Coordinated Exploration and Goal-Oriented Path Planning using Multiple UAVs

Christoph Rasche, Claudius Stern, Lisa Kleinjohann, and Bernd Kleinjohann
Faculty of Computer Science, Electrical Engineering and Mathematics
Department of Computer Science
University of Paderborn
Paderborn, Germany
crasche@c-lab.de, claudis@c-lab.de, lisa@c-lab.de, bernd@c-lab.de

Abstract—**Successful rescue operations after big accidents or natural disasters require a fast and efficient overview of the overall situation. Using aircrafts is an efficient method to achieve such an overview in relatively short time. With recent advances, unmanned aerial vehicles (UAVs) are more and more a viable choice under such circumstances.**

With the number of employed UAVs, the problem of coordination arises as well as proper task allocation among possibly heterogeneous UAVs. Coordination has to be done so that redundant exploration and collision of UAVs with each other are avoided.

This paper presents a hybrid approach for UAV coordination that covers the exploration of unknown terrains as well as goal-oriented coordination and simultaneous task allocation. The approach combines the simplicity of the gradient method with informed A* search and supports prioritized task assignment. It is based on the potential field theory using harmonic functions. Only one single configuration space for representing all relevant information, regarding the terrain and the UAVs is used. The system is suited for highly dynamic environments requiring frequent path recalculations.

Keywords-**Path Planning, Multiple UAVs, Exploration, Coordination, Potential Field Theory, Harmonic Functions**

I. Introduction

Over the last years, unmanned aerial vehicles (UAVs) received increasing attention in rescue operations. They can be used to explore terrains and are able to relieve humans from dangerous and risky tasks.

Through the use of cameras, it is for instance possible to detect victims or dangerous situations like fire nearby explosives. Such information can be used to coordinate the assistants at the accident site.

Another task, which can be solved is to take measurements in the air, e. g., after a volcanic eruption without risking the life of pilots.

A major task in rescue operations is the exploration of the disaster area. This is important to get an overview of the surroundings and to locate victims. It is necessary to obtain such an overview as fast as possible to start effective rescue operations and–in some cases–to avoid an escalation of the situation, e. g., when explosives are detected near a fire.

Exploratory navigation includes determining all obstacles and goals in a given environment. UAVs travel from their initial positions to one or more different goal positions, while avoiding obstacles and recognizing landmarks and objects. Therefore, the UAVs have to memorize the explored space to plan efficient paths to unknown territory.

In rescue scenarios often places can be identified by the staff where victims are very likely to be found. These areas will be explored first. To take this into account it is possible to mark areas as given goal areas.

The main challenge is to design a system, which covers all the following aspects: autonomous exploration, flying to given targets, and the coordination of multiple UAVs. So, the system to be designed has to manage exploratory navigation as well as goal-oriented planning. Also a task allocation amongst the UAVs is needed.

Another point is that the system has to be efficient. In medical emergencies often every second counts. A system, which needs a long time span to plan paths gives away important time for lifesaving. This means that every calculation, necessary for path planning, should be done without any noticeable delay.

In this paper, an approach combining exploratory navigation, goal-oriented path planning and simultaneous task allocation is presented. It gives a more detailed view of the work presented at the ICAS 2010 [1]. The presented approach is based on artificial potential fields (Section II-D). Path generation then utilizes an informed search algorithm in combination with a gradient method (Section VI). To obtain the desired efficiency, several methods (Section V-C), like a quadtree and an activation window, are used to reduce the calculation costs for path planning. The system ensures a complete exploration of unknown regions and scales well in relation to the number of operating UAVs.

The presented system has two modes of operation, which change automatically during runtime, dependent on the existence or absence of goals: *goal-oriented* and *non-goal-oriented* mode.

Every single UAV has an individual configuration (Section IV-B), amongst others containing its task list. In goal-oriented mode a UAV has one or more prioritized tasks (goals). In non-goal-oriented mode exploration of the complete territory will be done. The approach is cost-efficient because of the combination of the advantages of goal-oriented path planning with the simplicity of gradient methods for exploration. UAVs without a given goal always move to the nearest unexplored region to optimize exploration. Local as well as global information is used to coordinate multiple UAVs and to generate consistent trajectories. Areas, which probably contain important information can be explored first.

To even support exploration in goal-oriented mode, there is an adjustable tradeoff for the maximum allowed deviation from the shortest path to explore unknown regions (Section V-A). In many cases, gathering new information is also important and a small detour is worth a slight increase of flight time.

The paper is organized as follows. In Section II we show the basics of our approach and some differences to other research. Section III gives an overview of the system we developed. Section IV shows our configuration space, which is used as world model. In Section V we explain how we calculate the potential field using harmonic functions. In Section VI we show the path planning algorithms using the potential field to calculate smooth paths that guarantee obstacle avoidance and always lead to the goal if a path exist. In Section VII we show several results for exploratory navigation including the calculation costs. Finally, Section VIII gives a conclusion and a perspective of future work.

II. Related Work

This section describes related work in the fields of exploration, path planning, and task allocation. Afterwards, the theoretical background for the methods used in our approach is introduced. We start with the basic problem of path planning, which leads to the concept of a configuration space $\mathcal{C}$, and show an overview of the used approaches, based on $\mathcal{C}$.

Today the potential field theory [2] is used in several research areas, e. g., when autonomous robots have to explore terrains [3], [4], [5] or in game theory for path planning of units in strategy games [6]. Several different potential field techniques exist for the different applications.

Exploration of unknown terrain is one of the most important tasks of UAVs. Sawhney et al. [7] presented an exploration system for multiple robots and UAVs in 2D and 3D environments. Their work is focused on an asynchronous exploration strategy in unknown terrains. Asynchronous means that in contrast to synchronous allocation only UAVs, which have stopped are considered for the next exploration of unknown terrain.

Their main goal is to find several paths, which completely explore the terrain, such that the time needed for exploration is reduced as much as possible and the height of the path over any point is constrained to lie beneath an exposure surface. For this purpose they subdivide the terrain using an occupancy grid and plan their paths based on this data structure. In simulations they showed that their approach explores the terrain faster than a synchronous one.

Another part of UAV navigation is goal-oriented path planning. Jung et al. [8] showed a hierarchical path planning approach, including a control algorithm for one single UAV. The UAV had to accomplish the mission objective with limited computational resources. This was to reach a goal destination, while avoiding obstacles.

They start path planning using the A^*-algorithm as an informed search algorithm. Afterwards, a path smoothing takes place and the calculated path is followed by an auto pilot.

Like our approach, theirs is based on a multiresolution decomposition of the environment for path planning too, such that a coarser resolution is used far away from the agent, whereas fine resolution is used in the vicinity of the UAV.

For task allocation a scheduling was introduced. To increase the efficiency of their approach a real-time kernel was used. They demonstrated that the UAV with its limited resources managed the mission objective by using a real-time kernel.

Nikolos et al. [9] showed a goal-oriented approach based on evolutionary algorithms, which is capable to navigate through explored and unexplored terrain to a given goal. They distinguish two problems:

- UAV navigation using an offline planner, considering a known 3D environment
- UAV navigation using an online planner, considering a completely unknown 3D environment

In this context offline planning means that a complete path from the initial position of the UAV to the goal position is calculated in advance, whereas in online planning a nearly optimal path will be generated to an intermediate position within the sensors' range. During flight time this is repeated, until the goal position is reached.

They considered the path planning problem within an evolutionary algorithm context, in which B-Spline curves are used to represent the path line. The evolutionary algorithm models the coordinates of the UAV's control points as artificial chromosome genes.

They used the potential field theory combined with a grid-based cell decomposition method as underlying system on which path planning is done. Their results show that it is possible to find feasible paths efficiently by using evolutionary algorithms. This solution could be reached cost-efficiently since only a few iterations were needed to find a feasible solution. The most costly part was to optimize the solution.

All these approaches consider either goal-oriented or non-goal-oriented path planning. In this work we present a combined approach that considers three aspects: exploratory navigation, goal-oriented path planning and simultaneous task allocation. Our approach works in completely unknown environments as well as in (partially) known environments. Paths are always computed offline, which means that our approach calculate paths to given goals or to unexplored areas a priori and the UAVs follow them. If obstacles appear, which lie in the trajectory of any UAV, a recalculation is triggered.

For example in [7] no task allocation is done and the UAVs are considered to be homogeneous (at least in terms of exploration capability). In contrast, the UAVs in this work are considered to be heterogeneous and, in particular, some tasks can be solved only by a matching UAV. Additionally, in this work exploration of some areas is needed repeatedly as data can become obsolescent over time.

In contrast to most path planning approaches, which focus on the optimization of the path length or where the tests always end after one single exploration of the terrain, we try to lower the time for exploration of a terrain. In our work in some circumstances detours are allowed or even desired (Section V-A). The cases where detours are allowed are in goal-oriented mode. Taking a detour to explore some new terrain leads to an information gain, which can be important. To avoid paths where the detours increase the time needed to reach given goals too much the maximum length of the detours can be defined a priori (Section V-A). Additionally, as information can become obsolescent, explored terrain becomes unexplored again after some time.

A. Path planning basics

Path planning is a wide area of research. A lot of approaches to plan paths are described in the literature. But most approaches are based on a few basic methods. Most commonly used are the following three methods or a combination of them:

1) Roadmap method
2) Cell decomposition
3) Potential field theory

The roadmap method's basic idea is to create a roadmap, which reflects the connectivity of the free space. If such a roadmap exists, path planning is simple. Only the initial and the goal configuration must be connected to the roadmap. A feasible path is found by simply following the path in the roadmap, if both configurations can be connected to it. Otherwise, no feasible path exists. Kazemi et al. [10], e. g., combined the roadmap method with cell decomposition.

We combine cell decomposition with the potential field theory. For this a subdivision of the whole terrain into several subareas is done using a quadtree. Afterwards, each area gets a so-called potential value ϕ. Based on these values path planning can be done. A detailed description of this process is given in the further sections.

All methods are used to solve the basic problem to plan paths, which can be described as follows:

Let $\mathcal{A}$ be a UAV. It moves in a Euclidean space $\mathcal{W}$, represented as R^N (in our case $N = 3$). Additionally, $\mathcal{B}_1, \cdots, \mathcal{B}_r | \mathcal{B}_i \subset \mathcal{W}, (i = 1, \cdots, r)$ are fixed objects distributed in $\mathcal{W}$. Every single $\mathcal{B}_i$ is an obstacle. Assume that the geometry and the position of $\mathcal{A}$ is known. The $\mathcal{B}_i$ can be known but they don't have to be. Further we assume that $\mathcal{A}$ is not restricted in its movement through kinematic constraints. This leads to the following problem:

Given an initial position and direction plus a goal position and direction of $\mathcal{A}$ in $\mathcal{W}$, calculate a path τ consisting of a consecutive sequence of positions and directions of $\mathcal{A}$ under avoidance of contact with any $\mathcal{B}_i$. The path has to start at the initial position and must end at the goal position. Return an error, if such a path does not exist.

All path planning problems have in common that a path from the initial configuration to the goal configuration must be found. This leads to the concept of the configuration space [11].

B. Configuration space

The configuration space $\mathcal{C} \subset R^n$, with n describing the dimension of $\mathcal{C}$ is used to represent the UAV $\mathcal{A}$ within a terrain $\mathcal{W}$. A configuration $q \in \mathcal{C}$ is a tuple $q_1, \cdots, q_n$. The n independent variables can, e. g., represent the position, the role and tasks of a UAV. $\mathcal{A}$ comes with a specific role and tasks, which have to be fulfilled. In our work $\mathcal{A}$ is represented as a point $p \in \mathcal{W}$. The point p is also the position of $\mathcal{A}$ in $\mathcal{W}$. So, UAV $\mathcal{A}$ has a configuration $q_{\mathcal{A}} \in \mathcal{C}$ that consists of all relevant properties of $\mathcal{A}$, e. g., the current position p and its direction. Following the definition of Barraquand et al. [11] one can consider a geometrical application, which maps any configuration q to the point p in the terrain. This map

$$\begin{array}{rccl} X & : & \mathcal{C} & \rightarrow & \mathcal{W}, \\ & & q & \mapsto & p = X(q) \end{array}$$

is called the forward kinematic map. $\mathcal{C}$ is the union of all possible configurations q.

A Cartesian coordinate system $\mathcal{F}_{\mathcal{W}}$ is embedded in $\mathcal{C}$. In our work a configuration of $\mathcal{A}$ is the specification of the position and direction of $\mathcal{A}$ in consideration of $\mathcal{F}_{\mathcal{W}}$, plus its role, fuel level, maximum and favorite height and its tasks. The position of the UAV in $\mathcal{W}$ is called $p_{\mathcal{A}}$. Due to consistency $\mathcal{C}$ is discretized like the real space using cell decomposition.

1) Occupied space: The workspace of $\mathcal{A}$ contains a finite number of obstacles $\mathcal{B}_i$, $i = 1, \cdots, r$. Obstacles can, e. g., be skyscrapers, mountains or other UAVs, which lie in the trajectory of the UAV. Each obstacle $\mathcal{B}_i$ is in $\mathcal{CB}_i \subset \mathcal{C}$ of those configurations in which the UAV would take a position inside an obstacle $\mathcal{B}_i$. This means:

$$\mathcal{CB}_i = \{q \in \mathcal{C} | X(q) \in \mathcal{B}_i\}, \forall i = 1, \cdots, r.$$

Configurations including these positions are forbidden as they lead to damage of the UAVs.

2) Free space: Due to the existence of obstacles the UAVs are not able to take every position in $\mathcal{C}$. With the definition of $\mathcal{CB}_i$ it is possible to calculate the allowed configurations of the UAVs. This subset of configurations in which the UAVs do not take positions within an obstacle is called free space. This means:

$$\mathcal{C}_{free} = \mathcal{C} \setminus \bigcup_{i=1}^{r} \mathcal{CB}_i.$$

Every configuration $q \in \mathcal{C}_{free}$ is called a free configuration. Using this modeling a collision-free path from a start configuration q_{start} to a goal configuration q_{goal} is a consecutive map $\tau : [0,1] \to \mathcal{C}_{free}$ with $\tau_0 = q_{start}$ and $\tau_1 = q_{goal}$.

C. Cell decomposition

The main idea of cell decomposition is the subdivision of $\mathcal{C}$ into disjoint areas, called cells or leaves. An important part is the connectivity graph G. It captures the structure of $\mathcal{C}$. Each subarea is represented through a leaf. A distinction between exact and approximative approaches is made. We use an approximative approach due to cost reductions when dynamic changes take place.

The subdivision is made through the introduction of an internal data structure. This structure has to contain all relevant data, like positions and dimensions of the subareas, positions of obstacles and so on.

One of the most commonly used approaches are grids. They subdivide $\mathcal{W}$ into a number of equally sized subareas. One disadvantage of this approach is that it does not take into account neighboring areas with equal properties, e. g., several neighboring areas, which are occupied space. Neighbors of a current area are all areas, which have one or more border points in common with the current one. For cost reduction it is more efficient to combine such neighboring areas to one big area, so that only one big area has to be taken into account for calculations.

Therefore we use a quadtree for the subdivision of $\mathcal{W}$. The quadtree divides the complete space into four subspaces of equal size. These subspaces are recursively divided into four subspaces. Subdivision stops when the enclosed space of a leaf is of homogeneous type or a pre-defined maximum breakdown is reached. Homogeneous means that the space consists of only one type, e. g., occupied space. If the given breakdown is reached and the space is not homogeneous, an approximation is done: Every leaf which includes occupied space will be marked as occupied space. Otherwise, if the space includes different types of $\mathcal{C}_{free}$, it is marked with that type of space, which constitutes the largest part of the leaf. After subdivision, the complete terrain is represented through the leaves of the quadtree, i. e., the union of all leaves represents $\mathcal{W}$.

One disadvantage of a simple quadtree in contrast to a grid is, that finding neighbor leaves is more costly. It takes logarithmic time instead of constant time when using a grid. This becomes more important as neighbor finding is one of the operations needed most. It is necessary for potential calculations (Section V-B) and for finding paths (Section VI).

To take this into account, we extended the quadtree to a linear quadtree [12]. For transforming a quadtree into a linear quadtree each node of the tree gets a unique code and its level is saved. With this information it is possible to find neighboring nodes with constant time complexity on average.

The approach of cell decomposition is combined with the potential field theory in such a way, that a so-called potential value ϕ is assigned to each leaf of the quadtree.

D. Artificial potential fields for motion planning

The idea of using artificial potential fields for motion planning was introduced in 1985 by Khatib [2]. Using this technique a manipulator moves in a field of artificial forces. The idea is that paths can be obtained through linear superpositions of these fictitious forces or their potentials ϕ, which affect the UAV. The positions to be reached are modeled as attractive poles and obstacles are modeled as repulsive poles for the UAVs. So, a field of forces that affects the complete terrain has to be realized. This can be done using potential values. Therefore, the terrain is divided into sub-terrains, which get potential values. The difference of the potentials of two neighboring areas can be used to model forces. Two types of forces based on two types of potentials exist:

- Attractive forces, which represent goals and pull the UAVs towards them.
- Repulsive forces, which represent obstacles and push the UAVs away from them.

For a proper modeling of potential fields these attractive forces must fulfill two requirements:

1) they have to affect the whole configuration space and
2) they must always lead to the goal.

Figure 1 shows an example for an attractive potential. The goal to be reached is in the middle of the potential field shown in Figure 1 and all surrounding potential values are higher. So, following the descent gradient will always lead to the goal.

Suppose that the first condition is not fulfilled. Then there would exist areas in which no forces act. If the UAV started in such an area, it would never move. Another problem occurs if a UAV flies into such an area. In that case the UAV would usually stop flying. Reaching the goal is not possible then.

The second condition is crucial for reaching the goal, too. If there exist forces, which do not lead to the goal, it would be possible that the UAV selects a trajectory, which makes it impossible to reach the goal. Several possibilities to guarantee forces as demanded exist. One possibility is to

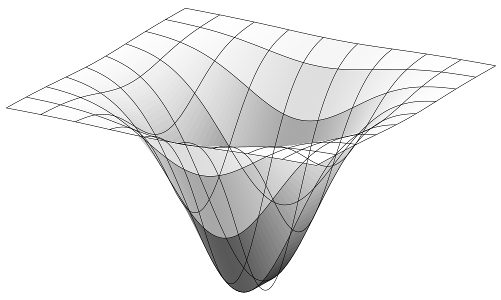


Figure 1. An example for an attractive potential.

adjust the potentials like a cone. The determining value can, e. g., depend on the Euclidean distance to the goal, as it is shown in Figure 1.

Repulsive forces have to fulfill two requirements, too:

1) they have to affect only a specified surrounding area of the obstacle and
2) they must always lead away from the obstacle.

Figure 2 shows an example for a repulsive potential. The obstacle is in the middle of the potential field shown in Figure 2 and all surrounding potential values are less than that of the obstacle. So, following the descent gradient will always lead away from the obstacle.

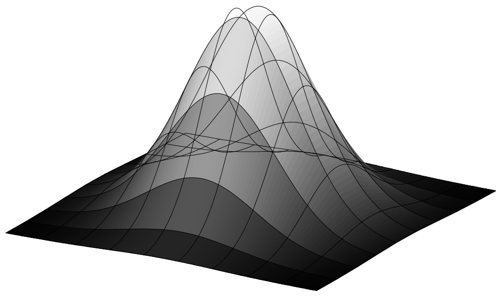


Figure 2. An example for a repulsive potential

The first condition is needed to ensure that obstacles do not affect the complete configuration space. An effect on the complete configuration space can affect the trajectories of the UAVs in such a way that they take unnecessary detours. The second condition ensures that UAVs will not be pulled toward obstacles, which would lead to crashes.

To ensure the two conditions repulsive potentials can be based, e. g., on the Euler's number, as it is shown in Figure 2.

In such a field of forces a path follows the descent gradient within the potential field. It keeps a UAV away from obstacles and pulls it towards the goal.

However, there are some problems when using such potential fields, especially in case of complex and highly dynamic environments. The basic movement in such a field is often done using a gradient method. In this case a UAV is pulled toward the goal. However, obstacles lying around the UAVs trajectory may cause the occurrence of so-called local minima. These minima are one of the most important problems.

A local minimum occurs if a single potential value is less than all surrounding potential values. In that case the driving force vanishes and the UAV gets trapped. Several methods to avoid and to get out of these minima exist. We use harmonic functions [13] for calculating one global artificial potential field. This avoids the generation of local minima.

E. Harmonic Functions

Using harmonic functions for potential value calculation was done first by Connolly and Burns [13] in 1990. One of the advantages of these functions is that one can prove that local minima can be avoided. Additionally, consistent and collision-free paths can be calculated. Harmonic functions satisfy Laplace's equation in n dimensions:

$$\nabla^2 \phi = \sum_{i=1}^{n} \frac{\partial^2 \phi}{\partial x_i^2} = 0.$$

Namely, ϕ must be two-times differentiable and the second partial derivatives of the potential ϕ must be zero. Additionally, ϕ must be strictly increasing, e. g., dependent on the distance to the goal. The value of ϕ is given on a closed domain Ω in the configuration space $\mathcal{C}$ (Section IV) and satisfies the min-max principle[1] and the uniqueness principle[2] [14], [15]. The min-max principle means that the potential function has its maximum and minimum values at its boundary points (in our case obstacles and goals). So the min-max principle can guarantee that no local minimum can have a potential value less than that of a global minimum and no saddle point can have a greater potential value than an obstacle. The uniqueness principle means that no two areas have the same potential value. This guarantees the absence of flat regions.

Connoly [16] showed that it is possible to generate paths without spurious minima. Combining harmonic functions with Dirichlet's boundary conditions leads to a value-restricted configuration space, which is important for calculations with discrete arithmetic. To respect Dirichlet's

[1]A harmonic function $f(x, y)$ defined on a closed domain Ω takes its minimum and maximum values on the boundary.

[2]If $f(x, y)$ and $g(x, y)$ are harmonic functions defined on Ω such that $f(x, y) = g(x, y)$ for every bound point, then $f(x, y) = g(x, y), \forall x, y$

boundary conditions we bound goal areas with the potential value 0 and occupied areas with the value 1.

Every harmonic function defined on a compact region $\Omega = \partial\Omega \cup \Omega$ satisfies three properties [13]:

1) Every harmonic function is analytic.
2) Select a point q_d to be a goal point with the constraint that $\phi(q_d) = 0$. Set all obstacle boundary points p to some constant $\phi(p) = c$. All harmonic functions satisfy the min-max principle, so ϕ is polar. That means that q_d will be the point at which ϕ attains its minimum value at Ω and all streamlines lead to q_d.
3) If c is set to 1, ϕ will be admissible in our sense. This is a simple normalization.

Additionally, harmonic functions have several valuable properties [13]:

- Completeness up to discretization error
- Fast surface normal computation
- Ability to exhibit different modes of behavior (grazing vs. avoidance)
- Robust control in the presence of unanticipated obstacles and errors
- Lack of spurious local minima
- Linear superpositioning
- Robustness with respect to geometrical uncertainty
- Continuity and smoothness of configuration space trajectories

Completeness in terms of path planning is given if one can guarantee that a path from each initial position to each goal position will be found if such a path exists. It is possible that a coarse discretization disables the finding of a path as shown in Figure 3, where the grey area represents an obstacle. Each leaf, which contains a part of the obstacle is treated as occupied space to ensure collision avoidance. Using a fine discretization as shown in Figure 3(a) makes a path planning from the start to the goal position possible. In Figure 3(b) the coarse discretization makes it impossible to find a route around the obstacle. This discretization errors cannot be compensated by harmonic functions.

Originally, harmonic functions were used for path planning only with a fixed map and one single known goal. But one can show that they also are well suited to dynamic environments with multiple UAVs and goals.

One problem of harmonic functions is their superpositioning. As Connolly mentioned [13], there is no guarantee that in complex or dynamic environments obstacles are avoided when superpositioning is used. The potential values in the neighborhood of an obstacle depend not only on that obstacle's potential, but also on every other obstacle's or possible goals' potentials. If the configuration or the strength changes, the path of a UAV can get infinitely close to the obstacle. The only structure that can be safely modeled as obstacle this way is a point itself, which cannot get affected.

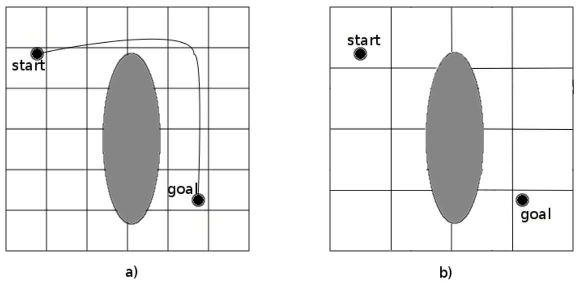


Figure 3. The figure shows two different discretizations of some terrain, including a grey area, which symbolize an obstacle. While in Figure a) a path from the start position to the goal position can be found, using the discretization in Figure b) it is not possible to plan a path from the start to the goal position.

III. Overview of the approach

This section gives a basic overview of our approach. For financial reasons the approach was implemented first as a simulation environment. The detailed description is given in the following sections.

In our work the presented path planning approach is considered to be a centralized one, in terms of using one global configuration space $\mathcal{C}$ to model the world in which the UAVs act. Path planning itself can be done in a centralized manner using a control station or in a decentralized way by the single UAVs.

On the one hand, UAVs have to explore unexplored parts of the terrain, while objects and landmarks must be recognized. Their position and size has to be noticed. Additionally, the UAVs must keep track of regions already visited to plan efficient paths to still unexplored regions of the territory. On the other hand, UAVs have to find collision-free paths from their initial positions to given goal positions. In both cases they act autonomously without any human interaction (except insertion and deletion of goals).

According to [17], it is possible to divide path planning into three general categories:

1) global path planning: A complete path from the position of the UAV to the goal will be computed.
2) local path planning: Several paths which lead towards the goal are planned and compared with each other.
3) reactive path planning: Only the next step is computed dependent on the current situation.

This work presents an approach combining global and reactive path planning for exploratory navigation. Additionally, the global approach is used to find paths to designated goal positions.

To ensure an efficient goal-oriented path planning using multiple UAVs, a coordination of the UAVs is indispensable. The coordination is done using a task allocation system. To ensure this, each UAV has a so-called role (Section IV-B) and schedules the given goals, based on this role.

To explore the terrain and to gain information about it, each UAV has one or two cameras. Different types of cameras, like infrared cameras, are used.

The simulation system currently used to evaluate our approach consists of two types of components, representing a single control station and the UAVs, respectively. Both make use of the same path planning library. The library contains the gradient method as well as the A*-algorithm and a neighbor finding algorithm. Figure 4 shows an overview of the system.

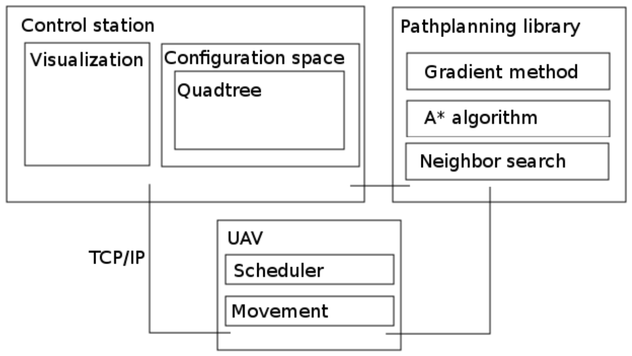


Figure 4. Overview of the designed system.

A. Control station

A control station was realized to represent the configuration space $\mathcal{C}$, which holds the data needed for path planning, including the properties of each UAV and the terrain, like positions of known obstacles, explored and unexplored space, position of the single UAVs, etc.

To distinguish between $\mathcal{C}_{free}$ and $\mathcal{CB}_i$ a subdivision of the terrain is needed. We use a quadtree, which is one of the most commonly used data structures to subdivide 2D terrains. By extending the quadtree to a linear quadtree we are able to find neighbors and to update the structure dynamically in constant time. Dynamic changing of the quadtree is needed to respect changes of the terrain due to exploration and aging of the information about subareas. The quadtree was designed in such a way that the union of the areas represented by all its leaves is the complete terrain.

Based on this subdivision it is possible to use the potential field theory for path planning. Therefore, each leaf gets a potential value, dependent on its distance to goals and obstacles.

The potential values ϕ are used to calculate paths, combining them with a gradient method for exploratory navigation and the A*-algorithm for goal-oriented path planning. For path planning only 2 dimensions are used. With respect to the different camera types of the UAVs, which have different properties, the third dimension, the height of the UAVs over ground is calculated by themselves.

The control station comes with a visualization of the terrain and the UAVs. For this purpose a 3D environment was implemented. It shows the positions of the single UAVs and is able to display their configurations. To display the terrain heightmaps are used. A heightmap is a monochrome picture where the value of each pixel represents the height of a single area of the terrain. The advantage of using heightmaps is that new terrains can be created easily. Additionally, the visualization allows an interaction between the user and the system. It is possible to set up new goals or obstacles and to delete them dynamically during simulation.

B. UAV

The UAVs were implemented as an external program. They consist of the UAV configuration $q_{\mathcal{A}}$ and schedule given goals, based on their role. Additionally, they are able to plan paths, based on a copy of $\mathcal{C}$, they can ask for. Hence, path planning can be decentralized through the UAVs or centralized, as a UAV can ask the control station to plan a path from the UAV position to each goal position.

The UAVs communicate with the control station using TCP/IP. They send their configuration periodically to the control station and can ask for path planning, a copy of the configuration space and additional information.

In our model, a UAV is unable to fly backwards, so it must be able to plan its motion, based on its direction and the goal direction. The movement calculation of a UAV for flying along the calculated path is done by itself.

Additionally, the UAVs have to calculate their height over the terrain. Each UAV has a favorite height with respect to its cameras. The resulting height over ground depends on a UAVs favorite height, the height of the terrain it is currently over and the height of the terrain it reaches next, as it is known so far. Considering also the height of the next terrain leads to a smoother flight when flying over terrains with huge height differences, if for instance steep faces lie in the trajectory of a UAV.

C. Pathplanning library

We developed a library, which can be used by the control station and the UAVs to calculate a path. This ensures that both a centralized path planning by the control station and a decentralized path planning by the single UAVs can be done. Using a library avoids a redundant implementation of the algorithms in the control station and the UAV. Therefore, the library implements the gradient method for exploratory navigation and the A*-algorithm for goal oriented path planning. Additionally, the neighbor finding algorithm is implemented, as it is needed to calculate paths.

IV. Construction of the configuration space

This section gives a more detailed description of the configuration space. We start with the cell decomposition approach. Afterwards, we present the configuration of a UAV, which is necessary for path planning.

In order to represent the complete scenario, the configuration space has to contain the coordinates and identities of all UAVs, goals, obstacles, known and unknown areas. This section, therefore, describes how the complete area is discretized into leaves using a quadtree. Afterwards, it is presented how additional information regarding the UAVs, like fuel level or individual tasks, are represented. Finally, the section discusses how the recalculation complexity of $\mathcal{C}$ can be decreased to a level at which online calculation is feasible.

A. Quadtree

As mentioned before, the configuration space is divided by a quadtree and each UAV is represented as a point in $\mathcal{C}$. The subdivision is done in several steps. First $\mathcal{C}$ is divided into free space $\mathcal{C}_{free}$ and occupied space $\mathcal{CB}_i$. When representing the UAV as a point in $\mathcal{C}$ the dimensions of the UAVs are not considered. However, to ensure a collision-free path it is necessary to consider their dimensions. This is done using obstacle growing, whereby the occupied space is expanded by the dimensions of the UAVs in all three dimensions.

Finally, the resulting $\mathcal{C}_{free}$ is subdivided into unknown, unexplored, explored and goal space. These space types are represented through different leaf types. This subdivision changes dynamically during runtime.

Note the distinction between unknown and unexplored space. Unknown space represents areas with no information about the terrain height. Unexplored space represents areas without up-to-date information. Hence, during a rescue mission the age of some data is of crucial importance. If the exploration time of a region exceeds a pre-defined threshold, the according region becomes unexplored again, but it can never become unknown again.

The resulting nodes store several properties as shown in Table I.

Property	Description
boundaries	Points to specify the position and the dimensions of the node.
space type	The type of space of the node, e. g., $\mathcal{C}_{free}$ or $\mathcal{CB}_i$.
location code	A unique code, used for neighbor finding with constant time complexity on average.
level	The level of the node (depending on the tree-depth of the node).
potential value	The potential value on which path planning is based.
exploration time	Time point at which the node was explored last time.

Table I
PROPERTIES OF THE QUADTREE NODES

To specify the positions of UAVs and obstacles, a Cartesian coordinate system $\mathcal{FW}$ was embedded into the configuration space $\mathcal{C}$. Hence, each UAV has an associated position and direction within the coordinate system.

The position and dimensions of each node are based on $\mathcal{FW}$, too. Each node is specified through 2 points in space spanning a rectangular region. When extending the quadtree to a linear quadtree a unique code and its level must be saved. In rescue scenarios up-to-date information is necessary. The exploration time is the point in time the node was explored. If the difference between the current time and the exploration time is above a given threshold, the node becomes unexplored again.

B. UAV configuration

Each UAV $\mathcal{A}$ has a configuration $q_{\mathcal{A}}$ within $\mathcal{C}$. It is represented as a 9-tuple $(q_1, \ldots, q_9)$ as shown in Table II.

Tuplenumber	Description
q_1	X - position in space
q_2	Y - position in space
q_3	Z - position in space
q_4	flight direction
q_5	favorite height
q_6	maximum altitude
q_7	role
q_8	task list
q_9	fuel level

Table II
UAV CONFIGURATION ($q_{\mathcal{A}}$)

The position of a UAV in $\mathcal{C}$ is described by three entries and must be unique, because if two UAVs had the same position in $\mathcal{C}$, they would have crashed. For path planning only two entries are used (q_1, q_2). The height of the UAVs (q_3) is neglected as the UAVs can have different favorite heights over ground. Of course the height is important to avoid collisions. It is also important for the cameras of the UAVs. The area the UAVs can explore at one moment depends on the flare angle of their cameras and the height of the UAVs above the terrain.

We distinguish two different height types. The favorite height (q_5) is the height above the underlying terrain a UAV wants to reach. The maximum altitude (q_6) describes the height of the UAV above sea level. In the following simulations (Section VII) sea level is modeled as the zero point of the program. The height of a UAV (q_3) is always the height above sea level.

The maximum altitude is used to partition the terrain into $\mathcal{C}_{free}$ and $\mathcal{CB}_i$. Each part of the terrain, which is higher than the maximum altitude, is $\mathcal{CB}_i$, the remaining terrain is $\mathcal{C}_{free}$.

The configuration of a UAV contains the flight direction (q_4). This direction can also be used to provide a preferred direction in which the UAV will fly first while exploring the terrain. The modeled UAVs can turn on the spot so the direction is not too important for motion planning. Previous tests showed that in this case the use of a preferred direction does not lead to better results concerning the exploration rate or speed. Hence, there is no favorite direction for the UAVs in the resulting system. Two UAVs cannot take the same position at the same time so the entries q_1, q_2 and q_3 are used to distinguish the single UAVs.

The task list includes all given goals. The types of goals specified so far are shown in Table III. A single task always contains one known goal. It can be UAV-specific or non-UAV-specific, e. g., refueling would be a UAV-specific task, whereas monitoring of areas would be a non-UAV-specific one. Goals can be points, which the UAV has to explore or areas, which have to be monitored with high priority. Each goal is associated with one task. When the fuel level reaches a given threshold, the UAV sets up a new task with a fuel station as goal and highest priority. This task will be executed before all other tasks to avoid damage.

Every UAV also has a role, which is used to realize a scheduling of all given tasks. In this work three roles are defined:

1) *explorer*: An explorer UAV is used mainly for terrain exploration. Goal-oriented path planning will be done only to avoid starvation.
2) *seeker*: A seeker UAV is used mainly to reach given goal points. As long as goals exist goal-oriented path planning will be done and exploration takes place only if no specific goals exist.
3) *surveillant*: A surveillant UAV is used mainly for goal-oriented path planning, too. The difference to the seeker UAV is that it monitors areas rather then flying to given goal points.

C. Decentralized task allocation for multiple UAVs

A role is associated with a specific priority scheme. Each task gets a priority $\mathcal{P} \in \{0, 1, \cdots, 9\}$. The higher $\mathcal{P}$ is the more important the task is. Depending on the role, the task list will be sorted according to the different task priorities. Explorer UAVs first explore the terrain and target specific goals with less priority. Seeker UAVs favor goal points over areas and exploration. Surveillant UAVs favor goal areas over points and exploration. The priority schemes of the seeker and the surveillant roles initialize tasks like goal points or goal areas with high priority to favor these tasks over exploration. Table III shows the implemented scheduling scheme.

Goal	Explorer	Seeker	Surveillant
Monitor areas	2	2	3
Exploration	3	0	0
Fly to goal point	1	3	2
Refueling	9	9	9
Landing	8	8	8

Table III
INITIAL PRIORITIES OF TASKS

The higher the priority the more important the task becomes. If several tasks have equal priority, the processing order of these tasks depends on the position of the corresponding goals. Tasks with goals closer to the UAV are executed first. This ensures fast completion of the tasks and additionally, they are scheduled in such a way that tasks with less distance to each other than to other tasks but equal priority will be processed consecutively.

We distinguish two types of tasks. High priority tasks ($\mathcal{P} > 4$) and low priority tasks ($\mathcal{P} \leq 4$). High priority tasks are tasks, which must be processed as fast as possible as not-processing them will lead to damage of the UAV.

However, this static role assignment does not avoid starvation. That means that if, e. g., only surveillant UAVs are in use and new tasks to monitor goal areas appear faster than the old ones can be finished, flying to goal points would never be executed. To avoid such a behavior the priorities of low priority tasks increase during time. To ensure that these tasks never preempt high priority tasks the increasing stops if $\mathcal{P} = 4$, which is higher than every initial priority of low priority tasks.

Scheduling of tasks, assignment and increasing of the priorities and creation of UAV specific tasks is always done in a decentralized manner by the UAVs.

V. CALCULATING THE POTENTIAL FIELD FOR PATH PLANNING

In this section we describe how the potential values are calculated. This is a two-step procedure. We start calculating a first approximation. Afterwards, we successively enhance the results of the first approximation until we get potential values, which are feasible for path planning. Our approach for potential field calculation has the disadvantage of being quite costly. Hence, to lower the calculation costs in such a way that we can make every calculation without noticeable delay we use several cost reduction techniques, which are described at the end of the section.

We assume that the control station, responsible for coordinating the UAVs, provides a single configuration space $\mathcal{C}$, which is used as a representation of the complete scenario. $\mathcal{C}$ is based on a quadtree (Section IV-A), whereby the leaves carry descriptive information, like a potential value. The quadtree subdivides the terrain in such a way that the union of the leaves represents the complete terrain. Each leaf of the quadtree gets one single potential value. Based on these values, the UAVs decide which leaves they use to find a trajectory to their designated goal position.

Additionally, $\mathcal{C}$ includes all UAVs with their configurations (Section IV-B), every goal and obstacle.

The intention is to maintain only one harmonic function, which describes the entire potential field, concerning multiple goals as well as multiple UAVs, avoiding the negative effects of linear superpositioning of harmonic functions described in [13].

Therefore we model the potential field as a discrete Dirichlet problem for harmonic functions, which always has a solution and we try to solve it using a relaxation technique. As a relaxation technique to transform the potential field update under Dirichlet boundary conditions the Gauss-Seidel method [18] is used.

To make use of the advantages of harmonic functions, in particular the absence of local minima, the potential values of the configuration space have to fulfill several requirements, e. g., potential values must strictly increase with the distance to the goals, the second partial derivatives have to vanish and so on. To take into account all these restrictions, the calculation process is modeled as an optimization problem to solve the discrete Dirichlet problem.

A. First Approximation

We distinguish two types of potential values ϕ. Bound and unbound values. Bound values are fixed values for goal and obstacle areas, which are set before the calculation starts and they will not be changed during the calculation. These values are the boundaries. Thereby we are able to ensure that obstacles have maximum and goals minimum values as harmonic functions satisfy the min-max principle. We use a potential value range from 0 to 1. Obstacles are bound with 1 and goals with 0.

Every other leaf $u(x, y)$ of the quadtree gets a so-called unbound potential value $\phi(x, y)$, which is greater than 0 and less than 1. These values are calculated for each leaf that is neither obstacle nor goal space. Therefore, we use the following equation. It depends on the distances to the nearest goal area, in sense of Euclidean distance.

$$\phi(x, y) = \xi \cdot \frac{\log\left(\tau\left(x, y\right)\right)}{\log(d)}. \quad (1)$$

Here $\tau(x, y)$ represents the Euclidean distance from the point $(x, y) \in \mathcal{C}$ to the nearest target point. The logarithm of the diagonal d of the complete terrain is used to normalize the values between 0 and 1. $\phi(x, y)$ satisfies Laplace's equation.

For goal-oriented mode it is possible to trade off shortest path calculation and gathering additional information by taking a detour over unexplored regions. To take this into account the ξ-Value was introduced. The value is between 0 and 1 and is set by the user. The lower the value of ξ is, the more attractive unexplored terrain becomes for the path planner. In case of non-goal-oriented path planning (exploratory navigation) ξ is set to 1.

As mentioned, Equation 1 is used to calculate a first approximation for every unbound potential in the environment. It is only a first approximation because not every bound value was respected. This leads to a linear system of equations.

B. Update equation

The first approximation does not consider obstacle areas. But that is necessary to ensure collision avoidance and to guarantee that the potential values are represented through a single harmonic function with its advantages, like reducing the number of local minima. To take this into account the first approximation of the unbound values given by Equation 1 is successively enhanced using a relaxation method. In our work the Gauss-Seidel algorithm is used for relaxation.

Numerical solutions for Laplace's equations can be found through finite differentiation methods. This is possible as the value of a harmonic function at each point is the arithmetic mean of the values of its surrounding points [15]. We use the neighbors of the following four of the existing eight neighbor directions to update the potential value of a leaf $u(x_i, y_j)$:

1) $u(x_{i+1}, y_j)$,
2) $u(x_{i-1}, y_j)$,
3) $u(x_i, y_{j+1})$,
4) $u(x_i, y_{j-1})$.

If we had only equally sized nodes, e. g., when using a grid with equal sized cells, we could calculate the potentials $\phi(x_i, y_j)$ using the following function:

$$\begin{aligned}\phi(x_i, y_j) = {} & \tfrac{1}{4}[\phi(x_{i+1}, y_j) + \phi(x_{i-1}, y_j) \\ + {} & \phi(x_i, y_{j+1}) + \phi(x_i, y_{j-1})].\end{aligned}$$

However, when using a quadtree to partition $\mathcal{C}$ as in our approach, the dimensions of the part of the terrain each leaf represents have to be considered. Another point is that through the use of a quadtree a leaf can have several neighbors per direction. In that case we make an approximation and use the mean distance and the mean potential value of the neighbor nodes. We take into account the following distances from the current leaf $u(x_i, y_j)$ to its used neighbors:

τ_{right}:	distance from $u(x_i, y_j)$ to $u(x_{i+1}, y_j)$,
τ_{left}:	distance from $u(x_i, y_j)$ to $u(x_{i-1}, y_j)$,
τ_{up}:	distance from $u(x_i, y_j)$ to $u(x_i, y_{j+1})$,
τ_{down}:	distance from $u(x_i, y_j)$ to $u(x_i, y_{j-1})$.

This leads to the following update equation, which is based on [19]:

$$\begin{aligned}\phi(x_i, y_j) = {} & \frac{\tau_{up}\tau_{down}\left[\tau_{right}\phi(x_{i+1}, y_j) + \tau_{left}\phi(x_{i-1}, y_j)\right]}{\tau_{up}\tau_{down}(\tau_{right} + \tau_{left}) + \tau_{right}\tau_{left}(\tau_{up} + \tau_{down})} \\ + {} & \frac{\tau_{right}\tau_{left}\left[\tau_{up}\phi(x_i, y_{j+1}) + \tau_{down}\phi(x_i, y_{j-1})\right]}{\tau_{up}\tau_{down}(\tau_{right} + \tau_{left}) + \tau_{right}\tau_{left}(\tau_{up} + \tau_{down})}. \quad (2)\end{aligned}$$

Neighboring leaves are all leaves, which have a border in common. Hence, possibly different leaf sizes are also taken into account. The potential value of the current leaf is represented through $\phi(x_i, y_j)$, the potential of the left neighbor through $\phi(x_{i-1}, y_j)$ and so on.

This modeling leads to a system where the UAVs are driven away from obstacles and led to the goals. Figure 5 shows a sample potential field, calculated using this optimization method. For better visualization the calculated potential field was mapped to a grid in Figure 5. Here the UAV is in the left upper corner, the goal to be reached is in the lower right corner. The higher parts of the terrain represent occupied space, whereas the lower ones are free space. The UAV will now follow the descent gradient, which always leads it to the neighboring space with the lowest potential value, until the goal is reached.

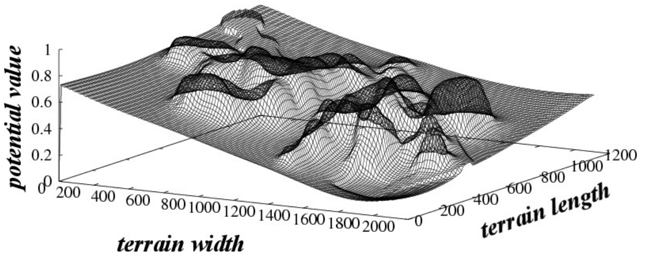


Figure 5. A sample potential field

To reach a given error rate with respect to a perfect harmonic function of $\epsilon \leq 10^{-p}$ when using M leaves for calculation, $\mathcal{O}(pM)$ iterations are needed. Intuitively, choosing a quadtree instead of a grid lowers the calculation costs. However, for large terrains, together with the demand for detailed resolution, this method is still very costly. Several methods reducing these costs are presented in the following section.

C. Cost reduction

The methods presented in the previous sections ensure complete exploration, reaching of given goals and task allocation. These methods are rather complex, especially the recalculation of $\mathcal{C}$. A first approach for cost reduction was an intelligent implementation (Section V-D). When calculating the first approximation a pointer array is created, which holds the leaves to be updated with pointers to their corresponding neighbors. So, iterative loops through this array are used for the update equation, instead of recursively cycling through the quadtree with all nodes. Saving pointers to the neighbors avoids redundant neighbor finding as the quadtree does not change during potential calculations.

However, without any cost reduction techniques the methods still do not well suit embedded systems. Another part for cost reduction is to make recalculations of $\mathcal{C}$ event based. So, recalculation will be done only for the following reasons:

- Insertion of new targets or obstacles
- Deletion of targets or obstacles
- Path-planning request after exploration

After a UAV changed the configuration space through exploration it requests a new path. In that case the central control station recalculates the potential values of the global configuration space. Obviously, even this leads to frequent recalculations of $\mathcal{C}$. In addition, the calculation load depends directly on the number of active UAVs. Therefore, the recalculation step must be performed efficiently.

The update equation is used mainly to lower the number of local minima. A complete calculation of the configuration space means in our case that no local minima are left. The lowering of the number of local minima is advantageous since trajectories are calculated in such a way that they lead to minima, even if they are only local minima. For the UAVs this results in a detour or in the worst case the UAVs get trapped in a local minimum. Leaving a local minimum is costly as it has to be detected and a safe path out of the minimum has to be calculated.

The costs to calculate the configuration space are directly related to the number of nodes to be updated. Hence, a quadtree was used instead of a grid, which is very common for representing a terrain when using potential fields. To reduce the costs for neighbor finding in the tree–which is necessary for the update equation and the path planning methods–the tree was extended to a linear quadtree. To transform a tree into a linear quadtree a unique code and the level of a node is saved for each leaf. An algorithm, which finds neighbors with constant time complexity on average was implemented. This is a further improved version of [20] with the focus on reducing tree editing costs. With only one configuration space for all UAVs, it is crucial to have a data structure, which can be modified easily–usually this is done several times per second. The modification must be done much more frequently than the potential calculations to respect changes in the terrain, like newly recognized obstacles. In contrast to [20] no additional information about the level difference to the neighbors is stored in the nodes. This significantly increases the editing speed.

Even with this cost reductions, a complete calculation of the configuration space is expensive. Prestes et al. showed [4] that a complete absence of local minima is not necessary for feasible path planning. They used a constant iteration depth to reduce the costs. Based on their results we established break conditions that also allow a dynamic number of leaves. The iterative calculation stops if:

- the potential values remain unchanged,
- the number of local minima is less than 0.5% of the number of updated leaves,
- the iteration depth is greater than or equal to 10% of the number of updated leaves.

The last condition guarantees the termination of the update method. Tests showed that these break conditions lead to an appropriate tradeoff between number of local minima and calculation costs.

Furthermore, leaves with potential values greater than or equal to 0.999 are not considered as local minima. Tests showed that such values usually occur in the neighborhood of newly explored obstacles. Usually such potential values are greater than the potential value of areas, which the UAVs take. This leads to a significant reduction of the calculation costs.

If a UAV gets trapped in a local minimum, the A*-algorithm is used to leave it. Therefore, the A* searches through the quadtree to find an unexplored area. It takes the first area found and sets it as the next target. Afterwards, a goal-oriented path planning to this area will be done.

For further cost reduction in non-goal-oriented mode, an activation window was established. First all nodes are inactive. After the exploration of an area the newly explored leaves are marked as active. After a given time, leaves can be marked as unexplored again and are no longer active. The first approximation and the following updates will be done only for active leaves.

D. Implementation details

The implementation of the calculation algorithm consists of three main methods. The first method, which is shown as pseudo-code in Listing 1, is used to start the first approximation and to ensure an iterative accomplishment of the update equation until the break conditions are reached. As our first approximation is based on goal distances we have to set up goals even in exploratory navigation. Therefore, we calculate the nearest unexplored or unknown space for each UAV and set this spaces as next targets to be reached. This is done in the method *SetNearestUnexploredAsNextGoal*. While processing the loop for the update equation we check the number of local minima each cycle. Therefor the method *CheckNumberMinima* is called, which checks for each node to be updated if there exists at least one neighbor node with a lower potential value. If one of the break conditions introduced in Section V-C is fulfilled the while loop, which calls the update equation, stops.

```
CalculatePotentialField()
{
  if(exploratory_navigation)
    SetNearestUnexploredAsNextGoal()

  leaf_number = 0
  FirstApproximation(Rootnode)

  max_minima = update_list.length() * 0.005
  number_minima = max_minima + 1
  max_depth = update_list.length() * 0.1
  iteration_depth = 0
  value_change = 1

  while(number_minima > max_minima &&
       iteration_depth <= max_depth &&
       value_change)
  {
    iteration_depth++
    value_change = UpdateEquation()
    number_minima = CheckNumberMinima()
  }

}
```

Listing 1. Pseudo-code of the main function for potential field calculations.

```
FirstApproximation(Treenode)
{
  if(Treenode not leaf)
  {
    for(i = 0; i < 4; i++)
      FirstApproximation(Treenode.child(i))
    return
  }

  if(Treenode.space == Target)
  {
    Treenode.potential = 0
    return
  }

  if(Treenode.space == Obstacle)
  {
    Treenode.potential = 1
    return
  }

  if((Treenode.space == Unexplored ||
      Treenode.space == Unknown)   &&
      exploratory_navigation)
  {
    Treenode.potential = 0
    return
  }

  update_list[leaf_number] = Treenode
  update_list[leaf_number].potential = log(
       target_distance)/log(terrain_diagonal)

  if(Treenode.space == Unexplored ||
      Treenode.space == Unknown)
    update_list[leaf_number].potential *= xi

  SaveNeighborNodes()

  for each(direction)
  {
    distance[direction] = SumDirections()
    distance[direction] /=
      num_neighbors[direction]
  }

  update_list[leaf_number].udr = distance[up]*
       distance[down]*distance[right]
  update_list[leaf_number].udl = distance[up]*
       distance[down]*distance[left]
  update_list[leaf_number].rlu = distance[right]*
       distance[left]*distance[up]
  update_list[leaf_number].rld = distance[right]*
       distance[left]*distance[down]
  update_list[leaf_number].direction =
      update_list[leaf_number].udr
    + update_list[leaf_number].udl
    + update_list[leaf_number].rlu
    + update_list[leaf_number].rld

  leaf_number++
}
```

Listing 2. Pseudo-code of the function to calculate the first approximation

The method used to calculate the first approximation (Listing 2) has several additional requirements. We move recursively through the complete tree but only the leaves are considered. First, we check if a node consists of target space, occupied space or in exploration mode unexplored space or unknown space. If so, we bound the potential value of the leaf with the corresponding value. If not, we need to calculate an unbound value for the leaf. In that case we use the first approximation equation from Section V-A.

We profit from the fact that there are no quadtree changes during the calculations. We use this property in several ways. As it makes no sense to move through the complete quadtree in each update iteration we first set up a list, where we save pointers to the leaves to be updated later, called *update_list*. As there are no tree changes also the neighbors of the leaves do not change. That makes it possible to calculate them once and save pointers to the neighbor nodes using a method called *SaveNeighborNodes*. It is possible for a node to have multiple neighbors in some directions. In that case we use an approximation for the following update equation as we only save the mean distance to the neighbor nodes in each direction. Finally, we calculate all relevant distances needed for the update equation once and save them, too.

```
UpdateEquation()
{
  value_change = 0
  for(i = 0; i < update_list.length(); i++)
  {
    old_potential = update_list[i].potential

    for each(direction)
    {
      potential[direction] = SumPotentials()
      potential[direction] /=
  update_list[i].num_neighbors[direction]
    }

    new_potential  = update_list[i].udr *
         potential[right]
    new_potential += update_list[i].udl *
         potential[left]
    new_potential += update_list[i].rlu *
         potential[up]
    new_potential += update_list[i].rld *
         potential[down]

    update_list[i].potential = new_potential /
         update_list[i].distances

    value_change+= |old_potential - new_potential|
  }

  return value_change
}
```

Listing 3. Pseudo-code of the function to calculate the update equation

As we made several calculations, needed for the update equation, which is shown in Listing 3, once after calculating the first approximation, the implemented update equation itself is not as complex as it seems to be in Section V-B. We now can use the *update_list* to access the nodes to be updated directly. In case we have several neighbor nodes in one direction we made an approximation in such a way that we use the mean potential value of the neighbors for calculation. After that we just have to multiply the resulting potential values with the relevant distances from Equation 2, sum them, and divide the result through the given complete distance value. Finally, we calculate the difference between the old and the new potential value to check if the potential values change or if a break condition is fulfilled.

VI. Path Planning Approach

In this section we present our approach for path planning. We distinguish between reactive path planning for exploratory navigation and global goal-oriented path planning. Additionally, in case of goal-oriented path planning a coordination method is introduced.

In contrast to the calculation of potential values ϕ, the path planner consider all leaves in all eight directions of the current leaf.

A. Planning paths for single UAVs

Based on the potential field stored in the configuration space $\mathcal{C}$ as described in the previous section, discrete path planning is possible. Here two different approaches are utilized:

1) Gradient based path planning for exploratory navigation (non-goal-oriented)
2) The A*-algorithm for goal-oriented path planning

Since the UAVs are considered to be heterogeneous, e. g., in terms of camera or other equipment, each UAV must be able to reach every point of the terrain. Therefore, it is impossible to assign subareas of the terrain to a single UAV.

In goal-oriented mode sometimes an additional exploration should be done, even if it leads to detours. The length of the detours can be limited. Therefore, two ξ-values are used. One is used during the potential field calculation (Section V-A), the other by the A*-algorithm. Depending on the priority of the goal, the algorithm uses Equations 3 or 4 to calculate the costs (the selection is explained below).

$$f = f_{pre} + \phi(p_{new}) \cdot \xi + \frac{\tau_{goal}}{2d}. \qquad (3)$$

$$f = f_{pre} + \tau_{pre} + \tau_{goal}. \qquad (4)$$

The costs to fly to the current leaf are f. The current leaf is the leaf, which is currently considered by the path planner. The costs for the start leaf are set to 0. The less f is, the more attractive the leaf becomes for the path planner. Furthermore, f_{pre} describes the costs to reach the predecessor. The potential value of the current node is $\phi(p_{new})$ and for unexplored regions $\xi | 0 < \xi \leq 1$ is used to lower the costs even more, dependent on the role of the UAV ($\xi = 1$ for explored regions).

Additionally, the algorithm uses two distances for the calculation: an estimated distance from the current leaf to the goal leaf, τ_{goal}, and the actual distance from the predecessor to the current leaf, τ_{pre}.

Equation 3 is used, if the goals are of low priority ($\mathcal{P} \leq 4$). In this case the costs depend mostly on the potential values and unexplored regions become more attractive. The complete diagonal of the terrain d is used to normalize the distance values to values between 0 and 1. Dividing by the double diagonal lowers the values even more, which makes the potential values more important for path planning.

Equation 4 is used, if the goals are of high priority ($\mathcal{P} > 4$). In this case it is necessary to reach the goal as soon as possible. The equation achieves this by setting the costs in such a way that the shortest path will be computed.

B. Goal-oriented coordination of multiple UAVs

It is possible that multiple UAVs set the same goal as the next task to process. This behavior should not be completely avoided as a UAV, which sets the goal later than others, may be closer to it, which reduces the time to reach it. After a UAV reached the goal the others are informed that the goal was reached and they remove it from their lists.

However, if multiple UAVs, which are close to each other, select nearly simultaneously the same goal as next goal, this behavior is not efficient. In this case these UAVs would calculate nearly the same paths. Parallel exploration would neither yield considerable information gain, nor would the goal be reached significantly earlier. To avoid this undesired behavior an error handling was implemented. For the handling $dist_{UAV}$ denotes the distance between a subsequent UAV$_i$ and the position of the UAV$_1$, which did the first calculation for this goal at time T_1. Additionally, $dist_{G_1}$ represents the distance of UAV$_1$ to the goal at T_1. t_t denotes the minimum time UAV$_1$ needs to reach the goal. If a UAV calculates a path to the goal position G_1 at time T_1 and another UAV calculates a path to G_1 at time T_i as well, the error handling starts. This handling avoids the calculation of nearly equal paths. It takes into account the points in time T_1 and T_i as well as the distances $dist_{UAV_i}$ and $dist_{G_1}$ in the following way:

$$\left(T_i - T_1 < \frac{t_t}{2}\right) \wedge \left(\frac{dist_{UAV}}{2} < dist_{G_1}\right).$$

If the formula becomes true, a subsequent UAV_i sets the priority for this goal to 0 and moves it to the end of its scheduling list. Hence, even if the UAV, which has set the goal first is not able to reach it, no starvation of that goal would occur.

A more simple way would be to avoid multiple path planning to a single goal. But a goal should be processed as soon as possible and as already mentioned, the first UAV, which makes a path planning, is not necessarily that one, which can reach the goal first. Goals with equal priority are sorted in an ascending order with respect to the UAV's distance to the goals, such that the possible detour to start flying to a goal, which is reached by another UAV first should be not quite as long.

The information about the used terrain can be obtained in two ways. One way is to get the data through third party applications like satellite images. Additionally, the UAVs have on-board cameras for the exploration of unknown areas and to update the information available so far.

VII. Results

For financial reasons the approach was implemented first as a simulation environment and several tests were done to check the costs for calculation and to check if it is able to fulfill all given requirements. The implementation consists of a control station, which models the world and visualizes it in 3D. It was implemented in C++ using OpenGL. Additionally, UAVs were simulated, which connect to the simulation environment using TCP/IP.

In this section we present several tests of the designed system. The focus of our tests was the exploration of terrains. Goal-oriented path planning with parallel exploration was neglected as those tests are presented in [21]. We mainly focus on the time needed to explore given terrains using different numbers of UAVs and the costs for the potential field calculations.

The used terrains were partitioned into $\mathcal{C}_{free}$ and $\mathcal{CB}_i$ in such a way that the UAVs were able to reach the complete free space.

The following tests were done to demonstrate that the system assures a complete exploration of a given terrain. Even by using only a single configuration space the use of multiple UAVs leads to faster exploration rates. Additionally, by using the cost reduction methods introduced in the previous section all calculations can be done online. For testing, a 3D simulation was created, which represents the terrain with the corresponding configuration space and the UAVs. The simulations were run on a desktop PC with a dual core 2.6 GHz CPU.

The tests include the calculation costs for $\mathcal{C}$. The time needed to calculate the configuration space during the different tests is shown in Figure 8(c), Figure 9(c) and Figure 10(c). Each calculation contains the following tasks:

- Assignment of the next terrains to be explored
- Bind goal space and occupied space
- Calculation of the distances to the next goal space
- Calculation of the first approximation
- Determining all neighbors of the relevant leaves
- Update of the potential values
- Check for break conditions

A. Used terrains

The tests were executed on two fictive maps represented as heightmaps (monochrome pictures, see Section III-A) with simulated UAVs. For exploratory navigation the following input parameter are relevant:

- Dimensions of the terrain
- Speed of the UAV
- Maximum altitude
- Size of the heightmap in pixels
- Maximum depth of the quadtree

- Distribution of $\mathcal{C}_{free}$ and $\mathcal{CB}_i$
- Number of UAVs

The first terrain (Figure 6) was represented through a heightmap with a resolution of 256×256 pixels. The underlying quadtree had a maximum resolution of 2×2 pixels per leaf, which led to a tree depth of 7. A terrain of $1000\ m \times 1000\ m \times 160\ m$ was simulated. The UAVs had a maximum altitude of $140\ m$. So, everything above $140\ m$ was treated as occupied space. In our test scenarios $\mathcal{C}$ was partitioned into $\mathcal{C}_{free}$ (72%) and $\mathcal{CB}_i$ (28%). The UAVs flew with a speed of $60\ km/h$ and had a favorite height of $40\ m$ over ground. To explore the terrain, the UAVs used a camera with a flare angle of $90°$. When using such a camera the UAVs were able to explore $5026.55\ m^2$ at one moment when they had reached their favorite height.

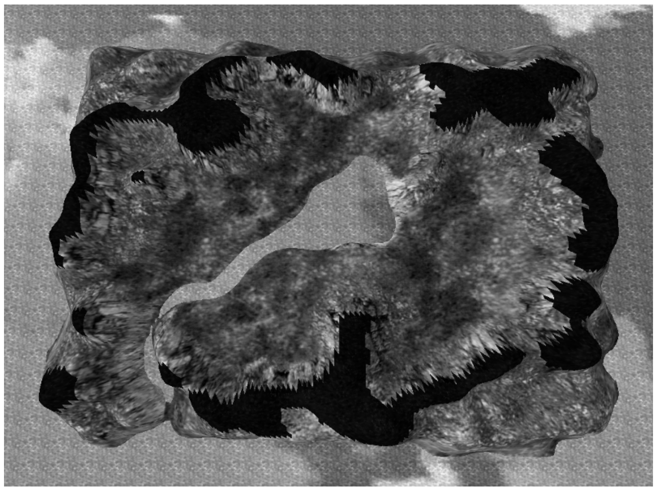


Figure 6. The first test terrain. The black areas are occupied space, the remaining areas are free space.

The second terrain (Figure 7) was represented through a heightmap with the same resolution. The underlying quadtree had a maximum resolution of 2×2 pixels per leaf. A terrain of $1000\ m \times 1000\ m \times 200\ m$ was simulated. The UAVs had a maximum altitude of $120\ m$. So, everything above $120\ m$ was treated as occupied space. In our test scenarios $\mathcal{C}$ was partitioned into $\mathcal{C}_{free}$ (65%) and $\mathcal{CB}_i$ (35%). The UAVs flew with a speed of $60\ km/h$ and had a favorite height of $20\ m$ over ground. To explore the terrain the UAVs used a camera with a flare angle of $90°$. When using such a camera the UAVs were able to explore $1256.64\ m^2$ at one moment when they had reached their favorite height.

The environment is able to consider the time points at which an area was explored. This ensures that areas, which were explored a given time ago become unexplored again to check whether they hold new information. During the following tests this behavior was disabled and areas explored once never became unexplored again.

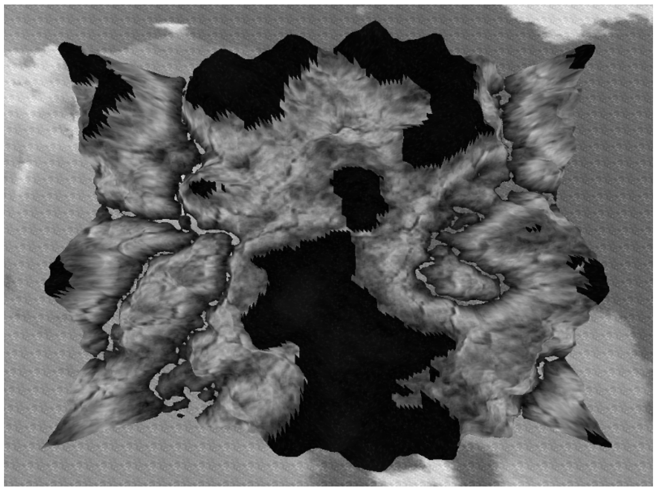


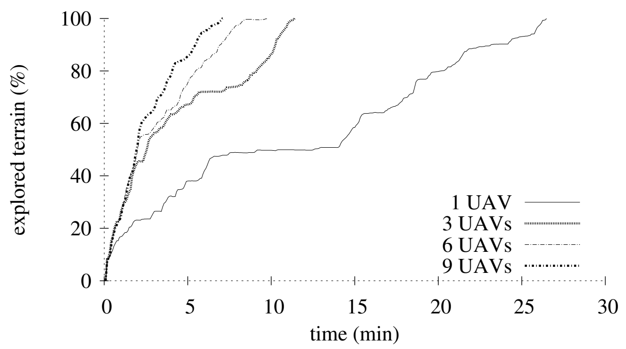
Figure 7. The second test terrain. The black areas are occupied space, the remaining areas are free space.

B. Test results: first heightmap

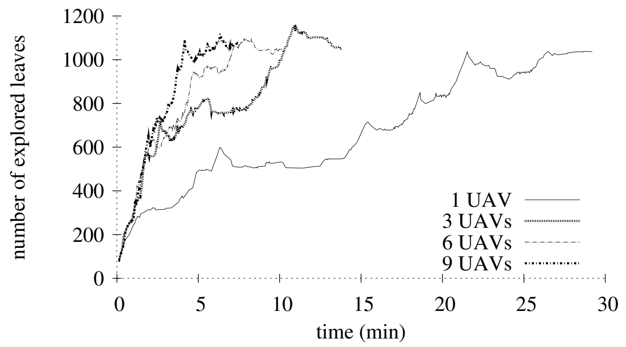
Several tests are presented in this paper. The first test series consists of exploring the terrain. The exploration was done by 1, 3, 6 and 9 UAVs. The partitioning of the terrain was unknown at the beginning. The UAVs had to recognize the occupied space through their cameras. For all tests the exploration was non-goal-oriented. Therefore, no areas had to be explored with high priority and the exploration continued as long as unexplored terrain reachable by the UAVs existed. Mostly, the gradient method was used to reduce the costs for path planning. Additionally, the A*-algorithm was used whenever UAVs got trapped in local minima, which occurred very rarely due to the use of harmonic functions for potential field calculation (Section V-A). Figure 8 depicts the results of the first test series.

Figure 8(a) shows the percentage of explored terrain covered relative to the exploration duration. In every test the complete terrain was explored. By using three UAVs instead of one only half the time for the exploration was needed. Increasing the number of UAVs always leads to a lower time needed for exploration. But tripling the number of UAVs again to nine did not reduce the exploration time by half again, because the more UAVs were used the more often they constrained each other and collisions had to be avoided. This needs time in which the UAVs are not available for exploration. Another point is that the less unexplored terrain is left the more UAVs start to fly to the same terrain to explore it.

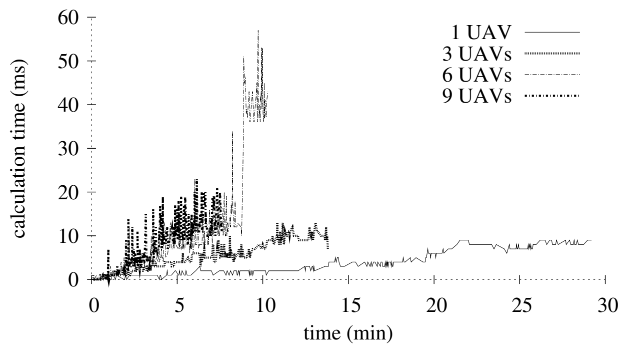
Additionally, the figure shows that until an exploration rate of 70% was reached there are only a few time spans in which the UAVs did not explore large areas of the terrain. This lack-of-exploration behavior occurred after the UAVs explored a large terrain and had missed a few small areas or when the next unexplored area was far away from the UAVs.



(a) Percentage terrain covered



(b) Number of utilized leaves for the calculations of $\mathcal{C}$



(c) Calculation times for single calculations of $\mathcal{C}$

Figure 8. Test result for the first test series

This shows that there is some room for further improvement for faster exploration.

Figure 8(b) shows the number of explored leaves related to the exploration duration. These leaves are those, which have to be updated. It shows that a maximum of about 1200 leaves had to be updated. The number of used leaves increased very fast at the beginning and decreased after a while because the the quadtree's recombining. As shown in the figure the maximum number of leaves remains nearly the same, independent of the number of used UAVs.

Figure 8(c) depicts the costs for the single calculations of the complete configuration space. The costs rise with the exploration rate because of the activation window. The more areas are explored, the more leaves are active and have to be considered for the calculations. But besides this, the costs are relatively constant except for a few spikes. They increase with the number of UAVs, which makes the environment more complex and leads to higher costs. For instance, recalculation is done every time a UAV requests a new path for exploration. So, more UAVs lead to more recalculations, which also increases the complete calculation costs. But the costs are in most cases below 20 ms, which allows online calculations.

A second test series was done for the first heightmap (see Figure 6). Six tests were made to check whether if it makes a difference if the terrain is known or not at the beginning. Additionally, it was checked if it makes a difference if all UAVs start from the same position or from different ones.

For the tests the following six scenarios were created:

1) Test 1: 1 UAV in known terrain.
2) Test 2: 3 UAVs in known terrain and all started from the same position.
3) Test 3: 3 UAVs in known terrain with three different start positions.
4) Test 4: 1 UAV in unknown terrain.
5) Test 5: 3 UAVs in unknown terrain and all started from the same position.
6) Test 6: 3 UAVs in unknown terrain with three different start positions.

Table IV gives an overview of the results. The following values were determined:

- test: The number of the test with the properties described above.
- test duration: The duration it took to explore the terrain.
- iteration depth ∅: The average number of iterations for the calculation of the configuration space.
- number calculations: The number of recalculations of the configuration space.
- costs complete: The complete time needed to make all recalculations of the configuration space.
- costs ∅: The average time a recalculation of the configuration space took.
- diff ∅: The average time between two recalculations of the configuration space.

As expected, the tests with unknown terrain (tests 4 - 6) needed more frequent recalculations of $\mathcal{C}$. A recalculation was done each time new occupied space was detected, what never would happen in known terrains as the occupied space is known at the beginning.

By the use of the break conditions and the introduction of an activation window (Section V-C), a single calculation of the configuration space took relatively little time. Except in the second test case the costs were on average less than 5 ms and this result shows that the calculations can be done before each path planning without causing a noticeable

test	test duration (min.)	iteration depth ∅	number calcula-tions	costs complete (ms)	costs ∅ (ms)	diff ∅ (s)
1	33.75	6.27	92	437	4.75	22.01
2	17.08	7.49	146	861	5.90	7.02
3	19.16	8.14	142	615	4.33	8.10
4	29.33	9.63	302	1337	4.43	5.83
5	13.83	7.99	303	1508	4.98	2.74
6	16.50	8.95	345	1672	4.85	2.87

Table IV
RESULTS OF THE SINGLE TEST CASES FOR THE SECOND TEST SERIES ON THE FIRST TERRAIN.

delay. The higher costs in test case two are the result of several repeatedly computed local minima, which needed many iterations of the update equation to vanish. Because of the higher number of explored nodes at the end of the exploration, the single iterations took a relatively long time.

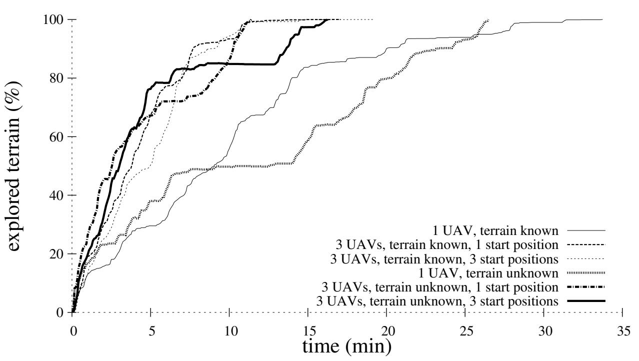
Since a recalculation of the configuration space takes place only before a new path for exploratory navigation is planned, the frequency of recalculating the configuration space was lowered. The first test case needed the fewest number of recalculations. In this test a recalculation was done on average every 22.01 seconds. In the worst case every 2.74 seconds (in test case five) such a recalculation took place.

The results show that the disadvantage of harmonic functions–the high calculation costs–compared to other methods for potential field calculation, which often use linear superpositioning, were reversed.

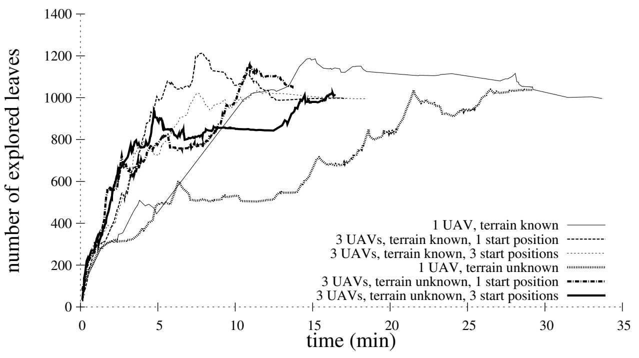
In Figure 9 the test results for the first heightmap are depicted. Figure 9(a) shows the exploration rate compared to the simulation time, which starts at 0 and is measured in real-time. Figure 9(b) shows the number of nodes, which are needed to calculate the configuration space at the single time points. This number has direct influence on the single calculation costs, which are shown in Figure 9(c).

One can see that the exploration was made relatively steady over time, especially during the tests with three UAVs. It made no big difference if all UAVs started from the same position or from three different start positions. The use of three UAVs made an advantage up to an exploration rate of 99% concerning the exploration speed, compared to the use of one single UAV. After a simulation time of, e. g., 7 minutes in the first test case, one single UAV had explored 39.65% of the terrain. When three UAVs made the exploration using the same start parameters they had explored 80.13% in test case two, respectively 81.39% of the terrain in test case three, in the same time span.

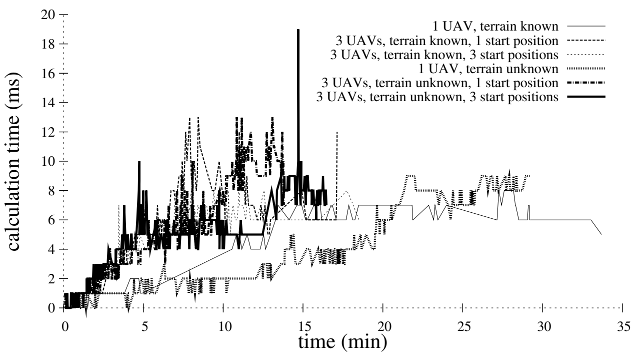
In test cases one and four, where only one UAV was used, it explored the terrain with a steady exploration rate until an exploration level greater than 85% was reached. After that, the exploration rate decreased dependent on time due to the fact that the UAV had to fly over explored terrains to reach the residual unexplored areas.



(a) Percentage terrain covered (first heightmap)



(b) Number of utilized leaves for the calculations of $\mathcal{C}$ (first heightmap)



(c) Calculation times for single calculations of $\mathcal{C}$ (first heightmap)

Figure 9. Test result for the first heightmap

In test case four such a steady exploration rate was not reached permanently. During these tests the UAV had to fly over explored terrains to reach the residual unexplored areas when only a relatively small portion of the terrain was explored. This behavior was observed only for a certain time span. After that, the exploration rate was steady again. Apart from that, a complete exploration of the free space was reached in each test.

Figure 9(c) depicts the costs to calculate the configuration space. One can see again, that the costs rise with the exploration rate of the terrain. This is due to the use of the activation window, which activates more leaves to be considered for the calculation of $\mathcal{C}$. This behavior can also be seen in Figure 9(b), which depicts the number of utilized leaves for the single calculations. As mentioned before, the behavior that explored terrains can become unexplored again was disabled. Therefore, the number of active nodes only decrease when they are combined.

The calculation costs for $\mathcal{C}$ are, apart from a few spikes, consistent. The only serious anomaly was in the sixth test with 19 ms. This anomaly was caused by several local minima, which needed several iterations to vanish. Except for this spike all other recalculations were between ≤ 1 ms and 10 ms. This shows that no noticeable delay of the exploration occurred, due to the calculations of the configuration space. Combined with the gradient method, which ensures that a route can be started after eight comparisons, this ensures fast path planning.

C. Test results: second heightmap

The six tests from the second test series were repeated on a second terrain (Figure 7) to ensure that they hold for other terrains as well. They are divided in the same six single tests as for the first heightmap.

Table V gives an overview of the results. The values that were determined are the same as for the first heightmap.

test	test duration (min.)	iteration depth ∅	number calcula-tions	costs complete (ms)	costs ∅ (ms)	diff ∅ (s)
1	35.50	7.18	113	334	2.96	18.85
2	19.50	8.12	155	624	4.03	7.55
3	21.16	10.91	160	659	4.12	7.94
4	22.00	12.61	146	646	4.42	9.04
5	7.75	9.40	141	710	5.04	3.30
6	6.66	5.25	150	625	4.17	2.67

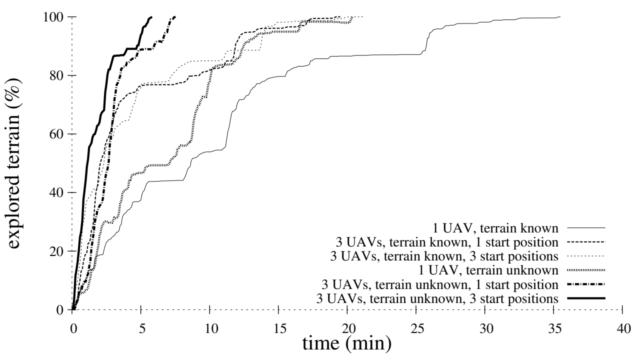
Table V
RESULTS OF THE SINGLE TEST CASES FOR THE SECOND TERRAIN.

The results are mostly the same as those we achieved on the first heightmap. Also, using this heightmap the time for calculating the configuration space does not cause any noticeable delay for path planning. On average the time was below 5 ms except for the fifth test. In the fifth test they were with an average of 5.04 ms not considerably higher.

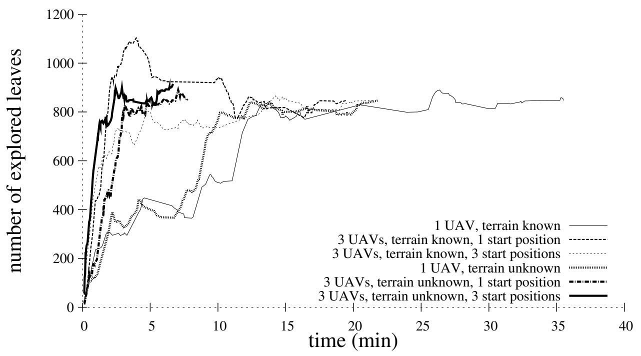
It should be noted, that when using three UAVs instead of one single UAV in unknown terrain, the exploration time was only one-third. This is not due to the fact that the single UAV had to fly much more often over explored terrain than in other tests. But the reason for this is that a high steady exploration rate was reached using three UAVs all the time (see Figure 10(a)).

When using this terrain, the frequency of recalculations of the configuration space was relatively low. It was similar to the first heightmap in such a way that the fewest recalculations were made in the first test case where only one UAV was used. A recalculation was done on average every 18.85 seconds. The most frequent recalculations were done in the sixth test case, where every 2.67 seconds a recalculation took place.

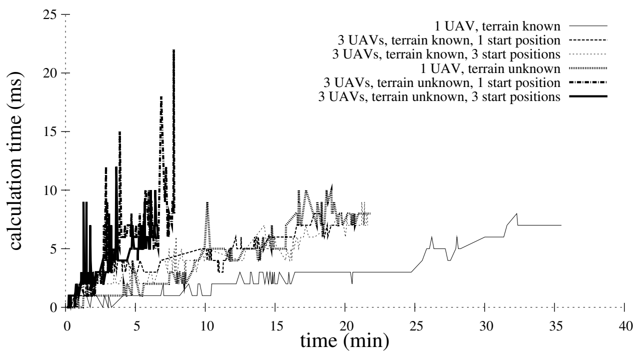
Figure 10 shows the results of the test cases for the second heightmap graphically. Figure 10(a) depicts the exploration over time, Figure 10(b) depicts the number of utilized leaves, which are relevant for the duration to calculate the configuration space. Finally, Figure 10(c) shows the duration of the single calculations of the configuration space.



(a) Percentage terrain covered (second heightmap)



(b) Number of utilized leaves for the calculations of $\mathcal{C}$ (second heightmap)



(c) Calculation times for single calculations of $\mathcal{C}$ (second heightmap)

Figure 10. Results for the second heightmap

Figure 10(a) shows the exploration rates of the terrain for the six test cases dependent on simulation time. It is shown that similar to the first heightmap, when using three UAVs, a steady exploration rate was reached in unknown terrain even during the complete exploration. In the test cases with known terrain the exploration rate decreased when an exploration level of nearly 80% was reached. The worst rate, which led to the longest time needed to explore the complete free space, was in test one, when using only one UAV in known terrain. Similarly to the tests done on the first heightmap, in every test a complete exploration of the free space was reached.

Figure 10(c) shows the costs to calculate the configuration space for the six test cases. It is shown that the calculation costs were relatively equal, except for the fifth test case with three UAVs in unknown terrain, where several spikes appear. In this test scenario the last calculation of the configuration space was the most costly one (22 ms). But even this is an acceptable value regarding a recalculation every few seconds.

When considering the results one can conclude that the disadvantage of calculating a potential field based on harmonic functions, the high calculation costs, vanished. This was due to the use of convergence criteria (Section V-C), the use of a quadtree (Section IV-A) to reduce the number of utilized leaves instead of using a grid and the introduction of an activation window (Section V-C). Hence, the advantages of harmonic functions can be exploited without suffering from their disadvantage. Always achieving an exploration rate of 100% of the reachable terrain shows that our path planning approach leads to good results even in complex and dynamic terrains.

The frequency of the recalculations of the configuration space shows that the system is a cheap one in respect of the calculation costs. This made it possible to use the designed approach in embedded systems with limited hardware.

Additionally, it was shown that the explored area increased with a steady rate in most cases and the UAVs had to fly relatively rare through explored terrain to reach unexplored terrain. In the cases where this behavior occurred, the exploration rate decreased. So, this is one issue to be solved, to decrease the time needed to explore terrains.

VIII. Conclusion and Future Work

The motivation for this paper was to design an efficient path planning system, which can be used to coordinate multiple UAVs to explore different disaster areas. The requirement was to create an efficient and robust system to coordinate multiple UAVs, including path planning, exploratory navigation and simultaneous task allocation, using only one global configuration space.

A hybrid approach for UAV coordination and efficient exploration of disaster areas was presented. It uses artificial potential fields, combined with an informed search algorithm, and a role system. Additional methods like a quadtree, an activation window, and break conditions were used to find a tradeoff between the number of local minima and computational costs. Until an exploration level of more than 70% was reached, a nearly steady exploration rate was achieved. Three UAVs need only half of the time for exploration in comparison to the time one single UAV needs.

The costs to compute the configuration space were decreased such that an online calculation without any noticeable delay was possible. This is important for highly dynamic environments, where the calculation has to be faster than the changes of the configuration space, in order to calculate efficient paths. In combination with the total exploration of the terrain, this leads to a robust and efficient system.

Future work is to lower the exploration time even more when using multiple UAVs. This can be done by more active coordination methods, e. g., explicit communication between the UAVs. Another focus of future work is to achieve greater autonomy. For this purpose an interaction of the UAVs with each other will be implemented to replace the central control station. This should be done in such a way, that using more UAVs leads to faster exploration of the complete terrain.

Another part of future work is to advance the potential field to n dimensions by including values like exploration times of the leaves and UAV properties into the single potentials. The next step is to extend the potential field to the third dimension of the terrain. Using a 3D potential field will allow the UAVs to move inside of buildings and fly, e. g., below bridges without any additional calculations.

An interesting point of investigation would be to decentralize the field such that each UAV calculates only that part of it, which is relevant for the UAV. In that case properties like fuel level could be included into the potential field to ensure that a UAV flies only to regions of the terrain from which it can fly back with its remaining fuel. Such a modeling of the potential field should reduce the number of methods, which are necessary to control a real UAV.

One step of our research project is to apply the approach to real UAVs. The achieved cost reduction enables the system to be implemented even on embedded systems. Of course, physical properties of real UAVs have to be considered even more. Additionally, sensor errors and the UAV behavior, e. g., in strong crosswind may lead to further need for adaptations when using real UAVs.

When applying the approach to physical UAVs we also need to consider non-holonomic constrains. This will lead to much smoother paths as the UAVs do not have to stop for course changes.

It is planned to extend the skills of the UAVs by introducing an inter-UAV communication, which makes decentralized coordination possible. This leads to the possibility that methods for formation flights can be realized. Several methods for this may be evaluated, e. g., the use of consensus finding [22] in communication graphs or adapting the potential field to hold information for building a formation, e. g., by using bifurcating potentials [23].

Acknowledgment

This work was conducted in the BMBF funded project SOGRO[3].

References

[1] C. Rasche, C. Stern, W. Richert, L. Kleinjohann, and B. Kleinjohann, "Combining autonomous exploration, goal-oriented coordination and task allocation in multi-uav scenarios," *ICAS, The Sixth International Conference on Autonomic and Autonomous Systems*, March 2010.

[2] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *IJRR*, vol. 5, no. 1, pp. 90–98, 1986.

[3] 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.

[4] E. Prestes, P. M. Engel, M. Trevisan, and M. A. P. Idiart, "Exploration method using harmonic functions," *Robotics and Autonomous Systems*, vol. 40, no. 1, pp. 25–42, 2002.

[5] M. Trevisan, M. A. P. Idiart, E. Prestes, and P. M. Engel, "Exploratory navigation based on dynamical boundary value problems," *Journal of Intelligent and Robotic Systems*, vol. 45, no. 2, pp. 101–114, February 2006.

[6] J. Hagelbäck and S. J. Johansson, "Using multi-agent potential fields in real-time strategy games," in *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 631–638.

[7] R. Sawhney, K. Madhava, and K. Srinathan, "On fast exploration in 2d and 3d terrains with multiple robots," in *AAMAS*. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2009, pp. 73–80.

[8] D. Jung, J. Ratti, and P. Tsiotras, "Real-time implementation and validation of a new hierarchical path planning scheme of UAVs via hardware-in-the-loop simulation," *Journal of Intelligent and Robotic Systems*, vol. 54, no. 1-3, pp. 163–181, March 2009.

[9] I. Nikolos, K. Valavanis, N. Tsourveloudis, and A. Kostaras, "Evolutionary algorithm based offline/online path planner for uav navigation," *IEEE SMC*, vol. 33, no. 6, pp. 898–912, Dec. 2003.

[10] M. Kazemi, M. Mehrandezh, and K. Gupta, "An incremental harmonic function-based probabilistic roadmap approach to robot path planning," *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, April 2005.

[11] 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, Mrz/April 1992.

[12] 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.

[13] C. Connolly, J. Burns, and R. Weiss, "Path planning using laplace's equation," *IEEE ICRA*, vol. 3, pp. 2102–2106, Mai 1990.

[14] 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.

[15] P. G. Doyle and J. L. Snell, "Random walks and electric networks," 2000. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:math/0001057

[16] C. Connolly, "Applications of harmonic functions to robotics," in *IEEE ISIC*, Aug 1992, pp. 498–502.

[17] D. Beck, A. Ferrein, and G. Lakemeyer, "Landmark-based representations for navigating holonomic soccer robots," in *RoboCup 2008: Robot Soccer World Cup XII*, ser. Lecture Notes in Computer Science, vol. 5399. Springer Berlin / Heidelberg, 2009, pp. 25–36.

[18] E. C. Zachmanoglou and D. W. Thoe, Eds., *Introduction to Partial Differential Equations with Applications*. Dover Publications, Inc., 1986.

[19] J. S. Zelek, "A framework for mobile robot concurrent path planning and execution in incomplete and uncertain environments," in *AIPS*, 1998.

[20] K. Aizawa, K. Motomura, S. Kimura, R. Kadowaki, and J. Fan, "Constant time neighbor finding in quadtrees: An experimental result," in *Communications, Control and Signal Processing, 2008. ISCCSP 2008. 3rd International Symposium on*, March 2008, pp. 505–510.

[21] C. Rasche, C. Stern, L. Kleinjohann, and B. Kleinjohann, "Role-based path planning and task allocation with exploration tradeoff for uavs," *ICARCV, 11th International Conference on Control, Automation, Robotics and Vision*, December 2010.

[22] W. Ren, W. R. Beard, and T. W. McLain, "Coordination variables and consensus building in multiple vehicle systems," in *Cooperative Control*, ser. Lecture Notes in Control and Information Sciences, vol. 309. Springer Berlin / Heidelberg, 2004, pp. 439–442. [Online]. Available: http://www.springerlink.com/content/m2gpyjh0mkq4uapc/

[23] 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, 2010, towards Autonomous Robotic Systems 2009: Intelligent, Autonomous Robotics in the UK. [Online]. Available: http://www.sciencedirect.com/science/article/B6V16-4XT3HPP-2/2/4338e40c022f12a0582b3655f7a51152

[3]"Sofortrettung bei Großunfall mit Massenanfall von Verletzten", supported by the Bundesministerium für Bildung und Forschung (BMBF) 13N10164.