2005 American Control Conference*, June 2005, pp. 1084 – 1089 vol. 2, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1109/ACC.2005.1470105.

[66] W. H. Huang, B. R. Fajen, J. R. Fink, and W. H. Warren, "Visual navigation and obstacle avoidance using a steering potential function," *Robotics and Autonomous Systems*, vol. 54, no. 4, pp. 288–299, 2006, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1016/j.robot.2005.11.004.

[67] J. H. Reif and H. Wang, "Social potential fields: A distributed behavioral control for autonomous robots," *Robotics and Autonomous Systems*, vol. 27, no. 3, pp. 171–194, 1999, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1016/S0921-8890(99)00004-4.

[68] D. E. Chang, S. C. Shadden, J. E. Marsden, and R. Olfati-Saber, "Collision avoidance for multiple agent systems," in *Proceedings of the 42nd IEEE Conference on Decision and Control*, vol. 1, December 2003, pp. 539–543, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1109/CDC.2003.1272619.

[69] P. Ogren, E. Fiorelli, and N. E. Leonard, “Cooperative control of mobile sensor networks:adaptive gradient climbing in a distributed environment,” *IEEE Transactions on Automatic Control*, vol. 49, no. 8, pp. 1292–1302, August 2004, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1109/TAC.2004.832203.

[70] M. R. D’Orsogna, Y. L. Chuang, A. L. Bertozzi, and L. S. Chayes, “Self-propelled particles with soft-core interactions: Patterns, stability, and collapse,” *Physical Review Letters*, vol. 96, no. 10, March 2006, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1103/PhysRevLett.96.104302.

[71] R. C. Arkin, “Motor schema – based mobile robot navigation,” *The International Journal of Robotics Research*, vol. 8, no. 4, pp. 92–112, 1989, [Online Available; Condition from 03. June 2013]. http://ijr.sagepub.com/cgi/doi/10.1177/027836498900800406.

[72] N. Esau, “Emotionale aspekte der mensch-roboter-interaktion und ihre realisierung in verhaltensbasierten systemen,” Dissertation, University of Paderborn, 2009.

[73] H. B. Duan and S. Q. Liu, “Non-linear dual-mode receding horizon control for multiple unmanned air vehicles formation flight based on chaotic particle swarm optimisation,” *Control Theory Applications, IET*, vol. 4, no. 11, pp. 2565–2578, November 2010, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1049/iet-cta.2009.0256.

[74] Z. Chao, L. Ming, Z. Shaolei, and Z. Wenguang, “Collision-free uav formation flight control based on nonlinear mpc,” in *Proceedings of the 2011 International Conference on Electronics, Communications and Control (ICECC)*, September 2011, pp. 1951–1956, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/ICECC.2011.6066578.

[75] B. Park, J. Choi, and W. K. Chung, “An efficient mobile robot path planning using hierarchical roadmap representation in indoor environment,” in *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA)*, May 2012, pp. 180–186, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/ICRA.2012.6225368.

[76] H. Oh, S. Kim, A. Tsourdos, and B. White, “Decentralised road-map assisted ground target tracking using a team of uavs,” in *Proceedings of the Data Fusion Target Tracking Conference (DF TT 2012): Algorithms Applications, 9th IET*, May 2012, pp. 1–6, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1049/cp.2012.0407.

[77] J. Berger, A. Boukhtouta, A. Benmoussa, and O. Kettani, “A new mixed-integer linear programming model for rescue path planning in uncertain adversarial environment,” *Computers & Operations Research*, vol. 39, no. 12, pp. 3420–3430, 2012, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1016/j.cor.2012.05.002.

[78] J. Q. Yu, S. Y. Li, G. C. Luo, and J. L. Wang, "Uav 3-d path planning with milp based on tightening constraint algorithm," in *Proceedings of the 2011 International Conference in Electrics, Communication and Automatic Control*, R. Chen, Ed. School of Aerospace Engineering, Beijing Institute of Technology, China: Springer New York, 2012, pp. 925–930, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/978-1-4419-8849-2_117.

[79] J. Guo, J. Wang, and G. Cui, "Online path planning for uav navigation based on quantum particle swarm optimization," in *Proceedings of the 3rd International Conference on Teaching and Computational Science (WTCS)*, ser. Advances in Intelligent and Soft Computing, Y. Wu, Ed. Department of Electric & Information Engineering, Zhengzhou Institute of Light Industry, Zhengzhou, China: Springer Berlin / Heidelberg, 2012, vol. 116, pp. 291–302, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/978-3-642-11276-8_37.

[80] H. Qingtian, C. Wenjing, and C. Jia, "Research of route planning based on genetic algorithm," in *Proceedings of the 2012 International Conference on Computer Science and Electronics Engineering (ICCSEE)*, vol. 2, March 2012, pp. 199–202, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/ICCSEE.2012.324.

[81] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/BF01386390.

[82] N. J. Nilsson, Ed., *Principles of Artificial Intelligence.* Springer Verlag, 1982.

[83] B. Horling and V. Lesser, "A survey of multi-agent organizational paradigms," *Knowledge Engineering Review*, vol. 19, no. 4, pp. 281–316, December 2004, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1017/S0269888905000317.

[84] M. Fox, *Organization structuring : designing large complex software.* Carnegie-Mellon University, 1979.

[85] A. Koestler, *The Ghost in the Machine.* Hutchinson Publishing Group, Arkana, 1967.

[86] K. Fisher, "Agent-based design of holonic manufacturing systems," *Robotics and Autonomous Systems*, vol. 27, no. 1–2, pp. 3–13, 1999, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1016/S0921-8890(98)00079-7.

[87] S. Breban and J. Vassileva, "Long-term coalitions for the electronic marketplace," in *Proceedings of the E-Commerce Applications Workshop*, 2001, [Online Available; Condition from 03. June 2012]. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.88.1197.

[88] C. Merida-Campos and S. Willmott, “Modelling coalition formation over time for iterative coalition games,” in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS ’04, vol. 2. Washington, DC, USA: IEEE Computer Society, 2004, pp. 572–579, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/AAMAS.2004.174.

[89] M. S. Fox, “An organizational view of distributed systems,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 11, no. 1, pp. 70–80, January 1981, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/TSMC.1981.4308580.

[90] C. H. Brooks, E. H. Durfee, and A. Armstrong, “An introduction to congregating in multi-agent systems,” in *Proceedings of Fourth International Conference on Multi-Agent Systems*, 2000, pp. 79–86, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/ICMAS.2000.858434.

[91] C. Dellarocas and M. Klein, “Civil agent societies: Tools for inventing open agent-mediated electronic marketplaces,” in *Agent Mediated Electronic Commerce II*, ser. Lecture Notes in Computer Science, A. Moukas, F. Ygge, and C. Sierra, Eds., vol. 1788. Sloan School of Management, Massachusetts Institute of Technology, USA: Springer Berlin / Heidelberg, 1999, pp. 24–39, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/10720026_2.

[92] G. Wiederhold, P. Wegner, and S. Ceri, “Toward megaprogramming,” *Communications of the ACM*, vol. 35, no. 11, pp. 89–99, November 1992, [Online Available; Condition from 03. June 2012]. http://doi.acm.org/10.1145/138844.138853.

[93] A. Koestler, “Online marketplaces,” in *Practical Handbook of Internet Computing*. CRC Press, 2004.

[94] M. P. Wellman, “A market-oriented programming environment and its application to distributed multicommodity flow problems,” *Journal of Artificial Intelligence Research*, vol. 1, pp. 1–23, 1993, [Online Available; Condition from 03. June 2012]. http://arxiv.org/abs/cs.AI/9308102.

[95] D. D. Corkill and S. E. Lander, “Diversity in agent organizations,” *Object Magazine*, vol. 8, no. 4, pp. 41–47, May 1998, [Online Available; Condition from 03. June 2012]. http://mas.cs.umass.edu/paper/267.

[96] T. Wagner and V. Lesser, “Relating Quantified Motivations for Organizationally Situated Agents,” in *Lecture Notes in Computer Science: Intelligent Agents VI. Agent Theories, Architectures and Languages.* Springer, Berlin, 2000, vol. 1757, pp. 334–348, [Online Available; Condition from 03. June 2012]. http://mas.cs.umass.edu/paper/136.

[97] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, June 2001, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/70.938381.

[98] M. Z. Aziz and B. Mertsching, "Survivor search with autonomous ugvs using multimodal overt attention," in *Proceedings of the IEEE International Workshop on Safety, Security & Rescue Robotics 2010*, July 2010, pp. 1–6, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/SSRR.2010.5981566.

[99] Microsoft Corporation, "Kinect for windows," [Online Available; Condition from 03. June 2012]. http://www.microsoft.com/en-us/kinectforwindows/.

[100] A. Macwan, G. Nejat, and B. Benhabib, "Target-motion prediction for robotic search and rescue in wilderness environments," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 41, no. 5, pp. 1287–1298, October 2011, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/TSMCB.2011.2132716.

[101] A. Macwan, G. Nejat, and B. Benhabib, "Optimal deployment of robotic teams for autonomous wilderness search and rescue," in *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, September 2011, pp. 4544–4549, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/IROS.2011.6094517.

[102] J. Meyer, P. Schnitzspan, S. Kohlbrecher, K. Petersen, M. Andriluka, O. Schwahn, U. Klingauf, S. Roth, B. Schiele, and O. von Stryk, "A semantic world model for urban search and rescue based on heterogeneous sensors," in *RoboCup 2010: Robot Soccer World Cup XIV*, ser. Lecture Notes in Computer Science, J. Ruiz-del Solar, E. Chown, and P. Plöger, Eds. Department of Mechanical Engineering, TU Darmstadt, Germany: Springer Berlin / Heidelberg, 2011, vol. 6556, pp. 180–193, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/978-3-642-20217-9_16.

[103] K. Shimaoka, K. Ogane, and T. Kimura, "An evaluation test field design for a usar robot related to a collapsed japanese house," in *Proceedings of the 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, November 2011, pp. 221–225, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/SSRR.2011.6106796.

[104] J. Ruiz-del Solar, E. Chown, and P. G. Ploeger, *RoboCup 2010: Robot Soccer World Cup XIV*, 1st ed. Springer, April 2011.

[105] T. Wisspeintner, T. van der Zan, L. Iocchi, and S. Schiffer, “Robocup@home: Results in benchmarking domestic service robots,” in *RoboCup 2009: Robot Soccer World Cup XIII*, ser. Lecture Notes in Computer Science, J. Baltes, M. Lagoudakis, T. Naruse, and S. Ghidary, Eds. Department of Mathematics and Computer Science, Freie Universität Berlin, Berlin, Germany: Springer Berlin / Heidelberg, 2010, vol. 5949, pp. 390–401, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/978-3-642-11876-0_34.

[106] N. Esau, M. Krüger, C. Rasche, S. Beringer, L. Kleinjohann, and B. Kleinjohann, “Hierarchical hybrid planning for a self-optimizing active suspension system,” in *Proceedings of the 7th IEEE Conference in Industrial Electronics and Applications*, IEEE. Singapore: IEEE, 18.–20. July 2012.

[107] F. Lingelbach, “Path planning using probabilistic cell decomposition,” Master’s thesis, Signaler Sensorer och System Stockholm, 2005.

[108] F. Bea, “Boston dynamics develops fastest record breaking four-legged robot, cheetah,” 6. March 2012, in Digital Trends [Online Available; Condition from 03. June 2012]. http://www.digitaltrends.com/cool-tech/boston-dynamics-develops-fastest-record-breaking-four-legged-robot-cheetah/.

[109] Robotnik, “Robotnik automation,” [Online Available; Condition from 03. June 2012]. http://www.robotnik.eu/.

[110] Designboom, “Hrp-4c humanoid robot on the runway at japan fashion week,” [Online Available; Condition from 03. June 2012]. http://www.designboom.com/technology/hrp-4c-humanoid-robot-on-the-runway-at-japan-fashion-week/.

[111] Botley and Harry, “Chemical caterpillar points to electronics-free robots,” 2011, [Online Available; Condition from 03. June 2012]. http://botley-and-harry.blogspot.de/2009/05/chemical-caterpillar-points-to.html.

[112] PSFK Conference, “Chembot: The military’s shape shifting robot blob,” 2011, [Online Available; Condition from 03. June 2012]. http://archive.psfk.com/2009/10/video-chembot-the-militarys-shape-shifting-robot-blob.html.

[113] Engadget, “Festo’s smartbird robot takes off with elegance,” 2011, [Online Available; Condition from 03. June 2012]. http://www.engadget.com/2011/03/25/festos-smartbird-robot-takes-off-with-elegance-doesnt-fight-s/.

[114] Engadget, “Festo shows off robot penguins and other visions of the future,” 2011, [Online Available; Condition from 03. June 2012]. http://www.engadget.com/2009/04/18/festo-shows-off-robot-penguins-and-other-visions-of-the-future/.

[115] Savateuse, “Aquajelly: A robotic jellyfish by festo,” 2011, [Online Available; Condition from 03. June 2012]. http://savateuse.tumblr.com/post/502245039/aquajelly-a-robotic-jellyfish-by-festo-for.

[116] Swiss UAV, "Swiss uav - autonomous unmanned aerial vehicles, made in switzerland," [Online Available; Condition from 03. June 2012]. http://www.swiss-uav.com/uav_systems.php.

[117] D. Dearing, "Rq-170 sentinel drone," [Online Available; Condition from 03. June 2012]. http://grabcad.com/library/rq-170-sentinel-drone.

[118] W. Yanyang, W. Tietao, and Q. Xiangju, "Study of multi-objective fuzzy optimization for path planning," *Chinese Journal of Aeronautics*, vol. 25, no. 1, pp. 51–56, 2012, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1016/S1000-9361(11)60361-0.

[119] S. Zhou, J. Wang, and Y. Jin, "Route planning for unmanned aircraft based on ant colony optimization and voronoi diagram," in *Intelligent System Design and Engineering Application (ISDEA), 2012 Second International Conference on*, January 2012, pp. 732–735, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/ISdea.2012.568.

[120] Y. Chen, J. Han, and X. Zhao, "Three-dimensional path planning for unmanned aerial vehicle based on linear programming," *Robotica*, vol. 30, no. 5, pp. 773–781, September 2012, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1017/S0263574711000993.

[121] H. Liu, X. Gu, J. Chen, and H. Liu, "Intelligent multi-objective receding horizon control for ucav mission planning," in *International Conference on Computer Science and Information Processing (CSIP)*, August 2012, pp. 1154–1158, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/CSIP.2012.6309063.

[122] C.-M. Lin, C.-F. Tai, and C.-C. Chung, "Intelligent control system design for uav using a recurrent wavelet neural network," *Neural Computing and Applications*, pp. 1–10, 2012, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/s00521-012-1242-5.

[123] M. Garratt and S. Anavatti, "Non-linear control of heave for an unmanned helicopter using a neural network," *Journal of Intelligent & Robotic Systems*, vol. 66, pp. 495–504, 2012, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/s10846-011-9634-9.

[124] M. R. Kindl and N. C. Rowe, "Evaluating simulated annealing for the weighted-region path-planning problem," in *26th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, March 2012, pp. 926–931, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/WAINA.2012.30.

[125] I. Palunko, R. Fierro, and P. Cruz, "Trajectory generation for swing-free maneuvers of a quadrotor with suspended payload: A dynamic programming approach," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2012, pp. 2691–2697, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/ICRA.2012.6225213.

[126] Y. V. Pehlivanoglu, "A new vibrational genetic algorithm enhanced with a voronoi diagram for path planning of autonomous uav," *Aerospace Science and Technology*, vol. 16, no. 1, pp. 47–55, 2012, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1016/j.ast.2011.02.006.

[127] S. Ross, N. Melik-Barkhudarov, K. Shaurya Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive uav control in cluttered natural environments," *ArXiv e-prints*, November 2012.

[128] M. Turpin, N. Michael, and V. Kumar, "Trajectory design and control for aggressive formation flight with quadrotors," *Autonomous Robots*, vol. 33, pp. 143–156, 2012, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/s10514-012-9279-y.

[129] Z. Xiang-Yin and D. Hai-Bin, "Differential evolution-based receding horizon control design for multi-uavs formation reconfiguration," *Transactions of the Institute of Measurement and Control*, vol. 34, pp. 165–183, April 2012.

[130] F. Rinaldi, S. Chiesa, and F. Quagliotti, "Linear quadratic control for quadrotors uavs dynamics and formation flight," *Journal of Intelligent & Robotic Systems*, pp. 1–18, 2012, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/s10846-012-9708-3.

[131] A. Yang, W. Naeem, G. W. Irwin, and K. Li, "A decentralised control strategy for formation flight of unmanned aerial vehicles," in *UKACC International Conference on Control (CONTROL)*, September 2012, pp. 345–350, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/CONTROL.2012.6334654.

[132] A. Kushleyev, D. Mellinger, and V. Kumar, "Towards a swarm of agile micro quadrotors," *RSS*, 2012.

[133] G. R. Madey, M. B. Blake, C. Poellabauer, H. Lu, R. R. McCune, and Y. Wei, "Applying dddas principles to command, control and mission planning for uav swarms," *Procedia Computer Science*, vol. 9, no. 0, pp. 1177–1186, 2012, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1016/j.procs.2012.04.127.

[134] A. Renzaglia, L. Doitsidis, A. Martinelli, and E. Kosmatopoulos, "Multi-robot three dimensional coverage of unknown areas," *International Journal of Robotics Research*, March 2012, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1177/0278364912439332.

[135] J. Worcester and M. Hsieh, "Task partitioning via ant colony optimization for distributed assembly," in *Swarm Intelligence*, ser. Lecture Notes in Computer Science, M. Dorigo, M. Birattari, C. Blum, A. Christensen, A. P. Engelbrecht, R. Groß, and T. Stützle, Eds. Springer Berlin Heidelberg, 2012, vol. 7461, pp. 145–155, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/978-3-642-32650-9_13.

[136] S. Hunt, Q. Meng, and C. Hinde, "An extension of the consensus-based bundle algorithm for group dependent tasks with equipment dependencies," in *Neural Information Processing*, ser. Lecture Notes in Computer Science, T. Huang, Z. Zeng, C. Li, and C. Leung, Eds. Springer Berlin Heidelberg, 2012, vol. 7666, pp. 518–527, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/978-3-642-34478-7_63.

[137] C. E. Pippin and H. Christensen, "Cooperation based dynamic team formation in multi-agent auctions," *Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR III*, vol. 8389, 2012, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1117/12.919551.

[138] P. B. Sujit and J. B. Sousa, "Multi-uav task allocation with communication faults," in *American Control Conference (ACC), 2012*, June 2012, pp. 3724–3729.

[139] Q. Lindsey, D. Mellinger, and V. Kumar, "Construction with quadrotor teams," *Autonomous Robots*, vol. 33, pp. 323–336, 2012, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/s10514-012-9305-0.

[140] Vicon, "Motion capture systems from vicon," [Online Available; Condition from 03. June 2012]. http://www.vicon.com/.

[141] M. Ozaki, L. Gobeawan, S. Kitaoka, H. Hamazaki, Y. Kitamura, and R. Lindeman, "Camera movement for chasing a subject with unknown behavior based on real-time viewpoint goodness evaluation," *The Visual Computer*, vol. 26, no. 6, pp. 629–638, 2010, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1007/s00371-010-0489-z.

[142] J. Barraquand, B. Langlois, and J. C. Latombe, "Numerical potential field techniques for robot path planning," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, no. 2, pp. 224–241, March/April 1992, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1109/21.148426.

[143] J.-C. Latombe, *Robot Motion Planning.* Norwell, MA, USA: Kluwer Academic Publishers, 1991.

[144] J. Goldsmith and J. Salmon, "Automatic creation of object hierarchies for ray tracing," *Computer Graphics and Applications, IEEE*, vol. 7, no. 5, pp. 14–20, May 1987, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/MCG.1987.276983.

[145] H. Moravec and A. Elfes, “High resolution maps from wide angle sonar,” in *Proceedings of the International Conference on Robotics and Automation*, vol. 2, March 1985, pp. 116–121, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/ROBOT.1985.1087316.

[146] R. Schumacher, R. Brand, M. Gilliland, and W. Sharp, “Study for applying computer-generated images to visual simulation,” Air Force Human Resources Laboratory, Training Research Division, Tech. Rep., 1969.

[147] J. L. Bentley, “Multidimensional binary search trees used for associative searching,” *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, September 1975, [Online Available; Condition from 03. June 2012]. http://doi.acm.org/10.1145/361002.361007.

[148] R. A. Finkel and J. L. Bentley, “Quadtrees a data structure for retrieval on composite keys,” *Acta Informatica*, vol. 4, no. 1, pp. 1–9, March 1974, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/BF00288933.

[149] M. Joselli, E. B. Passos, M. Zamith, E. Clua, A. Montenegro, and B. Feijó, “A neighborhood grid data structure for massive 3d crowd simulation on gpu,” in *Proceedings of the 2009 VIII Brazilian Symposium on Games and Digital Entertainment (SBGAMES)*, October 2009, pp. 121–131, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/SBGAMES.2009.22.

[150] G. Li, A. Yamashita, H. Asama, and Y. Tamura, “An efficient improved artificial potential field based regression search method for robot path planning,” in *Proceedings of the 2012 International Conference on Mechatronics and Automation (ICMA)*, August 2012, pp. 1227–1232, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/ICMA.2012.6283526.

[151] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, “Path planning for autonomous driving in unknown environments,” in *Proceedings of the Eleventh International Symposium on Experimental Robotics (ISER-08)*, ser. Springer Tracts in Advanced Robotics, O. Khatib, V. Kumar, and G. Pappas, Eds. Toyota Research Institute AI & Robotics Group Ann Arbor MI 48105: Springer Tracts in Advanced Robotics (STAR), July 2008, vol. 54, pp. 55–64, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1007/978-3-642-00196-3_8.

[152] A. Masoud, “Managing the dynamics of a harmonic potential field-guided robot in a cluttered environment,” *Industrial Electronics, IEEE Transactions on*, vol. 56, no. 2, pp. 488–496, February 2009, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/TIE.2008.2002720.

[153] Q. Li, L. Wang, B. Chen, and Z. Zhou, "An improved artificial potential field method for solving local minimum problem," in *Proceedings of the 2nd International Conference on Intelligent Control and Information Processing (ICICIP)*, July 2011, pp. 420–424, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1109/ICICIP.2011.6008278.

[154] D. Yagnik, J. Ren, and R. Liscano, "Motion planning for multi-link robots using artificial potential fields and modified simulated annealing," in *Proceedings of the ASME International Conference on Mechatronics and Embedded Systems and Applications (MESA)*, July 2010, pp. 421 –427, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1109/MESA.2010.5551989.

[155] K. Sato, "Collision avoidance in multi-dimensional space using laplace potential," in *Proceedings of the 15th Conference of the Robotics Society Japan*, 1987, pp. 155–156.

[156] K. Sato, "Deadlock-free motion planning using the laplace potential field," *Advanced Robotics*, vol. 7, no. 5, pp. 449–461, 1992, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1163/156855393X00285.

[157] B. Epstein and M. Schiffer, "On the mean-value property of harmonic functions," *Journal d'Analyse Mathématique*, vol. 14, pp. 109–111, 1965, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/BF02806381.

[158] J. O. Kim and P. K. Khosla, "Real-time obstacle avoidance using harmonic potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 338–349, June 1992, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/70.143352.

[159] S. Ponnusamy and H. Silverman, *Complex Variables with Applications*, 1st ed. Birkhäuser Boston, September 2006.

[160] Karnik, M., Dasgupta, B., and Eswaran,V., "A comparative study of dirichlet and neumann conditions for path planning through harmonic functions," *Future Gener. Comput. Syst.*, vol. 20, no. 3, pp. 441–452, 2004, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1016/j.future.2003.07.008.

[161] C. I. Connolly, "Applications of harmonic functions to robotics," *Journal of Robotic Systems*, pp. 498–502, August 1992, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1109/ISIC.1992.225141.

[162] E. Dynkin and S. Kuznetsov, "Solutions of lu=u α dominated by l-harmonic functions," *Journal d'Analyse Mathématique*, vol. 68, pp. 15–37, 1996, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/BF02790202.

[163] D. J. Bennet and C. R. McInnes, “Distributed control of multi-robot systems using bifurcating potential fields,” *Robotics and Autonomous Systems*, vol. 58, no. 3, pp. 256–264, March 2010, [Online Available; Condition from 03. June 2013]. http://www.sciencedirect.com/science/article/pii/S092188900900195X.

[164] R. U. Seydel, *Practical Bifurcation and Stability Analysis*, 3rd ed. Springer, 2009.

[165] R. P. P. P. G., H. L. J. van der Maas, and E. J. Wagenmakers, “Fitting the cusp catastrophe in r: A cusp-package primer,” in *Journal of Statistical Software*, vol. 32, November 2009, [Online Available; Condition from 03. June 2012]. http://EconPapers.repec.org/RePEc:jss:jstsof:32:i08.

[166] C. Zhangxin, C. Shui-Nee, and L. Kaitai, *Bifurcation Theory & its Numerical Analysis*, 2nd ed. Springer, July 1998.

[167] D. J. Bennet and C. R. McInnes, “Verifiable control of a swarm of unmanned aerial vehicles,” *Journal of Aerospace Engineering*, vol. 223 (7), pp. 939–953, 2009, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.1243/09544100JAERO508.

[168] A. R. Mosteo and L. Montano, “A survey of multi-robot task allocation,” *Challenges*, pp. 1–27, 2010, [Online Available; Condition from 03. June 2012]. http://www.mosteo.com/papers/mm10-i3a.pdf.

[169] D. Miller, P. Dasgupta, and T. Judkins, “Distributed task selection in multi-agent based swarms using heuristic strategies,” in *Proceedings of the 2nd international conference on Swarm robotics*, ser. SAB’06, E. Sahin, W. Spears, and A. Winfield, Eds., vol. 4422. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 158–172, [Online Available; Condition from 03. June 2012]. http://dl.acm.org/citation.cfm?id=1763837.1763848.

[170] B. Gerkey and M. Mataric, “Multi-robot task allocation: analyzing the complexity and optimality of key architectures,” in *ICRA ’03. IEEE International Conference on Robotics and Automation. Proceedings*, vol. 3, September 2003, pp. 3862–3868, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/ROBOT.2003.1242189.

[171] G. P. Das, T. M. McGinnity, S. A. Coleman, and L. Behera, “A fast distributed auction and consensus process using parallel task allocation and execution,” *Proceedings of the 2011 IEEERSJ International Conference on Intelligent Robots and Systems*, pp. 4716–4721, 2011, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/IROS.2011.6094948.

[172] D. Landén, F. Heintz, and P. Doherty, "Complex task allocation in mixed-initiative delegation: A uav case study," in *Principles and Practice of Multi-Agent Systems*, ser. Lecture Notes in Computer Science, N. Desai, A. Liu, and M. Winikoff, Eds., vol. 7057. Department of Computer and Information Science, Linköping University, Sweden: Springer Berlin / Heidelberg, 2010, pp. 288–303, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/978-3-642-25920-3_20.

[173] L. Guanghui, Y. Tamura, and H. Asama, "Dynamical task allocation and reallocation based on body expansion behavior for multi-robot coordination system," in *Proceedings of the International Conference on Mechatronics and Automation (ICMA)*, August 2011, pp. 537–542, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/ICMA.2011.5985718.

[174] T. Li, D. Baumberger, D. A. Koufaty, and S. Hahn, "Efficient operating system scheduling for performance-asymmetric multi-core architectures," in *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing. SC '07*, November 2007, pp. 1–11, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1145/1362622.1362694.

[175] Q. Hui, W. M. Haddad, and S. P. Bhat, "Semistability, finite-time stability, differential inclusions, and discontinuous dynamical systems having a continuum of equilibria," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2465–2470, October 2009, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/TAC.2009.2029397.

[176] A. M. Lyapunov, *General Problem of the Stability Of Motion.* CRC Press, 1992.

[177] S. Shankar Sastry, R. M. Murray, and Z. Li, *A Mathematical Introduction to Robotic Manipulation.* Crc Pr Inc, 1994.

[178] J. Lasalle, "Some extensions of liapunov's second method," *Circuit Theory, IRE Transactions on*, vol. 7, no. 4, pp. 520–527, December 1960, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/TCT.1960.1086720.

[179] Free Software Foundation, Inc., "Gcc, the gnu compiler collection," [Online Available; Condition from 03. June 2013]. http://gcc.gnu.org/.

[180] D. H. Eberly, *3D Game Engine Design, Second Edition: A Practical Approach to Real-Time Computer Graphics*, 2nd ed. Morgan Kaufmann, November 2006.

[181] J. S. Gutmann, M. Fukuchi, and M. Fujita, "A floor and obstacle height map for 3d navigation of a humanoid robot," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005. ICRA 2005*, April 2005, pp. 1066–1071, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/ROBOT.2005.1570257.

[182] K. L. Murdock, *3ds Max 2012 Bible*, 1st ed. John Wiley & Sons, August 2011.

[183] W. Donnelly and A. Lauritzen, “Variance shadow maps,” in *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, ser. I3D ’06. New York, NY, USA: ACM, 2006, pp. 161–165, [Online Available; Condition from 03. June 2012]. http://doi.acm.org/10.1145/1111411.1111440.

[184] Jet Propulsion Laboratory, “Shuttle radar topography mission - the mission to map the world,” [Online Available; Condition from 03. June 2012]. http://www2.jpl.nasa.gov/srtm.

[185] OpenStreetMap Foundation, “Openstreetmap,” [Online Available; Condition from 03. June 2012]. http://www.openstreetmap.org.

[186] R. Seidel, “Reprint of: A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons,” *Computational Geometry*, vol. 43, no. 6–7, pp. 556–564, 2010, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1016/j.comgeo.2010.03.003.

[187] L. Bishop, D. Eberly, T. Whitted, M. Finch, and M. Shantz, “Designing a pc game engine,” *Computer Graphics and Applications, IEEE*, vol. 18, no. 1, pp. 46–53, January/February 1998, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/38.637270.

[188] E. Prestes, M. E. Paulo, M. Trevisan, and M. A. P. Idiart, “Exploratory navigation based on dynamical boundary value problems,” *Journal of Intelligent and Robotic Systems*, vol. 45, no. 2, pp. 101–114, February 2006, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/s10846-005-9008-2.

[189] T. Lozano-Pérez and M. A. Wesley, “An algorithm for planning collision-free paths among polyhedral obstacles,” *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, October 1979, [Online Available; Condition from 03. June 2013]. http://doi.acm.org/10.1145/359156.359164.

[190] M. Agrawala, A. C. Beers, and M. Levoy, “3d painting on scanned surfaces,” in *Proceedings of the 1995 symposium on Interactive 3D graphics*, ser. I3D ’95. New York, NY, USA: ACM, 1995, pp. 145–ff., [Online Available; Condition from 03. June 2012]. http://doi.acm.org/10.1145/199404.199429.

[191] K. Aizawa, K. Motomura, S. Kimura, R. Kadowaki, and J. Fan, “Constant time neighbor finding in quadtrees: An experimental result,” *3rd International Symposium on Communications, Control and Signal Processing, 2008. ISCCSP 2008.*, pp. 505–510, March 2008, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/ISCCSP.2008.4537278.

[192] A. S. Glassner, *An introduction to ray tracing.* Morgan Kaufmann, Juni 1989.

[193] D. Warneke and C. Dannewitz, "Statistics-based id management for load balancing in structured p2p networks," in *LCN 2009. IEEE 34th Conference on Local Computer Networks*, October 2009, pp. 273–276, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/LCN.2009.5355086.

[194] H. Frey and I. Stojmenovic, "On delivery guarantees and worst-case forwarding bounds of elementary face routing components in ad hoc and sensor networks," *IEEE Transactions on Computers*, vol. 59, no. 9, pp. 1224–1238, September 2010, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/TC.2010.107.

[195] C. Scheideler, A. Richa, and P. Santi, "An o(log n) dominating set protocol for wireless ad-hoc networks under the physical interference model," in *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, ser. MobiHoc '08. New York, NY, USA: ACM, 2008, pp. 91–100, [Online Available; Condition from 03. June 2012]. http://doi.acm.org/10.1145/1374618.1374632.

[196] H. J. Bremermann, "On a generalized dirichlet problem for plurisubharmonic functions and pseudo-convex domains," in *Transactions of the American Mathematical Society: Characterization of Šilov boundaries*, 1959, pp. 246–276, [Online Available; Condition from 03. June 2012]. http://www.ams.org/journals/tran/1959-091-02/S0002-9947-1959-0136766-9.

[197] M. A. K. Jaradat, M. H. Garibeh, and E. A. Feilat, "Autonomous mobile robot dynamic motion planning using hybrid fuzzy potential field," *Soft Computing*, vol. 16, pp. 153–164, 2012, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/s00500-011-0742-z.

[198] D. M. Rivera, F. A. Prieto, and R. Ramirez, "Trajectory planning for uavs in 3d environments using a moving band in potential sigmoid fields," in *Robotics Symposium and Latin American Robotics Symposium (SBR-LARS), 2012 Brazilian*, October 2012, pp. 115–119, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/SBR-LARS.2012.26.

[199] P. Kim and D. Kurabayashi, "Forming an artificial pheromone potential field using mobile robot and rfid tags," in *System Integration (SII), 2011 IEEE/SICE International Symposium on*, December 2011, pp. 621–626, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/SII.2011.6147520.

[200] A. A. Masoud, "A harmonic potential approach for simultaneous planning and control of a generic uav platform," *Journal of Intelligent & Robotic Systems*, vol. 65, pp. 153–173, 2012, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/s10846-011-9570-8.

[201] E. Prestes, M. E. Paulo, M. Trevisan, and M. A. P. Idiart, "Exploration method using harmonic functions," *Journal of Neurophysiology*, vol. 40, 2002.

[202] G. Dvali, G. Gabadadze, and M. Porrati, "Metastable gravitons and infinite volume extra dimensions," *Physics Letters B*, vol. 484, no. 1–2, pp. 112–118, 2000, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1016/S0370-2693(00)00631-6.

[203] E. C. Zachmanoglou and D. W. Thoe, Eds., *Introduction to Partial Differential Equations with Applications.* Dover Publications, Inc., 1986.

[204] C. Kanzow, Ed., *Numerik linearer Gleichungssysteme.* Springer Verlag, 2005.

[205] E. Prestes, M. E. Paulo, M. Trevisan, and M. A. P. Idiart, "Exploration technique using potential fields calculated from relaxation methods," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 4, pp. 2012–2017, 2001, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/IROS.2001.976368.

[206] A. Sheldon, B. Paul, and R. Wade, *Harmonic Function Theory*, 2nd ed. Springer, 2001.

[207] G. D. Smith, *Numerical Solution Of Partial Differential Equations: Finite Difference Methods*, 3rd ed. Oxford University Press, Januar 1986.

[208] W. F. Ames, *Numerical Methods for Partial Differential Equations*, 3rd ed. Academic Press, September 1992.

[209] J. S. Zelek, "A framework for mobile robot concurrent path planning and execution in incomplete and uncertain environments," in *Proceedings of the AIPS-98 Workshop on Integrating Planning, Scheduling and Execution in Dynamic and Uncertain Environments*, 1998, [Online Available; Condition from 03. June 2013]. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.53.4440.

[210] S. Wongthanavasu and C. Lursinsap, "A 3d ca-based edge operator for 3d images," in *Proceedings of the International Conference on Image Processing (ICIP '04)*, vol. 1, October 2004, pp. 235–238, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/ICIP.2004.1418733.

[211] S. Masoud and A. Masoud, "Motion planning in the presence of directional and regional avoidance constraints using nonlinear, anisotropic, harmonic potential fields: a physical metaphor," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 32, no. 6, pp. 705–723, November 2002, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/TSMCA.2002.807030.

[212] D. Koditschek, "Exact robot navigation by means of potential functions: Some topological considerations," in *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, vol. 4, March 1987, pp. 1–6, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/ROBOT.1987.1088038.

[213] D. J. Bennet and C. R. McInnes, “Spacecraft formation flying using bifurcating potential fields,” in *59th International Astronautical Congress*, September 2008, [Online Available; Condition from 03. June 2012]. http://strathprints.strath.ac.uk/id/eprint/7331.

[214] V. I. Arnold, *Mathematical Methods of Classical Mechanics*, 2nd ed. Springer New York, February 2010.

[215] C. Warren, “Multiple robot path coordination using artificial potential fields,” in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, vol. 1, May 1990, pp. 500–505, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/ROBOT.1990.126028.

[216] C. Bentes and O. Saotome, “Dynamic swarm formation with potential fields and a* path planning in 3d environment,” in *Robotics Symposium and Latin American Robotics Symposium (SBR-LARS), 2012 Brazilian*, October 2012, pp. 74–78, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/SBR-LARS.2012.19.

[217] L. Blazovics, B. Ilsinszki, K. Csorba, B. Forstner, and C. Hassan, “Aspects of formation control for swarm robots,” in *Carpathian Control Conference (ICCC), 2012 13th International*, May 2012, pp. 46–51, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/CarpathianCC.2012.6228614.

[218] R. D. Garcia, L. Barnes, and M. A. Fields, “Unmanned aircraft systems as wingmen,” *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, vol. 9, no. 1, pp. 5–15, 2012, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1177/1548512910391217.

[219] D. J. Bennet, C. R. Mcinnes, M. Suzuki, and K. Uchiyama, “Autonomous three-dimensional formation flight for a swarm of unmanned aerial vehicles,” *Journal of Guidance, Control and Dynamics*, vol. 34, pp. 1899–1908, 2011, [Online Available; Condition from 03. June 2013]. http://dx.doi.org/10.2514/1.53931.

[220] W. Ni and D. Cheng, “Leader-following consensus of multi-agent systems under fixed and switching topologies,” *Systems & Control Letters*, vol. 59, no. 3–4, pp. 209–217, 2010, [Online Available; Condition from 03. June 2013]. http://www.sciencedirect.com/science/article/pii/S0167691110000186.

[221] Deutsches Institut für Normung e. V, “Deutsches institut für normung,” [Online Available; Condition from 03. June 2012]. http://din.de/.

[222] ISO, “Iso - international organization for standardization,” [Online Available; Condition from 03. June 2012]. http://www.iso.org/.

[223] B. P. Gerkey and M. J. Matarić, “A formal analysis and taxonomy of task allocation in multi-robot systems,” *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1177/0278364904045564.

[224] F. Delle Fave, A. Rogers, Z. Xu, S. Sukkarieh, and N. Jennings, “Deploying the max-sum algorithm for decentralized coordination and task allocation of unmanned aerial vehicles for live aerial imagery collection,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, May 2012, pp. 469–476, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/ICRA.2012.6225053.

[225] M. Lujak, H. Billhardt, and S. Ossowski, “On mobile target allocation with incomplete information in defensive environments,” in *Agent and Multi-Agent Systems. Technologies and Applications*, ser. Lecture Notes in Computer Science, G. Jezic, M. Kusek, N.-T. Nguyen, R. J. Howlett, and L. C. Jain, Eds. Springer Berlin Heidelberg, 2012, vol. 7327, pp. 4–13, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1007/978-3-642-30947-2_4.

[226] T. Huntsberger, G. Rodriguez, and P. S. Schenker, “Robotics challenges for robotic and human mars exploration,” in *Proceedings of the 4th International Conference and Exposition/Demonstration on Robotics for Challenging Situations and Environments, Albuquerque, NM; UNITED STATES*, March 2000, pp. 340–346, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1061/40476(299)45.

[227] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, *Global Positioning System: Theory and Practice*, 3rd ed. Springer, Oktober 1994.

[228] M. S. Grewal, L. R. Weill, and A. P. Andrews, *Global Positioning Systems, Inertial Navigation, and Integration*, 2nd ed. John Wiley & Sons, February 2007.

[229] J. J. van Zyl, “The shuttle radar topography mission (srtm): a breakthrough in remote sensing of topography,” *Acta Astronautica*, vol. 48, pp. 559–565, 2001.

[230] A. Roth, “Terrasar-x: a new perspective for scientific use of high resolution spaceborne sar data,” in *Remote Sensing and Data Fusion over Urban Areas, 2003. 2nd GRSS/ISPRS Joint Workshop on*, May 2003, pp. 4–7, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/DFUA.2003.1219947.

[231] S. Buckreuss, R. Werninghaus, and W. Pitz, “The german satellite mission terrasar-x,” *Aerospace and Electronic Systems Magazine, IEEE*, vol. 24, no. 11, pp. 4–9, November 2009, [Online Available; Condition from 03. June 2012]. http://dx.doi.org/10.1109/MAES.2009.5344175.

[232] InterArtCenter.net, “Buddhist temple wooden building,” [Online Available; Condition from 03. June 2012]. http://artist-3d.com/free_3d_models/dnm/model_disp.php?uid=2531.

[233] InterArtCenter.net, “Walking bridge curve garden viaduct,” [Online Available; Condition from 03. June 2012]. http://artist-3d.com/free_3d_models/dnm/model_disp.php?uid=431.

[234] InterArtCenter.net, “Triumphal arch classical monument,” [Online Available; Condition from 03. June 2012]. http://artist-3d.com/free_3d_models/dnm/model_disp.php?uid=3182.

[235] InterArtCenter.net, “Paris famous eiffel tower,” [Online Available; Condition from 03. June 2012]. http://artist-3d.com/free_3d_models/dnm/model_disp.php?uid=3289.