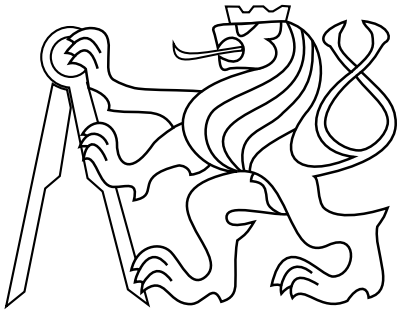
Czech Technical University in Prague

Faculty of Electrical Engineering
Department of Cybernetics
Multi-robot Systems



World management and coverage path planning in the MRS UAV System

Bachelor's Thesis

Azat Mukhametshin

Prague, May 2024

Study programme: Open Informatics
Branch of study: Software

Supervisor: Ing. Pavel Petráček

Acknowledgments

I would like to express my gratitude to my supervisor Ing. Pavel Petráček for his guidance, responsiveness, and valuable improvements throughout the work on this thesis. I thank my family for their unwavering support and encouragement during my studies. Also, I am grateful to my friends, whose companionship and positivity brightened up my study time. Last but not least, I want to thank me for believing in me, for doing all this hard work, for never quitting and for being me at all times.

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Mukhametshin Azat** Personal ID number: **507203**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Science**

Study program: **Open Informatics**

Specialisation: **Software**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

World management and coverage path planning in the MRS UAV System

Bachelor’s thesis title in Czech:

Správa světa a plánování cest s maximálním pokrytím v MRS UAV System

Guidelines:

The aim of this bachelor thesis is to extend the MRS UAV System [1] by adding (A) a world manager that allows for configuring, visualizing, and interacting with a world and obstacles in the Robot Visualization (RViz) tool of ROS1, and (B) planning of collision-free maximum coverage paths for UAVs [2-3] deployed in the world with obstacles (A). (C) The final objective of the thesis is to perform an experiment with a UAV using (A) and (B) in an outdoor environment with obstacles. The experiment shall be performed either in a virtual or real world (the domain will be specified by the supervisor based on the availability of hardware UAV platforms).
(A) The world manager should:
1. contain a world representation using a non-convex polygonal structure in the horizontal plane and variable height bounds,
2. incorporate non-convex obstacles of different heights and sizes,
3. visualize GPS-based photomaps (in outdoor cases),
4. allow defining, loading, and saving of the world with the obstacles,
5. enable visualizing flight telemetry of UAVs,
6. allow interaction of both virtual and real UAVs within the same visualization (includes: individual and multi-UAV selection, waypoint and key bindings navigation),
7. be integrated into the MRS UAV System [1].
(B) In addition, the task is to implement at least one path planning algorithm that maximizes the total coverage inside the non-convex world specified in (A). This task should:
1. review and compare at least three state-of-the-art coverage path planning algorithms suitable for UAVs [2-3],
2. implement the most suitable coverage path planning algorithm within (A),
3. planning coverage paths inside the entire world (A) as well as in user-definable safe zones inside (A),
4. interactive specification and parametrization of the path planning algorithm inside RViz,
5. loading and saving the coverage paths for a given world,
6. offering the possibility for integration of other coverage path planning methods (e.g., [4]).
(C) Perform experiment with the MRS UAV System either in a virtual world using the Gazebo simulator or in the real world. The experiment shall use (A) and (B) to fly an autonomous UAV along the planned path. The performance of the implemented algorithm should be analyzed and discussed both qualitatively and quantitatively.

Bibliography / sources:

1. T. Baca, M. Petrlik, M. Vrba, V. Spurny, R. Penicka, D. Hert, and M. Saska, The MRS UAV System: Pushing the Frontiers of Reproducible Research, Real-world Deployment, and Education with Autonomous Unmanned Aerial Vehicles, Journal of Intelligent Robotic Systems, 2021.
2. E. Galceran, M. Carreras, A survey on coverage path planning for robotics, Robotics and Autonomous Systems, 2013.
3. T. M. Cabreira, L. B. Brisolara, P. R. Ferreira Jr., Survey on Coverage Path Planning with Unmanned Aerial Vehicles, Drones, 2019.
4. D. Datsko, F. Nekovar, R. Penicka, M. Saska, Energy-aware Multi-UAV Coverage Mission Planning with Optimal Speed of Flight, IEEE Robotics and Automation Letters, 2024.

 © ČVUT v Praze, Design: ČVUT v Praze, VIC

Name and workplace of bachelor's thesis supervisor:

Ing. Pavel Petráček Multi-robot Systems FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **15.02.2024** Deadline for bachelor thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

Ing. Pavel Petráček Supervisor's signature	Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

 © ČVUT v Praze, Design: ČVUT v Praze, VIC

Declaration

I declare that presented work was developed independently, and that I have listed all sources of information used within, in accordance with the Methodical instructions for observing ethical principles in preparation of university theses.

Date ....

Abstract

This thesis designs and implements a user interface for the MRS UAV System to offer a more convenient way of interacting with the system. The user interface allows managing (loading, changing, saving) world boundaries with inlying zones of danger and safety without rebooting the system, controlling the Unmanned Aerial Vehicles (UAVs) remotely, setting multiple navigation goals, visualizing UAV telemetry and flight data, displaying satellite maps, and finally planning coverage paths in the set-up zones defined as both convex and concave polygons with holes. In addition, the thesis studies and compares four coverage path planning algorithms, further implementing the three most suitable for UAVs and evaluating them on several problems of varying complexity. The evaluation reveals the use cases of each algorithm and helps better understand their advantages and drawbacks. The coverage path planning is integrated into the developed user interface and utilized in an experimental deployment using a real UAV.

Keywords Unmanned Aerial Vehicles, Automatic Control, Coverage Path Planning

Abstrakt

Tato práce navrhuje a implementuje uživatelské rozhraní pro MRS UAV Systém, které nabízí pohodlnější způsob interakce se systémem. Uživatelské rozhraní umožňuje spravovat (načítat, měnit, ukládat) světové hranice s přilehlými zónami nebezpečí a bezpečnosti bez restartování systému, ovládat bezpilotní prostředky na dálku, nastavovat více navigačních cílů, vizualizovat telemetrii a letová data bezpilotních prostředků, zobrazovat satelitní mapy a také plánovát cesty pokrytí v nastavených zónách definovaných jako konvexní i konkávní polygony s dírami. Vedle toho práce studuje a porovnává čtyři algoritmy plánování cest pokrytí, dále implementuje tři z nich nejrelevantnější pro bezpilotní prostředky a vyhodnocuje je na několika problémech různorodé komplexity. Vyhodnocení odhaluje případy užití každého algoritmu a pomáhá lépe pochopit jejich výhody a nevýhody. Plánování trasy pokrytí je integrováno do vyvinutého uživatelského rozhraní a využíváno při experimentálním nasazení na skutečném bezpilotním prostředku.

Klíčová slova Bezpilotní Prostředky, Automatické Řízení, Plánování Cesty Pokrytí

Abbreviations

AI Artificial Intelligence

API Application Programming Interface

BFS Breadth-First Search

CPP Coverage Path Planning

DARPA SubT Defense Advanced Research Projects Agency Subterranean

DD Diagonal decomposition-based approach

DFS Depth First Search

EA Energy-aware algorithm

ENU East-North-Up

FCU Flight Controller Unit coordinate

GA Genetic Algorithm

GNSS Global Navigation Satellite System

GPS Global Positioning System

MD Morse decomposition-based approach

ROS Robot Operating System

RRT Rapidly-exploring Random Tree

RViz Robot operating system Visualization

SM Stride method

UAV Unmanned Aerial Vehicle

UI User Interface

UTM Universal Transverse Mercator

VRP Vehicle Routing Problem

Contents

Chapter 1

Introduction

Unmanned Aerial Vehicles (UAVs) attract a lot of attention for their omnipresent utilization varying from capturing breathtaking aerial footage in the film industry and delivering packages to monitoring agricultural fields for precision farming. One of the properties of UAVs is the capability to obtain image surveys on a large area, which is highly useful in such fields, as precision agriculture, ecological research and disaster management. The image-generating problem is not trivial and can be decomposed into two different tasks: the coverage path planning problem and map reconstruction from a set of overlapping geo-referenced images recorded during the flight mission of the UAV. In this thesis, we focus on implementing different coverage path planning algorithms and comparing their efficiency and suitability for integration into the MRS UAV System.

The MRS UAV System is multirotor UAV control and estimation system [8]. It makes replicable research available through realistic simulations and real-world experiments. However, most of the interface provided by the system is represented by Robot Operating System (ROS) topics and services, which are extremely convenient to implement but not very user-friendly for sending data to the program. For example, if a user wants to set a reference point near an obstacle, they must know the coordinates of the reference precisely, or else an UAV will fly in the wrong direction. This is where a convenient user interface plays a crucial role. Since the user can see the position of set references or objects relative to others, the User Interface (UI) lowers the risk of human error by reducing the cognitive load (see Fig. 1.1). Furthermore, the user interface frees one from memorizing topic names and switching between windows with complex and technical information.

(a) Setting point in the Robot operating system Visualization (RViz).

```
rosservice call /uav1/trajectory_generation/path "path:
  header:
    seq: 0
    stamp: {secs: 0, nsecs: 0}
    frame_id: 'uav1/world_origin'
  input_id: 0
  use_heading: true
  fly_now: true
  stop_at_waypoints: false
  loop: false
  max_execution_time: 0.0
  max_deviation_from_path: 0.0
  dont_prepend_current_state: false
  override_constraints: false
  override_max_velocity_horizontal: 0.0
  override_max_acceleration_horizontal: 0.0
  override_max_jerk_horizontal: 0.0
  override_max_velocity_vertical: 0.0
  override_max_acceleration_vertical: 0.0
  override_max_jerk_vertical: 0.0
  relax_heading: false
  points:
  - position: {x: 10.369, y: 27.688, z: 1.5271}
    heading: 0.0253897"
```

(b) Setting the same point in a terminal window.

Figure 1.1: Example of setting point in RViz and in a terminal

This thesis is organized as follows. In Sec. 2 the constraints of the MRS UAV System and the considered algorithms are introduced. In Sec. 3 the analysis of the algorithms is pre-

sented, describing their advantages and drawbacks. Additionally, the implementations of the algorithms are tested in the simulation. The results of the experiments are summarized and discussed, revealing the most suitable algorithms in different cases. In Sec. 4 we design, implement and describe the functionality of the user interface, which allows for setting navigation references and defining world boundaries as well as obstacles and objects within. It eliminates the need for memorizing complex ROS topics. In Sec. 5 the behavior of the implemented features is tested in real-world conditions. Finally, conclusions and further improvement opportunities are presented in Sec. 7.

1.1 Related works

The MRS UAV System is described in [8]. It is a highly modular, open-source system streamlining replicable research in simulations as well as real-world experiments. The implementation is well-documented and actively maintained by the Multi-robot Systems Group at the Faculty of Electrical Engineering at Czech Technical University. The system is complementary with hardware solutions presented in [3], [7]. It has been shown to be a particularly powerful framework in multiple real-world applications, such as Defense Advanced Research Projects Agency Subterranean (DARPA SubT) Challenge [5], decentralized swarming [11], autonomous cooperative wall building [2], flying objects detection [6], and digital documentation of large interiors [4].

Visualization of the MRS UAV System is built upon RViz. RViz [13] is a tool to visualize robotic data and ROS-based systems (such as MRS UAV System) build on the ROS network structure. RViz allows for adding different plugins, such as tools, displays, and panels. Tools are focused on interacting with the RViz 3D world visualization, providing Application Programming Interface (API) for processing mouse movements and key events. The purpose of displays is to demonstrate custom data from topics in the visualization. Panels serve to hold application-specific elements, such as buttons, image viewers and control inputs.

The plugin allows for community-based extensions and development of RViz, which itself is an open-source project. One example of an RViz plugin is the *rviz_satellite* [9], which supports automatic placement of the loaded tiles as well as a manual definition for increasing noise resistance. This plugin has been used to visualize map overlays in this thesis, as described in Sec. 4.5.

A significant part of this thesis also focuses on Coverage Path Planning (CPP). CPP methods are well summarized in [12], [18]. These works cover a large number of different algorithms that can be used in varying environments, but most of them are not applicable in the MRS UAV System due to the constraints introduced in Sec. 2. In the following summary, the focus lies on the applicable approaches.

One of the major concerns about the CPP problem is to guarantee complete coverage. To achieve this, the area of interest is splitted into cells in order to simplify the coverage. The most common decomposition methods used in CPP involving UAVs are exact and approximate cellular decompositions. Exact cellular decomposition divides the area of interest into subareas, whose union exactly occupies the workspace. This approach reduces the CPP problem to visiting every cell of the decomposition. Individual cells then can be covered using simple motions such as back-and-forth [15], [26], [28]. Approximate cellular decomposition [14], [19], [20] splits the workspace into a set of homogeneous-sized cells, that can be covered by taking exactly one picture. This approach simplifies the CPP problem to visiting all the waypoints.

Coverage path planning task using UAVs is sometimes simplified to deal with convex polygons only as done in [25]. When considering a concave environment, researchers focus on path construction regardless of the shape of an area as in [16], where the algorithm is run recursively in sub-areas created by concavities. The method introduced in [21] deals with concave polygons without holes, however, the path endpoint is given by the algorithm, which does not need to be in the field of view of a human flying the UAV back. The algorithm presented in [15] solves the problem of an unreachable finish point by defining the landing point manually. More importantly though, [15] can plan coverage paths in polygons containing holes. For polygon decomposition of concave polygons with holes, [15] uses diagonal decomposition presented in [24]. This CPP approach and the decomposition are further discussed in Sec. 2.1.1.

The coverage path planning problem often contains the polygon decomposition as a sub-problem. Since the greater amount of cells the decomposition produces, the less efficient the resulting path is, it is crucial to decompose an area into the largest partitions possible. The boustrophedon decomposition introduced in [28] does not require the partitions to be convex, and thus results in a lower number of cells. The Morse decomposition presented in [26] generalizes the boustrophedon decomposition by using Morse theory [29]. This approach results in higher flexibility of the algorithm and therefore different coverage paths can be generated for different environments, addressing the requirements of a particular mission.

Another CPP approach is presented in [22], which uses the approximate cellular decomposition for coverage planning. The algorithm generates valid paths but does not consider obstacles, which makes it disadvantageous with respect to other methods. An improved algorithm to [22] was presented in [19], which avoids obstacles within the area of interest while reducing the number of turning points. The improvement is achieved by introducing Breadth-First Search (BFS) for searching the path to the nearest unvisited cell and prioritizing straight-line motions.

The approximate cellular decomposition transforms the coverage path planning problem into the motion planning problem. A large amount of motion planning strategies have been studied, mostly focusing on optimality and obstacle avoidance. To tackle this problem, the Rapidly-exploring Random Tree (RRT) algorithm was presented in [27]. However, this approach did not guarantee any optimality, and therefore RRT* was introduced in [23], resulting in a more efficient path search but requiring much more memory. The memory usage problem was solved in the Fixed Nodes RRT* algorithm, retaining the same probabilistic guarantees on optimality and completeness as RRT*. This approach was used in [14] to plan a coverage path in three steps. First, Fixed Nodes RRT* is used to find the shortest path to the neighbors avoiding obstacles. Second, Genetic Algorithm (GA) is used to plan the shortest path visiting every cell. Third, the savings-based Vehicle Routing Problem (VRP) algorithm divides the path into a minimal set of the shortest paths.

1.2 Statement on the usage of artificial intelligence tools

Artificial Intelligence (AI) tools (in particular ChatGPT) have been used in the following ways. Firstly, to address specific problems that require a detailed understanding of library, command, or programming language documentation. For example, converting a roll-pitch-yaw rotation into a quaternion using a specified class or generating an explanation for a Bash command with numerous typographical symbols. Secondly, to name chapters, services, variables, and other elements to ensure their names reflect their purposes while remaining concise. Finally, to generate key discussion points to verify whether essential details have been covered. All of the information provided by AI tools has been critically analyzed and(or) tested.

Chapter 2

Coverage Path Planning

This chapter delves into the review and comparison of four coverage path planning algorithms suitable for UAVs. The selection of these algorithms was guided by several key criteria aimed at identifying solutions that address the characteristics of the World Manager (see Sec. 4.1) and the MRS UAV system in general. Among the criteria considered during the selection process were:

1. Rotary-wing UAVs. The MRS UAV System is designed with multi-rotor helicopters, which can perform maneuvers with low turning radius.
2. Non-convex polygons with holes. The World manager enables adding obstacles to the environment, therefore the algorithm must be able to avoid no-flight zones and yet complete the coverage.
3. Shared workspace and zones of interest. Many algorithms assume a UAV can fly outside of the zone of interest (typically in a rectangular zone), which is undesirable in the proposed methodology.
4. Full knowledge of the workspace. An algorithm must take the polygonal area as input and compute the coverage path before a mission starts.

Considering the mentioned constraints, the following coverage path planning algorithms are introduced and compared:

1. Diagonal decomposition-based approach (DD) is described in Sec. 2.1.1.
2. Morse decomposition-based approach (MD) is described in Sec. 2.1.2.
3. Energy-aware algorithm (EA) is described in Sec. 2.2.1.
4. Stride method (SM) is described in Sec. 2.2.2.

As it has been mentioned, one of the major metrics when considering the CPP problem, is coverage completeness. To guarantee completeness, the target area is divided into a set of partitions in order to simplify the coverage. The algorithms described in this thesis use two decomposition approaches: exact and approximate cellular decomposition. The exact cellular decomposition splits the workspace into sub-areas, also known as cells, whose union exactly occupies the workspace. Resulting partitions of exact cellular decomposition methods can not be covered using a fixed focal length camera, therefore different motion patterns, such as back-and-forth, are used to cover the partitions. On the other hand, the approximate cellular decomposition decomposes the area of interest into a homogeneous-sized set of partitions, that can be covered by taking pictures of the whole cell. This way the problem is simplified to visiting each waypoint represented by cells. The selected methods of exact and approximate decompositions are described below in Sec. 2.1 and Sec. 2.2 respectively. To improve the clarity of the descriptions, the mathematical notation introduced in Table 2.1 is used hereafter.

Symbol	Description
v_i	A vertex of a polygon or a hole within it.
h_i	A vertex of a hole within a polygon.
$e = v_i v_j$	An edge of a polygon joining vertices v_i and v_j.
$d = v_i v_j$	A diagonal of a polygon joining vertices v_i and v_j.
cp	A critical point of a function, i.e., a point where the function takes its extremum.
$\mathbf{p}$	A 2D point, $\mathbf{p} \in \mathbb{R}^2$.

Table 2.1: Overview of the mathematical notation.

2.1 Exact cellular decomposition

2.1.1 Diagonal decomposition-based approach

An exact cellular decomposition approach, considering concave polygonal areas, is explored in [15]. Although the presented decomposition is only suitable for concave polygons without holes, the algorithm is suitable for polygonal areas with holes once the decomposition into convex partitions is done. The authors refer to [24], where such an algorithm is provided.

The decomposition algorithm tries to generate as big a convex partition as possible, iterating through the border of a polygon. The partition is defined by a diagonal $d = v_1 v_n$. In case the partition contains a hole or a part of it, the diagonal $d = v_1 v_n$ is changed to $d = v_1 h_1$, where h_1 is the closest[1] vertex of the holes inside the partition. Then the list of vertices $\{h_1, \ldots, h_n, v_1\}$ is placed right after the vertex v_1 and the operation starts from the next notch vertex. That way, several diagonals, that decompose the polygon into convex parts, are created. However, redundant diagonals may occur and can be removed without violating convex decomposition. To decrease the number of partitions, merging is performed as the last step of the algorithm. Such an approach is proven to find a solution with the number of diagonals no more than four times greater than the optimal solution [24]. However, in the computations on polygons without holes provided by the authors, the ratio of the number of generated partitions to the optimal one is within 1.18, which is far from the upper bound of 4.0.

After the decomposition, DD determines the best sweep direction, which results in four optimal coverage paths (see Fig. 2.1) per partition. To find the minimal path length, all permutations of polygons are generated and the optimal solution is selected. This approach leads to $O((k!/2) \cdot 4^k)$ time complexity, where k is the number of partitions after the decomposition. This complexity is enormous and therefore, if the number of cells is too big, Depth First Search (DFS) is used to find a sub-optimal permutation. The DFS only considers adjacent cells as long as it is possible. If all adjacent cells are visited, the algorithm moves to each of the remaining cells. This approach reduces time complexity to $O(b^d)$ where b is the branching factor of the graph and d is the number of edges in the shortest path through the graph. Combined with polygon decomposition, the total time complexity of the algorithm is $O(b^d + NR^2(\frac{N}{2} - R + \frac{5}{2}) - 3R^2 + Nn_H^2)$, where N is equal to number of all vertices including holes, R is number of notches from the point of view of the polygon to be decomposed and n_H is number of vertices of holes.

[1] View procedure DrawTrueDiagonal in [24].

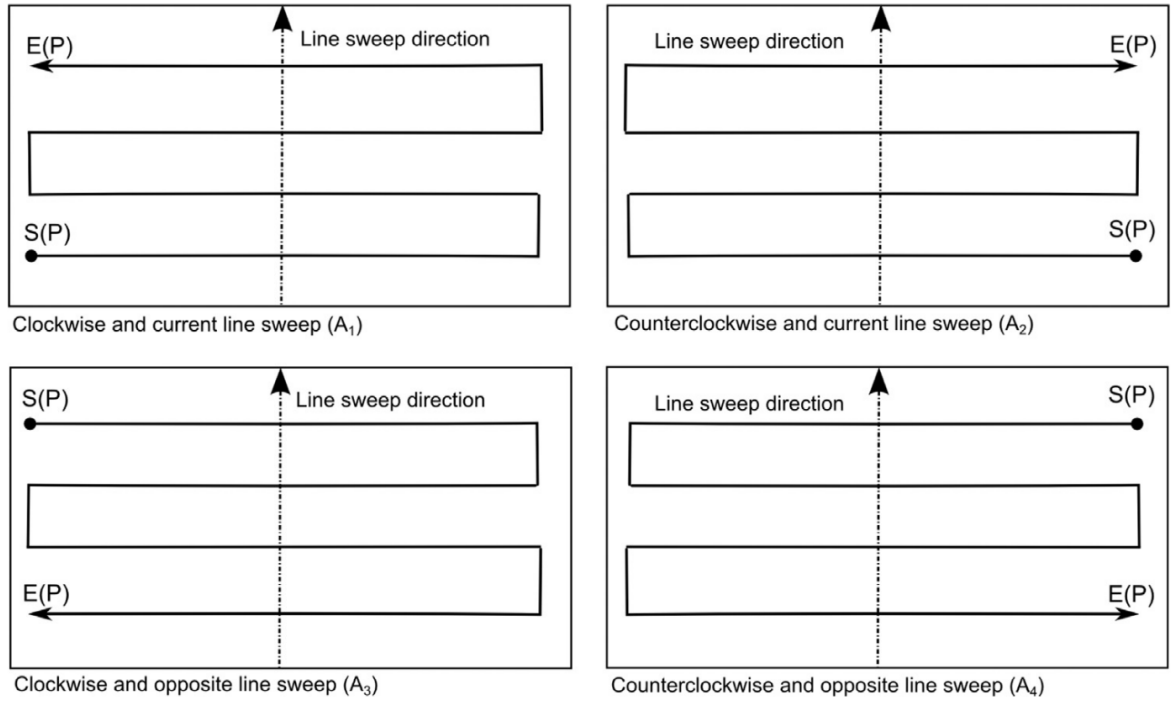
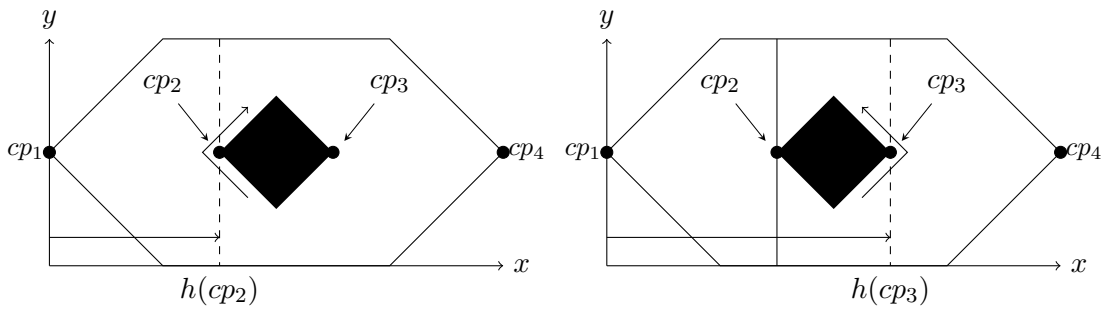


Figure 2.1: Coverage alternatives of a partition [15].

2.1.2 Morse decomposition-based approach

An analytical approach to decompose area and plan coverage path is described in [26]. To determine the cell decomposition, the methodology of [26] sweeps a slice through the target space, where the slice is a codimension one manifold. As the slice is swept, it is separated into smaller pieces (see Fig. 2.2a) when it encounters an obstacle or smaller pieces are merged into larger ones (see Fig. 2.2b) as it passes an obstacle. It is proven, that changes in the connectivity of the slice (i.e., the change in the number of pieces) occur at critical points of a Morse function[2] limited to the boundaries. According to Morse theory [29], no obstacle occurs between the critical points, therefore the space between them can be trivially covered by motions corresponding to the slice. Once the decomposition is complete, the task simplifies to finding an exhaustive walk through the adjacency graph (see Fig. 2.3).



(a) Splitting. (b) Merging.

Figure 2.2: The restriction of the slice function $h(x) = x_1$ to the obstacle boundaries takes a local minimum at cp_2 and a local maximum cp_3. Since $h(x)$ takes its extremes at cp_2 and cp_3, they are the critical points.

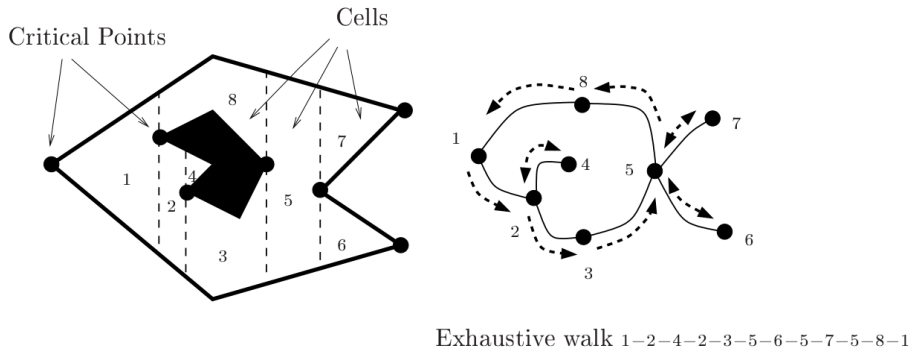


Figure 2.3: Example of Morse decomposition using $h(x) = x_1$ function and its adjacency graph [26].

In the MRS UAV System, all the boundaries are non-smooth, and thus the critical points must be found algorithmically. The algorithm for finding critical points is described in Alg. 1. Since the critical points can lie not only on the border but also on hole boundaries,

[2] A Morse function is one whose critical points are nondegenerate, i.e., critical points are isolated from one another. An example of a non-Morse function is $x^5 \cdot cos(\frac{1}{x})$, as it has the infinite number of critical points on interval $[-\epsilon; \epsilon]$ where $\epsilon > 0$.

Alg. 1 must be run both on the border and the holes of the polygon. Since critical points of simple functions (that are most commonly used for Morse decomposition) such as $h(x) = x_1$ or $h(x) = x_1^2 + x_2^2$ can be found in constant time using an analytical approach, the time complexity of finding critical points and thus cellular decomposition is $O(n)$ where n is the number of vertices of all boundaries. Combined with DFS for an exhaustive walk through the adjacency graph, the total time complexity is $O(n + b^d)$, where b is the branching factor of the graph and d is the number of edges in the shortest path through the graph.

Algorithm 1 Search for critical points

Input:
P ▷ input polygon without holes (array of vertices)
$M(\mathbf{p}) : \mathbb{R}^2 \to \mathbb{R}$ ▷ Morse function for a 2D point $\mathbf{p} \in \mathbb{R}^2$
$M'(e = v_i v_j) : \mathbb{R}^4 \to \{\mathbf{p}\}$ ▷ function that returns critical points of the Morse function bounded to the edge
Output:
$CPs = \{\mathbf{p}\}$ ▷ critical points

Begin:
$CPs \leftarrow \emptyset$
$L \leftarrow$ [] ▷ array of potential critical points and their function values
▶ Fill the list of potential critical points
for v_i in P **do** ▷ iterate each vertex exactly once
    L.append(v_i)
    $j \leftarrow (i + 1)\%|P|$
    $e \leftarrow v_i v_j$
    for each $cp \in M'(e)$ **do** ▷ compute critical points of the Morse function bounded to the edge and iterate them exactly once
        if $cp \neq v_i$ **and** $cp \neq v_{i+1}$ **then**
            L.append(cp)
        end if
    end for
end for
▶ Filter the critical points
for each $i \in \mathbb{Z}, 0 \leq i < |L|$ **do** ▷ iterate potential critical points exactly once
    $cp_{i-1} \leftarrow L[i-1]$ ▷ if $(i-1) < 0$, the last value is taken
    $cp_i \leftarrow L[i]$
    $cp_{i+1} \leftarrow L[i+1]$ ▷ if $(i+1) \geq |L|$, the first value is taken
    if $(M(cp_i) > M(cp_{i-1})$ **and** $M(cp_i) > M(cp_{i+1}))$ **or**
       $(M(cp_i) < M(cp_{i-1})$ **and** $M(cp_i) < M(cp_{i+1}))$ **then**
        CPs.append(cp_i)
    end if
end for
return CPs

2.2 Approximate cellular decomposition

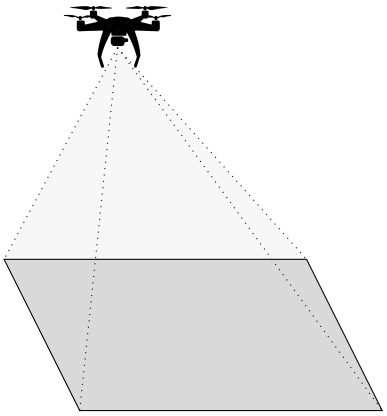
In contrast with land robot coverage, the size of cells does not fit the dimensions of a vehicle when considering aerial coverage using UAVs. In this case, the size of cells is equal to

the footprint of a UAV camera (see Fig. 2.4a). Assuming the camera footprint is square, the side of the taken image is

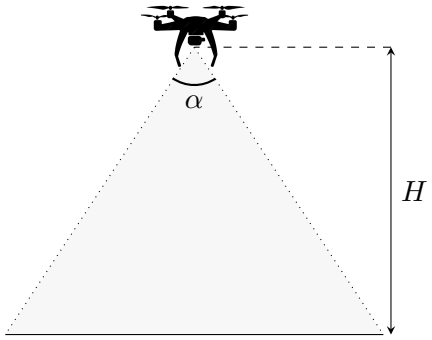
$$s = 2H \cdot tan(\frac{\alpha}{2}), \tag{2.1}$$

where s (m) is the size of the cell side, α (rad) is the camera angle of view and H (m) is the flight altitude from the ground (see Fig. 2.4). To ensure the image overlap, the distance between cells is adjusted to $D = s - w$ (see Fig. 2.4). Since $w = rs$ where $r \in [0, 1]$ is the overlap rate, the final distance between cells is

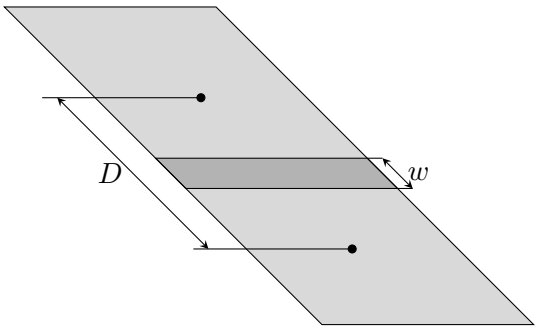
$$D = (1 - r)s \tag{2.2}$$



(a) A projection footprint of the UAV camera field of view.



(b) Front view of the UAV.



(c) Parameters of the overlapping images.

Figure 2.4: Camera-related parameters of the coverage path planning.

2.2.1 Energy-aware algorithm

An optimal CPP algorithm with a quadrotor UAV is presented in [14]. The mission is planned in two steps. In the first phase, the path from $\mathbf{p}_i$ to $\mathbf{p}_j$ for every i and j is computed using RRT-based algorithm named Fixed Nodes RRT*. Considering that the vehicle must return to the initial position, the problem is treated as the traveling salesman problem and the shortest path is computed using GA and the savings-based VRP algorithm. GA results in $O(n^2 + gnm)$ time complexity, where n is number of waypoints, g is number of generations in GA, and m is the GA population size. The savings-based VRP algorithm has at least quadratic in the number of waypoints time complexity.

2.2.2 Strides-based approach

The authors of [19] present a coverage path planning algorithm that increases efficiency by reducing the number of turns. The main concept of the algorithm is a stride — a sequence of consecutive adjacent cells with no turns. Stride starts at the current cell and several conditions determine, where it ends. The algorithm starts at the defined cell, moves along the longest stride and iterates if any unvisited cell exists. In case a dead end is reached (i.e., all 4 neighboring cells are either not reachable or already visited), we employ a BFS to find the nearest unvisited cell adjacent to a visited one. As a final step of the algorithm, the clean-up is performed on the generated path. If the path contains a sequence of revisited cells that connect two adjacent cells, the sequence of revisited cells is removed from the path. Assuming that the algorithm does not end up at dead ends frequently, the time complexity is $O(n)$, where n is the total number of cells.

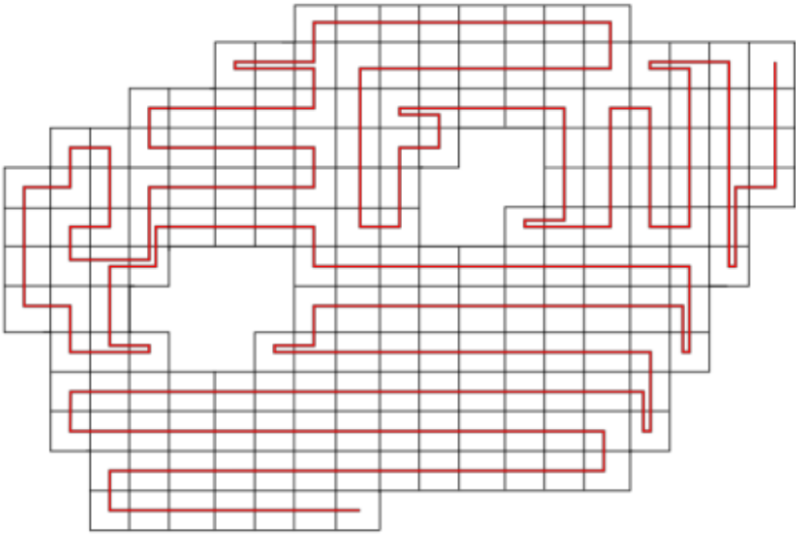


Figure 2.5: Example of coverage path computed by the Stride-based algorithm [19].

Chapter 3

Comparing CPP methods

3.1 Algorithmic analysis

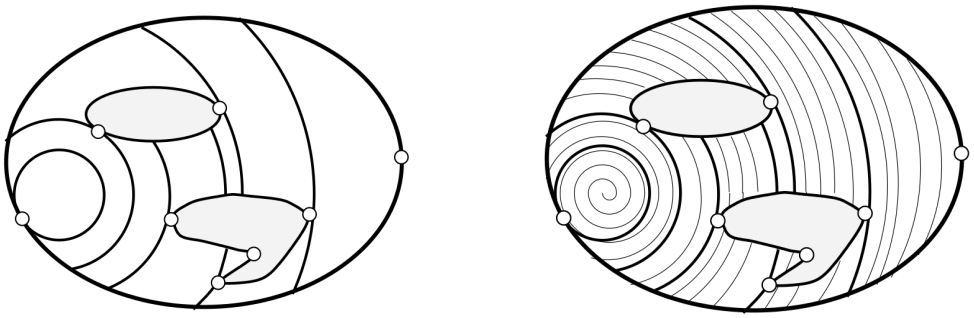
A properly implemented exact cellular decomposition approach guarantees the complete coverage of the workspace and near-optimal coverage path. On the other hand, it has a more difficult implementation and higher computation complexity than the approximate cellular decomposition approach. Meanwhile, the approximate cellular decomposition approach trades off accuracy for reduced implementation and computation complexity. Choosing between exact and approximate cellular decomposition results in searching for a compromise. Since the world configurations in the MRS UAV System tend to remain simple (i.e., the vast majority of vertices do not represent reflex angles and the number of obstacles remains low), the time complexity does not play a crucial role. Moreover, since paths can be precomputed and saved, the problem of time-consuming computations can be solved for complex environments by precomputation. Taking everything into consideration, exact cellular decomposition approaches are more suitable for our usage and therefore prioritized over approximate decompositions.

In contrast with DD, MD has several advantages. Firstly, MD is much faster in polygon decomposing and searching for the optimal path. MD has exponential time complexity for path search and decomposition time complexity is linear in the number of vertices Meanwhile, DD takes a quadratic time for decomposition. However as it has been mentioned, time complexity does not play a big role in our case. Secondly, MD is flexible, as different Morse functions result in different decompositions (see Fig. 3.1). It is also possible to rotate the function to obtain a better decomposition. Thirdly, using the linear Morse function in MD results in paths similar to the DD if individual partitions are considered. Moreover, MD does not require the partitions to be convex, which may result in a better path.

DD also has its advantages. The flexibility of MD results in a need to test different Morse functions and their variants to find the optimal one. Moreover, MD requires a start point, where the decomposition is started from, whereas DD finds the optimal start point in a given decomposition. The diagonal decomposition-based coverage algorithm finds the best coverage path for individual convex partitions, while the approach based on Morse decomposition may follow the worst sweep direction of partition (see Fig. 3.2).

To sum up, both DD and MD have their advantages and drawbacks. The efficiency of the algorithms depends on different parameters, such as the number of reflex angles in the polygon, the number of holes and their positioning relative to each other. Therefore, to decide which one suits the conditions of the MRS UAV System, both algorithms have been implemented and quantitatively compared in the Sec. 3.2

Among the approximate decomposition-based methods, SM has low time complexity in contrast with EA. Therefore it is suitable for fast path planning on large areas. However, the advantage of the two-step algorithm is that it considers waypoints instead of cells. This



(a) Decomposition. (b) Coverage path.

Figure 3.1: Cellular decomposition for $h(x) = (x_1)^2 + (x_2)^2$ Morse function and its associated spiral coverage pattern. The slices are the circles that are the pre-images of h. At the critical points, demarked with little circles (not to be confused with the slices), the circle-shaped slices become tangent to the obstacles. Rather than moving along circular paths and stepping outward, the robot follows a spiral pattern [26].

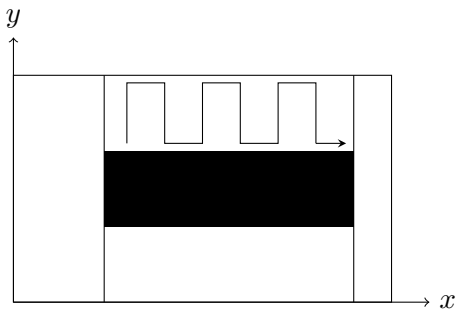


Figure 3.2: Example of MD with function $h(x) = x_1$ choosing the worst sweep direction for a cell.

provides new ways of generating them (e.g., random generation or adding a few points manually for better coverage). Also, EA always generates a path that lies within the safety area thanks to RRT-based algorithm, whereas SM assumes, that all adjacent cells can be reached with straight line movement, which does not have to be always true and therefore has to be explicitly handled. Finally, EA considers energy limitations and divides the path into a set of paths, that can be covered either by single UAV or by several UAVs. The energy limitation can be worked around as SM can be extended for multiple UAV usage [20], but the number of paths has to be set manually in order to both achieve a minimal set of paths and stick to the limitations.

3.2 Performance

3.2.1 Implementation details

It is important to point out, that the implementations do not fully correspond to the mentioned algorithms and the list of changes is presented below. The reason is that the algorithms are challenging to implement due to many hidden caveats and paper-omitted details. Although our implementations are suboptimal, the algorithms can still be compared and integrated into the MRS UAV System. If the implementations are discussed, the ending "-impl" is added, so that the reader does not confuse them with the strictly theoretical algorithms.

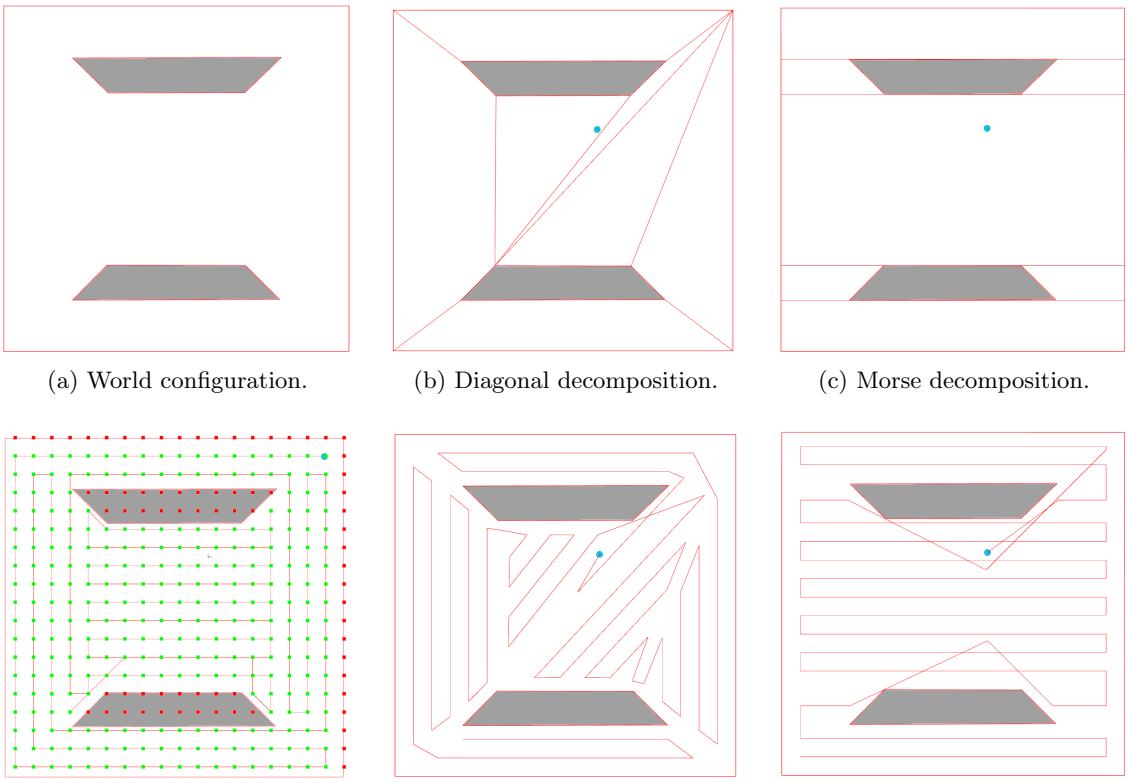
The differences between theoretical descriptions and our implementations are as follows. SM-impl [19] lacks the heuristic on which direction to choose in case two or more strides in different directions have equal lengths, SM-impl also lacks final clean-up of the path. DD-impl [15] uses the corrected search algorithm regardless of the number of partitions. When generating permutations of paths of individual partitions, only the path that starts closest to the last point of the generated path is considered. As for the decomposition algorithm, DD-impl uses the Mp3 algorithm introduced in [24]. Mp3 algorithm was modified to start from the first vertex of the polygon and not from the last considered one. The MD-impl [26] uses $h(x) = ax_1 + bx_2$ Morse function, where a and b are scalars. The only change MD-impl has is that the vertices of the polygon are slightly shifted in the range [-0.02; 0.02] to avoid edge cases of searching for critical points. The distances between cells in SM-impl and between sweeps in DD-impl and MD-impl are computed according to Eq. (2.2).

3.2.2 Evaluation

Several illustrative examples have been developed to compare the outputs of the algorithms. In the first one (see Fig. 3.3), the coverage path over a simple environment with two obstacles is shown. In the second example (see Fig. 3.5) coverage path is planned in a complex concave polygon without holes. The third world configuration (see Fig. 3.6) is taken from [15]. The last one (see Fig. 3.7) demonstrates the extremes of different methods. We use the number of turns in the path as the metric of optimality. As stated in [15], for fixed distances, the time is increased when the rotor-craft turns because it has to completely stop before it starts moving in a different direction. The wastes time while the robot slows down and accelerates. Additional acceleration also increases energy consumption, which is a crucial parameter in rotary-wing UAVs.

The blue circle on the images (see Fig. 3.3, 3.5, 3.6, 3.7) is the drone's position, where the coverage path starts. If the start point of SM-impl is not set on the edge of a polygon, the algorithm follows a straight line to the edge, marking the cells as visited. Since visited cells are equivalent to invalid ones when computing a stride, this generates an unnecessary obstacle, that can propagate and require a larger number of turns in the path. Therefore the start position is set on an edge, which significantly decreases the required number of turns.

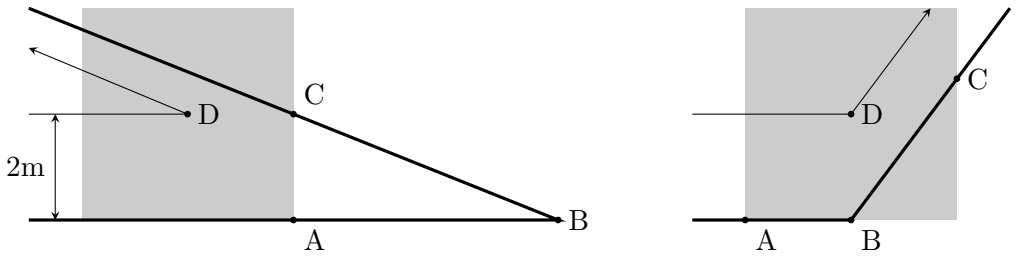
The coverage path for a simple polygon with two holes generated using exact cellular decomposition methods is shown in Fig. 3.3. The total number of turning maneuvers is the following: 47 for DD-impl, 36 for MD-impl and 41 for SM-impl (all the quantitative results are summarized in Table 3.1). One can see that some regions are missed out by DD-impl. This happens because of acute angles being formed by the decomposition (see Fig. 3.4). However, this can be partially solved by either decreasing the indent from partition bound or by introducing a merging process.



(a) World configuration. (b) Diagonal decomposition. (c) Morse decomposition.

(d) Coverage path generated by SM-impl. (e) Coverage path generated by DD-impl. (f) Coverage path generated by MD-impl.

Figure 3.3: Polygon decompositions (b, c) and coverage paths (d-f).

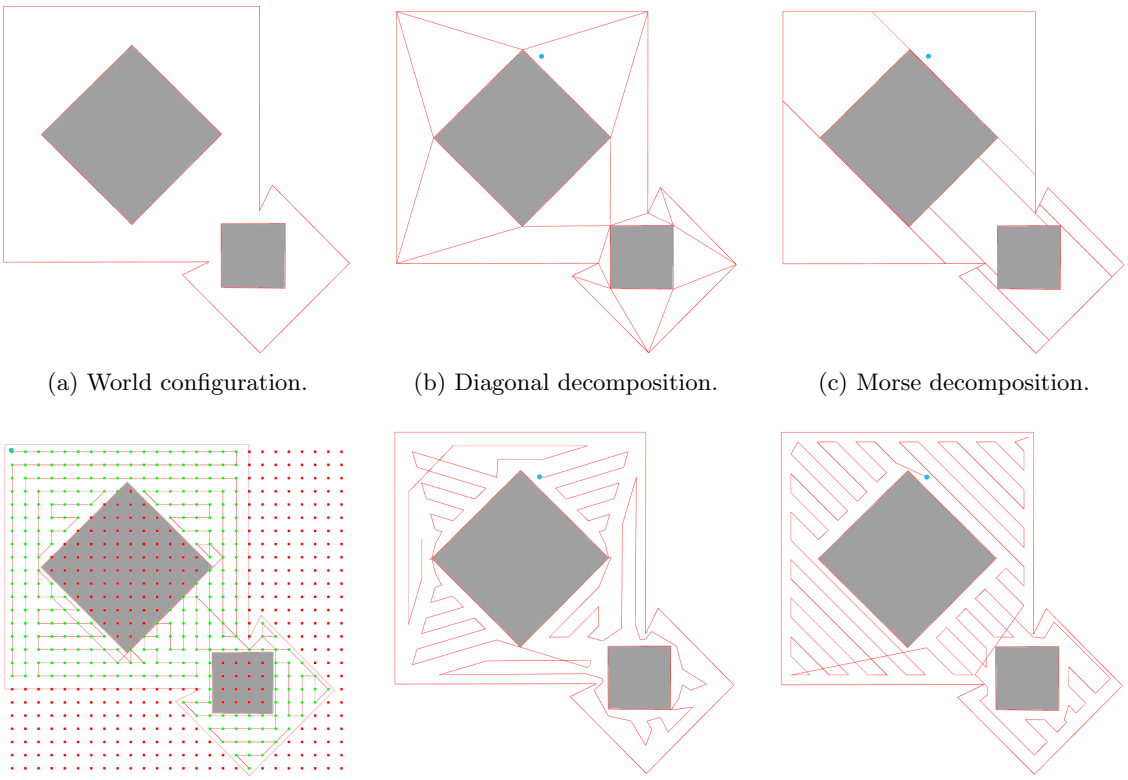


(a) An acute angle causes incompleteness by leaving the triangle ABC uncovered.

(b) Obtuse angle does not cause coverage incompleteness.

Figure 3.4: Example of a partition with angle ABC. The image overlap rate is zero, the camera footprint (denoted by the grey shadow) is 4 m square and the boundary indent is equal to the distance between sweeps, the rightmost position of the drone following a sweep is point D.

Another environment covered with each of the implemented algorithms is shown in Fig. 3.5. The number of turning maneuvers is 92 for DD-impl, 95 for MD-impl and 132 for SM-impl. As it can be noticed, no partition in Fig. 3.5b can be merged with any other in order to make a bigger partition. Therefore, the only way to get rid of gaps in coverage is to decrease the indent. In contrast, MD-impl does not leave visible gaps if proper parameters are set. Furthermore, diagonal decomposition results in 17 cells, while Morse decomposition has only 11 cells, and this leads to a significant difference in computation time. Our DFS-based exhaustive walk search algorithm iterates over 23k permutations in MD-impl and 107mil permutations in DD-impl. Note that the search algorithm is the same in both cases and only the number of partitions and their neighbors matter. Therefore Morse-based coverage path planning can take more time in particular cases.

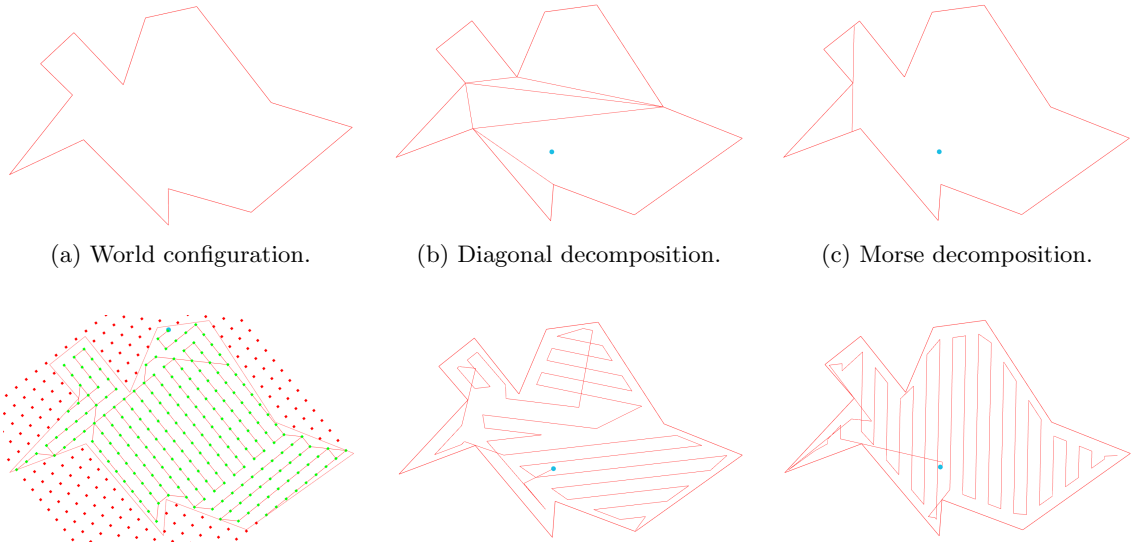


(a) World configuration. (b) Diagonal decomposition. (c) Morse decomposition.

(d) Coverage path generated by SM-impl. (e) Coverage path generated by DD-impl. (f) Coverage path generated by MD-impl.

Figure 3.5: Polygon decompositions (b, d) and coverage paths (c, e, f).

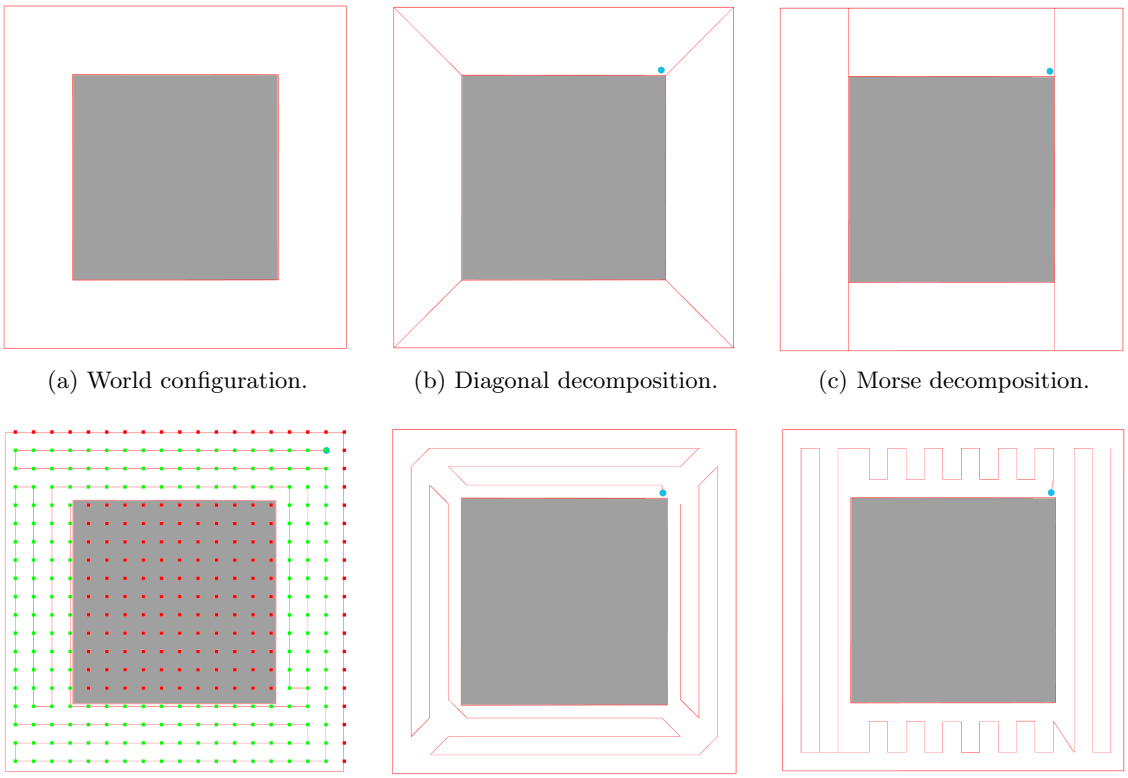
The next example shown in Fig. 3.6 is taken from [15] and [21]. DD-impl generated a path with 53 turns, MD-impl required 52 turns and SM-impl turned 70 times. Although the turn number is almost equal between the first two algorithms, it is clear that DD-impl misses even larger parts of the polygon due to the higher number of triangular partitions and acute angles. As it was mentioned, the merge process can decrease the number of partitions and thus the coverage percentage, however the next example will demonstrate, that sometimes the problem cannot be solved with merging only.



(a) World configuration. (b) Diagonal decomposition. (c) Morse decomposition.

(d) Coverage path generated by SM-impl. (e) Coverage path generated by DD-impl. (f) Coverage path generated by MD-impl.

Figure 3.6: Polygon decompositions (b, d) and coverage paths (c, e, f).



(a) World configuration. (b) Diagonal decomposition. (c) Morse decomposition.

(d) Coverage path generated by SM-impl. (e) Coverage path generated by DD-impl. (f) Coverage path generated by MD-impl.

Figure 3.7: Polygon decompositions (b, c) and coverage paths (d-f).

The last example shown in Fig. 3.7 is provided in order to show the drawback of the Morse-based coverage planning. In case the polygon can be decomposed into "long" convex partitions that are perpendicular to each other, the MD-impl (which can only cover a partition using one pre-defined sweep direction) requires too many turning maneuvers (see Fig. 3.7f). DD-impl finds the best sweep direction for each partition and therefore can find paths with fewer turns (Fig. 3.7e). The number of turns is the following: 24 for DD-impl, 56 for MD-impl and 23 for SM-impl. However, in this particular example using $f(x) = |x_1| + |x_2|$ function in Morse decomposition can generate a coverage path that will leave no gaps and yet have the number of turning maneuvers similar to DD-impl.

3.2.3 Summary

	Turn number			Path length (meters)			Visible gaps		
Polygon	DD	MD	SM	DD	MD	SM	DD	MD	SM
Simple	47	36	41	1343	1500	1647	Yes	No	No
Concave	53	52	70	1058	1170	1288	Yes	No	No
Complex	92	95	132	1229	1458	1890	Yes	No	No
Ring	24	56	23	895	908	1157	No	No	No

Table 3.1: Performance summary for the implemented algorithms.

The quantitative data of the four presented problems are summarized in Table 3.1. Simple polygon is shown in Fig. 3.3, concave polygon in Fig. 3.6, complex polygon in Fig. 3.5 and ring polygon in Fig. 3.7. As the reader can see, MD-impl on average generates more efficient paths with a bigger coverage percentage. Certainly, it does not mean that the MD-impl coverage is always complete, but in our system, where the polygon tends to have a small number of large edges, MD-impl demonstrates the best results. In contrast, the path that is generated by SM-impl is far from optimal in the complex polygon. However, SM-impl is useful if an environment is complex enough and a user does not have time to plan the path using any other method, as the time complexity of this algorithm is linear.

Chapter 4

User Interface and World Manager

Defining safety areas for drones, i.e., the zones a drone is allowed to fly within, is crucial for preventing the drones from colliding with each other or with other objects as well as for complying with legal regulations. Therefore, the Control Manager [30] of the MRS UAV System has point and line verifications ensuring that a drone flies within the safety area. However, the implementation of the boundaries was not flexible and was hard to maintain, which resulted in simplifying the safety zone by deleting obstacles from it. Moreover, every change in the safety area needed a rebooting of the entire system. To solve these problems, the Safety Area Manager was implemented, taking on the responsibility of controlling the world configuration and verifying the positions of a drone.

As has been mentioned in Sec. 1, ROS topics and services constitute a great API, but they are not convenient for a person to use from outside of a program. RViz plugins aim to increase the convenience of the MRS UAV System usage by introducing the UI to the safety area management. Together with visualizing data from various topics, the UI simplifies mission planning and increases situational awareness during missions.

The World Manager tool (discussed in Sec. 4.1) is an RViz plugin aimed to simplify user communication with the Safety Area Manager. It allows for adding, deleting and modifying the world boundaries as well as obstacles within. Furthermore, saving and loading world configurations are implemented. Other plugins implemented within this thesis are:

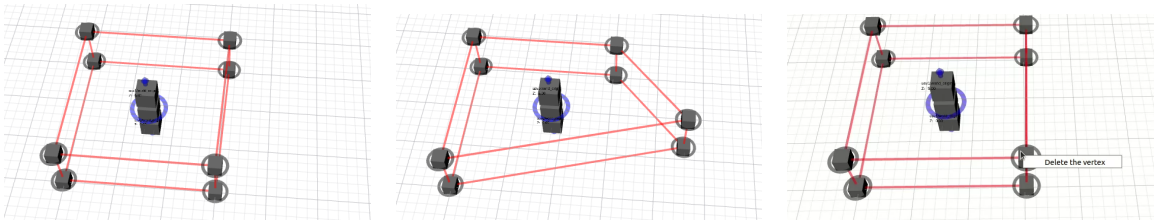
- Control tool for controlling UAVs remotely (discussed in Sec. 4.2)
- Waypoint Planner for setting multiple navigation goals (discussed in Sec. 4.3)
- Status display for monitoring UAV telemetry (discussed in Sec. 4.4)
- Satellite Overlay display for visualizing map tiles (discussed in Sec. 4.5)
- Coverage Path Planner (discussed in Sec. 4.6)

An example of the plugin usage in simulation is added to [1] to demonstrate the implemented UI.

4.1 Safety Area Manager

The Safety Area Manager operates with prisms and so-called inlying polygons. The inlying polygon is a polygon without holes and can be located at a specified height. The polygon does not influence point validity but can be used by other tools, such as the coverage path planner described in Sec. 4.6. A prism is a polyhedron with 2 parallel polygonal bases which are connected by parallelograms. The bases of a prism are strictly horizontal and the side edges are strictly vertical. There are two kinds of prisms: a safety zone and an obstacle. For a point to be considered valid, it must lie within the safety zone and must not lie within any obstacle. All of the prisms and inlying polygons can be customly changed using several kinds of interactive markers that are described below.

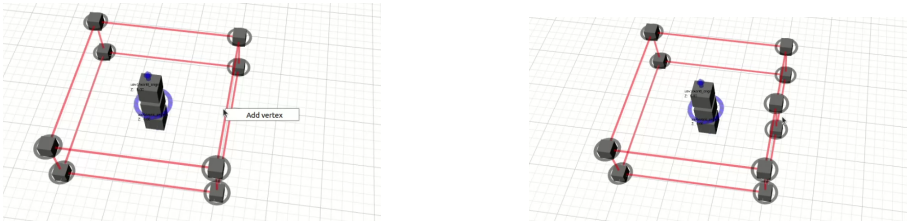
First, the vertex interactive marker (shown in Fig. 4.1) enables moving and deleting a vertex of a figure. To prevent computation errors, a vertex cannot be moved in a way that makes the polygon of a prism or inlying polygon invalid. A polygon is valid, if it has no internal intersections, its vertices are defined in clockwise order and it has at least 3 vertices. If a user tries to perform the action that makes the prism invalid, the prism stays in the last valid state until a new action is performed.



(a) A prism before moving the vertex. (b) The same prism, after moving the vertex. (c) Delete menu option.

Figure 4.1: Examples of interacting with a vertex marker.

Second, the edge interactive marker (shown in Fig. 4.2) displays individual edges and allows adding a new vertex in the middle of the horizontal edge. This functionality allows for defining complex prisms and inlying polygons, including setting convex polygons or approximating smooth shapes.



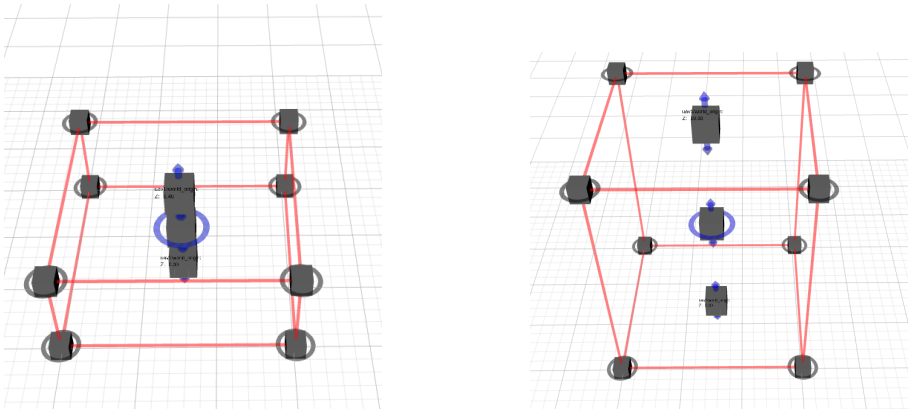
(a) Add vertex option. (b) Added vertex.

Figure 4.2: An example of interacting with an edge marker.

Third, bound interactive markers (shown in Fig. 4.3) handle the height of the prisms. Each prism has two interactive markers responsible for lower and upper bounds. These markers can be used to set the height of a prism or to move the whole figure in a 2D plane. Markers also provide the "Delete prism" option. However, the safety zone prism cannot be deleted, as no point is considered valid without it. Note that bound interactive markers are not used to visualize and configure inlying polygons, as the polygons do not have minimal and maximal height.

Fourth, the center interactive marker (shown in Fig. 4.4) enables rotating the entire figure, setting its height, moving it in a 2D plane, and deleting the entire prism or inlying polygon. Similar to the bounds interactive markers, the center interactive marker does not allow deleting the safety zone prism.

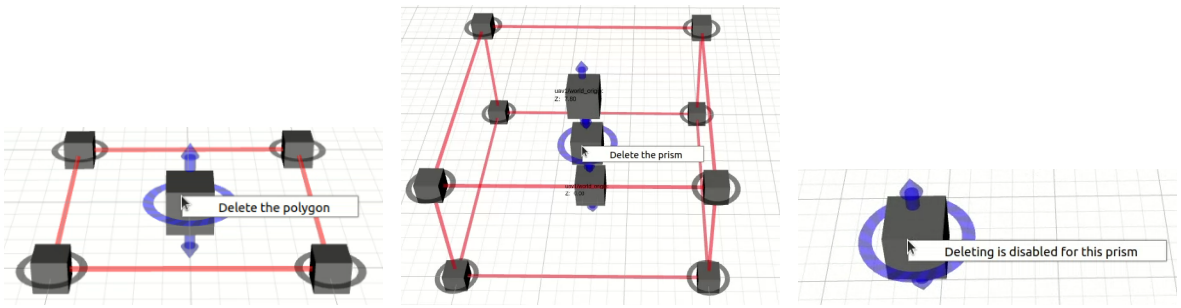
Fifth, a static edge marker complements all of the above to provide pure interactionless visualization. Since the interactive marker server implemented in ROS does not allow publishing a marker if it has not been changed, the static edge marker overcomes this issue by continuously publishing the markers to the network. The static edge marker mimics the state of the interactive elements and visualizes it cleanly.



(a) A prism before changing its upper bound.

(b) The same prism after an adjustment.

Figure 4.3: An example of interacting with a bounds marker.



(a) Delete an inlying polygon option.

(b) Delete a prism option.

(c) If a prism cannot be deleted, the info message occurs.

Figure 4.4: An example of interacting with a center marker.

To avoid setting world properties from scratch every time the system has been rebooted, Safety Area Manager provides a service for saving a current configuration into a file. Saved configuration can be loaded later as well as set as default configuration on the system start. A configuration file is a *.yaml* file, that includes the following parameters: (i) world origin, which defines the coordinate frame representing the Global Navigation Satellite System (GNSS)-based frame, (ii) coordinates of the vertices of the prisms and their upper and lower bounds. The data of the prisms are split into 2 sections: safety zone and obstacles. The safety zone section must be present in every world configuration, while the obstacles are optional.

The Safety Area Manager provides several ROS services for adding an obstacle, adding an inlying polygon, saving configurations and loading them. To avoid using them directly through a terminal, we implemented an RViz tool. The tool provides a menu with the services on right-click (see Fig. 4.5). Additionally, since the tool's class inherits from the `rviz::InteractiveTool`, it can also be used to interact with interactive markers (e.g., for setting up the prisms and the inlying polygons). Several examples of world configurations set up by this tool are presented in Fig. 4.6.

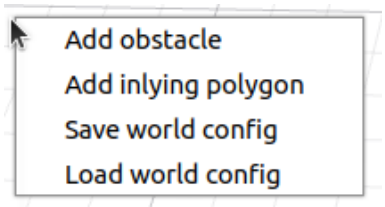
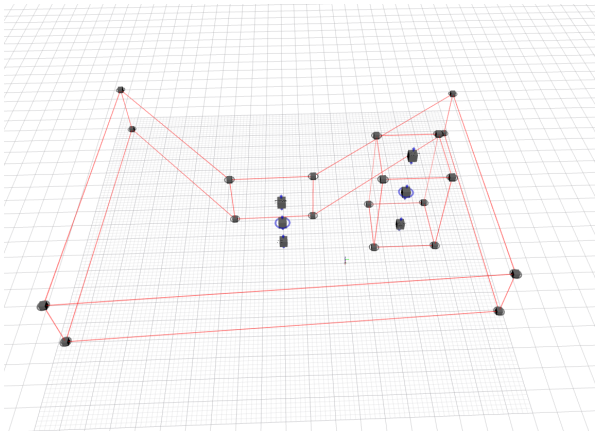
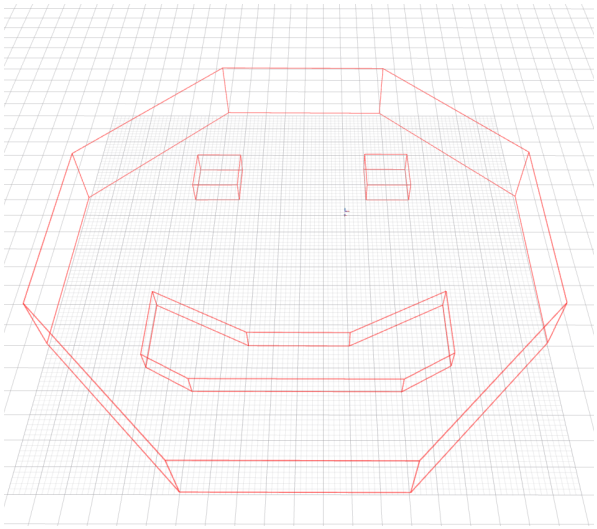


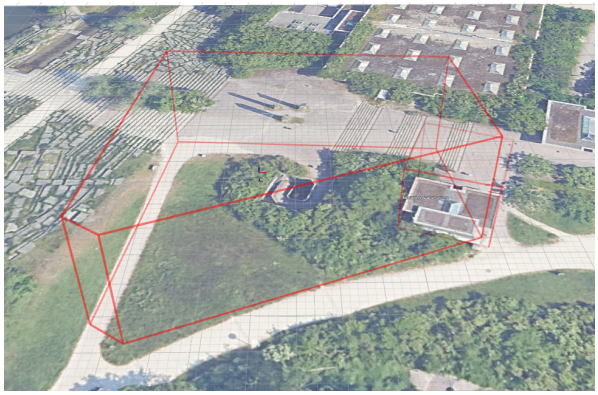
Figure 4.5: An example of a tool menu for the World Manager tool.



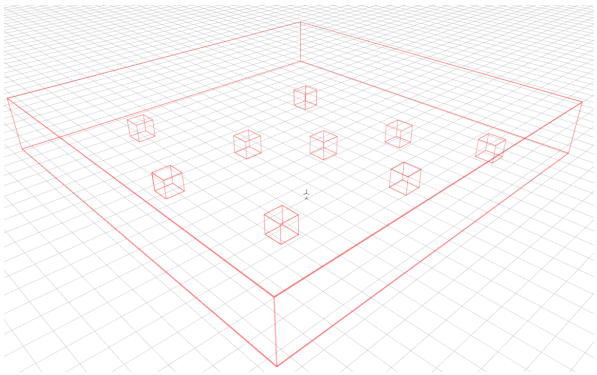
(a) Concave world with one simple obstacle. Supporting markers are turned on.



(b) Convex world with a concave obstacle.



(c) World that has been configured according to satellite maps. The building is considered to be an obstacle.



(d) Convex world with many obstacles.

Figure 4.6: Several examples of worlds configured with the Safety Area Manager and the World Manager tool. Supporting markers are turned off for (b-d), so the demonstrations are not overloaded.

4.2 Control tool

The Control tool enables controlling an UAV remotely through RViz. To control a drone, it must be first selected with a click-and-drag movement. The selection of multiple drones is

available. Due to the numerous keyboard bindings (see Table 4.1), convenient informational messages inform the user within the RViz status (located at the bottom left of the window).

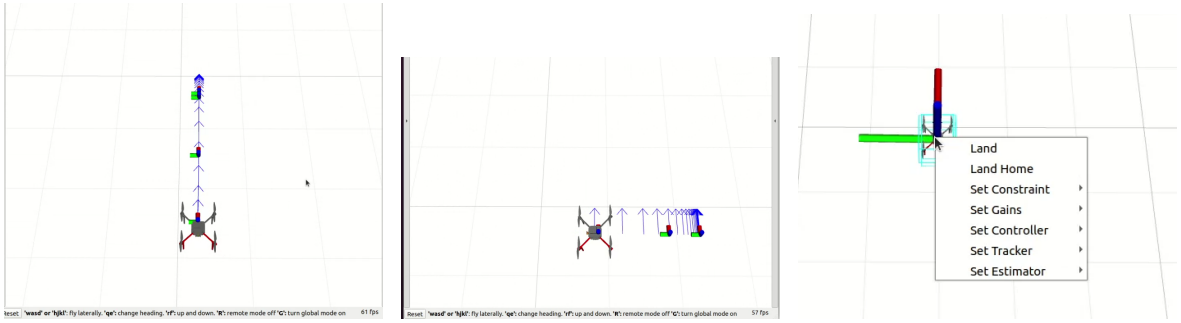
Key	Action
'wasd' or 'hjkl'	Fly laterally.
'qe'	Change UAV's heading.
'rf'	Fly up and down.
'R'	Turn the remote mode on/off.
'G'	Turn the global mode on/off.

Table 4.1: Control tool key bindings.

While in the remote mode, a user can fly the selected UAVs with the keyboard. By default, UAVs follow the commands in their Flight Controller Unit coordinate (FCU) frame, also known as the body frame of the UAV. If the global mode is activated, the commands are interpreted in the current world frame, i.e., X and Y axes are independent of the UAV heading.

A drone menu is available on right click. If the UAV is flying, a user can call the "land" or "land_home" services. If the UAV is on the ground, a user can call the "takeoff" service. Also, a user can change the controller, tracker, estimator, controller gains and tracker constraints. Furthermore, one can add custom services to the drone's menu by publishing a message to the topic *mrs_uav_status/set_trigger_service*. The message is a *std_msgs/String* type and has to consist of two entries separated by spaces: a service name (*uav_manager/land_home*) and a name to be displayed in the menu (Land Home). Unless the service name begins with "/", the namespace of the UAV will be added automatically (*uav_manager/land_home* will be remapped to */uav1/uav_manager/land_home*).

If several UAVs were selected, the common menu occurs on "m" key pressing. The common menu only shows the services that are common for all the selected UAVs, excluding custom services. The remote mode works the same as for one UAV.



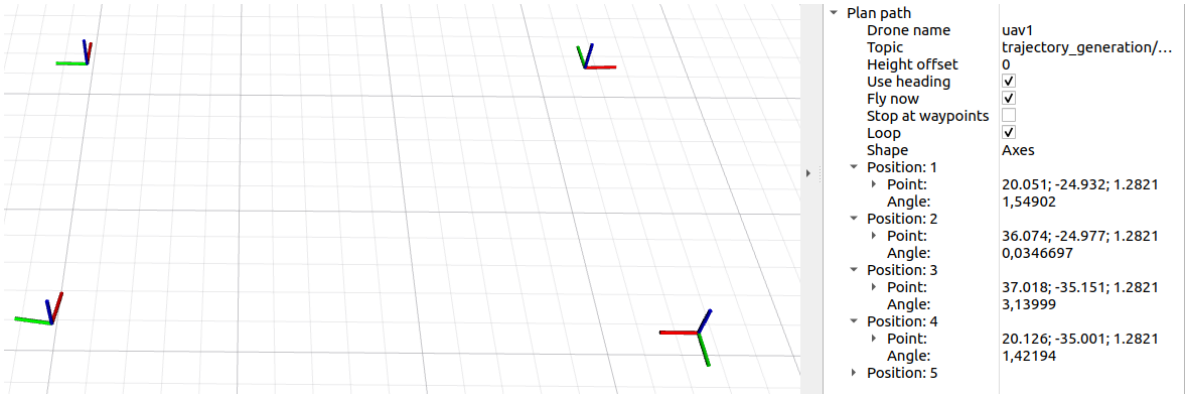
(a) Fly forward. (b) Fly right. (c) Drone menu example.

Figure 4.7: Examples of using Control tool.

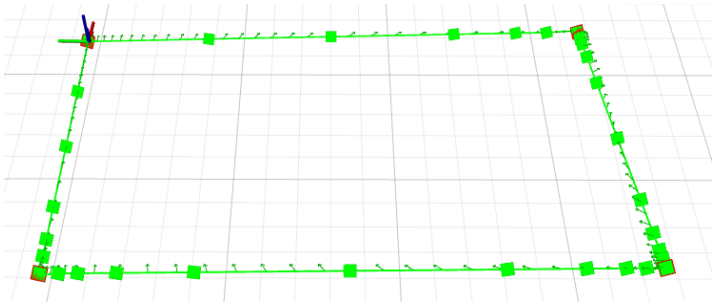
4.3 Waypoint planner

The Waypoint planner is an RViz tool that allows sending a sequence of waypoints to a drone. The click-and-pull input supplies a 2D position with a heading (a waypoint). The

tool options allow for changing the flight height, fixing the UAV heading, generating looping waypoint paths and waiting for an operator trigger to start the flight.



(a) Setting looped references.



(b) A drone following the looped trajectory.

Figure 4.8: Example of usage of the Waypoint Planner.

4.4 UAV Status display

UAV Status display brings the functionality of *mrs_uav_status* package [31] into the RViz. The plugin displays useful information about the UAV state and sensors (see Fig. 4.9). It also can show custom *std_msgs/String* messages published to the topic *mrs_uav_status/display_string*. Finally, the plugin can monitor the rates of different ROS topics and warn the user if the topic is published less or more frequently than required, or not published at all.

4.5 Satellite overlay

To make configuring the world even more convenient, we added visualization of worlds overlaid on Global Positioning System (GPS)-specified OpenStreet maps [9] (see Fig. 4.10). This plugin displays satellite maps loaded from the internet and is highly customizable. The plugin receives *sensor_msgs/NavSatFix* messages and loads corresponding map tiles. The map tiles are cached to *$HOME/.cache/rviz_satellite* and therefore the display can be used even without the Internet connection. Transformation of tiles to RViz fixed frame can be done in two ways that are configured using the Map Transform Type option:

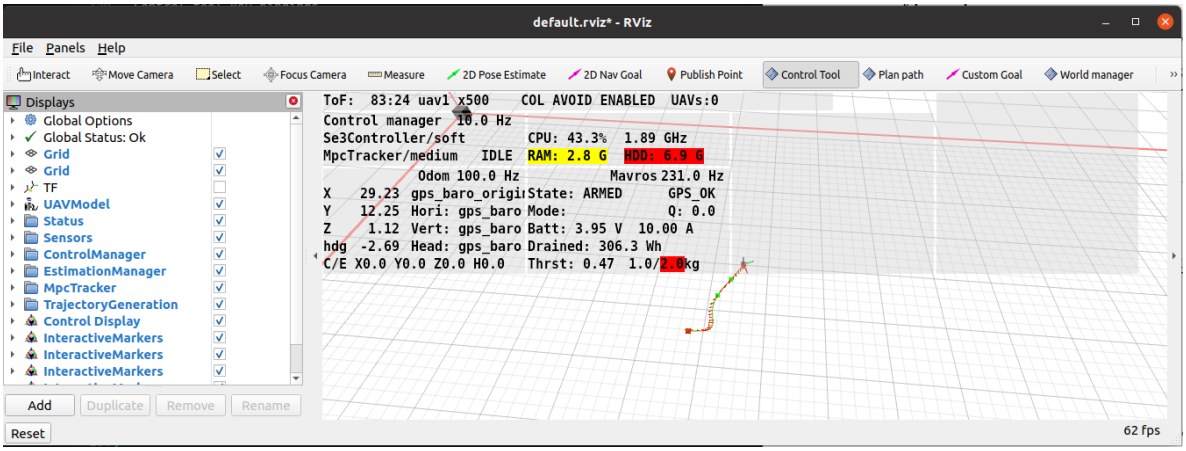


Figure 4.9: Example of Status display.

- Specify a Map frame, which is an East-North-Up (ENU)-oriented frame in which your robot localizes.
- Specify Universal Transverse Mercator (UTM) frame (and possibly UTM zone). In this mode, no map frame is required and the tiles are directly placed on their UTM positions. The subscribed NavSatFix messages are only used to determine the tiles to download, so small inconsistencies between the NavSatFix frame and the measured latitude/longitude are not a big problem.

Additionally, plugin options allow for changing tile transparency, resolution, number of adjacent blocks to load and the offset of displayed tiles in the Z coordinate. Finally, the tile server has to be specified using the form http://server.tld/{z}/{x}/{y}.jpg, where the tokens {z}, {x}, {y} represent the zoom level, x coordinate, and y coordinate respectively. If an API requires a pair of latitude and longitude values instead of x and y tile coordinates, the form http://server.tld/{z}/{lat}/{lon}.jpg is required, where {lat} and {lon} represent the latitude and longitude values of the requested location. It was decided to use https://mt1.google.com/vt/lyrs=y&x={x}&y={y}&z={z} as it has the most recent tiles and our usage is in line with the terms of usage [10].

4.6 Coverage path planner

One of the tasks that can be performed using UAVs is obtaining geo-referenced high-resolution aerial images. For that purpose, a proper CPP algorithm must be applied. The Coverage Path Planner implements several most suitable CPP algorithms, which are discussed in Sec. 2 and Sec. 3, and enables planning and visualizing the coverage missions. Saving and loading of a computed path is possible, as computations can take a long time in particular cases. Several parameters can be set in tool properties, such as the used method, height of the flight, view angle of a camera, photo overlap rate and start point. Furthermore, each of the methods can have parameters that influence the computation and affect the path optimality.

To plan a coverage path over a part of the world, a user can define an inlying polygon using the World Manager tool (described in Sec. 4.1) and choose it in Coverage Path Planner tool properties (see Fig. 4.11b). The intersection of the chosen inlying polygon and the safety area is computed and the resulting polygon is sent to the chosen coverage method. An example of partial world coverage is shown in Fig. 4.12.

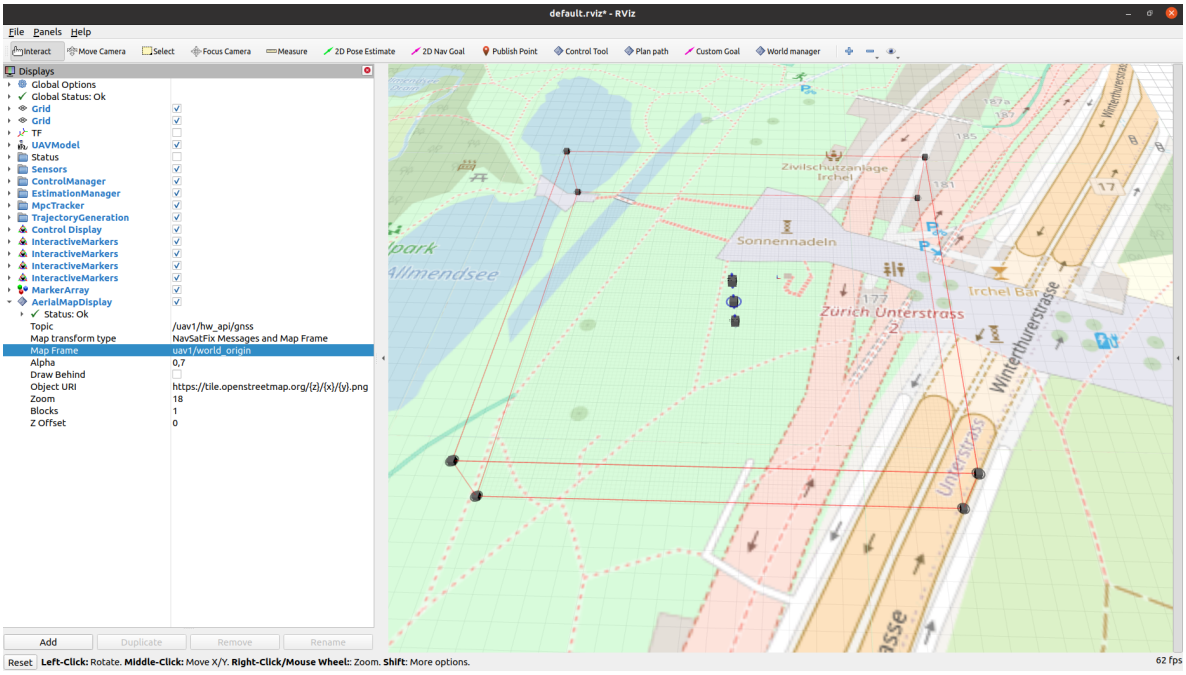
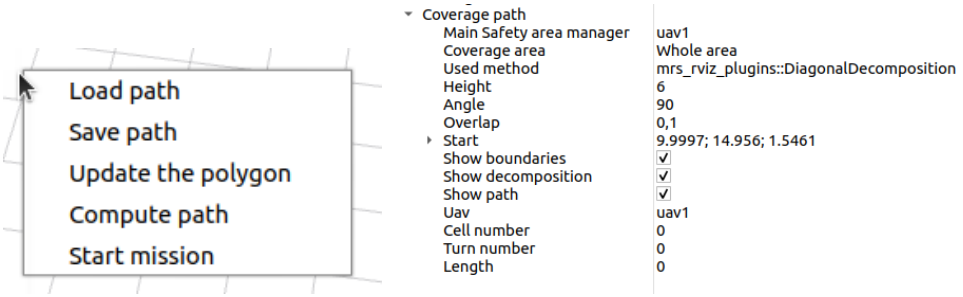


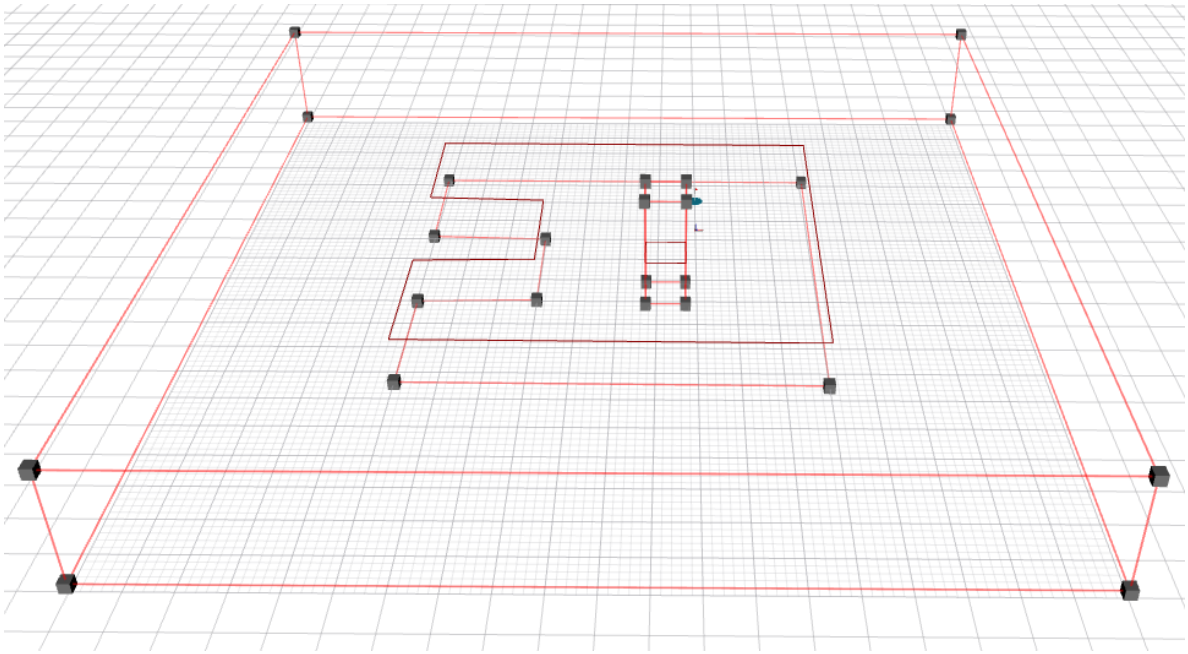
Figure 4.10: Example of the satellite display.

Coverage methods are loaded using ROS pluginlib package [17]. The base class is `mrs_rviz_plugins::CoverageMethod` and therefore it offers the possibility for integration of other coverage path planning methods. The plugin requests the Safety Area Manager for a safety polygon at a defined height and sends it to the chosen coverage method. As computations can take a long time, the "Update polygon" button was added, so a user can verify, if the current world configuration and coverage parameters generate the required coverage zone. The menu and configuration options are demonstrated in Fig. 4.11.

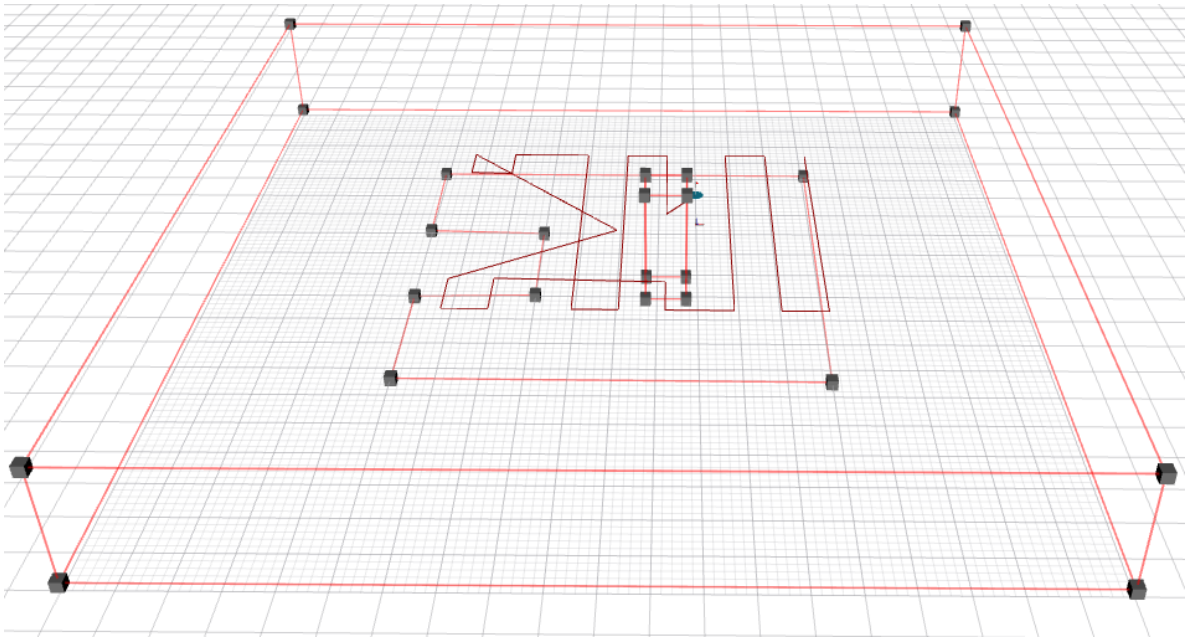


(a) Right-click menu. (b) Tool properties.

Figure 4.11: Coverage path planner example.



(a) The polygon considered for coverage



(b) The planned coverage path.

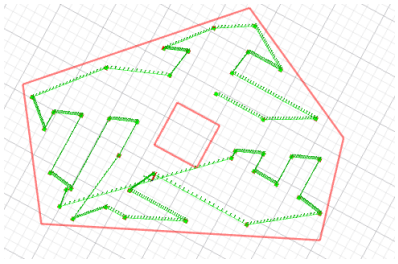
Figure 4.12: Coverage path planned over a part of the world using MD-impl algorithm (see Sec. 2.1.2 and Sec. 3.2.1).

Chapter 5

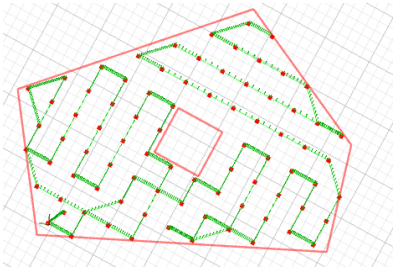
Real-world experiments

In order to test the robustness of Safety Area Manager and thus capability of the plugin to plan coverage paths in real-world environments, we performed experiments with a real drone. Also, other plugins mentioned in Sec. 4 were used in order to control the UAV and monitor its parameters. Waypoint planner (see Sec. 4.3) was used to navigate the UAV around the field, Control tool (see Sec. 4.2) was used for landing and remote control, and Status display helped to keep track of the flight parameters of the vehicle.

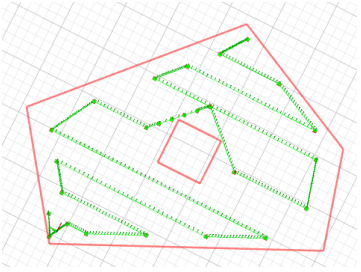
Safety Area Manager and RViz plugins performed admirably. Several bugs causing unexpected behavior have been found during the experiments but none of them was critical or made the drone fall. All of these software issues have been corrected in the final implementations. However, the coverage path planning algorithms had errors in their computations, which resulted in non-optimal paths during real-world experiments. The errors have been fixed and therefore each experiment is supplemented by paths generated by correct algorithms (marked as "(corrected)" in the following figures). These corrected paths were generated later on the same worlds. Moreover, each experiment was recorded on camera and the videos can be found in [1]. Links to particular experiments are added to the corresponding figures.



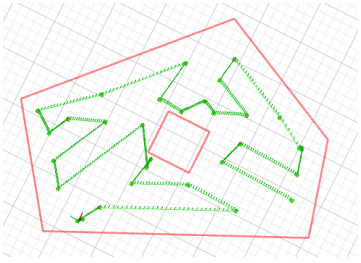
(a) Diagonal decomposition (flown).



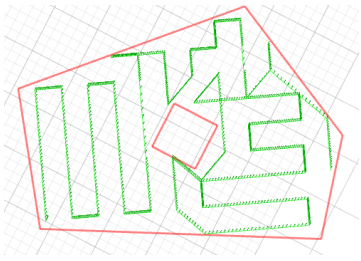
(b) Stride method (flown).



(c) Morse decomposition (not flown).

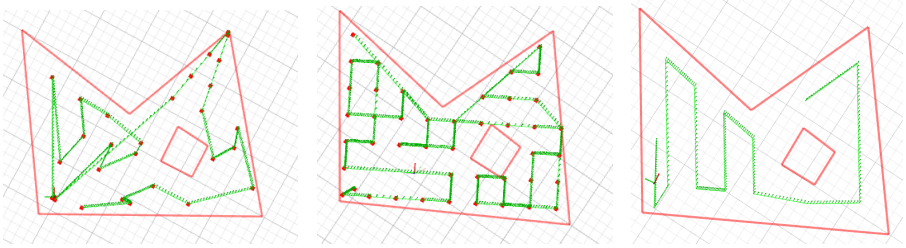


(d) Diagonal decomposition (corrected).



(e) Stride method (corrected).

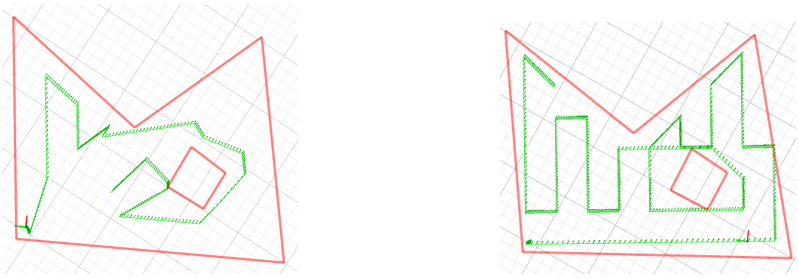
Figure 5.1: Experiment 1.



(a) Diagonal decomposition (flown).

(b) Stride method (flown).

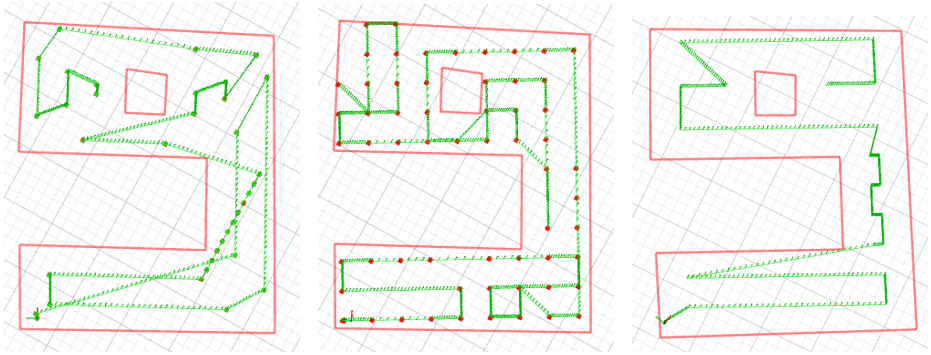
(c) Morse decomposition (not flown).



(d) Diagonal decomposition (corrected).

(e) Stride method (corrected).

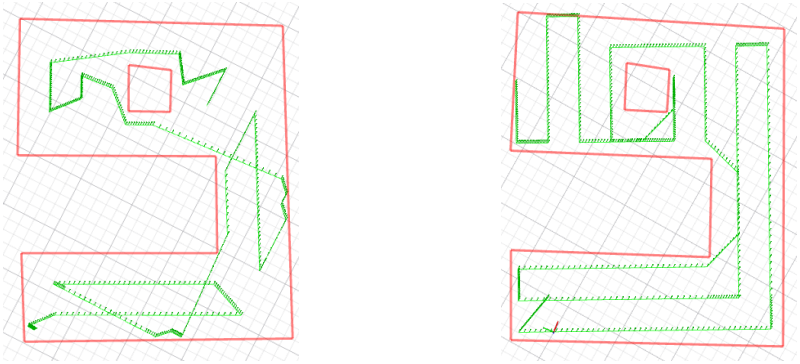
Figure 5.2: Experiment 2.



(a) Diagonal decomposition (flown).

(b) Stride method (flown).

(c) Morse decomposition (not flown).



(d) Diagonal decomposition (corrected).

(e) Stride method (corrected).

Figure 5.3: Experiment 3.

The coverage flight time (in seconds) is presented in Table 5.1. The postfix "-old" corresponds to the algorithms used to paths in the real-world experiments, i.e., subfigures a-b in Fig. 5.1, 5.2, 5.3. The algorithms with "-impl" are corrected algorithms presented in subfigures c-d in Fig. 5.1, 5.2, 5.3.

	DD-old	SM-old	MD-impl	SM-impl	DD-impl
Experiment 1	155	199	98	95	140
Experiment 2	89	148	55	53	118
Experiment 3	111	156	96	85	113

Table 5.1: Coverage time for performed experiments.

To sum up, the designed UI remarkably improves the user experience, eliminating the need to memorize ROS services and their parameters. Interactive safety area boundaries enabled the performing of the series of experiments on different world configurations without rebooting the UAV. The implemented plugins have shown high robustness and therefore can be used in other mission planning.

Chapter 6

Achieved objectives

In this thesis, the following objectives have been achieved:

(A) Multiple plugins and the Safety Area Manager were implemented, which
   1. contain a world representation using a non-convex polygonal structure in the horizontal plane and variable height bounds (described in Sec. 4.1),
   2. incorporate non-convex obstacles of different heights and sizes (described in Sec. 4.1),
   3. visualize GPS-based photomaps in outdoor cases (described in Sec. 4.5),
   4. allow defining, loading, and saving of the world with the obstacles (described in Sec. 4.1),
   5. enable visualizing flight telemetry of UAVs (described in Sec. 4.4),
   6. allow interaction of both virtual and real UAVs within the same visualization, including individual and multi-UAV selection, waypoint and key bindings navigation (described in Sec. 4.3 and Sec. 4.2),
   7. are integrated into the MRS UAV System (described in Sec. 1 and Sec. 4).

(B) In addition, several path planning algorithms that maximize the total coverage inside the non-convex world are implemented. This task included:
   1. reviewing (see Sec. 2) and comparing (see Sec. 3) four state-of-the-art coverage path planning algorithms suitable for UAVs,
   2. implementing three most suitable coverage path planning algorithms (described in Sec. 3.2),
   3. enabling planning coverage paths inside the entire world (A) as well as in user-definable safe zones inside (A) (described in Sec. 4.6),
   4. enabling interactive specification and parametrization of the path planning algorithm inside RViz (described in Sec. 4.6),
   5. enabling loading and saving the coverage paths for a given world (described in Sec. 4.6),
   6. offering the possibility for integration of other coverage path planning methods (described in Sec. 4.6).

(C) Finally, the experiments with the MRS UAV System in the real world are performed (see Sec. 5). During the experiments, (A) and (B) are used to fly an autonomous UAV along the planned path. The performance of the implemented algorithm are analyzed and discussed both qualitatively and quantitatively (see Sec. 3.2).

Chapter 7

Conclusion

In summary, the implemented features significantly enhance the functionality and usability of the MRS UAV System. We have introduced numerous tools for configuring world properties, remotely controlling UAVs, setting multiple navigation goals, and efficiently visualizing on-board UAVs's data on remote stations. The display of satellite maps, which was included in the plugin list, allows the mentioned features to be used with greater precision. Stand-alone Safety Area Manager improves the maintainability and usability of the MRS UAV System. Additionally, several most suitable coverage path planning algorithms were implemented and compared. To the extent of our knowledge, our plugin is the first open-source non-discrete implementation of Morse decomposition, that can be extended with any other Morse function. Last but not least, we tested the implemented features in real-world experiments, which showed the high robustness of the Safety Area Manager as well as the implemented plugins.

There are still opportunities to improve the user experience of the MRS UAV System. For example, the Rviz Satellite display waits for GPS data to be received from a drone. Since not all of the drones are equipped with such hardware and the world position is set in advance, the plugin could request such data from the Safety Area Manager directly in order to initialize the image. Moreover, inlying polygons are not currently a part of the world config. This is left for future development.

Chapter 8

References

[1] A. Mukhametshin. (2024). Additional materials to the thesis, [Online]. Available: https://mrs.fel.cvut.cz/theses/mukhametshin2024 (visited on May 17, 2024).

[2] T. Baca, R. Penicka, P. Stepan, M. Petrlik, V. Spurny, D. Hert, and M. Saska, "Autonomous Cooperative Wall Building by a Team of Unmanned Aerial Vehicles in the MBZIRC 2020 Competition," *Robotics and Autonomous Systems*, vol. 167, p. 104 482, Sep. 2023.

[3] D. Hert, T. Baca, P. Petracek, V. Kratky, R. Penicka, V. Spurny, M. Petrlik, M. Vrba, D. Zaitlik, P. Stoudek, V. Walter, P. Stepan, J. Horyna, V. Pritzl, M. Sramek, A. Ahmad, G. Silano, D. Bonilla Licea, P. Stibinger, T. Nascimento, and M. Saska, "MRS Drone: A Modular Platform for Real-World Deployment of Aerial Multi-Robot Systems," *Journal of Intelligent & Robotic Systems*, vol. 108, pp. 1–34, 64 2023.

[4] P. Petracek, V. Kratky, T. Baca, M. Petrlik, and M. Saska, "New Era in Cultural Heritage Preservation: Cooperative Aerial Autonomy for Fast Digitalization of Difficult-to-Access Interiors of Historical Monuments," *IEEE Robotics and Automation Magazine*, pp. 2–19, 2023.

[5] M. Petrlik, P. Petracek, V. Kratky, T. Musil, Y. Stasinchuk, M. Vrba, T. Baca, D. Hert, M. Pecka, T. Svoboda, and M. Saska, "UAVs Beneath the Surface: Cooperative Autonomy for Subterranean Search and Rescue in DARPA SubT," *Field Robotics*, vol. 3, pp. 1–68, 2023.

[6] M. Vrba, V. Walter, V. Pritzl, M. Pliska, T. Báča, V. Spurný, D. Heřt, and M. Saska, *On onboard LiDAR-based flying object detection*, 2023. arXiv: 2303.05404.

[7] D. Hert, T. Baca, P. Petracek, V. Kratky, V. Spurny, M. Petrlik, M. Vrba, D. Zaitlik, P. Stoudek, V. Walter, P. Stepan, J. Horyna, V. Pritzl, G. Silano, D. Bonilla Licea, P. Stibinger, R. Penicka, T. Nascimento, and M. Saska, "MRS Modular UAV Hardware Platforms for Supporting Research in Real-World Outdoor and Indoor Environments," in *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2022, pp. 1264–1273.

[8] T. Baca, M. Petrlik, M. Vrba, V. Spurny, R. Penicka, D. Hert, and M. Saska, "The MRS UAV System: Pushing the Frontiers of Reproducible Research, Real-world Deployment, and Education with Autonomous Unmanned Aerial Vehicles," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 26, pp. 1–28, 2021.

[9] G. Cross and A. Schröder. (2020). Rviz_satellite, [Online]. Available: https://github.com/nobleo/rviz_satellite (visited on May 17, 2024).

[10] Google. (2020). Google maps platform terms of service, [Online]. Available: https://cloud.google.com/maps-platform/terms?_gl=1*l38tlk*_ga*NDEzMzc2MDE4MC4xNjg4MzMxMDkx*_ga_NRWSTWS78N*MTcwNjYxMjUzNC4xLjEuMTcwNjYxMzAyOC4wLjAuMA.. (visited on May 17, 2024).

[11] P. Petráček, V. Walter, T. Báča, and M. Saska, "Bio-Inspired Compact Swarms of Unmanned Aerial Vehicles without Communication and External Localization," *Bioinspiration & Biomimetics*, vol. 16, no. 2, p. 026 009, 2020.

[12] T. M. Cabreira, L. B. Brisolara, and P. R. Ferreira Jr., "Survey on coverage path planning with unmanned aerial vehicles," *Drones*, vol. 3, no. 1, 2019.

[13] D. Hershberger, D. Gossow, J. Faust, and W. Woodall. (2018). Rviz package, [Online]. Available: http://wiki.ros.org/rviz (visited on May 17, 2024).

[14] Y. Bouzid, Y. Bestaoui, and H. Siguerdidjane, "Quadrotor-uav optimal coverage path planning in cluttered environment with a limited onboard energy," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 979–984.

[15] M. Torres Anaya, D. Pelta, J. Verdegay, and J. Torres, "Coverage path planning with unmanned aerial vehicles for 3d terrain reconstruction," *Expert Systems with Applications*, vol. 55, Feb. 2016.

[16] C. Di Franco and G. Buttazzo, "Energy-aware coverage path planning of uavs," in *2015 IEEE International Conference on Autonomous Robot Systems and Competitions*, 2015, pp. 111–117.

[17] E. Marder-Eppstein, T. Foote, D. Thomas, and M. Shah. (2015). Pluginlib package, [Online]. Available: `http://wiki.ros.org/pluginlib` (visited on May 17, 2024).

[18] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[19] E. Santamaria, F. Segor, I. Tchouchenkov, and R. Schönbein, "Path planning for rapid aerial mapping with unmanned aircraft systems," in *ICONS 2013 : The Eighth International Conference on Systems*, Feb. 2013, pp. 82–87.

[20] E. Santamaria, F. Segor, and I. Tchouchenkov, "Rapid aerial mapping with multiple heterogeneous unmanned vehicles," in *International Conference on Information Systems for Crisis Response and Management*, 2013.

[21] Y. Li, H. Chen, M. Joo Er, and X. Wang, "Coverage path planning for uavs based on enhanced exact cellular decomposition method," *Mechatronics*, vol. 21, no. 5, pp. 876–885, 2011, Special Issue on Development of Autonomous Unmanned Aerial Vehicles.

[22] F. Segor, A. Bürkle, M. Kollmann, and R. Schönbein, "Instantaneous autonomous aerial reconnaissance for civil applications," in *International Conference on Systems (ICONS) 2011*, Jan. 2011.

[23] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 7681–7687.

[24] J. Fernández, B. G.-Tóth, L. Cánovas, and B. Pelegrín, "A practical algorithm for decomposing polygonal domains into convex polygons by diagonals," *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, vol. 16, pp. 367–387, Feb. 2008.

[25] I. Maza and A. Ollero, "Multiple uav cooperative searching operation using polygon area decomposition and efficient coverage algorithms," in. Jan. 2007, vol. 6, pp. 221–230.

[26] E. Acar, H. Choset, A. Rizzi, P. Atkar, and D. Hull, "Morse decompositions for coverage tasks," *I. J. Robotic Res.*, vol. 21, pp. 331–344, Apr. 2002.

[27] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

[28] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon decomposition," in *Proceedings of 1st International Conference on Field and Service Robotics (FSR '97)*, 1997, pp. 216 –222.

[29] J Milnor, "Morse theory.," *Princeton, New Jersey: Princeton University Press.*, 1963.

[30] T. Baca. (). Control manager, [Online]. Available: `https://github.com/ctu-mrs/mrs_uav_managers` (visited on May 17, 2024).

[31] D. Hert. (). Mrs_uav_status, [Online]. Available: `https://github.com/ctu-mrs/mrs_uav_status` (visited on May 17, 2024).