Adaptive Path Planning for Navigation and Sensing of Micro Aerial Vehicles

by

Seyed Abbas Sadat Kooch Mohtasham

M.Sc., Simon Fraser University, 2010
B.E., Iran University of Science and Technology, 2008

Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
School of Computing Science
Faculty of Computing Science Science

SIMON FRASER UNIVERSITY
Summer 2016

Approval

Name:	**Seyed Abbas Sadat Kooch Mohtasham**
Degree:	**Doctor of Philosophy (Computing Science)**
Title:	***Adaptive Path Planning for Navigation and Sensing of Micro Aerial Vehicles***
Examining Committee:	**Chair:** Nick Sumner Assistant Professor

Richard Vaughan
Senior Supervisor
Associate Professor _____

Brian Funt
Supervisor
Professor _____

Greg Mori
Internal Examiner
Professor
School of Computing Science _____

Gregory Dudek
External Examiner
Professor
School of Computer Science
McGill University _____

Date Defended: 3 August 2016

Abstract

Micro Aerial Vehicles (MAVs) have gained much attention as data collection and sensing platforms. Professionals in many fields can access rich sensory data with fast update rates by performing automated aerial surveys using MAVs. However, these robots have limited payloads and short flight times. Therefore, it is useful to perform a task with light and low-powered sensors and as quickly as possible. In this proposal, we consider some of the fundamental tasks performed by MAVs and propose methods by which a MAV can achieve these tasks more efficiently and robustly.

In the first part of this thesis, we consider the task of navigation in which a MAV, using visual Simultaneous Localization and Mapping (SLAM) to map the environment and localize itself within it, moves from its current location to a goal location. As SLAM is highly dependent on the visibility of visual features, we propose an adaptive path planning approach that avoids visually-poor regions of the environment and can generate safe trajectories for the MAV to perform the navigation task.

In the second part, an aerial coverage task is considered where the MAV must map interesting regions with initially unknown locations. Rather than using an exhaustive 'lawnmower' coverage pattern that sweeps the entire region uniformly, we propose non-uniform coverage strategies that adaptively cover all the interesting regions with high resolution and coarsely sweep the rest of the area. The proposed methods generate shorter trajectories compared to a uniform lawnmower pattern.

In the last part of the thesis, we assume a time/energy budget for the vehicle and consider specific costs for different manoeuvres of the MAV. We introduce a novel problem of guaranteeing complete coverage of an area at low resolution, while identifying regions of interest (ROIs), and locally surveying as much of the ROIs at high resolution as the battery allows. Three different policies are proposed to decide when to switch between the coarse survey and high resolution imagery data collection.

Keywords: Micro aerial vehicle, path planning, active SLAM, arial coverage

To my parents,
Mina and Mohsen

Acknowledgements

First, I would like to thank my advisor Dr. Richard Vaughan for his great support during all the years I have been at SFU. He was a kind and patient teacher and always provided valuable guidelines for me. I will remember everything I learned from him in my future careers. I would also thank Dr. Brian Funt and Dr. Greg Mori for their precious reviews and comments. I am thankful to Dr. Jens Wawerla for his helpful suggestions during the past 6 years.

I am thankful to all my friends and colleagues at the Autonomy Lab. Mani, Shokoofeh, Jake, Jack, Jacob, Lingkang, and Sepehr, for the fruitful discussions that we had in our lab meetings. Also, their valuable comments increased the quality of this work. Besides, I grateful for all the moments they shared with me during these years. I also would like to thank Kyle Chutskoff and Damir Jungic who worked with me in summer 2013. A part of this thesis became possible with their help which I am grateful for.

During my PhD studies, I was lucky to be a member of the NSERC Canadian Field Robotics Network (NCFRN) and attend the annual field trials for 3 years. I had the chance to meet excellent researchers in my field and make a lot of friends that share the same passions. They taught me a lot about field experiments and that how exciting it can be. Here, I would like to thank all the members of the NCFRN, specially the director of the network, Dr. Gregory Dudek for making this collaboration possible.

Finally, I am very grateful to my family for their support and sacrifices through all these years. I would like to specially thank my wife, Neda, for all her dedications and caring supports without which I could never get here.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Micro Aerial Vehicles (MAVs)[1] have gained much attention as data collection and sensing platforms. Professionals in many fields can access rich sensory data with fast update rates by performing automated aerial surveys using MAVs. Aerial imagery with unmanned aerial vehicles (UAVs) in natural resource management has been the focus of researchers for applications such as precision agriculture, natural disaster management, aquatic system management, polar remote sensing, and wildlife research [1]. In surveillance applications, UAVs are used to track ground targets or perform continuous patrolling in a certain area [2, 3]. Industrial inspection tasks that are hazardous to humans have also been a potential application for UAVs that can fly almost anywhere without any need for further infrastructure [4]. Moreover, due to their fast response time, UAVs have been a useful platform to acquire data for post-disaster assessment and environmental monitoring [5].

In contrast to ground robots which typically have adequate power and can carry various sensors and batteries of high capacity, MAVs can carry limited payloads and the heavier they become, the shorter the flight time. Extra payload will also affect the flight stability of MAVs [6]. Hence the sensor suite on-board these robots should be light-weight and consume little power. Therefore, when used for data collection, a camera is generally the most important and widely used sensor on-board these robots due to its low power consumption, weight, and rich sensory data. In addition to the constraints in the on-board sensors, MAVs are subject to short flight times. Although fixed-wing UAVs can fly for relatively long time (50 min), typical quadrotors have a shorter flight time (25 min).

In the last decade, small-scale aerial vehicles have evolved into robots with high degree of autonomy. General purpose flight controllers are now available off-the-shelf[2] and have been reliably used in various vehicles, enabling low-level autonomy. However, long term trajectory planning and high-level behaviour generation can help to deal with challenges that arise due to the constraints in sensor types, insufficient on-board processing power, and short flight time. In this document, we consider some of the problems that arise due to the limitations of these platforms, and propose methods that can help MAVs accomplish autonomous navigation and

[1] We use MAV and UAV (*unmanned aerial vehicle*) interchangeably in this thesis.

[2] DJI NAZA (www.dji.com), 3DR Pixhawk (www.3dr.com), Open Pilot CC3D (www.opwnpilot.com) and etc.

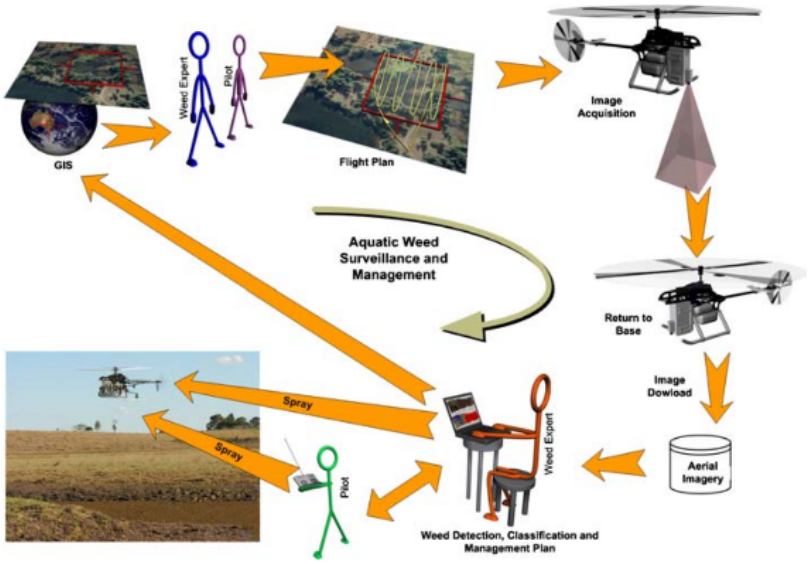


Figure 1.1: Weed surveillance and management process flow. [7]

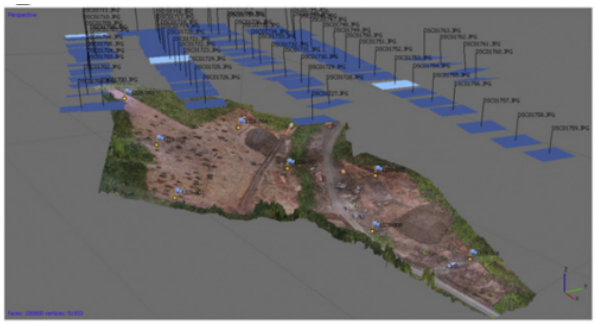
survey tasks more reliably and efficiently in spite of the mentioned limitations in the platform. The tasks we consider here involve path planning in an initially unknown environment and re-planning upon availability of new information. We show that the robustness, efficiency, and productivity of MAVs in performing a task can be increased by the proposed adaptive path planning methods.

1.1 Example Applications

One of the UAV tasks that we consider in this thesis is high resolution coverage of unknown regions in a specified area. The goal of this task is to sweep all the interesting regions from a predefined distance. This can be assumed to be a solely data collection or imagery mission. Alternatively, the MAV can perform an action that is not merely sensing. For example, consider aquatic weed surveillance and management using MAVs [7]. In the initial flights of MAV, images of the target region are collected and transferred to the base. Human experts supervise the automated weed detection algorithms and finally a weed infestation map is generated. In subsequent flights, the MAV is used to spray chemicals on potential regions (see Figure 1.1). This step is done from a safe distance close to the ground that ensures the accuracy of spraying. Additionally, in precision agriculture, farmers have started using MAVs to increase yields and decrease crop damage [8]. Crop dusting, seeding, and monitoring plant quality are among the MAV applications in farms (see Figure 1.2a).

Autonomous micro aerial vehicles promise to reduce labour costs and human errors and improve the quality of performing some tasks. SalamÃŋ *et al.* [9] organize UAV applications in vegetation remote sensing into three categories: (i) passive applications, where the collected data are used to generate maps for long term use, (ii) proactive applications, in which the gathered data are immediately processed off-line and used for short-term decision-making (the

(a) DJI's MG-1 drone is capable of spraying chemicals over the crops through four downward-facing nozzles, enabling automatic crop-dusting [10]



(b) UAVs are used to collect aerial images from an area of interest. A map of the region is generated from the images using offline photogrammetry methods [11]

Figure 1.2: Example MAV applications.

process shown in Figure 1.1 falls in this category), and (iii) reactive applications where the UAV is capable of decision-making and acting upon detected situations in real-time. The purpose of this thesis is to provide planning algorithms and methods that increase the productivity and efficiency of MAVs in proactive and reactive applications. Specifically, parts of this thesis focus on how the trajectory of the vehicle should change upon availability of new information such that the short flight time is used effectively.

In aerial mapping and photogrammetry, UAVs are used to collect images while flying over the area of interest [11]. Mission planning software provides tools for offline waypoint generation and trajectory design. Typically, the flight altitude is fixed and the preplanned flight path is not modified during the mission (Figure 1.2b). The collected images are used in an offline process to generate a height-map of the area. The map can also be extended to include information such as material type, soil characteristics, etc. [1]. Subsequent missions are executed in order to collect higher-resolution images from sections that require more accurate observations. For example, consider the regions with vague classification of materials or frequent hight change. With additional data from closer distance, these areas can be modelled more precisely. In this thesis, we focus on methods that adaptively change the survey path of the MAV such that the resolution of the collected data is adapted online based on necessity, and hence eliminate the need for multiple data-gathering missions.

1.2 Contributions

The contributions of our research can be summarized as follows:

- To improve robustness, we propose a path planning method for MAVs with vision-only Monocular SLAM. Current planning methods assume reliable performance of Visual SLAM in all environments and hence, the MAV fails upon moving to a region with poor visual saliency. Conversely, our proposed method generates safe paths to a goal according to the information richness of the environment. The planner runs on top of monocular SLAM

and uses the available information about the structure of the environment and feature visibility to find trajectories that maintain visual contact with feature-rich areas. The MAV continuously re-plans as it explores and updates the feature-points in the map. In real-world experiments we show that our system is able to avoid paths that lead into visually-poor sections of the environment by considering the distribution of visual features. If the same system ignores the availability of visually-informative regions in the planning, it is unable to estimate its location accurately and fails to reach its goal.

- To improve efficiency, we propose a novel method for non-uniform terrain coverage using MAVs. The existing methods for coverage path planning consider a uniformly interesting target area and hence all the regions are covered with high resolution. However in many real world applications items of interest are not uniformly distributed but form patches of locally high interest. The proposed coverage strategies generate shorter trajectories compared to lawnmower, and achieve sparse sampling of uninteresting sections of the environment and high resolution sampling of interesting patches.

- To improve productivity, we introduce the problem of guaranteeing complete coverage of an area at low resolution, while identifying regions of interest (ROIs), and locally surveying as much of the ROIs at high resolution as the flight time allows. The existing methods decompose this task into two separate subtasks which are accomplished separately, postponing the high resolution data collection. However, our proposed policies accomplish these two subtasks simultaneously, which is important in applications where the reward is time-discounted. We assume a time/energy budget for the vehicle and consider specific costs for different manoeuvres of the MAV. Different policies are proposed to decide when to switch between the coarse survey and high resolution imagery data collection.

1.3 Thesis Outline

In Chapter 2, we present the related work in the area of visual navigation and active localization and mapping. Also, coverage path planning methods for mobile robots and specifically UAVs are reviewed. In Chapter 3, we introduce our approach towards robust navigations of MAVs using monocular SLAM and report the results of real-robot experiments. In Chapter 4 we propose coverage strategies that consider the non-uniformity of the environment, and present simulation and field experiments that show effectiveness of our adaptive approach. In Chapter 5, we introduce active aerial survey and propose and compare various policies to achieve this task. Finally, we provide a conclusion in Chapter 6 and discuss the possible directions of future research.

Chapter 2

Background

This chapter contains a short summary of the prior work in localization-aware path planning, aerial coverage, and other related topics on which the rest of the thesis is built on. We begin with an introduction to SLAM and perception-aware path planning. We then present conventional coverage path planning methods and discuss the main methods used for aerial coverage.

2.1 Visual Navigation

Autonomous navigation of mobile robots in an environments requires the availability of pose information. For outdoor applications, GPS can be used to localize the robot. However, the accuracy of GPS decreases when the robot is close to buildings or trees. For indoor applications, external motion capture devices (e.g. Vicon[1]) can be used to obtain high frequency and accurate position estimations. However, they require modification of the environment. *Simultaneous localization and mapping* (SLAM) methods have been developed in the last decade and successfully used to estimate the pose of a robot. Various sensors such as range sensors [12], RGBD cameras [13], stereo cameras [14], and monocular cameras [15, 16, 17] has been used with SLAM. Although active sensors such as range sensors and RGBD cameras have been successfully deployed on-board MAVs for localization and mapping [18, 19, 20], they reduce the flight time due to their weight and power consumption. On the other hand, monocular visual SLAM seems to have become the preferred approach for MAV localization as cameras are light, consume relatively little power and have rich sensory data [21, 22, 23].

2.1.1 Monocular Visual SLAM

Given a sequence of the images captured by a moving camera, *structure-from-motion* (SFM) can reconstruct the 3D scene, while estimating the camera trajectory [24]. SFM employs optimization techniques, such as bundle adjustment [25], to solve for both scene structure and the camera motion (see Figure 2.1). While structure-from-motion is traditionally considered in an offline fashion, visual SLAM is intended to achieve both mapping and localization in real-time.

[1] www.vicon.com

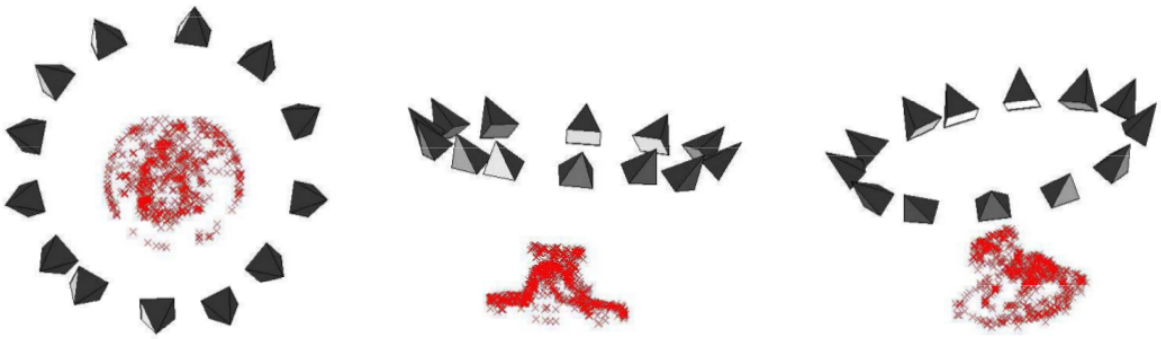


Figure 2.1: Structure-from-motion can estimate the camera poses while reconstructing the 3D object [26].

Monocular visual SLAM was originally solved by a filtering approach [27, 28], where the location of the map points and the camera trajectory were jointly estimated by a filter at each frame. However, this approach suffered from waste of computational resources on frames with little new information and the accumulated linearization errors. Conversely, keyframe-based methods construct the 3D scene using a subset of the frames [29, 15]. These method take advantage of the costly bundle-adjustment optimization to estimate the map accurately, but at lower update rate compared to the frame-rate. It has been shown [30] that the keyframe-based approach to SLAM outperforms filtering, as it provides the most accuracy per unit of computing time.

With their landmark work, G. Klein and D. Murray introduced parallel tracking and mapping (PTAM), in which the idea of separating mapping (scene reconstruction and camera trajectory estimation) from tracking (camera localization) is used. Consequently, PTAM can provide the pose of the camera relative to the current map in real-time, while the map is reconstructed at a lower pace. PTAM is shown to be successful for real-time augmented reality applications in small workspaces. Many other visual SLAM methods have been proposed that correct the scale drift [31], operate at high frame-rates [16], are scalable [32, 33], provide semi-dense maps [34], and show robustness to the camera motions and the environment [17].

In SLAM, it is assumed that the camera is a passive sensor or the input is a sequence of recorded images. However, as proposed by Bajcsy [35], control strategies can improve the quality of sensor data. Hence, many works in the literature exists that focus on the problem of *active perception* in various settings. In the following, we summarize some of the previous works on active reconstruction and localization.

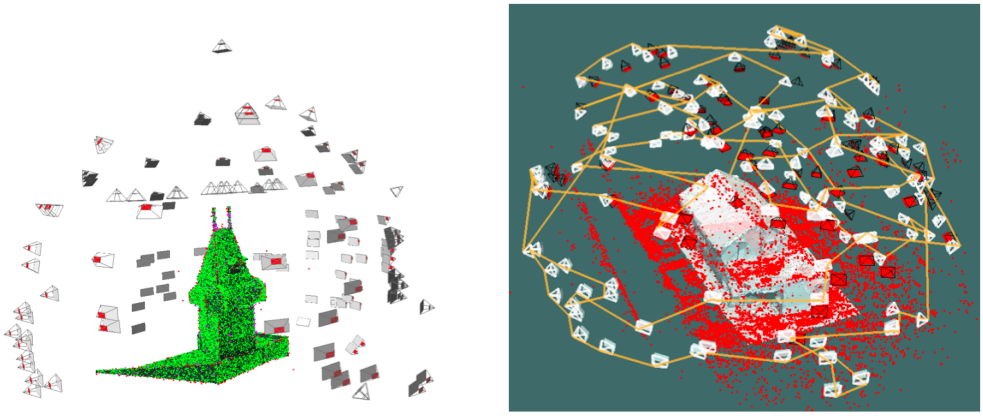
2.1.2 Active Reconstruction and Mapping

The task of producing an accurate and complete map of the environment using a mobile robot is usually decomposed to two parts: (1) finding observation locations of high value and (2) planning a trajectory for making some observations. For the first step, a measure of information is often defined that shows where the new measurements should be taken in the environment.

Bourgault *et al.* use the expected information gain (reduction in entropy) to decide where to explore next in the environment [36]. They use occupancy grid to model the environment in which each grid cell is assigned a probability of being occupied, initialized to 0.5. Using a range sensor, the robot can make observations at different locations and the expected posterior entropy is obtained by integrating the posterior entropy over possible measurements. Stachniss and Burgard combine the information gain and the distance to choose the next observation location [37]. The new metric which considers both the uncertainty and the travelled distance, is shown to produce accurate maps with short overall path length, yielding a good trade-off. Makarenko *et al.* propose an integrated utility measure that considers uncertainty reduction in map, uncertainty in the robot pose, and the navigation cost [38]. The candidate destination that has the highest utility value is then selected to visit. Sim and Roy argue that approaches based on variance minimization, which is equivalent to minimizing the product of the eigen values of the covariance matrix (called *D-optimality*), are not accurate for map building [39]. They propose a metric based on the trace of the covariance matrix (mean of the eigen values, called *A-optimality*). Moreover, in contrast to the previous methods that select a single location based on a local, greedy approach, they use breadth-first-search to find trajectories that yield the maximum information gained along the path. The space of robot positions is discretized to a grid and the search tree is constantly pruned to make the problem tractable.

In visual 3D reconstruction, the problem of *next-best-view* (NBV) deals with finding the next viewpoint for the camera to achieve high quality reconstruction of the scene. Wenhardt *et al.* consider this problem in a probabilistic framework and various information gain metrics from the literature are experimented and compared to solve the NBV problem [40]. Additionally, they integrate the visibility probability of each feature point into their criteria to account for non-visible landmarks. However, they do not handle self occlusion in their problem formulation. Dunn and Frahm propose a new metric for NBV that aggregates the uncertainty direction vector of the reconstructed points and the resolution and saliency of the visible surface patch [41]. Hence, they consider the geometric uncertainty reduction and the ability to match features at the same time.

Hoppe *et al.* consider the problem of automated image acquisition using MAVs to fully reconstruct an *a priori* available 3D scene [42]. A viewpoint quality measure is defined that considers the uncertainty of the reconstructed points based on triangulation angles. They use clustering to remove redundant viewpoints and plan an efficient tour to visit each sampling location (see FIgure 2.2). In another work [43], they propose to improve the manual image acquisition done by a human by performing sparse SFM to obtain the scene mesh and verifying if the model is complete. Through the feedback, the user can know if more images are required to satisfy the quality of the reconstruction. They show that, in comparison to the batch-based SFM, their method can achieve complete reconstruction using fewer number of images.



(a) Final set of camera viewpoints. (b) Flight path of the UAV.

Figure 2.2: Photogrammetric camera network design. Figure reproduced from [42].

2.1.3 Localization-aware Path Planning

Navigating to a goal location is a simple task for robots provided that the location of the robot at any time is known (with some tolerable uncertainty). However, if the quality of localization depends on the location of the robot in the environment, performing the navigation task and keeping the uncertainty in robot pose within some bound at the same time will be challenging. Generally speaking, in the localization process the state of a robot is estimated by measuring the location of some observable landmarks (feature points). If the robot fails to make enough observations, the localization fails. Therefore, if the environment is already explored and the locations of the landmarks are known, one can predict the quality of pose estimation anywhere in the environment and navigate accordingly. For example, Moon *et al.* plan safe paths such that enough landmarks are visible [44] to avoid growth in position uncertainty. They used stereo vision to measure the relative position of *a priori* known landmarks which enabled them to localize the robot in the known map. Fonatelli *et al.* use a topological mapping framework based on pure visual appearance and propose an image-based optimal planner that generates manoeuvres that keep the visual features in the field of view of the camera [45]. The map data is assumed to be available from previous explorations.

Davison and Murray used an active stereo camera on a mobile platform to estimate the position of the robot [47]. The trajectory of the vehicle was preplanned. However, the localization was performed online by observing and tracking features. In order to cope with the position uncertainty, they actively tracked features as the robot was moving, and refined the position estimation. Moreover, some observations were made from viewpoints that were perpendicular to the axis of the highest uncertainty in the feature location. Bianco *et al.* produce a potential field based on the landmark observability and used it to navigate around the environment [48]. The displacement and size of the landmarks, compared to when viewed at the goal place, is used to produce the the artificial forces in the field. Only distinct landmarks are used and unreliable ones are discarded. Davison *et al.* actively predict where to search for a specific feature in the

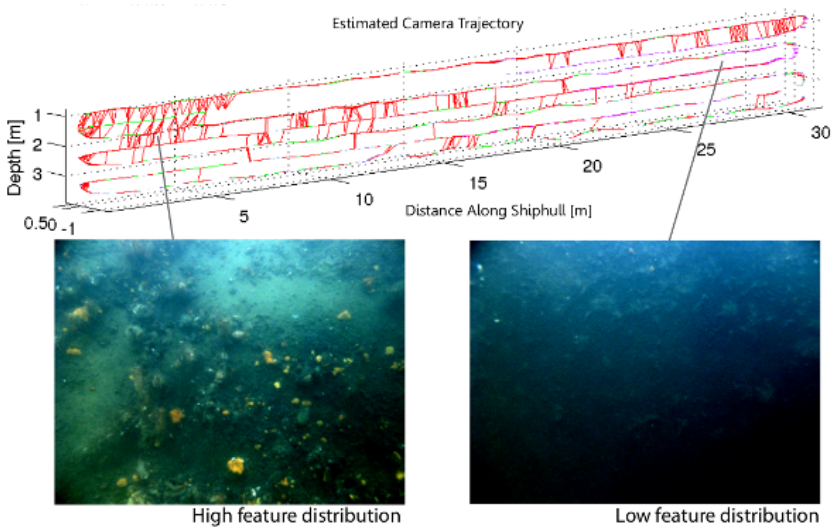


Figure 2.3: Estimated trajectory of an underwater robot mapping the below- water surface of a ship hull. (top) Trajectory of the robot with successful cross-track camera registrations depicted as red lines. (bottom) Representative images indicative of the image feature content within that area. Note that the density of pose-graph links is spatially correlated with feature content [46].

image to avoid computationally expensive operations in the matching phase on the *MonoSLAM* [28]. Additionally, since only a few (10-20) feature measurements could be afforded in their system, only the most informative features were observed. Frintrop and Jensfelt introduce an approach to active camera control for visual SLAM [49]. The focus of their work is on three behaviours: (i) a tracking behaviour which tracks useful features, preventing them from leaving the field of view, (ii) a re-detection behaviour, which is intended to trigger loop closing by directing the camera to regions where previously mapped features are expected, and (iii) an exploration behaviour that is activated when enough landmarks are in the field of view. It is shown that active gaze control, compared to passive camera, can yield informative features with a wide baseline, frequent loop closing, and a more uniform distribution of mapped points in the environment.

In a ship hull inspection application, Hover *et al.* describe an *autonomous underwater vehicle* (AUV) that incorporates odometry, sonar, and camera constraints into a pose-graph formulation of SLAM in order to estimate its trajectory [51]. The camera constraints are created by matching features between two overlapping images of the hull and performing a two-frame bundle-adjustment. Kim and Eustice propose a measure of local saliency based on bag-of-words representation of the image to predict successful camera registrations [46]. Moreover, a global saliency metric is defined to measure the likelihood of loop closure at previously observed locations. They use these measures in an integrated perception-driven navigation where the AUV balances between inspection of the ship hull and revisiting globally salient locations for loop closure [52]. When revisiting a location, a visually salient trajectory is planned using the local saliency metric to prevent the uncertainty from further growth. Although lack of visual features

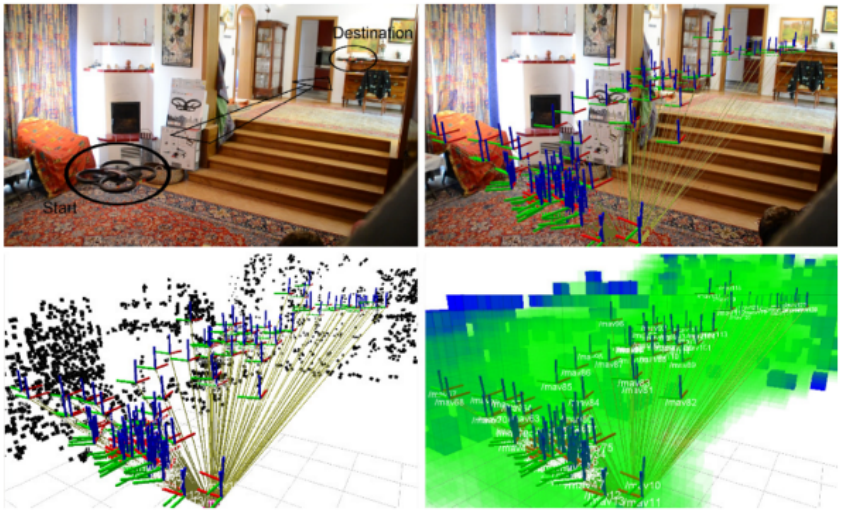
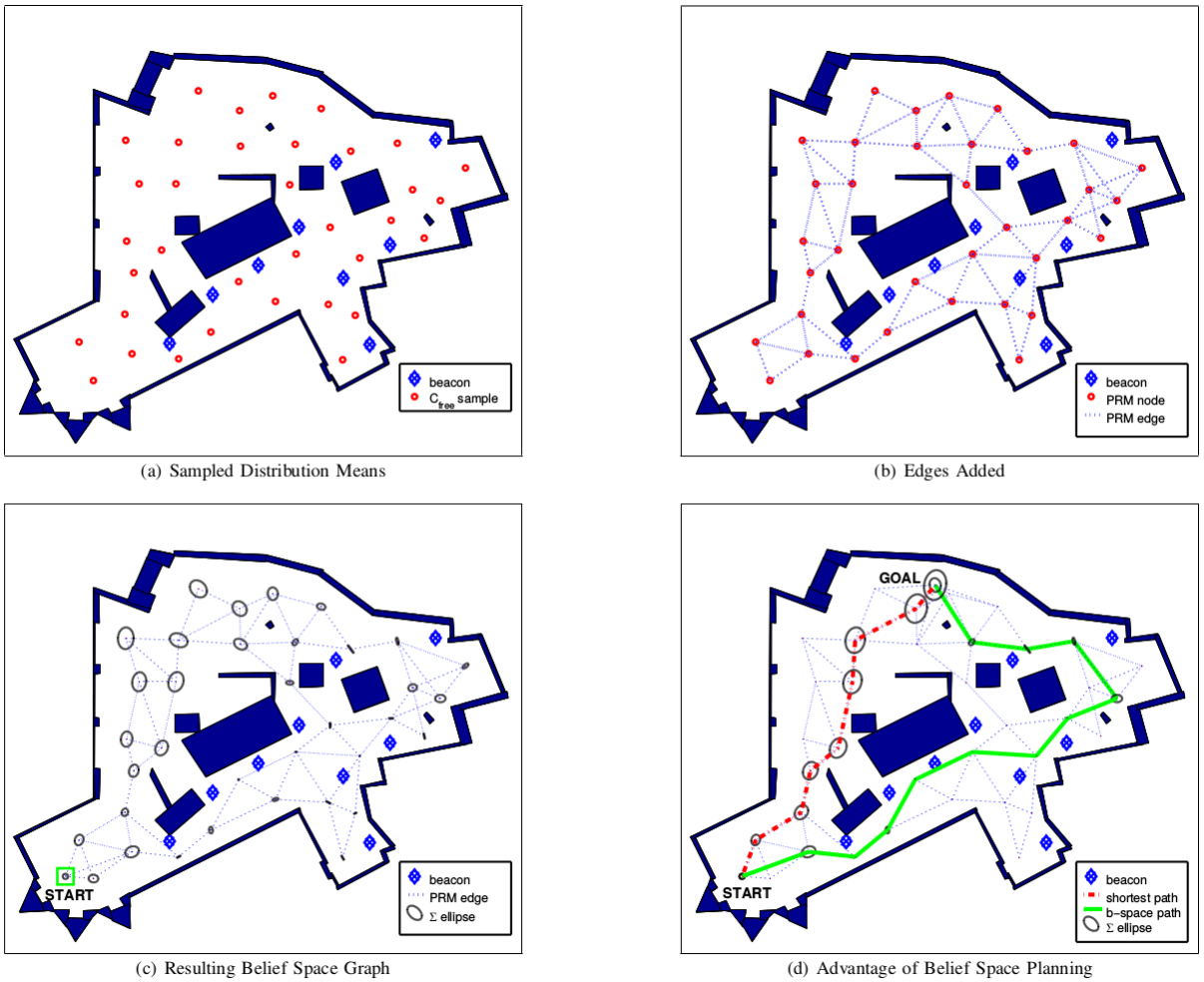


Figure 2.4: Active monocular localization method for exploration [50].

leads to increasing uncertainty, due to the availability of other other sensors, short-term failure of feature matching does not break the localization in their AUV.

In aerial robotics, Mostegel *et al.* proposed an active monocular localization method for exploration [50]. Similar to us, they have focused on visual SLAM based on bundle adjustment. They define a localization quality metric for a candidate camera view point based on the quality of the map point and the viewing angle. To predict the quality of potential map points that can be extracted by triangulating new tracked 2D features, they use the distribution of the depth values of visible map points. Local movements are used to decrease the distance between current location and the goal (see Figure 2.4). Although they use similar measures to ours, in order to evaluate the pose estimation quality of the SLAM, their approach does not reason about visibility of a map point in candidate view point. Moreover, the use of local planning, though very computationally cheap, may results in local minima. Conversely, our proposed approach integrates the localization system needs into a global planning approach that avoids local minima.

Localization-aware path planning can also be viewed as a problem in *Planning in Information Space*, where Partially Observable Markov Decision Processes (POMDPs) or graph-based searches are used to find a solution in the belief space [54]. However, these frameworks suffer from exponential growth in complexity when the number of possible actions and observations increases. Roy *et al.* made some simplifying assumptions and proposed *Coastal Navigation* to generate trajectories that reduce the average uncertainty in the position of the robot as it navigates to a goal in a known environment [55]. Using Markov Localization and a 360° laser scanner, they compute the expected information gain at each discrete cell according to the prior position and the sensor model. A trajectory is then planned based on this information map such that a combined cost of travelling and uncertainty is minimized. In order to keep the problem tractable, they only simulate the most likely sensor measurements and track the position of the robot using odometry which results in a prior Gaussian distribution. Prentice and Roy intro-



(a) Sampled Distribution Means

(b) Edges Added

(c) Resulting Belief Space Graph

(d) Advantage of Belief Space Planning

Figure 2.5: Belief Roadmap. (a) The free space is sample and (b) the edges are added to the graph when there is no collision. (c) The resulting uncertainty a each node is calculated and (d) a path is found that minimized the length and the pose uncertainty at the same time [53].

duced Belief Roadmap (BRM) which is shown to generate safe paths substantially faster than conventional methods of belief space planning [55]. Using the *Extended Kalman Filter* (EKF) for belief estimation, they obtained a covariance transfer function that can predict the posterior covariance of the state from one node of the graph to the other. Hence, the optimal path in the road map can be found efficiently (see Figure 2.5). They also showed the feasibility of their method using experiments with laser-equipped MAVs in an indoor environment [56]. Bry and Roy extended these ideas and proposed Rapidly-exploring Random Belief Trees (RRBT) to consider the dynamic constraints of the vehicle in belief space planning [57]. Achtelik *et al.* used RRBT with a visual-inertial state estimation method for MAVs [58]. They assume a prior map of the environment is available consisting of the mapped feature points. The pose estimation output of the visual SLAM is considered as an observation in the EKF and the state covariance is computed by minimizing the re-projection errors of the 3D map points across all images. However, the entire planning is performed offline and is computationally expensive. In order to increase the efficiency of the above stochastic planners, adaptive sampling strategies have been

proposed that prevent dense sampling of regions with poor sensor information [59]. Similarly, Bryson and Sukkarieh use the EKF to estimate the state of the UAV, and, by multiple applications of the prediction step of EKF, the expected pose uncertainty is estimated at a landmark location [60]. The information gain of observing the landmark is used as the utility of visiting it. However, for exploration, they rely on the assumption that the density of the landmarks on the ground is sufficiently high. Their simulation experiments show that the uncertainty in the UAV pose can be maintained small while exploring the area.

Conclusion

The problem of active localization has been considered in different settings. Some of the previous work assumes that a map of the environment is *a priori* available [44, 45, 55, 56, 57, 58]. Such a map might contain dense information about the structure of the environment [55, 56, 57]. Conversely, the map may be in form of a sparse collection of feature points (landmarks) with known positions [44, 45]. Moreover, in case of unknown map, there might be assumptions on the distribution of landmarks in the area [60]. These methods cannot be applied to situations were there is no prior information about the environment. Additionally, in contrast to methods that restrict the camera position to a fixed trajectory (as in [47]), or a 2D manifold in the 3D space (e.g. [61]), the full control of the camera is highly favourable in order to exploit the available information in the environment.

The camera has been used as complementary to other sensors such as laser and sonar in pose estimation [52]. In these systems, the localization can tolerate the lack of visual features for a limited period of time without any failure. Similarly, in some cases it is assumed that an odometry sensor exists that can be used to predict the effect of motion commands [62]. Specifically, EKF-based methods use the dynamics and control efforts to propagate the pose uncertainty in a trajectory [60, 56, 57]. However, we are interested in localization-aware planning methods that only rely on the visual information obtained by a camera without any extra sensor or assumptions on the dynamics of the vehicle.

Finally, for navigation tasks, the visual information should be integrated with a global planner to avoid local minimums. Although, the effect of local camera motions on the performance of the mapping and localization is studied in the literature (e.g. [50]), these methods are not applicable to navigation tasks in a relatively complex environment.

2.2 Aerial Coverage

Coverage path planning (CPP) is the problem of finding a trajectory for a mobile robot such that a target area is completely swept by the sensor footprint. CPP is different from the sensor deployment or *static coverage* problem in which the goal is to place a number of sensors in the environment to satisfy a sensing requirement such as complete coverage or sensor overlap minimization [63, 64]. In the following, we first present an overview of the conventional CPP

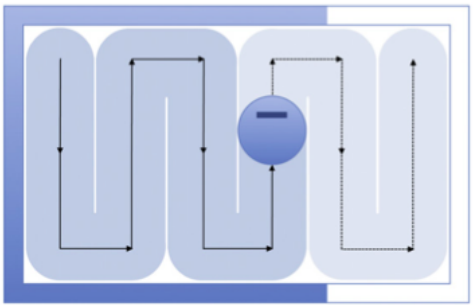


Figure 2.6: Lawnmower pattern. Figure reproduced from [65].

methods for mobile robots. Later, we summarize the previous work on aerial coverage path planning.

2.2.1 Coverage Path Planning for Mobile Robots

Finding an optimal coverage path is NP-hard for even a simple polygon [66]. Therefore existing approaches try to find an efficient approximate solution. For 2D coverage, some methods decompose the target region into simpler polygons which are simpler to cover. Other methods use a grid-based representation which leads to an approximate coverage. Also, some methods have been proposed to improve the efficiency of 3D coverage. Finally, closed-loop control methods avoid explicit representation of the target area. We provide an overview of the main approaches to coverage path planning. There are many extensions and modifications in the literature to improve the performance of these methods. However, we do not include them here. For more detailed surveys see [67, 65].

Exact Cellular Decomposition

One of the main approaches in area coverage path planning is based on the divide-and-conquer strategy [68]. In these methods the environment, which is often represented by a polygon, is decomposed into a number of simple subregions or *cells*. Each cell then can be easily covered using simple back and forth motion as illustrated in Figure 2.6. This type of motion is called the *Lawnmower pattern*. To achieve a complete coverage of the environment, all the cells are visited and covered accordingly. In order to minimize the cost of traveling between cells, a graph is constructed and an appropriate tour is computed to visit all cells. Furthermore, as we will see later, in some multi-robot scenarios cellular decomposition is used to break the coverage task into a number of smaller tasks and each robot is assigned to a separate cell.

Trapezoidal decomposition divides a target area into trapezoids [69]. This is achieved simply by sweeping a vertical line through the 2D plane. Upon reaching each vertex of the environment polygon, the required edges are added to create trapezoids. Note that it is possible that some triangles also emerge during this process. Two cells sharing a line segment are defined to be adjacent and accordingly an adjacency graph is produced. Then an exhaustive walk is found that visits all the nodes in the graph to achieve complete coverage. A practical application of

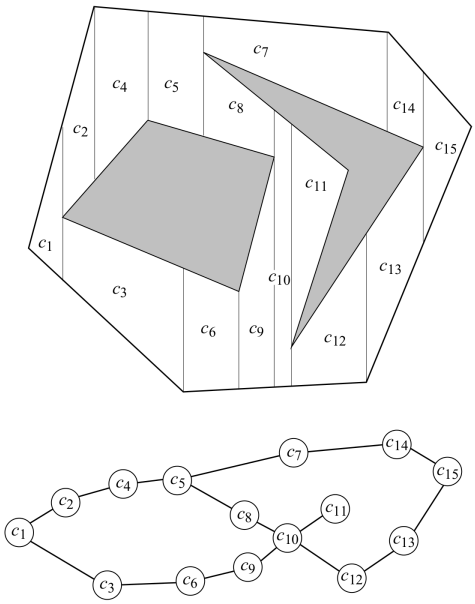


Figure 2.7: Trapezoidal decomposition. Figure reproduced from [68].

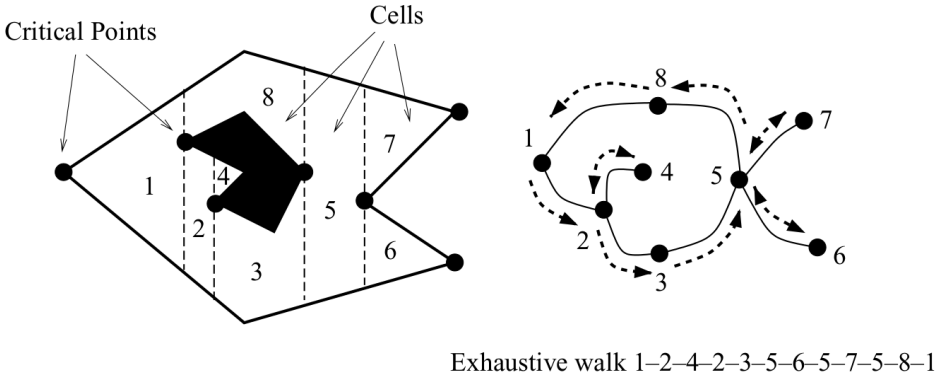


Figure 2.8: Boustrophedon decomposition. Figure reproduced from [68].

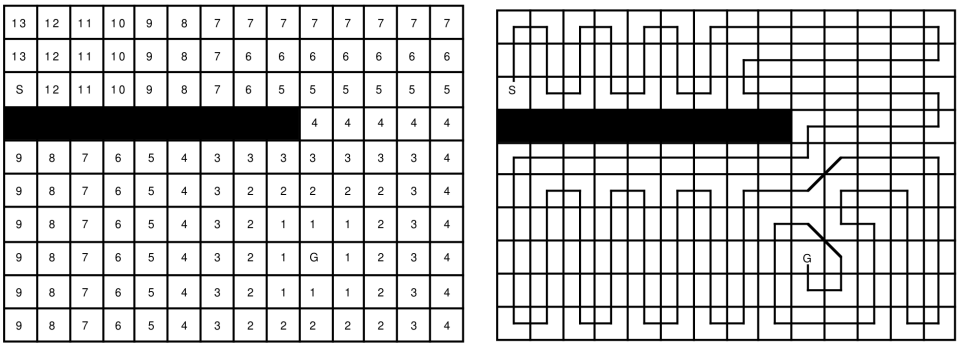
this decomposition method can be found in the work of Oksanen *et al.* ([70]) where coverage paths are generated for agricultural field machines. However, using some heuristic methods they propose to merge some neighbour trapezoids to form fewer large yet simple cells. Also, a local search is performed to find a direction for the sweeping line when decomposing the area so that a cost function, which includes trajectory length and the number of turnings, is minimized.

In trapezoidal approach many neighbour cells could be merged together into one cell resulting in a shorter coverage path. In [71], Choset *et al.* proposed the *Boustrophedon* method to achieve a better decomposition. As in the trapezoidal method, a vertical line is swept through the configuration space. However it only stops at vertices where the line can be extended both up and down (see Figure 2.8). These vertices are called *critical points*. As before, by visiting all the nodes in the adjacency graph the area is covered completely. Zhou *et al.* recently used this approach in agricultural operations planning [72].

In summary, offline exact cellular decomposition guarantees a complete coverage of the environment. Although each cell can be covered using a simple Lawnmower pattern, an efficient tour in the adjacency graph must be found to complete the coverage trajectory. Complete coverage is also guaranteed in the online version of this method at the expense of sensor overlaps and revisiting regions.

Grid-based Methods

Many coverage path planning methods represent the environment using a collection of grid cells of the same shape. Each cell has a binary or probability value stating if it is occupied or empty. This is an approximate cellular decomposition as the grid cells approximate the shape of the obstacles. The coverage algorithms that use this representation make sure that all the grid cells are visited and the travelling cost is minimized. An indirect way to minimize the coverage path length is to (*a*) avoid revisiting a grid cell as much as possible and (*b*) move from one cell to a neighbour cell only. Proposed by Zelinsky *et al.*, the *Wave-front* method forms a discrete



(a) The labels of each cell in the Wave-front method.

(b) Final coverage path in the Wave-front method.

Figure 2.9: Wave-front method. Figure reproduced from [73].

potential field using a start and goal cell. It first assigns 0 to the goal cell [73]. Then all the unmarked cells neighbouring cell 0 are marked 1. This process is repeated so that all the cells are assigned a value (Figure 2.9a). Then the robot at the start location (or any cell) visits the neighbour cell with highest label value (randomly picking one if multiple options exist). Consequently the robot visits cells with equal potentials and then moves to cells with lower potentials and finally ends up in the goal cell. An example coverage path produced by this method is shown in Figure 2.9b. A nice feature of this method is that the trajectory ends at a specific goal location. However, in some situations this method can not avoid revisiting cells. This happens when the robot visits a cell whose neighbours are already visited.

The *Minimum-Spanning-Tree* (MST) method only considers the cells that are completely unoccupied by obstacles [74]. First, a grid with cells 4 times the robot sensor footprint is constructed. Then a graph is created by representing each cell with a node and connecting two nodes if they are neighbour cells. The minimum spanning tree of this graph then is computed. In order to achieve complete coverage of the environment the robot circumnavigates the spanning

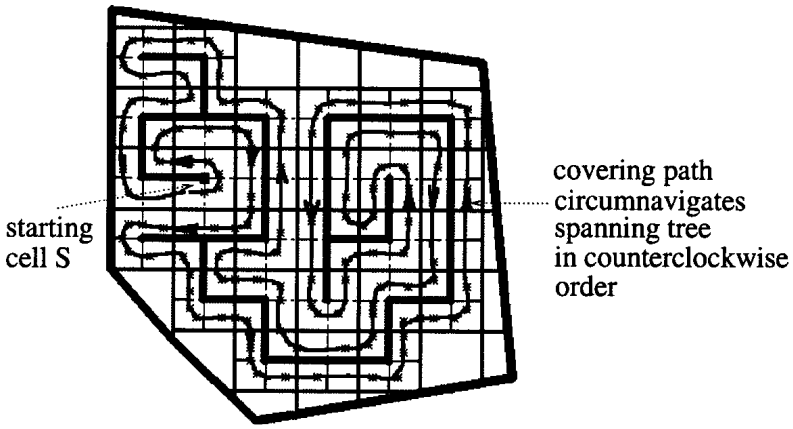


Figure 2.10: Spanning tree method. Figure reproduced from [74].

tree visiting quadrants of the cells as shown in Figure 2.10. Unlike the wave-front method, this algorithm never revisits a cell and hence produces an optimal solution in terms of path length.

3D Coverage

The target surface of a coverage task may not always be in form of a simple plane as is assumed in most of the CPP methods. For example in [75], Galceran *et al.* consider using *Autonomous Underwater Vehicles (AUVs)* to perform a survey of seabed floor that contains high-relief terrain. As the regular Lawnmower pattern requires extensive variation in the depth of the vehicle and hence is expensive, they propose to decompose the target surface into separate regions at approximately same height. Each region is then effectively treated as a planar area and covered with a typical coverage method.

However, in some situations the complexity of the target surface does not allow such simple decompositions. Hover *et al.* studied such a coverage task in the context of ship hull inspection using AUVs [61]. Their proposed approach samples the free space and incrementally creates a roadmap by which the low-resolution ship hull mesh can be covered. Then an approximate solution of the *Travelling Salesman Problem (TSP)* is found to traverse the roadmap completely which results in the coverage of the 3D structure.

Closed-loop Control Methods

Many closed-loop control strategies have been proposed to achieve a coverage task with a single or a team of mobile robots. These methods usually formulate the coverage task as an optimization problem in which the goal is to minimize (e.g., sensor overlap) or maximize (e.g., sensing of uncovered area) a metric. This formulation is used in both static and dynamic coverage problems and techniques such potential fields, and gradient descent are used to tackle it. For example Howard *et al.* in [76] developed a method in which a continuous potential field is used to accomplish a coverage task using a team of mobile sensors. Sensors are repelled by obstacles and other team members and are consequently spread out in the area. This is an example of

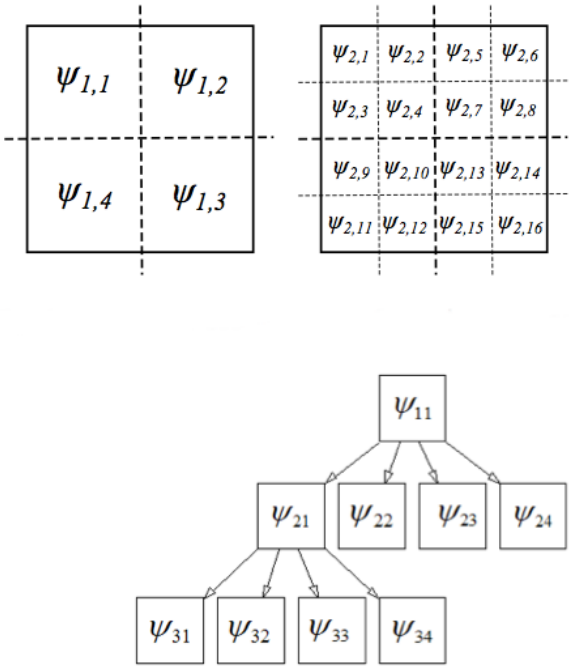


Figure 2.11: Hierarchical decomposition of the workspace. Figure reproduced from [77].

static coverage in which the problem is how to redeploy the robots in the environment to satisfy a coverage criteria.

In other other similar approaches, the control laws direct the robots in a direction with maximum coverage benefit and are guaranteed to converge to a complete area coverage [78]. As opposed to task-based divide-and-conquer approaches, these methods suffer from being local and greedy and fail to generate efficient global coverage trajectories. Franco *et al.* tried to alleviate this problem by continuously weighting the global and local strategies [77]. Therefore the control laws are composed of two components: i) a local component that is dependent on the coverage status of a local neighbourhood, designed to direct the robot towards regions with high local coverage benefit and ii) a global component that depends on the global coverage status and directs the robot towards a point from where the robot can cover uncovered regions. To achieve this, they hierarchically decomposed the workspace into 2×2 squares (Figure 2.11). The local control component is induced by the current cell at the leaf level. The global component is defined by a position P_{obj} in an uncovered region of the environment. At the beginning, P_{obj} is the center of the current cell. When the current cell is completely covered, the parent cell is checked. If it is not fully covered, one of its children is selected as P_{obj}. This process is recursively continued in case the parent cell is completely covered. This global order of regions enforced in computing the coverage control commands avoids unnecessary travelling of the robots to far apart regions in order to satisfy the global objective and thus leads to energy savings.

The methods above do not consider particularities of UAVs and aerial coverage and are designed for general mobile robots. In the next chapter we show how these methods are (modified and) used for aerial coverage.

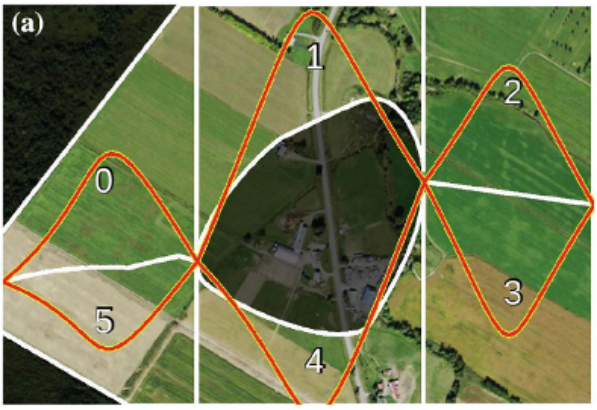


Figure 2.12: Duplicate edges of the Reeb graph. Figure reproduced from [79].

2.2.2 Aerial Coverage Using UAVs

The general CPP methods introduced in Chapter 2.2.1 have been applied directly to aerial coverage often in an offline setting. In these works, the target area is specified as a polygon or grid with GPS coordinates of the polygon nodes or grid center. It is usually assumed that the UAV is flying at an altitude which is safe and hence there is no risk of colliding with an obstacle. However sometimes some parts of the environment are treated as no-flying zones which should be avoided by the UAV. Therefore in CPP for UAVs, these subregions are dealt with as obstacles. Another type of region that might be present in aerial coverage is uninteresting regions. Flying in these regions is allowed but does not contribute to the coverage goal i.e. covering them is of no value. For example in a outdoor crop mapping task with a quadrotor, covering the lake is unnecessary but flying over it (for example to reach the other side of the lake) is allowed.

As mentioned before, due to the very short flight time of UAVs, generating efficient coverage paths is useful. To reach efficiency, existing methods try to avoid unnecessary coverage, i.e. they try to minimize the overlap in the sensor footprint along the produced trajectory. This goal will consequently reduce the path length. Another element considered by some existing methods is to minimize the number of turns in the flight trajectory. Reducing the number of turnings will consequently produce trajectories that consist of long straight stripes, as is the case in Lawnmower pattern. Therefore, when a robot follows the trajectory it can maintain a constant velocity in a large part of the coverage path and it only accelerates or de-accelerates when it is turning. The overall outcome is less power consumption. Such efficient coverage trajectories can be planned offline when the environment is known *a priori*. Some of these methods, aimed for aerial coverage, are presented in the following section which includes planning for both single and multi-robot systems. However, when no prior knowledge about the target area is available, the planning has to be done online based on the real-time sensory data.

In the following, we present the existing methods of coverage trajectories planning for unmanned aerial vehicles. We partition the approaches into two main groups: a) methods that precompute the trajectory based on a priori knowledge of the environment and b) methods that adaptively re-plan the trajectory based on on-line sensory data.

Offline Planning Methods

Some of the coverage methods rely on the prior knowledge of the environment and generate a coverage trajectory offline. Then the UAV executes that plan. However, the trajectory is not changed during the mission. Various methods of this type are presented in the following sections.

Aerial coverage using cellular decomposition When the target area is known in advance, cellular decomposition can be used to plan coverage trajectories for UAVs in the same way as was introduced in the previous chapter. For instance, Xu *et al.* proposed a method for efficient coverage of a known environment using a fixed-wing UAV in [79]. Boustrophedon cellular decomposition is first applied on the area polygon to partition it into simple sub-regions. Then, a Reeb graph is constructed in which nodes represent the critical points and edges represent cells (see Section 2.2.1). They determine the best tour through the cells so that all the cells (edges) are visited only once. They achieve this by finding an Eulerian circuit in the corresponding Reeb graph. However, the essential and necessary condition for the existence of an Eulerian circuit in a graph is that all the nodes have to be of even degree. Therefore a subset of edges must be duplicated in the Reeb graph. This subset is found through an integer programming to minimize the number of duplications and the added cost. When performing the actual cell coverage using a Lawnmower pattern, in order to avoid repeated coverage of a cell, a duplicated cell is partitioned into two sub-cells that each corresponds to a single edge (see Figure 2.12). They performed simulation and real UAV experiments to show the efficiency of their method and examined the effect of parameters such as sweeping direction and wind direction on the coverage quality. A similar Boustrophedon approach is used by Moon *et al.* in [80] where they use small helicopters for crop-dusting.

Cellular decomposition has also been used as a way of allocating regions to each robot in a multi-UAV coverage scenario [81]. Araujo *et al.* consider coverage planning for multiple UAVs with kinematic constraints [82]. They first find the a sweeping direction that leads to minimum number of turns (Figure 2.13). This optimal sweep direction is aligned with the direction of two parallel lines with minimum distance bounding the area polygon. Then, assuming a given relative capability for each UAV, the polygon is partitioned with a line along the sweep direction so that the relative area of each cell corresponds to the relative capability of the assigned UAV. Then Lawnmower-like lanes are constructed to cover each sub-area. In order to perform persistent surveillance, a method to choose the next Lawnmower lane is proposed. The prioritization is based on factors such as kinematic constraints, uncertainty of the lane and distance to the nearest endpoint of the lane. Likewise, Mehdizade studied the problem of area decomposition and allocation for a heterogeneous team of UAVs [83]. He used multi-objective optimization techniques to generate solutions that minimize both fuel consumption and mission time.

Aerial coverage on grids A grid-based representation of the environment has also been used in aerial CPP. Since cameras are a very popular sensor for data collection and the visible region in a camera image is a rectangle/square (assuming a downwards facing camera and after

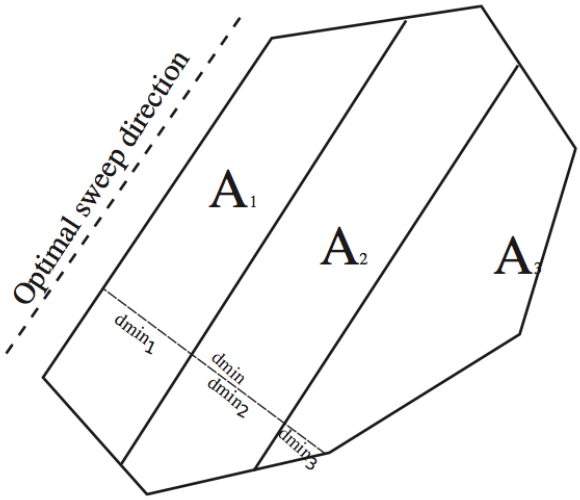
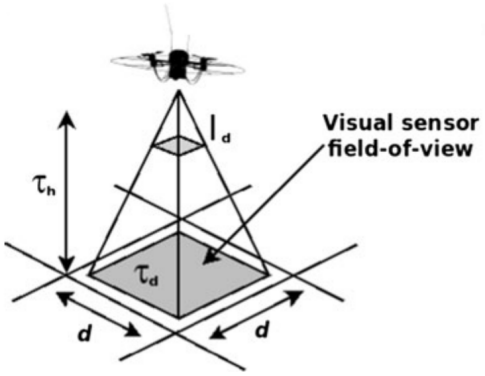


Figure 2.13: Optimal sweeping direction. Figure reproduced from [82].

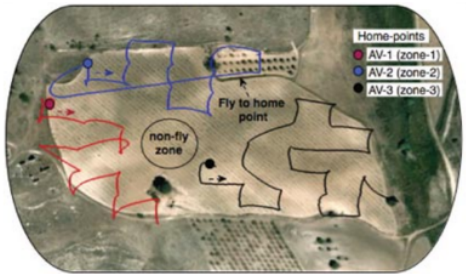
rectifying and cropping the image), a grid-based representation with square building blocks is preferred as it makes the problem simpler. Such a representation is used by Valente *et al.* in [84, 85]. They present a mission planner for coverage path planning using mini quad-rotors. Using on-board cameras, a set of images are collected that contains the whole area and is then used to create a large image of the farm using image stitching process. They assume that the environment is known in advance, i.e. the boundary of the farm is given in form of a polygon. The input also includes sub-polygons which indicates no-flying zones. The output of the planner is a set of GPS waypoints that can be visited by the UAVs. These waypoints are basically the positions at which a cell of the uniform grid can be covered completely by the camera. Although the X-Y coordinates of the waypoints correspond to the specific grid cell, the height of it, τ_h, can be chosen from a range of possible options. This height is maintained during the whole mission to ensure a uniform sampling of the whole field and is calculated according to the resolution requirements by the following equation:

$$\frac{\tau_d}{\tau_h} = \frac{I_d}{l}$$

in which τ_d, I_d and l are respectively the target grid cell width, the image width and the focal length of the camera (see Figure 2.14a). This height is modulated later to produce the desired overlap in the images so that image registration becomes possible in the stitching process. The coverage path planning is done on the adjacency graph induced by the grid cells. To make sure the UAV starts from and ends in specific locations, they use the distance transform similar to the Wave-front approach (2.2.1). Then, starting from the take-off cell, a depth-limited search is employed to find all possible coverage paths while following a gradient descent on the grid. Among them a path with minimum number of turns and no repeated visitation of a cell is chosen as the final solution. They performed real experiments with a quadrotor to demonstrate the feasibility of their proposed method. This work later was extended to multi-UAV coverage in which the target area was partitioned and allocated to each individual UAV through negotiation

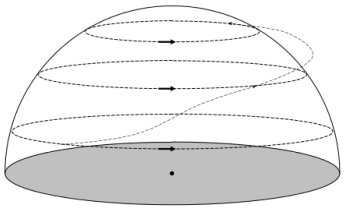


(a) UAV altitude and the imaging requirements. Figure reproduced from [84].

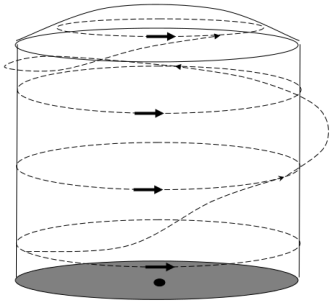


(b) Multi-UAV coverage using a grid-based approach. Figure reproduced from [86].

Figure 2.14: Aerial coverage in an agricultural application



(a) Covering a hemisphere.



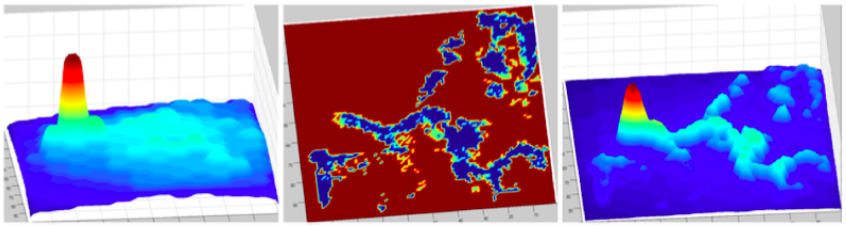
(b) Covering a cylinder.

Figure 2.15: Covering simple models in optimal time. Figure reproduced from [87].

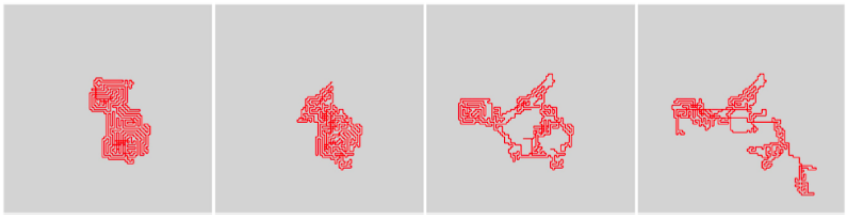
and then each UAV used the described planner to find a coverage path for its sub-area [86] (see Figure 2.14b).

Aerial coverage of more complex structures Some of the existing approaches to aerial CPP consider more complex target surfaces. Cheng *et al.* in [87], proposed a method of time-optimal UAV trajectory planning for 3D urban structure coverage. However, they used abstract models such as hemispheres and cylinders to simplify the 3D urban structure. The proposed trajectory pattern consists of horizontal circles at different altitudes on the corresponding abstract models. The UAV flies on these circles maintaining a constant altitude. After traversing the complete circular trajectory, the UAV will optimally fly to the next circle in a higher altitude (see Figure 2.15a, 2.15b). They prove that, using this trajectory pattern, the duration of the coverage mission is within a constant factor of the lower bound of the duration of the optimal coverage trajectory.

Aerial coverage as a search strategy Sensing all the points in an area has also been used as a method of searching for a static target. However, in most of these scenarios a likelihood map



(a) From left to right: a priori probability distribution of target location, sensor detection probability map and target detection probability distribution.

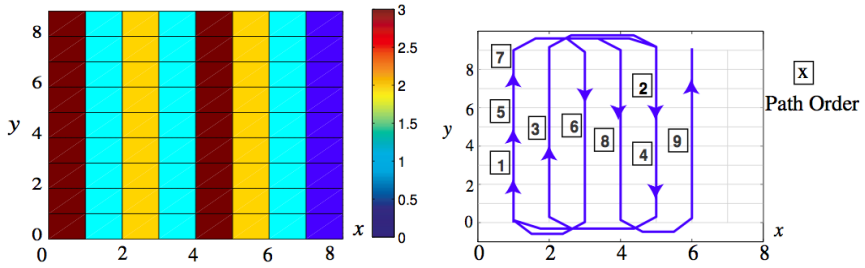


(b) The two left trajectories are produced using greedy methods and the other two on the right are produced with the proposed clustering based method.

Figure 2.16: Searching an area by coverage path planning. Figure reproduced from [88].

is given *a priori* that indicate the probability distribution of the target location. By exploiting these priors, often the coverage trajectory is planned so that the most likely places have some kind of priority. Note that here we present the methods that assume the likelihood map does not change during the search and therefore the path is generated offline. For example, in [88] Lin *et al.* propose Hierarchical Heuristic Search which is a CPP method to find a target of interest in the environment using an aerial vehicle. Given a prior probability map indicating where the target might be and another map which represents the probability of sensor detection, a method for planning a path with high cumulative probability of detection in a given finite time is proposed. They first obtain the posterior probability of detection by bayesian filters based on the two known probability maps. Then using a mixture of Gaussians model, the environment is clustered into a number of regions with high probability of finding the target. The distance to the center of the Gaussian distributions, the volume, and the area of the distribution surface within three standard deviations are used to prioritize the regions. To generate the actual trajectories, a spiral pattern is used. They demonstrate that generating trajectories of limited length using this prioritization of partitions with high probability of detection is more efficient than other greedy algorithms with one or more lookahead step.

Similarly, Ousingsawat *et al.* considered searching an area by planning a Lawnmower coverage path over a grid [89]. In their problem each grid cell has a prior importance value (Figure 2.17a) which is interpreted as the uncertainty of the cell and is decreased by 1 when observed by the UAV. They propose a method to find the number of times and order of visiting each Lawnmower track to maximize the coverage (reduce the total uncertainty to ≈ 0) and, at the same time, minimize the mission time (Figure 2.17b). Using a sensor with FOV of 2 cells, they compute the number of visitations each lane needs taking into account the overlap in sensor footprint



(a) Uncertainty map (b) Order of the Lane visitation

Figure 2.17: Modified Lawnmower pattern. Figure reproduced from [89].

when the UAV flies along neighbour lanes. Then they examine all the permutations of the lanes (with repeated lanes according to the required visitation number) to find the ordering with minimum cost. To manage the complexity, they prune low quality permutations using some heuristics. The cost of an ordering is computed according to the distance between consecutive lanes. Furthermore, sharp turns are penalized so that the selected solutions satisfy the turning constraints of the UAV. They show that their method can achieve the required coverage level more efficiently than an approach where a complete Lawnmower pattern is repeated a number of times.

Adaptive Trajectory Planning

When the target regions in a coverage task are not known in advance, one can not rely on offline trajectory generation. Although an exhaustive Lawnmower pattern can be used to sweep the whole area which will include uninteresting and interesting regions, more efficient trajectories can be obtained using real-time sensor data and adaptive re-planning. In the following online methods, though a probability map of the target regions might be available in advance, the trajectory is generated online based on updated belief about the location of target regions. The common procedure in these methods is that based on the current knowledge, a one or more step action is planned that has the most benefit in terms of the completing the task. After performing these actions and integrating the sensed data into the state of the task, new steps are planned and so on. This problem setting might be similar to a search task since in both problems the ROIs should be localized and covered. However, as we will show later, in most of the the search tasks, the path is planned so that the probability of detecting the ROIs is maximized, i.e. it visits regions of high target detection probability. This strategy might not generate an efficient path due to the high cost of switching between these regions. On the other hand in an online coverage task, having in mind that all the ROIs should be found and covered, the strategy is to systematically cover and then leave each region (and never come back) which will ultimately generate a more efficient trajectory.

POMDP-based approach Murtaza *et al.* proposed a priority-based coverage path planning algorithm in [90]. They consider a search and rescue scenario in which aerial vehicles are deployed to find all the victims as soon as possible. A partially observable Markov decision

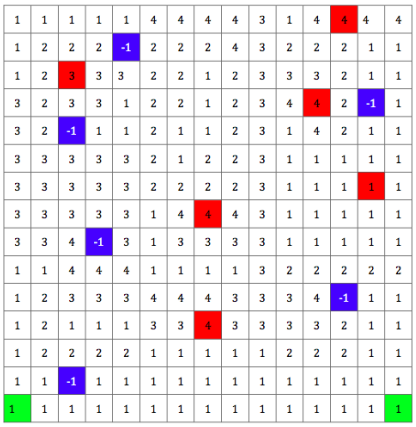


Figure 2.18: Grid-based representation of the area. The labels indicate the prior relative importance of the cells. The red, blue, and green colours respectively indicate the victims, obstacles, and starting locations of the UAVs. Figure reproduced from [90].

process (POMDP) is used to formulate the problem and the decisions of where to visit next are made by a POMDP-based solution. Using *a priori* knowledge of the possible locations of victims, a grid is initialized in which each cell is assigned a probability indicating the belief about that cell's containing a victim and the sum of the probabilities of all the non-obstacle cells is 1 (see Figure 2.18). The set of states of the POMDP are the total grid cells and actions are visiting each cell. Finding a cell with a victim has a reward of 1 and 0 otherwise. Each UAV broadcasts the result of sensing a cell so that all the robots have the current state of the world. The authors compare the proposed method to a greedy and a potential-based approach and show that in cases where the initial belief is close to the true locations of the victims, their approach finds them in a shorter time than the other two methods. It is easy to see that this method will not work as expected in case of no a priori knowledge.

Non-uniform search An interesting work is by Carpin *et al.* who proposed a method based on probabilistic quadtrees to search for a single or multiple targets in a grid [91, 92]. To the best of our knowledge, this is the only work in the literature that considers multi-resolution sensing by flying at different altitudes. In a quadtree, the root node represents the whole area (assumed to be in square shape) and each node has 4 children each of which cover a quadrant of the parent cell, with a higher resolution sensing (see Figure 4.2). At each level of the tree, each cell c of the grid is associated with a binary random variable X_c with $P(X_c = 1) = p_c$ which refers to probability that a target is located in the cell. The number of targets are not known in advance and consequently the cells are independent of each other. Accordingly, if the children of a node c_p in the quadtree are c_1, c_2, c_3, c_4 then the probability of the presence of a target in the parent cell is:

$$p_p = 1 - (1 - p_1)(1 - p_2)(1 - p_3)(1 - p_4)$$

They use a binary sensor model considering the false negative and false positive rates of the sensor:

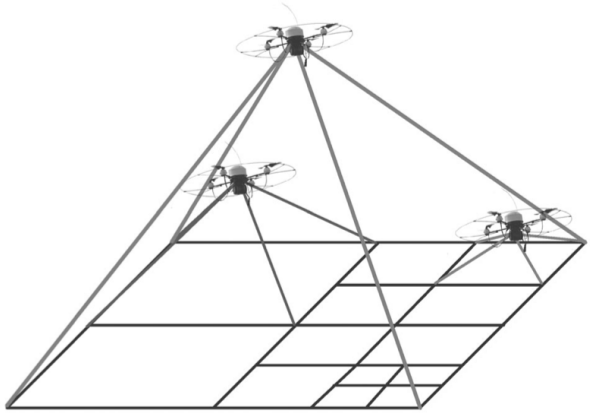


Figure 2.19: Quadtree-based search. Figure reproduced from [91].

$$P(Z_n = 1|X_n = 0) = \alpha(d(n))$$
$$P(Z_n = 0|X_n = 1) = \beta(d(n))$$

where $d(n)$ is the altitude of the node n. Upon sensing at a quadtree node c, the new p_c is calculated based on recursive bayesian update rule. Then the probabilities associated with the descendants and ancestors are updated.

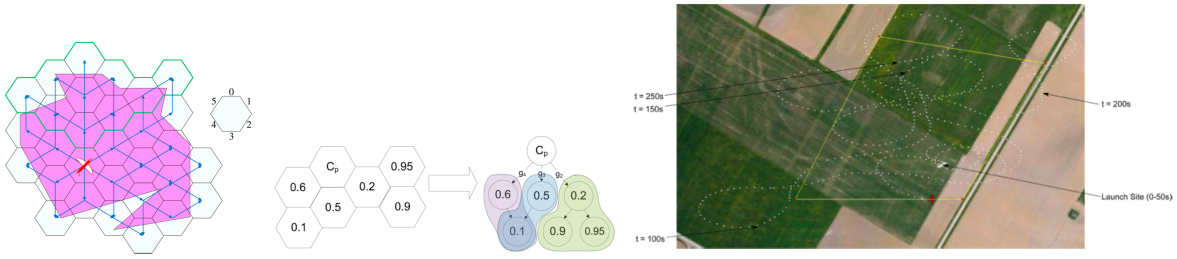
Initially, the quadtree is created based using the *a priori* information. To reduce subsequent computation time, the tree is pruned initially as follows: all the internal nodes are sorted based on the difference between the minimum and maximum probability of its children, called *disparity*. Then starting from the nodes with smallest disparity, the child nodes are pruned until the desired number of nodes remain in the tree.

Having developed a way of probabilistically modelling the interesting area, they use a combination of information gain and the distance to a node to choose the next node to visit. Here information gain is defined as the expected reduction in entropy of the leaf nodes (grid cells with highest resolution) when sensing occurs at a node n:

$$I(n) = H(T) - E_{Z_n}[H(T|n)]$$

The next node to visit can be located at any level of the tree. They compare their approach to a method that searches a uniform grid with the Lawnmower pattern. They show that in a given limited time period, their probabilistic method can find all the targets in the region while in 10% of the trials the Lawnmower pattern failed to cover the targets in the given time. However they do not discuss the case when no prior information is available, i.e. $p_c = 0.5$ for all leaf nodes c, which is the fair configuration when comparing this method to Lawnmower pattern. Instead they report the results of applying the same criteria for choosing a cell in a single resolution grid (no tree) and show that their method performs better.

Branch entropy The interesting regions do not always correspond to a real target on the surface. For instance, when performing an online mapping, regions of high uncertainty in the



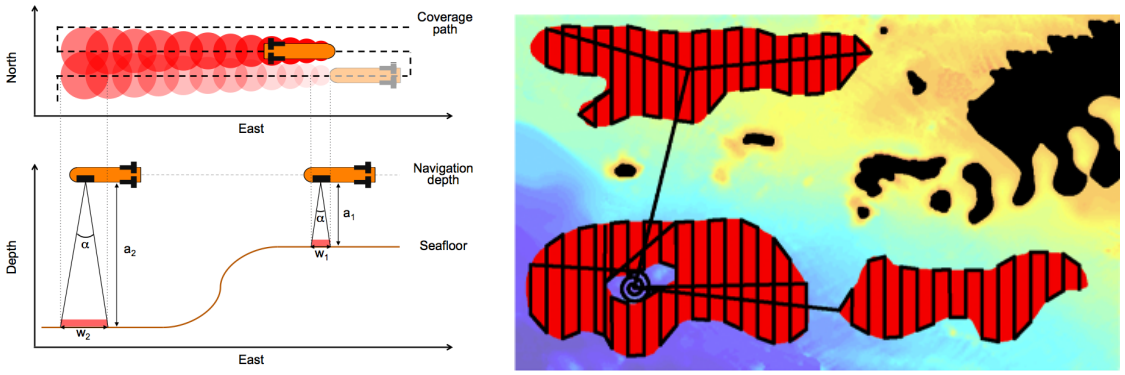
(a) Hexagonal grid (b) Branch entropy (c) trajectory of the UAV

Figure 2.20: UAV trajectory planning in a sensor-driven approach based on information gain. Figure reproduced from [93].

map need to be covered again. Paull *et al.* consider a similar case in [93], where a sensor-driven coverage path planning method is proposed for fixed-wing vehicles (originally developed for AUVs in [94]). In this work, they do not assume that the vehicle follows the planned trajectory correctly. Therefore, after completing a short straight trajectory, the heading of the UAV is computed in real-time based on the online sensor readings. A grid-based coverage map is maintained during operation in which a binary random variable is assigned to each cell which shows if that cell is visually covered sufficiently so that it can be classified as a target object or a non-target object. Then to choose the next heading direction, a weighted multi-objective function is minimized which is composed of two parts. The first objective function directs the robot towards a direction that yields maximum sum of expected information gain in the belief about the sensed cells. In order to obtain an efficient coverage path and avoid a greedy-first search in a unknown environment, the second part of the objective function incorporates the branch entropy concept. For this, a separate hexagonal grid is considered as shown in Figure 2.20a. Then a directed acyclic graph (DAG) is created based on the cell that the UAV currently occupies. The branch entropy is defined for each neighbour and is the weighted average of the mean entropy at each level of a tree in the DAG as shown in 2.20b. The weighting is determined so that the closer cells (upper in the tree) have more effect on the branch entropy. Therefore, the final heading is determined considering both the short-term reduction in entropy and the long-term coverage goal. They tested their method in simulations and validated the results with real fixed-wing vehicle (Figure 2.20c). It is shown that while the UAV performing a Lawnmower pattern sometimes misses parts of the area, the proposed approach completely covers the area by online planning of the flying direction. However, the final trajectories are long and the overlap in the sensor footprint is high which indicates the disadvantage of this method.

Underwater Coverage for Barometric

Autonomous Underwater Vehicles have been used to perform surveys for bathymetric or marine habitat mapping. Although some of the CPP methods for AUVs deal with problems that are specific to under water applications, most of them are concerned with the general coverage trajectory planning and can be inspiring for aerial setting.



(a) Sensor footprint overlap changes due to varying seabed height

(b) Coverage path of the interesting regions of the sea floor

Figure 2.21: Coverage path planning for underwater vehicles. Figure reproduced from [95] and [96].

Galceran *et al.* noticed unnecessary overlap of the sensor footprint in consecutive lanes of Lawnmower when covering seabed floor [95]. This problem occurs when a surface or underwater vehicle with a downwards looking sensor is sweeping a surface with varying height at a constant depth as demonstrated in Figure 2.21a. Therefore they segment the environment into regions of approximately same height. Then for each region, sweeping in a direction orthogonal to the surface gradient, the inter-lap spacing is computed based on the sensor footprint on the surface. This is an efficient coverage planning when the environment is known a priori. In a different work, Galceran *et al.* consider the problem of covering subregions of interest extracted from previous surveys to enable marine habitat monitoring [96]. They first use an approximate TSP solver to find a visiting order of the subregions and then generate required trajectories to cover each subregion using online decomposition-based CPP (Figure 2.21b). Similarly, Hollinger *et al.* have presented a method for generating survey trajectories that are targeted for certain areas of interest [97]. In their case, after a complete maximum-coverage mapping of the underwater surface with an AUV, a successive partial survey is required to improve the reconstruction quality in regions with high local uncertainty. The new survey is generated by greedily choosing a set of diving directions from a pre-defined set. They show that using this adaptive 2-stage survey produces a map with lower uncertainty than a complete coverage with maximum overlap in the sensor footprint.

In [98], Williams considers selecting a set of Lawnmower tracks for an AUV in an offline setting. Assuming a prior map of the type of seabed and the probability of mine detection based on the seabed type, his algorithm iteratively chooses a track that maximizes the increase in the probability of detection. Furthermore, the algorithm allows the displacement of the predefined tracks in an orthogonal direction to improve the greedy initial set of tracks. Also the tracks are traversed in the order of spatial proximity as opposed to detection benefit. He shows that his method produces higher probability of detection compared to uniformly-spaced Lawnmower tracks. In a later work, Williams assumes that the seabed type is not given in advance. Therefore he uses a ripple recognition algorithm to find regions with ripples. For this he proposed a method

to select the best Lawnmower track online based on the current state of task [99]. Each predefine track is evaluated in terms of the ripple detection benefit and travelling cost to reach the track. The best track is then selected and executed and the track selection process repeats again with the updated data. Since the shadows produced by the sand ripples obscure image data, a second survey is generated afterwards to cover the regions with these shadows hence the need for robust detection of them at the first survey.

Conclusion

A summary of the CPP methods developed for aerial coverage is shown in Table 2.1. It indicates that most of the these methods are designed to be used in an offline setting and the trajectory does not adapt to the environment as the UAV collects more information during the mission. The few online approaches that are developed for UAVs, suffer from the fact that they do not produce short trajectories. Also, most of them rely on some kind of a priori knowledge about the regions of interest and are not effective in a completely unknown environment. This can easily be seen in the probabilistic quadtrees method ([92]) where the strategy to cover the regions of interest is according to a utility function that prefers visiting locations with high information gain and small distance. This heuristic may be effective to increase the probability of finding the ROIs but can not fulfill basic requirements of achieving an efficient coverage trajectory such as minimizing sensor overlap and hence the UAV might travel to a part of the environment a number of times before covering it completely. This switching between possible ROIs results in long trajectories. Therefore, as somehow noted briefly in [77] and [98], searching for the ROIs should be combined with a coverage strategy to produce suitable trajectories.

Another possible way of improving the efficiency of online aerial coverage is to consider the spatial properties of the ROIs. In all the works presented in this report, the target objects of interest (or interesting cells) are supposed to be independent of the neighbours. This is a sensible assumption when the target fits in a grid cell or is small. However, in many situations the interesting targets are in form of patches that occupy a big section of the area and exhibit a form of locality. Consequently finding an interesting region should increase the belief of the robot about the interestingness of the neighbouring regions and therefore a more effective strategy of discovering RIOs can be achieved. Furthermore, in designing effective methods for online aerial coverage of ROIs, one important rule of thumb is to avoid uninteresting sections of the environment. Since these regions are not known in advance, they can not be avoided completely. However, an alternative is to cover them with a lower sensing/navigation cost. We examin both of these ideas in Chapter 4 and 5.

Method	**Online/Offline**	**Coverage/Search**	**Remarks**	**Ref.**
Cellular decomposition	offline	coverage	requires *a priori* knowledge	[79, 80, 82, 81]
Grid-based aerial coverage	offline	coverage	requires *a priori* knowledge	[84, 86]
Complex structures coverage	offline	coverage	simplifies the scene	[87, 42]
Modified Lawnmower search	offline	search	only uses pre-defined path tracks; restricted prior form	[89]
Hierarchical heuristic search	offline	search	only covers the most probable regions	[88]
Marine habitat mapping	offline	coverage	requires *a priori* knowledge	[96]
Adaptive Lawnmower track selection	online	coverage	pre-defined path tracks; unnecessary revisits	[98, 99]
Active dive planning	online	coverage	2-stage coverage	[97]
Priority-based coverage	online	search	uses POMDP; needs prior knowledge	[90]
Probabilistic quadtrees	online	search	revisits regions many times	[91, 92]
Sensor-driven coverage	online	coverage	long trajectories and sensor overlap	[94, 93]

Table 2.1: Summary of the coverage methods used on various platforms

Chapter 3

Robust Navigation of MAVs with Monocular SLAM

Autonomous operation of MAVs requires accurate pose information. For indoor applications, external motion capture devices can be used to obtain the location of the vehicle. However, this involves modification of the working environment. Moreover, satellite-based GPS is not available when the view of the sky is obstructed. *Simultaneous localization and mapping* (SLAM) methods have been developed in the last decade and successfully used to estimate the pose of a robot. Active sensors such as laser range finders or RGB-D sensors, though successfully used on-board MAVs for SLAM, pose a problem for quadrotors with very limited payload and energy reserves. On the other hand, a passive camera provides rich sensory information, consumes relatively little power and is very low mass, which makes it a suitable sensor on MAVs for localization and mapping [100, 22]. For these reasons, we restrict our work to single-camera sensing.

Visual SLAM, however, is critically dependent on the availability of feature-rich areas in the environment. The accuracy of localization can be high when there are enough feature points in the camera view but, as the camera moves to a texture-less area, few measurements can be made and the uncertainty in pose estimate grows and SLAM fails. Therefore, for navigating to a goal, a trajectory along which the vehicle stays in feature-rich areas is a safer path. However, this requires a map of the environment with information about the distribution of feature points which is not always available. We propose an approach to navigate safely to a goal location by iterative re-planning. The quadrotor starts with an initialized monocular SLAM system and continuously updates its estimate of the visual feature distribution and re-plans a path to the goal accordingly. Therefore, the proposed method is able to accomplish the navigation task, in an environment with irregular feature distribution (as thus unsafe regions), by generating paths that, though longer, are in visually rich part of the environment. The same planner cannot reliably perform the task when it ignores feature visibility.

(a) Experimental arena. With no initial map, and using only mono-SLAM, the MAV must navigate around obstacles to reach a goal point. The left-hand part of the room has few visual features and is unsafe for navigation.

(b) A sparse 3D reconstruction of the room. The left-hand side is a poor reconstruction.

Figure 3.1: The target environment

3.1 Feature-rich Path Planning

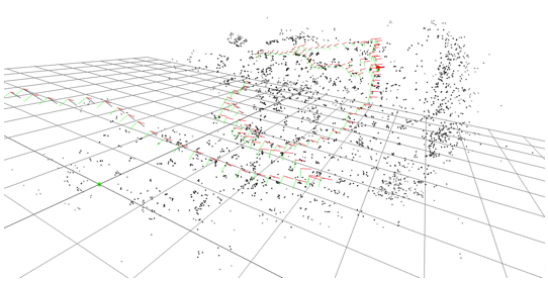
We consider a quadrotor equipped with a camera, performing visual SLAM[1] to obtain its current position. To perform the navigation task, we generate a number of candidate collision-free paths using a sampling method, and evaluate each path in terms of localization quality. This is achieved based on the current state of the already mapped regions (as explained in section 3.1.3). A candidate path is then selected to be executed based on a cost function. During the navigation, the quality of pose estimation is monitored online and in case of poor performance, the system re-plans to use the new information.

The next two sections provide more details of the proposed method.

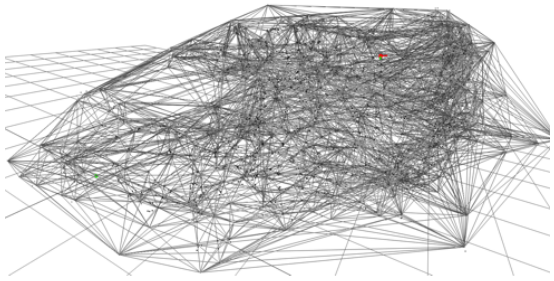
3.1.1 Localization and Mapping

We use Parallel Tracking and Mapping (PTAM) [15] as our monocular visual SLAM system. PTAM has been successfully used in aerial vehicle navigation [22, 101]. It divides SLAM into tracking and mapping, each running in a separate thread. The tracking component tracks salient features (FAST corners [102]) in the camera image by matching them with the feature points already stored in the map. For matching features, first a motion model is applied to the camera pose which results in a prediction of the camera pose. Then the map points are projected back onto the camera frame and compared with the extracted salient features in the camera frame by local search in the image. After feature matching, the new pose of the camera is calculated by minimizing the error between the position of the observed features in the image and the projection of the map points onto the camera frame. Tracking is performed in every frame and, based on some heuristics, some images are selected and inserted in the map as keyframes. The mapping component performs local and global bundle adjustments to refine the pose for all keyframes (except the first keyframe) and all the points in the map.

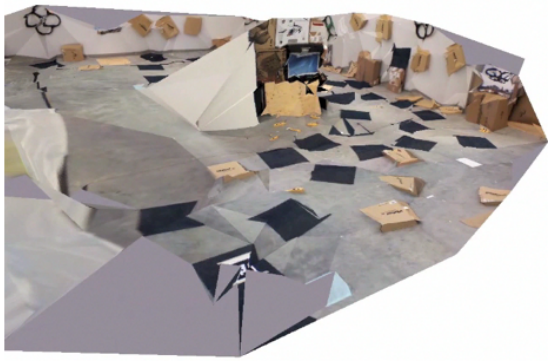
[1] All the computation for this system is done off-board. But in principle it can be performed on-board too.



(a) Mapped feature points



(b) 3D delaunay triangulation of the points.



(c) Final 3D mesh.

Figure 3.2: Sparse reconstruction of the environment. (a) The output of the PTAM is a collection of map points and key-frames. (b) A 3D delaunay triangulation of the feature points is created. The true surface of the environment is embedded in this mesh. (c) the final mesh is obtained by minimizing an objective function using graph-cuts optimization.

Note that the tracking component localizes the camera with respect to the map points that are generated by matching features between key-frames. Hence, as the camera moves, the map is extended by inserting key-frame. Therefore localization becomes possible as the camera is moved to unmapped regions. For key-frame insertion, PTAM uses a distance criterion to other key-frames. However, the camera can be close to other key-frames and at the same time no mapped features are visible (e.g. when the camera is close to a surface). Subsequently, to permit flexible map expansion in exploratory behaviours, we use visual change cue as a criteria for key-frame insertion, i.e. if the number of observed (matched) map points becomes less than 50, a key-frame is created to extend the map. Such heuristics for key-frame insertion have been proposed in more recent visual SLAM systems [17].

3.1.2 Sparse 3D Reconstruction

PTAM represents the map of the environment by a cloud of feature points that are measured in two or more key-frames. To produce a mesh of the environment, we perform a sparse 3D reconstruction of the explored area using the method introduced in [103]. This approach is suitable for scene reconstruction using features generated by *structure from motion* (like PTAM), and produces acceptable results even when using sparse point clouds. It first creates a 3D

Delaunay triangulation of the map points[2]. For this step, we only use the map points that have been observed from at least 4 key-frames. Therefore, the reconstructed mesh and the prediction of localization quality (as we will see later) use the map points that have high probability of existence and unlikely to be spurious. Then using graph-cuts, it extracts the surface mesh embedded in the Delaunay triangulation that minimizes an energy function. The energy function is based on the visibility constraints imposed by the line of sight of each feature. To reduce the computational cost, we do not include the photo-consistency part of the energy function (see [103] for details).

We use the resulting surface mesh in two ways: (i) to avoid obstacles; and (ii) to estimate the feature richness of different regions in the environment (described in the next section). For both these purposes, we use ray-tracing to find out if a line segment $\overline{p_s p_e}$ collides with the surface. We efficiently perform ray-tracing using the original Delaunay triangulation which embeds the surface mesh as follows: The tetrahedron that contains the start point p_s is looked up (already an efficient operation). Then, the next tetrahedron that collides with the line is found by checking all neighbour cells (two tetrahedra are neighbours if they have a common triangle). This process is done iteratively and terminates upon reaching the tetrahedron that contains the line end point. If during this process the common triangle between two consecutive tetrahedron belongs to the surface mesh, it implicates a collision. For line segments where either of the end points is outside the convex hull of the point cloud (and is thus not contained by any tetrahedron), we perform ray tracing using the line segments formed by the middle point of the line and either one of the end points. The same process is used if both the end points are outside the convex hull unless the length of the line segment is smaller than some threshold. Note that the ray tracing queries are performed very efficiently by moving along a line segment in non-uniform steps because most of the queries encountered during planning are almost completely contained in the convex hull of the feature cloud.

3.1.3 Planning and Navigation

We propose a novel approach in which the MAV navigates to its goal safely by iterative evaluation of availability of feature-rich regions and re-planning. At the beginning a sparse mesh is produced using the initial feature cloud of PTAM. Then a set of candidate plans is generated using Rapidly-Exploring Random Trees (RRT*) [104]. We eventually select the plan that optimizes a cost function that includes both path length and the expected amount of features along the path (explained later). A plan P consists of a list of waypoints $p_i = [x_i, y_i, z_i, \theta_i]$ which specifies a 3D point at location $[x_i, y_i, z_i]$ in the world coordinate system and a rotation of θ_i along the world Z axis. The drone visits the waypoints one by one using a PID position controller. At each waypoint the whole scene is reconstructed and the 3D mesh is updated. The drone re-plans after a fixed part of the plan has been executed (visited 4 waypoints in our experiments). This process is repeated until the drone reaches the goal (within a radius of 1 m). The reason behind re-planning is that when the drone moves along the plan, new points are added to the map and

[2]We used the CGAL library for this purpose

the 3D mesh of the scene is changed. Therefore, re-planning might generate a better path than the current plan since it uses the most up-to-date information about the world.

Detecting a small number of useful features or predicting collisions will result in re-planning as follows: as the drone is following the waypoints, we track the number of features that are successfully matched to map points and used towards pose estimation in PTAM. If it is lower than a threshold (50 features), the drone changes its current target waypoint to the previous waypoint and re-plans when it reaches that waypoint. Detecting a future collision along the plan (after updating the mesh) triggers the same behaviour. In both cases, moving to the last visited waypoint provides free space for the drone to manoeuvre and get out of unsafe situations encountered during exploration of unknown environments.

Next, we describe how the expected amount of visual features is estimated and used in the planning cost function.

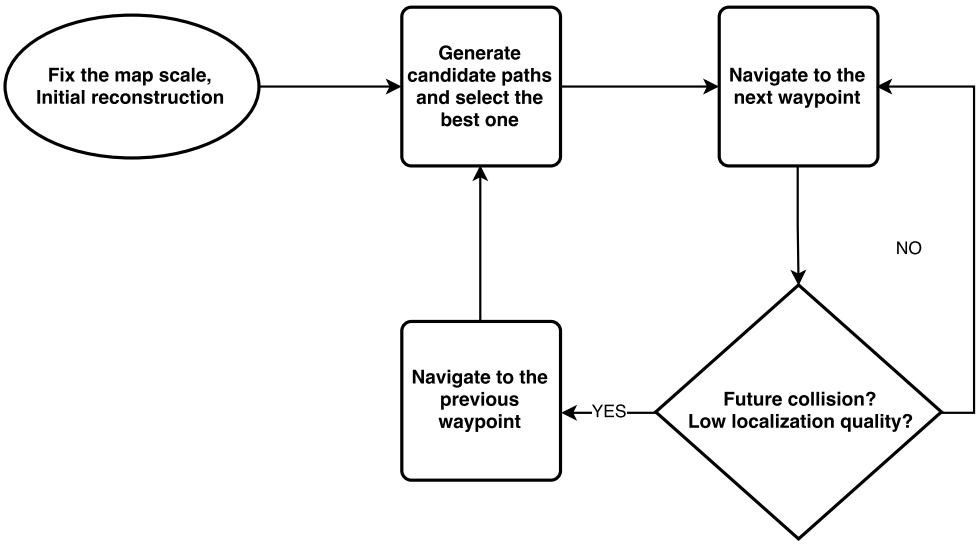


Figure 3.3: The behaviour of the MAV.

Viewpoint Score Generally in key-frame based visual SLAM systems, the view points that are very close to key-frames in terms of distance and viewing angle, with a high chance can extract and match many features to the map points. But this criterion is very restrictive and does not provide a way to predict if a view point far from the key-frames is texture rich enough for SLAM to work. Therefore we devised a more general way using local density of the reconstruction mesh triangles to predict the expected number of detectable features in an image taken from a viewpoint.

The scoring function contains two parts:

$$s(P) = D(P)V(P)$$

where $P \in SE(3)$ is a 6-DoF pose containing translation $P_t \in \mathbb{R}^3$ and rotation $P_r \in SO(3)$ that specify the viewpoint. We define $D(P)$ as a measure of the texture quality and $V(P)$ as an estimate of how well the target area is visible from the view point. The calculation of these functions is as follows: a ray is cast from P along the viewing angle to find the point $c \in \mathbb{R}^3$ on the mesh that will be in the center of the camera view. Then the set T of all the triangles on the mesh with centres within a radius r (1 m) of c are found. Next, the normal of the local surface $\overrightarrow{n}$ is estimated by fitting a plane to the triangles (Figure 3.5). Now we define $D(P)$ as:

$$D(P) = qm$$

Here m is the density of the triangles in T, that is number of triangles divided by the sum of their area. Also q, with $0 \leq q \leq 1$ is the quality of the plane fitting, where 0 means that the variance is the same along any plane going through the best fitting line, and 1 means that the variance is null orthogonally to the best fitting plane.

The second component of the scoring function, $V(P)$, penalizes the view points as follows:

$$V(P) = \cos(\alpha) \exp(-|d_{pref} - \|\overrightarrow{cP_t}\||).$$

Here α is the angle between the local surface normal $\overrightarrow{n}$ and $\overrightarrow{cP_t}$ and d_{pref} is the preferred viewing distance. Therefore, view points at the preferred distance and fronto-parallel to the target surface receive high scores. On the contrary, viewing angles away from surface normal or very close to or far from the surface are assigned low scores.

With the above $s(P)$ function a very dense patch of triangles might lead to a very high score, therefore we define an upper bound for $s(P)$ and normalize the score as $s_{norm}(P)$. We use a kernel function $f(x) = \dfrac{a}{1 + e^{bx+c}}$, shown in Figure 3.4, to increase the contrast between high and low scores which leads to the final form of the scoring function:

$$Score(P) = \begin{cases} \dfrac{a}{1 + e^{bs_{norm}(P)+c}} & P \text{ is close to key-frames} \\ s_{default} & \text{Otherwise} \end{cases}$$

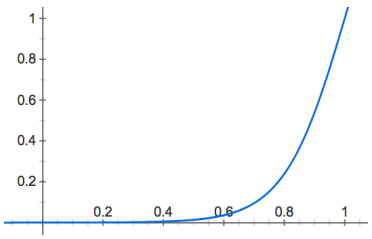


Figure 3.4: The kernel function used in scoring viewpoints.

Note that, since the viewpoints that are far from the key-frames (translational distance more than $4m$ or yaw distance more than 90°) cannot be reliably scored using the above method, they are assigned a default score $s_{default}$.

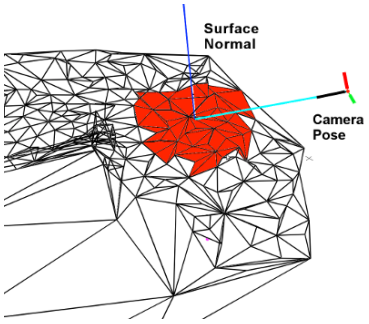


Figure 3.5: Finding the local density of the mesh. The density of triangles in the red area is used to obtain the score of the camera viewpoint.

Planning Planning is based on RRT*. The main idea is to generate a set of candidate paths and then select the one that has minimum cost. The cost function includes both travelling distance and viewpoint score of the waypoints along the path. The Feature-Rich RRT* (FR-RRT*) is described in Algorithm 1.

Algorithm 1 Feature-Rich RRT*

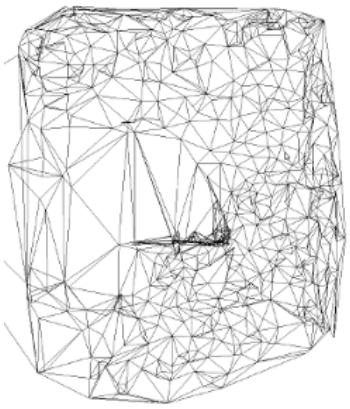
Input: Reconstructed mesh M, workspace X_{all}, motion constraints K, start pose $\mathbf{x}_{start}$, goal position $\mathbf{p}_{goal}$, number of iterations N.
Output: A set of candidate paths.

$V \leftarrow \{\mathbf{x}_{start}\}$, $E \leftarrow \emptyset$, $Tree \leftarrow (V, E)$
$C \leftarrow \emptyset$ \\ *The set of candidate nodes.*
for $i = 1...N$ **do**
  $\mathbf{x}_{sample} \leftarrow Sample(X_{all})$
  $\mathbf{x}_{near} \leftarrow Nearest(\mathbf{x}_{sample}, V \backslash C)$
  $\mathbf{x}_{new} \leftarrow Steer(\mathbf{x}_{near}, \mathbf{x}_{sample}, K)$
  if $NoCollision(\mathbf{x}_{near}, \mathbf{x}_{new}, M)$ AND $AcceptableScore(\mathbf{x}_{near}, \mathbf{x}_{new})$ **then**
    $V \leftarrow V \bigcup \{\mathbf{x}_{new}\}$
    $E \leftarrow E \bigcup \{(\mathbf{x}_{near}, \mathbf{x}_{new})\}$
    $Tree \leftarrow Rewire(\mathbf{x}_{new}, Tree)$
    if $IsCandidate(\mathbf{x}_{new}, \mathbf{p}_{goal})$ **then**
      $C \leftarrow C \bigcup \{\mathbf{x}_{new}\}$
    end if
  else
    $Delete(\mathbf{x}_{new})$
  end if
end for
return $ExtractPlansToNodes(C, Tree)$

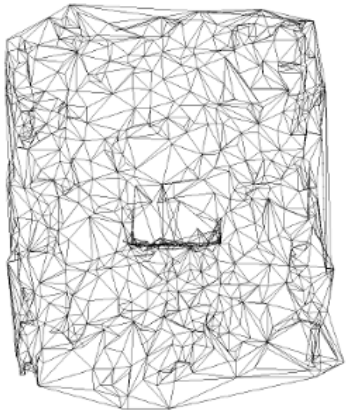
The algorithm extends the tree incrementally as follows: It samples a point from the configuration space and generates a new node using the *Steer*() function to extend the nearest node to the sample. The *Steer*() function satisfies constraints defined for the drone's motion. Specifically, the drone is allowed to move only in the direction of the camera (within an angle

(a) Feature-poor

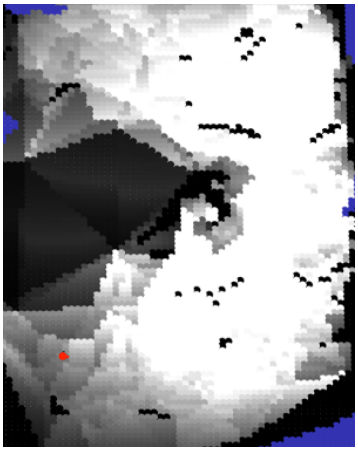
(b) Feature-rich



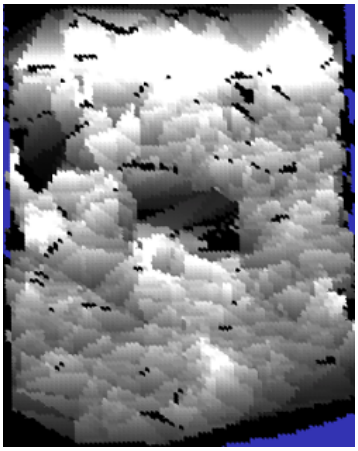
(c) Feature-poor. Note the low triangle density in the low feature region



(d) Feature-rich. The triangle density is high and uniform in the environment.



(e) The viewpoint score of feature-poor environment.



(f) The viewpoint score of feature-rich environment.

Figure 3.6: Two configurations of the room. (a,b): 3D reconstruction of the room. (c,d): The generated mesh of the room. (e,f): A map of the feature-richness score. The direction of all the samples are towards the top of the image.

of 45°) unless it is moving to a place that it has been before. This is to ensure that the drone can detect obstacles that are on its way. The tree is extended with the new node if the resulting edge does not collide with the mesh ($NoCollision()$ function) and the viewpoint score along the edge is above a threshold ($AcceptableScore()$ function). When a new node is added to the tree, we maintain the optimality of the tree in travelling cost using the $Rewire()$ function. This function implements the approach used in [104] for RRT*.

A newly added node enters the $Candidate$ set if either it is within some radius of the goal or it satisfies all the following conditions: a) it has a default viewpoint score b) the line connecting the node to the goal is collision free and the angle it makes with the direction of the node is less than 90°. The nodes in the $Candidate$ set are not expanded during the subsequent samplings. Therefore the candidate plans might not go all the way to the goal but stop at waypoints from where a straight line will result in a collision-free path. Note that these decisions are based on the currently available and usually incomplete information about the world. In case new obstacles are encountered during the execution of the plan, the drone re-plans as explained in Section 3.1.3.

The above algorithm produces a set Q of plans. Each plan $P_i \in Q$ is a list of N_i waypoints p_{ij} (with current pose at the start of the list). We select the plan that minimizes the following cost function as the best plan:

$$Cost(P_i) = \sum_{j=1}^{N_i-1} \frac{Dist(p_{ij}, p_{ij+1})}{MinScore(p_{ij}, p_{ij+1})} + EstimateDist(p_{iN_i}, \mathbf{p}_{goal})$$

In the above equation, $Dist(.,.)$ calculates the distance between two waypoints considering both the translation and yaw (4 DoF). $MinScore$ finds the smallest viewpoint score along the edge connecting two points by some sampling viewpoints in fixed intervals along the edge. $EstimateDist(.,.)$ estimates the travelling distance from the last waypoint to the goal considering the motion constraints mentioned before.

3.2 Experiments Setup

We tested our proposed method with a real MAV in a $8m \times 6m$ room. The commercially available Parrot AR.Drone 2.0 quadrotor was used as our platform. The only sensor that is used in our experiments is the forward camera which has a resolution of 640x320. We changed the camera angle so that it has a pitch of $\approx 45°$ towards the ground to capture images from both forward and bottom areas. The robot communicates with an off-board computer over Wi-fi and all the sensors' data are read using *ardrone_autonomy*, a ROS driver for AR.Drone quadrotors. Control commands are sent as (x, y, z, θ) velocities.

For map initialization, PTAM starts to track features upon receiving a *begin-tracking* signal. It expects that the camera moves while the features are being tracked such that an appropriate baseline is formed. Upon the second signal (*end-tracking*), PTAM triangulates the features using the start and end frames and the initial map points are created. Note that PTAM requires plannar initial scene and therefore we use the room floor for this purpose. In order to automate

Figure 3.7: AR.Drone 2.0 used in the experiments.

PTAM initialization, after the MAV takes off, we use a fiducial marker tracker [3] to localize the vehicle and perform the map initialization by flying sideways. After PTAM is initialized, we use the same fiducial marker to fix the scale of the map and transform it to a predefined coordinate system. For this, we make sure that the map initialization is done using the features around the fiducial marker such that the initial map is a plannar surface around the marker. Consequently, the height of the MAV with respect to the marker is used to scale the MAV height provided by PTAM and subsequently the whole map is transformed accordingly.

The task was to navigate from a predefined starting position to a fixed 3 DoF goal position specified with respect to the fiducial marker. The drone is unable to sense the goal directly, which precludes visual servoing. The 3D reconstruction of the room is shown in Figure 3.8. The start and goal positions are also indicated with a square and a star shape respectively. A wall of cardboard boxes was placed in the middle of the room as an obstacle. This world configuration provides two paths from the start to the goal location. A short, direct path on the left of the obstacle and a larger path on the right. It is worthwhile pointing out that the right path requires a lot more turning manoeuvres and hence is a lot more challenging for the drone. We performed experiments with two different visual feature distributions in the room (see Figure 3.6a, 3.6b): The first one has low amount of texture on the left side of the obstacle whereas the second one is approximately uniformly textured. For each room configuration we performed 10 trials using FR-RRT* and RRT*. If the drone reached the goal (within a radius of $1m$), the trial was *successful*. A trial was marked a *failure* if the drone lost track of the features, therefore failed to estimate its pose and subsequently was unable to navigate.

3.3 Results and Discussion

The results are shown in Table 3.1. In the first environment configuration the shortest path found by RRT is always passing through the left side of the room which is a texture-less area and therefore 70% of the time the vehicle fails to reach the goal (Table 3.1a). However, RRT

[3] http://www.ros.org/wiki/ar_track_alvar

Figure 3.8: The MAV task starts from the location marked by the indicated square and navigates to the goal location indicated by the star.

sometimes generated a sub-optimal path on which some of the texture patches behind the wall were captured in camera images and therefore PTAM was able to continue pose estimation (see Figure 3.9). By contrast, using FR-RRT* the drone only failed 20% of the time to reach the goal. This was achieved by choosing a safer, texture-rich path on the right side of the room (see Figure 3.10). In the fully-textured environment, FR-RRT* performs almost the same as RRT* (Table 3.1b).

	Success	Failure
FR-RRT*	8	2
RRT*	3	7

(a) Experiment 1: Irregularly-textured room

	Success	Failure
FR-RRT*	9	1
RRT*	10	0

(b) Experiment 2: Uniformly-textured room

Table 3.1: The results of the experiments with two different configurations.

Experiment 1: irregularly-textured environment To show the behaviour of the drone, sample plans generated by FR-RRT* are illustrated in Figure 3.10. They indicate that the initial plans are close to the shortest path. But, as the vehicle visits some waypoints close to the texture-less area and updates the 3D reconstruction, the next plans deviate from the

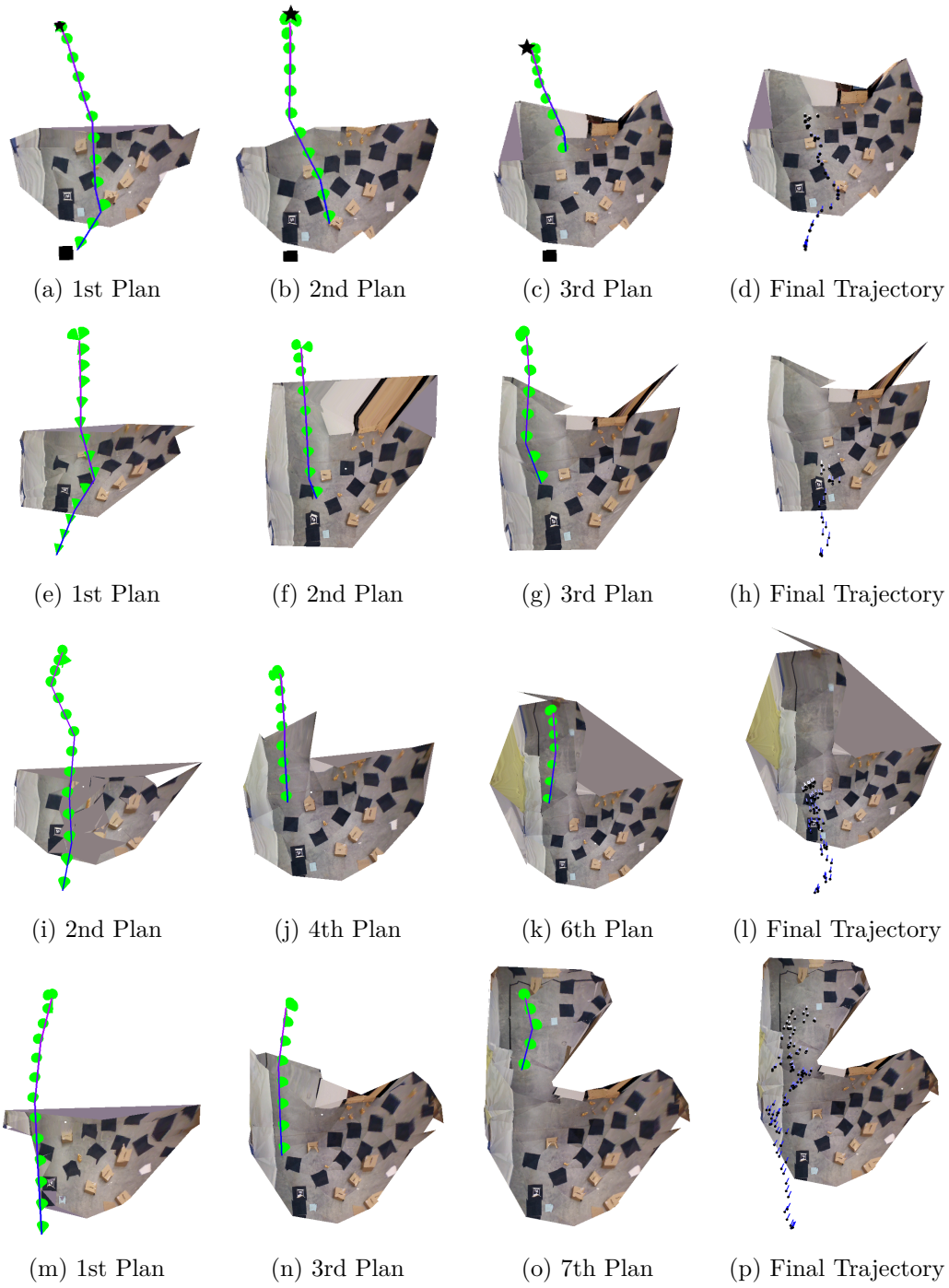
shortest path and are more inclined to pass through areas with high texture due to high cost of waypoints in texture-poor areas (Figure 3.10b, 3.10c). The last row of figure 3.10 (m-p) shows a case where the MAV failed to reach the goal using FR-RRT*. In our implementation, since the quadrotor was not able to perform translation and rotation commands well at the same time, we first correct the translation error in the position of the robot and the waypoint position, and then correct the heading of the quadrotor. Therefore, as shown in Figure 3.10o, when the MAV was navigating to the second waypoint of the 3rd plan, the camera was pointing at the texture-less floor of the room, and hence PTAM failed. Using a more capable vehicle that can simultaneously perform rotation and translation movements, can avoid such failures.

In contrast to our method, RRT* paths are uninformed about the saliency of the environment texture and hence only optimize for travelled distance. As shown in Figure 3.9, RRT* paths are unsafe and unable to direct the vehicle to the goal in 70% of the experiments. Figure 3.9p demonstrates a example trial where the localization did not fail and the quadrotor reached the goal location. This happened because the MAV moved to the left of the room while the camera was pointing at the floor behind the wall and PTAM was able to map some of the feature points in that region early in the experiment. Hence, sufficient number of map points were observed in the subsequent waypoints and the localization did not fail.

Experiment 2: regularly-textured environment When the area was uniformly-textured, the MAV was able to reach the goal location using RRT* in all the trials (see Figure 3.11). Since the optimal route to the goal was an straight path forward, the camera was pointing to the floor and PTAM could track and map enough features to localize the UAV.
Our method is also successful to direct the MAV to the goal in 90% of the trials. As shown in Figure 3.12, The waypoints generated by FR-RRT* are more exploratory compared to RRT*. However, there was a failure case in our method, which is shown in Figure 3.12 (e-h). In this trial, the MAV moves backward to the second waypoint of the 7th plan (Figure 3.12g). Since the front camera is rigidly attached to the quadrotor, as soon as the movement starts, the camera pointed at the plain wall on the left of the room and consequently the localization failed. Mounting the camera on a gimbal can mitigate this problem.

Mapping Another observation is that when images captured by the camera are not feature-rich enough, PTAM needs more key-frames to be able to provide enough map-points for pose estimation. This leads to a high density of key-frames in low-textured area and an increase in memory consumption. On the other hand, if the number of features found in key-frames is high (i.e. moving in texture-rich region) there is no need for a high density of key-frames and PTAM is able to measure features in the local space without inserting new key-frames. This means fewer key-frames and lower memory use.

During the experiments we observed that in some cases the pose estimate is not accurate enough to avoid collisions. This could be due to inaccurate scale estimation during initialization. Since the safety distance for the drone is in absolute scale, there is a possibility of collision with obstacles in these situations. We terminated the trials in which the drone was too close to



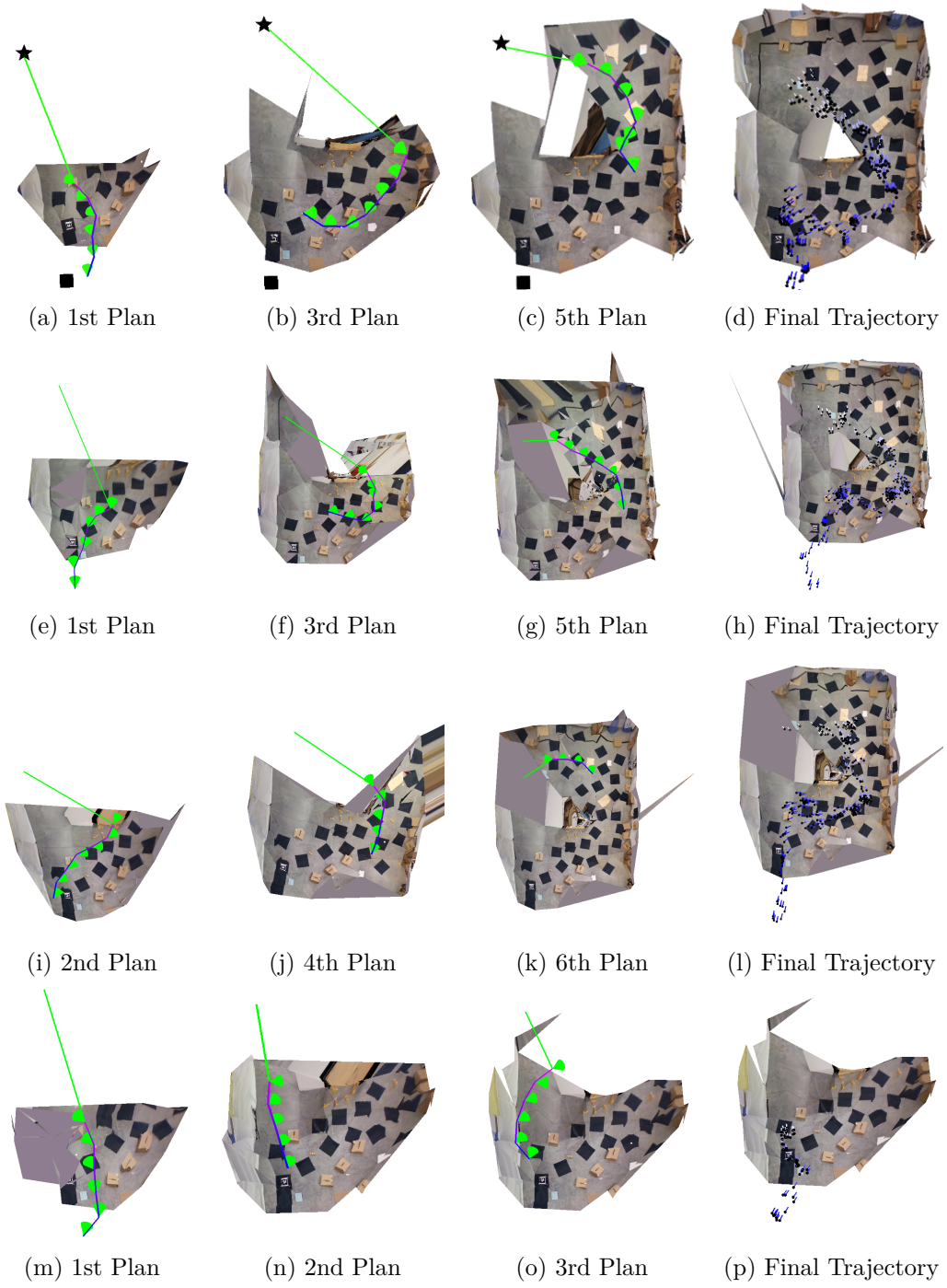
(a) 1st Plan (b) 2nd Plan (c) 3rd Plan (d) Final Trajectory

(e) 1st Plan (f) 2nd Plan (g) 3rd Plan (h) Final Trajectory

(i) 2nd Plan (j) 4th Plan (k) 6th Plan (l) Final Trajectory

(m) 1st Plan (n) 3rd Plan (o) 7th Plan (p) Final Trajectory

Figure 3.9: Selected plans generated by RRT* in irregularly-textured environment (30% success). (a-l) The MAV does not reach the goal, as it is unable to localize when it reaches a feature-poor area. (m-p) A sample trial where the MAV managed to reach the goal. At the beginning of the experiment, the MAV was able to mapped some feature points on the back of the room from the left side of the take-off spot (see the final trajectory (p)). Consequently, flying straight to the goal, it did not fail to localize.



(a) 1st Plan (b) 3rd Plan (c) 5th Plan (d) Final Trajectory

(e) 1st Plan (f) 3rd Plan (g) 5th Plan (h) Final Trajectory

(i) 2nd Plan (j) 4th Plan (k) 6th Plan (l) Final Trajectory

(m) 1st Plan (n) 2nd Plan (o) 3rd Plan (p) Final Trajectory

Figure 3.10: Selected plans generated by FR-RRT* in irregularly-textured environment (80% success). Each row corrsponds to a sample trial. (a-l) The MAV reaches the goal, after replanning to avoid a feature-poor region. (m-p) A sample failure case. The MAV lost track of features while moving towards the second waypoint of the 3rd plan (i.e (o)). Since in our controller the heading of the robot is corrected when the MAV is at the goal waypoint, the camera was pointing at the texture-poor region of the room.

(a) 1st Plan (b) 2nd Plan (c) 4th Plan (d) Final Trajectory

Figure 3.11: Plans generated by RRT* regularly-textured environment (100% success). (a-d) The MAV moves directly to the goal, while observing enough features to perform mapping and localization.

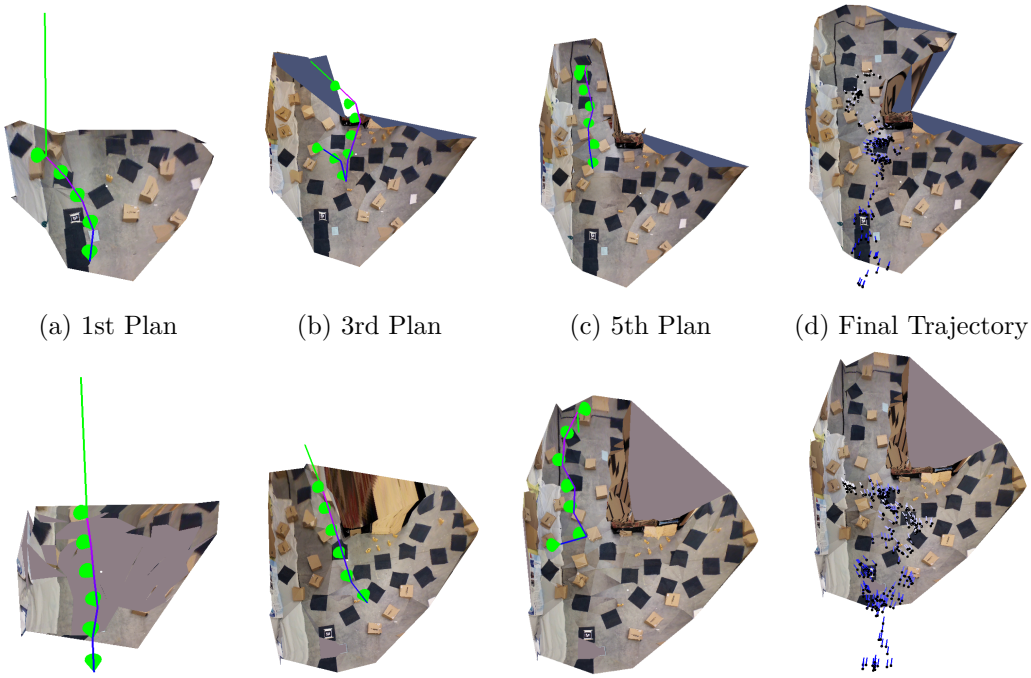
obstacles (9 cases in all 54 experiments). Also in 3 cases the drone ended up too close to obstacles (e.g by turning and seeing a new nearby obstacle) such that no valid paths could be found since it was already in a collision state. All these trials were excluded from the analysis as they are not caused by our method but the limitations of the vision-only SLAM and navigations.

3.4 Conclusion

Visual SLAM seems to have become the preferred way of mapping and localization with small UAVs due to the light-weight and little power consumption of the cameras. Deploying MAVs to unknown GPS-denied environments requires localization-aware path planning to provide safe paths between two locations. In this chapter, we proposed a path planning method that generates paths that pass through visually-rich regions of the environment. Therefore the monocular SLAM is able to observe enough features to map the environment and localize the robot. The real robot experiments show that the proposed method is able to generate longer but safer paths for the UAV compared to an unaware conventional planner.

The proposed planner uses the point cloud generated by the visual SLAM to obtain a sparse mesh of the local environment. The mesh is used for obstacle avoidance and predicting the localization quality of a camera view-point. We indirectly estimate the informativeness of candidate viewpoints by defining a view-point score. This score measures (a) the texture richness and planarity of the observed surface and (b) how well the camera is looking at that surface. The viewpoint score defined in section 3.1.3 can be extended to include other criteria such as local saliency [52] and map point uncertainty [50] (as will be discussed in Chapter 6)[4].

[4]The work in this chapter was published at ICRA 2014 [105].



(a) 1st Plan (b) 3rd Plan (c) 5th Plan (d) Final Trajectory

(e) 1st Plan (f) 3rd Plan (g) 7th Plan (h) Final Trajectory

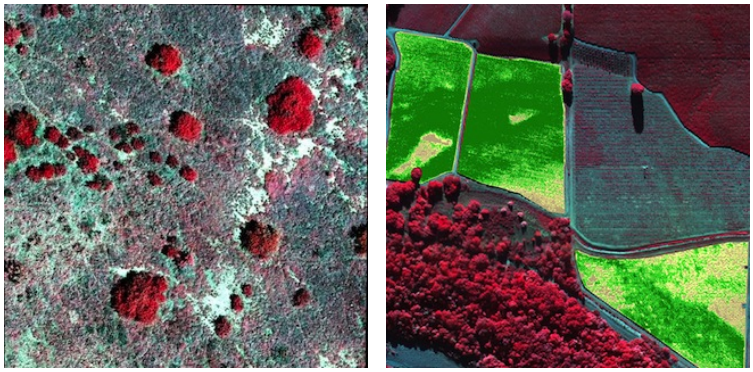
Figure 3.12: Plans generated by FR-RRT* in a regularly-textured environment (90% success). (a-d) The MAV demonstrates some exploratory behaviour while navigating to the goal. (e-h) A failure case where the UAV is unable to observe enough feature points. This happened in the 7th replanning (see (g)) where the MAV was moving backwards to the second waypoint of the path. As soon as the MAV started to move back, the camera pointed at the plain wall and no measurements were taken. Mounting the camera on a gimbal can help avoid this problem.

Chapter 4

Non-Uniform Aerial Coverage

Aerial imaging is one of the canonical tasks performed by UAVs. The collected images are excellent sources of data for experts and can be obtained quickly in an highly automated process. To perform such surveys, usually the user specifies a region of interest (ROI) on a map using mission planner software with graphical user interface (GUI). The parameters such as image resolution and overlap are also defined by the user according to the application requirements [106, 107]. The planning system generates a trajectory satisfying the input constraints [108]. The UAV then flies above the given target region according to the generated plan and records camera images so that each point in the region is visible in at least one of the images. The problem of finding such trajectory for a general mobile robot has been an active part of research in robotics. As mentioned in Chapter 2, coverage path planning is the problem of finding a trajectory for a mobile robot equipped with a sensor such that all the points in the target area is swept by the sensor footprint. This task is important in many robotics applications such as inspection, surveillance, de-mining, search and rescue, agriculture and etc.

As discussed in section 2.2, many algorithms have been developed to solve the coverage path planning problem, but the area to be covered is assumed to be uniformly interesting and consequently the robot moves within a specific distance from the surface and sweeps the entire



(a) Searching for trees in a desert-like environment

(b) Aerial sampling of fields with particular crop.

Figure 4.1: Real environments used in our simulations

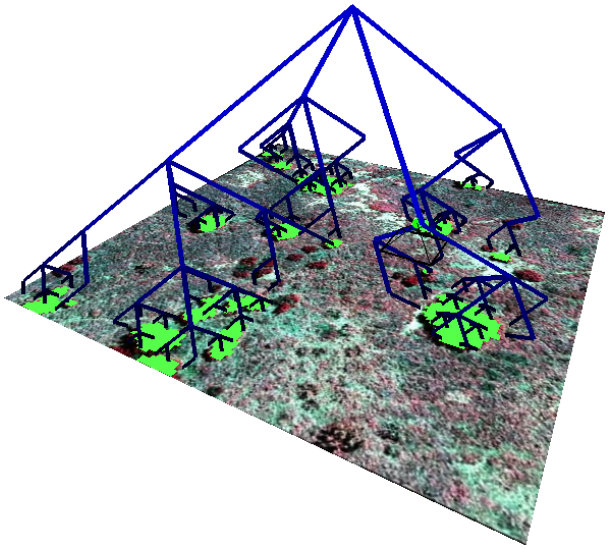


Figure 4.2: Simulated environment for area coverage with UAV. Interesting regions are coloured green. The lines show the interesting branches of the coverage tree.

region. However, in many applications, as a result of non-uniformity in the environment (Fig. 4.1), different parts of the target area can be covered with different resolutions, for example, by flying at different altitudes. This may allow the path planner to produce shorter paths due to the fact that the sensor footprint sweeps a bigger area as the distance between the sensor and the target surface increases [109]. In many real-world applications the distribution of interesting sub-areas is not known in advance. But we may be able to use the available data to classify a section as possibly interesting or uninteresting. At high altitude we can decide online whether there is a need to cover a sub-region from closer viewpoints or not.

We achieve non-uniform coverage by using a tree structure similar to [91]. In this tree each node covers a section of the area with a specific resolution. The resolution increases as one traverses down the tree so that a region covered by a parent node is completely covered by its children nodes with some higher resolution (see Figure 4.2). By visiting some subsets of the nodes of this tree, the entire area is covered but with non-uniform resolution. In our method, the UAV visits the tree nodes (based on the strategies presented in section 4.2) so that the resolution requirement is satisfied, i.e. it must visit the leaf nodes (highest resolution) in the interesting regions. Therefore the robot travels in the environment visiting nodes at different depths of the tree in order to accomplish the coverage task. Four different strategies are presented to cover the area by traversing the coverage tree. All strategies are complete, i.e. they will cover the target area up to the required resolution for each sub-region. At the same time, each strategy has features which might make it the better choice in different situations.

Note that, if the MAV is capable of very stable hover, instead of changing the height to modulate the observation resolution and region, we could rely on a zoom camera and maintain the altitude while flying in the x-y plane. However, apart from the effect of weight and power

consumption of zoom cameras[1] on flight time, we do not restrict the application of the proposed method to image acquisition only and consider the tasks, such as crop-dusting, where the UAV is required to act on the target regions from close distance (see section 1.1).

4.1 Coverage Tree

Let us assume that the target area A is $m \times m$ meters and free of obstacles. Also, for simplicity, assume that the shape of the sensor footprint is a square with length of $l(h)$ where h is the distance of the sensor to the ground. The coverage tree embedded in the metric space is recursively created as follows:

i The root of the tree R, is located at the center of A with a height of $h_R = l^{-1}(m)$ (where $l^{-1}(.)$ is the inverse of the function $l(.)$), i.e. the sensor footprint covers A at the root node.

ii Let h_n and A_n be the height of node n and the area covered by the sensor at node n respectively. Then, for a *branching factor*[2] $b \geq 2$, A_n is decomposed into a $b \times b$ grid with cells of length $l_c = \frac{l(h_n)}{b}$ and therefore $h_c = l^{-1}(l_c)$. For some threshold h_t, if $h_c \geq h_t$ then there is a node for each grid cell, with node n as parent, at the center of the cell and with height h_c, i.e. the sensor covers the grid cell at the child node.

According to the above definition, for a $m \times m$ area A, the root node will be at the center and at a height such that the whole area is covered by the footprint. Then, assuming $b = 2$, the root node will have 4 child nodes forming a 2×2 grid, at lower height $h^{'} = l^{-1}(\frac{m}{2})$ so that each child node covers $\frac{1}{4}$ of A and so on.

In coverage trees, the child nodes cover exactly the same area as the parent does but with a higher resolution which depends on the branching factor b. Also no nodes with a height less than a threshold h_t exist. The parameter h_t determines the lowest height at which the sensor can sufficiently cover a region at the finest resolution. Therefore conventional path planners for area coverage use this height to produce a lawnmower pattern (assuming no obstacles in the area) to sweep the whole area. This simple pattern is equivalent to visiting all the leaf nodes of the coverage tree in an order that yields the minimum path length.

If we know that a region covered by the children of a node is not interesting, it is enough (and possibly more efficient) to cover that region with lower-resolution sensing by only visiting the parent node. Since this information in not available *a priori*, one can visit the parent node first and then decide whether or not the sub-regions covered by each child node are interesting enough to be visited. We assume that the sensor data can always correctly indicate which parts of the footprint are interesting and need closer coverage, i.e., our interestingness sensor has no false negatives. For example, images captured by a camera can be processed to recognize some

[1] Typical zoom cameras, such as FPV-10X [110] consume 2x power and are 3x heavier than a regular camera, e.g. Point Grey FireFly camera [111].

[2] *Branching factor* is usually used to refer to the number of children of each parent in a tree. However, here we use it to denote the parameter b.

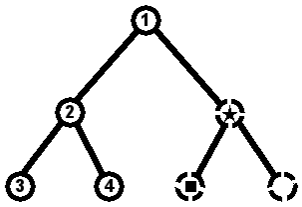


Figure 4.3: Both DF and SH strategies visit nodes 1, 2, 3 and 4 in order. DF then visits the next unvisited node in the tree (star), whereas SH visits its nearby child (square) opportunistically. If (square) is interesting the rest of the children are visited. In case it is uninteresting the UAV visits the parent node (star) recursively.

particular vegetation that makes the green parts in Figure (4.1b) interesting and the red parts uninteresting.

Here we try to keep the notion of interestingness as general as possible. For instance, there does not have to be an explicit part of the sensor data that shows the interestingness of a region directly. It can be the result of complex probabilistic reasoning that leads to the need for closer coverage of a region. For example, if the goal of the coverage is to find people in an unknown area, regions with buildings/structures are more interesting than bushes, and as the UAV descends in the coverage tree, direct people recognition can be used as the interestingness metric. Similarly, when the UAV is mapping the underlying terrain, regions with more uncertainty or local variation in height can have priority for closer coverage than the rest of the area. In case of false positives on interestingness, path length may be increased but coverage is still guaranteed.

The branching factor b controls the rate of increase in coverage resolution between a parent node and its children. A large branching factor will produce a coverage tree composed of nodes with coarse coverage at high altitudes and children that are close to the surface of the area. This configuration of the tree will be practical if the estimation of the interestingness is very accurate. Otherwise a smaller branching factor will be more viable.

The shape of the sensor footprint is assumed to be square to simplify the systematic decomposition of the area. If the footprint of a sensor is in another shape e.g. circle, then one can use the largest square that fits inside the footprint and use it as the square footprint assumed in coverage trees. For instance, when the sensor is a camera with wide angle of view lens, the largest square in the middle of the undistorted image is used as the effective footprint (see for example [112]). Both the sensor footprint and the target area can have an arbitrary shape and the only requirement is a method to decompose the area into a number of footprint shapes with minimum overlap. Everything else about the coverage tree remains the same.

In case of a rectangular target area A, we construct the coverage tree for the smallest square that contains A. Then we prune every node k for which $A_k \cap A = \emptyset$, i.e. the sensor footprint at node k does not intersect with the target area. Therefore we can accommodate non-square areas as well.

Note that if the environment has obstacles which prevent the UAV from flying close to the surface, the corresponding nodes will not be feasible and can be removed from the tree. In this way we can easily relax the assumption that no obstacles are present in the area.

The following section describes three proposed strategies to traverse a coverage tree.

4.2 Basic Traversal Strategies

As mentioned earlier, one way to cover the area is the lawnmower pattern at h_t in which all the leaves of the coverage tree are visited. However one can start at a higher altitude in the tree and visit the lower nodes if they happen to be interesting. In this case the range of the sensor should also be considered when choosing the highest altitude of the sensor, i.e. the robot should not go to a height at which the sensor can no longer distinguish the interesting child nodes from uninteresting ones. Let us call this highest level in the tree l_{max} which refers to a depth in the tree. Note that all the following algorithms use the procedure $Visit\_Refine$ in Algorithm 2 in which the UAV visits the input node and saves all the interesting children.

Algorithm 2 Visit a node and prune the uninteresting children and keep the rest

```
1: procedure VISIT_REFINE(Node n, Container c)
2:     Visit(n)
3:     c.Insert(Children of n that needs visitation)
4: end procedure
```

We propose 3 approaches to traverse the coverage tree:

Breadth-First (BF) In this strategy the UAV starts at l_{max} and visits all the nodes in that level by performing a lawnmower pattern. Upon reaching each node, the corresponding child nodes are labelled as interesting or uninteresting. After the last node of l_{max} is visited, the UAV descends to the next depth in the tree and visits all nodes that were marked interesting in the previous traverse. Again all children are labelled. For path planning at each level we use a 2-approximation TSP tour. This process repeats until there is no node in the next depth of the tree.

By each call of $Visit(node)$, a new goal location is sent to the UAV flight controller. Hence, here we are interested in the computational complexity of the algorithm between consecutive calls of $Visit\_Refine$. Assuming the number of grid cells at the finest level if the tree is N, $Visit\_Refine$ is $O(N)$. The most expensive part of BF algorithm is finding the tour which depends on the number of interesting cells at each level of the tree, but in the worse case it has a complexity of $O(NlogN)$.

Depth-First (DF) In the DF strategy, similar to BF, the UAV starts performing the lawnmower trajectory at l_{max}. However, after visiting each node, the UAV descends to the next depth to visit all interesting children (if any). This behaviour is repeated recursively upon visiting each child node (see Fig 4.3).

Algorithm 3 Breadth-first coverage path planning

```
1: procedure BF
2:     nodeQueue ← Lawnmower(l_max)
3:     while nodeQueue NOT empty do
4:         node ← nodeQueue.dequeue()
5:         Visit_Refine(node, nextNodeQueue)
6:     end while
7:     if nextNodeQueue NOT empty then
8:         nodeQueue ← Tour(nextNodeQueue)
9:         nextNodeQueue ← ∅
10:        Goto 3
11:    end if
12: end procedure
```

Algorithm 4 Depth-first coverage path planning

```
1: procedure DF
2:     nodeStack ← Lawnmower(l_max)
3:     while nodeStack NOT empty do
4:         node ← nodeStack.pop()
5:         Visit_Refine(node, nodeStack)
6:     end while
7: end procedure
```

The most expensive part of DF is $Visit\_Refine$ which is $O(N)$.

Shortcut Heuristic (SH) This strategy is the same as DF with one difference: Assume the UAV has visited a node and the next node to visit, n_{next}, is located at some height above the current node. According to DF the UAV flies directly to n_{next}. However in SH, the UAV visits the nearest child ($n_{nearest}$) of n_{next}. Now there are two possible situations: $n_{nearest}$ is interesting or it is uninteresting. In case it is interesting, we recursively visit all other unvisited children of n_{next} (see Fig 4.3). Note that we do not have to visit n_{next}. Alternatively if the child node is uninteresting, the UAV visits the parent of $n_{nearest}$.

In terms of complexity, $Path$ is $O(logN)$ since one can ascend up in a tree to find a path from a node to an ancestor. Therefore SC has a complexity of $O(N)$

Analysis In the Breadth-First strategy, the whole area is covered initially with coarse resolution. The paths at higher levels are relatively short since the footprint of the sensor sweeps a larger area. Then the sub-regions that might be interesting in the next levels are determined and visited. It is worthwhile to note that this strategy provides a complete but coarse coverage of the whole area after a short time and gradually provides higher resolution coverage of the interesting regions. Previous work has assumed that such initial coarse coverage is available to be used as input to a planner [113, 114, 115], but our approach is a generalization of this 2-level method.

Algorithm 5 Shortcut-heuristic coverage path planning

```
1: procedure SC
2:     nodeStack ← Lawnmower(l_max)
3:     while nodeStack NOT empty do
4:         node ← nodeStack.pop()
5:         if node.isWaiting then
6:             if node has interesting descendent then
7:                 nodeStack.push(Children of node that need visitation)
8:                 Goto 3
9:             else
10:                Visit_Refine(node, nodeStack)
11:            end if
12:        else
13:            if node.depth < current depth then
14:                nodeList ← Path(node, closest descendent of node to current position )
15:                Mark nodes in nodeList as waiting
16:                nodeStack.push(nodeList)
17:                Visit_Refine(nodeStack.pop(), nodeStack)
18:            else
19:                Visit_Refine(node, nodeStack)
20:            end if
21:        end if
22:    end while
23: end procedure
```

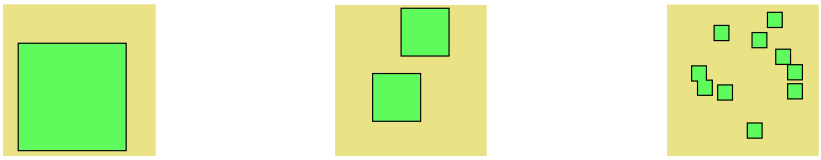


Figure 4.4: Sample synthetic environments that are generated for the first experiment. The squares show interesting areas.

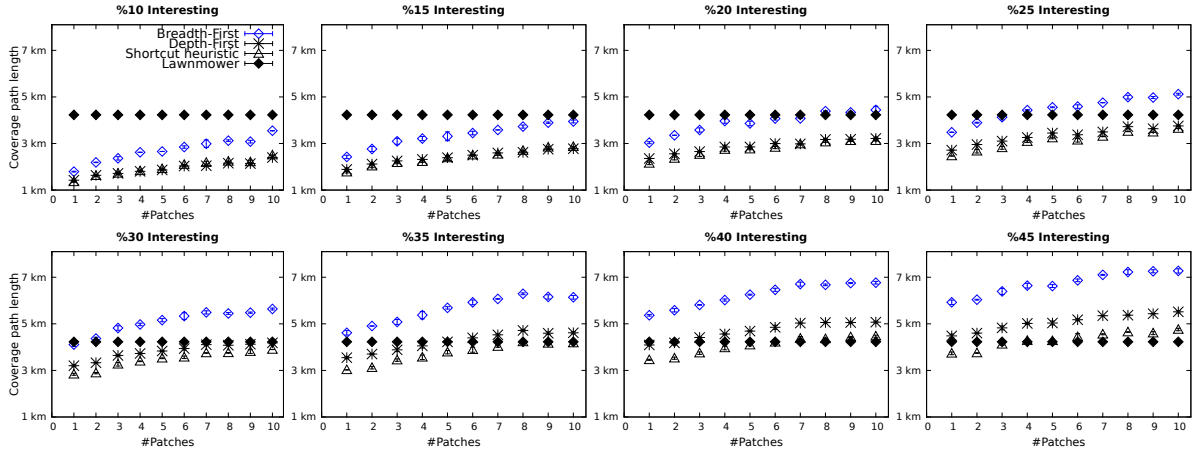


Figure 4.5: Results of the simulations: each graph shows the average length of the coverage paths generated by each strategy for different distribution of interesting regions.

The Depth-First and the Shortcut Heuristic, on the other hand, provide the finest necessary coverage of a sub-region and then move on to the remaining parts of the area. Consequently, with DF and SH, the UAV never goes back to a previously visited region for higher resolution coverage. Intuitively, BF provides a complete but low resolution coverage as early as possible - an "anytime" strategy - while BF and SH provide the high resolution coverage in the shortest time. In many applications interesting regions form big patches, e.g. see Fig (4.1b). In such situations, the probability that one node is interesting provided a sibling node is interesting, will be high. This *locality* property is exploited by the Shortcut Heuristic. This causes the UAV to remain close to the lowest altitude in the interesting regions and unlike the Depth-First strategy, it eliminates unnecessary visits to the corresponding parent nodes.

4.2.1 Experiments and Results

We performed two different sets of experiments in simulation. In the first set the interesting regions are generated at random from a known distribution. In the second set, the interesting regions are identified in images of natural terrains.

Synthetic Environment

In these experiments an 128×128 m^2 area, free of obstacles, is considered (Figure 4.4). The simulated UAV is equipped with a sensor pointing downwards with $l(h) = h$, i.e., the field-of-view of 90°. In order to examine our methods in many environment configurations, two parameters

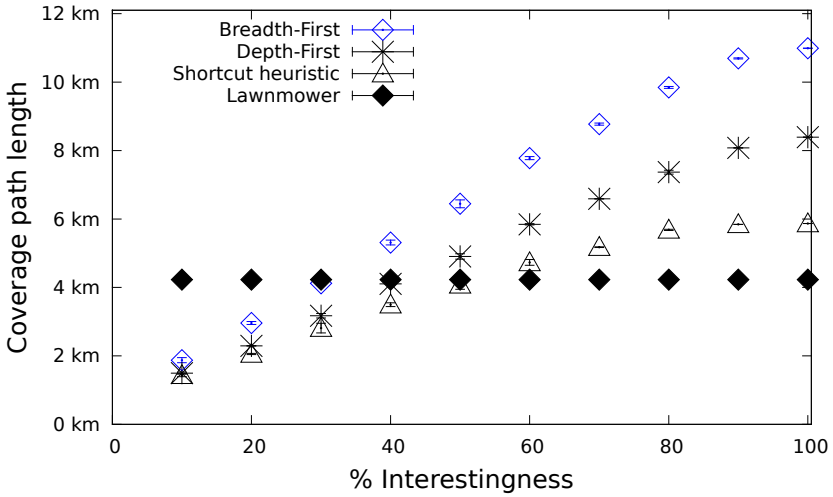


Figure 4.6: Results of the experiments with a single patch.

are used to synthetically generate distributions of interesting regions: p, the percentage of the whole area that is interesting and c, the number of interesting segments. We assume that the interesting regions have the same size. For each pair of (p, c), $p \in \{10, .., 100\}$ and $c \in \{1, .., 10\}$, 10 random environments are generated and the four strategies (BF, DF, SH, lawnmower) are used to cover the area. For the traversal strategies we use $b = 2$, $l_{max} = 1$ and coverage tree with maximum depth of 5. For each strategy, the average of the total distance that the robot travelled is used as a performance metric.

Figures 4.5 and 4.6 show the results of these experiments. In all experiments the variance was below 189 m (the error bars are very small and difficult to see in the figures).

The results indicate that if the interesting portion of the environment is less than 20% and forms few patches, (almost) all coverage tree strategies perform better than the lawnmower pattern. For example if 25% of the area is interesting then the shortcut heuristic takes a little more than half the length of the lawnmower pattern to cover the area. This means that the UAV could finish the coverage in half the time. As the number of interesting regions increases, the BF strategy becomes inefficient quickly. This is because the BF strategy visits all interesting nodes at the first tree level, then descends to the next level and revisits all nodes again and so forth. This also means uninteresting areas between interesting patches are revisited at higher resolutions. The revisiting and switching between patches of interest increases the travel cost quickly. In contrast DF and SH perform one high level (short distance) traverse and immediately exploit the available information to visit only interesting areas.

An area with large segments of interesting regions (i.e. small c) results in a high locality property in which SH outperforms the lawnmower and DF strategies. However, as the distribution of interesting regions varies from few large segments to a more uniform one, i.e. c becomes large, the locality property decreases and lawnmower becomes superior to both DF and SH. This is due to the fact that the lawnmower pattern never revisits a location.

Strategy	Environment 1	Environment 2
Breadth-First	0.86	1.04
Depth-First	0.56	0.81
Shortcut Heuristic	0.59	0.69
Lawnmower pattern	1	1

Table 4.1: Ratio of coverage paths lengths generated by the coverage tree strategies to the length of the lawnmower-like pattern.

Figure 4.6 shows the path length of each strategy for different percentages of interesting area in the environment. Only one single patch is used in these experiments. As the interesting patch grows, we expect that the Shortcut Heuristic keeps the UAV close to the ground to cover the interesting neighbourhood which yields a better performance compared to Depth-First. As the fraction of interesting area grows, the Shortcut Heuristic is eventually outperformed by the lawnmower which is optimal when 100% of the environment is interesting.

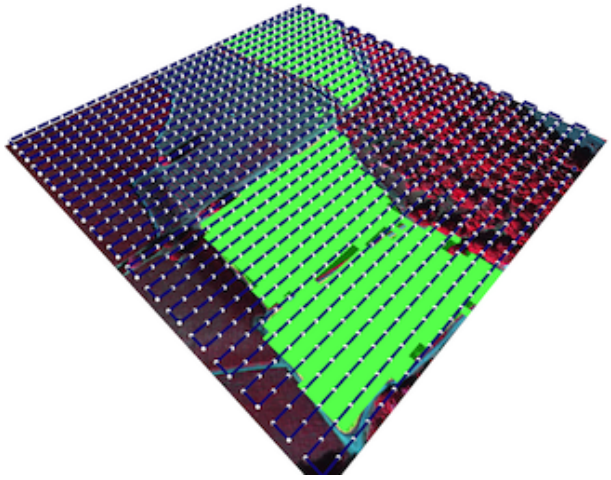


Figure 4.7: The Lawnmower-like pattern is used as the baseline approach. The area is covered with a uniform resolution.

Simulated Environment

We also used example images of natural environments for our simulated UAV. Figure 4.1 shows two sample environments. In Figure 4.1a the bushes and Figure 4.1b fields with a particular vegetation are considered interesting respectively. The interestingness detector checks if the color of a pixel falls in a certain color range. The area shown in Fig.4.1a represents an environment with a number of small interesting segments (Environment 1) whereas Figure 4.1b demonstrate an area with few large continuous interesting regions (Environment 2). We ran the same experiments as described in the previous section on these two environments, with results shown in Table 4.1.

The results indicate that the lawnmower pattern performs worse than the DF and SH in these maps. This is because the majority of the area is uninteresting and the two mentioned

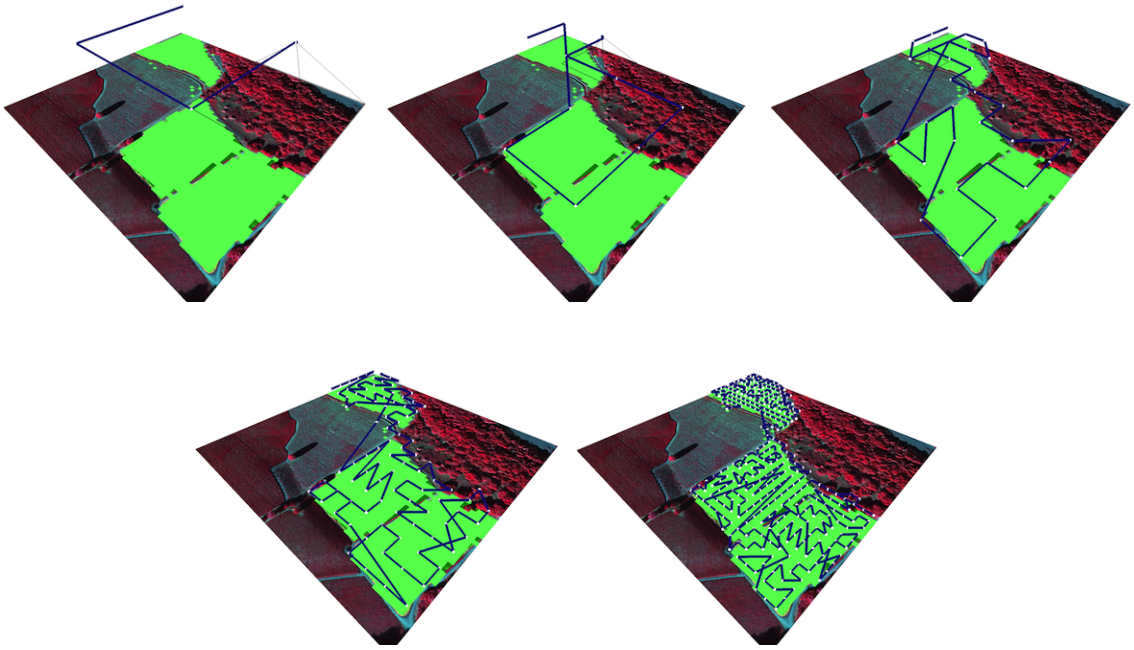


Figure 4.8: Sample real environments are simulated in the second experiment. These figures show the coverage plan (generated by TSP 2-approximation) for each level of the tree in the BF strategy. At the first level with no prior interestingness information, coarse lawnmower pattern is performed. Light green (light grey) pixels are interesting.

strategies are able to behave accordingly. In the first environment (Fig. 4.1a), Depth-First performs a little better than the Shortcut Heuristic due to the low locality. However in the second map (Fig. 4.1b) the large interesting regions are covered more efficiently with the SH strategy compared to DF.

4.3 Hilbert-based Traversal Strategy

In this section we propose a new online traversal algorithm based on the Hilbert space filling curve (SFC) that improves the performance of the method through enhancing the local coverage plans and exploiting locality of the interesting regions. The new method out-performs all the previous strategies. Preliminary results show that with some distributions of interesting regions, Hilbert-based strategy generates shorter coverage paths compared to the regular Lawnmower pattern, independent of the total interesting area size.

Spires *et al.* applied the Hilbert SFC to exhaustive geographic search with a swarm of robots [116]. Starting at random locations in the region, each robot follows the Hilbert curve and performs the sensing action at each grid cell. A robot has accomplished its task upon reaching the starting cell of another robot. However, in contrast to our approach, they only rely on a single order of Hilbert curve. SFCs have also been used to generate tool paths for CNC machines [117]. However in these applications, the regions of interest are known *a priori*.

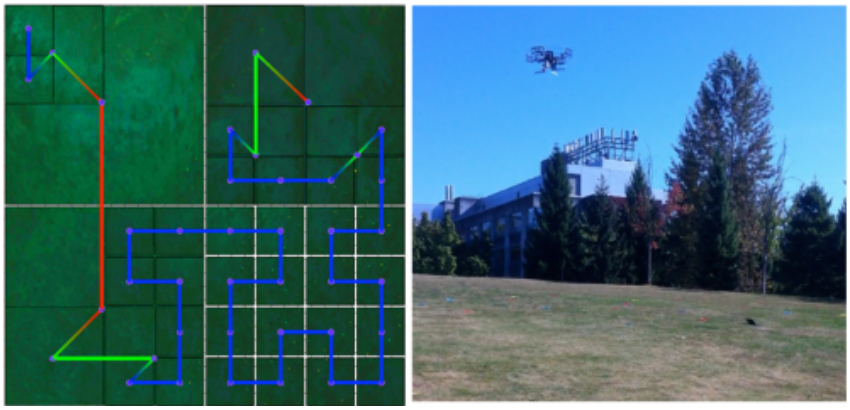
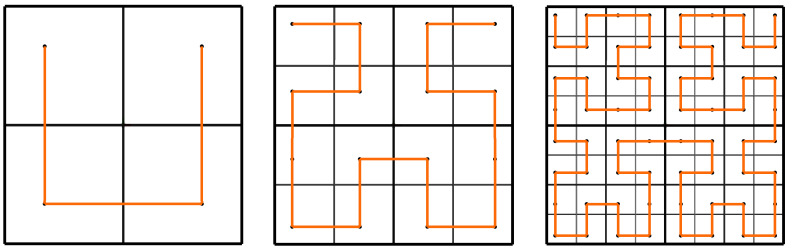


Figure 4.9: Non-uniform coverage with real UAVs. Interesting regions (bottom right of the left figure) are covered with higher sensor resolutions than uninteresting area.



(a) 1st order (b) 2nd order (c) 3rd order

Figure 4.10: Hilbert curves of 1st, 2nd and 3rd order.

Here our goal is to reduce the redundancy in coverage and and exploit the locality of the interesting targets better. Consequently, we need a systematic ordering π of the nodes at each level so that two nodes appearing beside each other in π cover neighbour grid cells. We propose to use Hilbert space filling curve to impose such an ordering on the nodes. The Hilbert curve, H, is a fractal space-filling curve shown in Figure 4.10. The Hilbert curve of Nth order, denoted by H_N, fills a grid of $2^N \times 2^N$. It is a useful curve since it provides a mapping between 1D and 2D (nD) spaces which preserves locality adequately well [118]. Informally speaking, it means that a robot following a Hilbert trajectory stays close to the places that it has recently visited. This property exists in the Hilbert curve of any order.

We use the Hilbert curve to impose an ordering on the nodes at each level of the coverage tree. Note that H_1 provides an ordering on the 2×2 grid which corresponds to the nodes at depth 1 of the coverage tree when $b = 2$ (see Figure 4.12a, 4.12b) [3]. Similarly, H_2 and H_3 impose an ordering on nodes at depth 2 and 3 of the coverage tree (see Figure 4.12c, 4.12d). More generally, if $b = 2^s$, $H_{d \times s}$ can be used to sequentialize the nodes at depth d. Therefore a coverage tree can be seen as a tree with 1-D ordered nodes at each level as shown in Figure 4.11.

[3]We define the top left end of the curve as start and the top right end as the end of the curve.

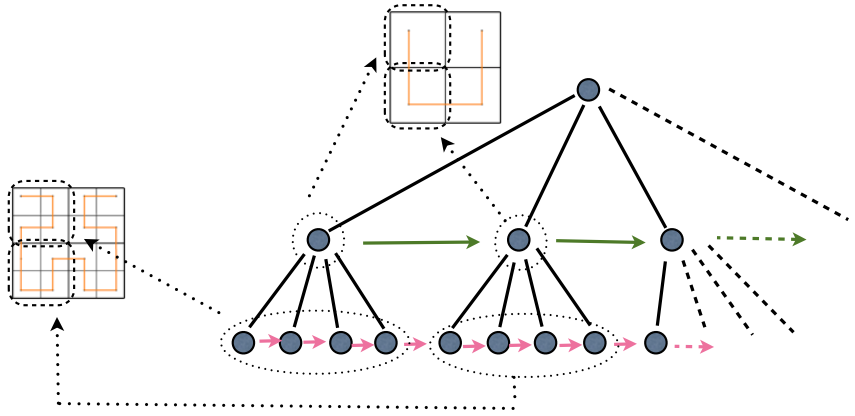


Figure 4.11: Coverage tree with Hilbert-based ordering of nodes at each depth and its relationship to grids in different resolutions

Our approach to non-uniform coverage is to visit the nodes based on H. Upon visiting an interesting node, we increase the coverage resolution by descending to the next depth or stay in the same depth of the tree. If the visited node was uninteresting, the resolution of the coverage is decreased by going one level up (decrease depth) in the tree. In either case, we continue to visit the nodes based on the appropriate Hilbert ordering. The intuition is that, when an interesting region is observed, one can opportunistically assume that the next node will also be interesting due to the locality preserving feature of the Hilbert curve, i.e. if the interesting region forms a big patch, with high chance, the next node will remain in the patch and the overhead of visiting the parent nodes is avoided.

Our proposed traversal strategy is formally presented in Algorithm 6. It returns the next node that should be visited by the UAV according to the current state of the tree.

In this algorithm, $Children(n)$ returns the children of node n and $Parent(n)$ returns the parent of node n. $Depth(n)$ returns the depth of the tree at which node n exists. The function $NeedVisit(n)$ returns $true$ unless visiting n is not necessary (note that $NeedVisit(null) = false$ and $NeedVisit(leaf) = false$ if $leaf$ has already been visited or classified as not interesting). This is the case when there is an ancestor node of n that is classified as uninteresting. Also, when none of the children of node n need to be visited then $NeedVisit(n) = false$. This function is required to prune out nodes when all their interesting descendant leaves have been visited. Additionally, $Next(n)$ returns the next node of the tree (or $null$ if n is the last node) at the same depth of n in the order imposed by $H_{Depth(n)}$. Finally, first (last) child of a node is defined as the child node that appears first (last) in the corresponding Hilbert ordering.

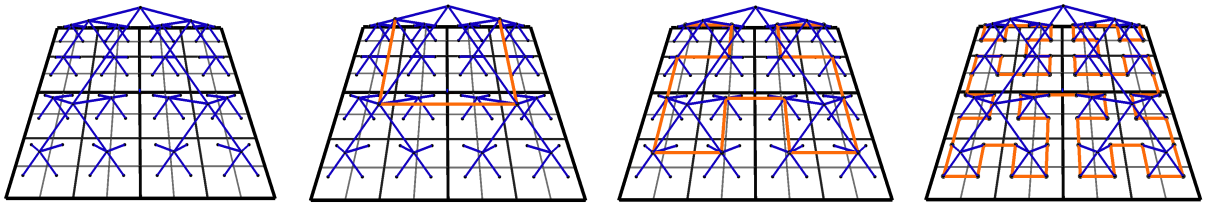
Using this algorithm, the traversal starts by visiting the leaf node on the top left corner of the environment (line 2). Upon visiting a node n, if it is interesting and at least one of its children needs to be visited, it visits the interesting children according to H (lines 4-5). In case the node is uninteresting, it ascends to the parent node provided that n is not at the maximum height (lines 6-7). Then, lines 9-20 make sure that we prune the nodes that do not need to be visited. Here, as the nodes are being skipped, if the last child(ren) of a node is pruned (because it was classified as uninteresting when visiting the parent node) we go one level up to adhere to

Algorithm 6 Hilbert-based coverage path planning

```
1: procedure HI(T)                    ▷ T: coverage tree with the implicit order of the nodes
2:     Visit(n, the first leaf of T)
3:     while n <> null do
4:         if Interesting(n) AND NeedVisit(Children(n)) then
5:             n ← first child of n
6:         else if !Interesting(n) AND Depth(n) > 1 then
7:             n ← Parent(n)
8:         end if
9:         if NeedVisit(n) then
10:            Visit(n)
11:        else if n is the last child of Parent(n) then
12:            n ← Parent(n)
13:        else
14:            if Next(n) <> null then
15:                n ← Next(n)
16:                Go to 9
17:            else
18:                n ← null
19:            end if
20:        end if
21:    end while
22: end procedure
```

the general strategy. Figure 4.14 shows such a situation. At the top left, the 2×2 red square, s is pruned when the UAV visits the node marked by the triangle. Since s is the last child of the triangle node (by the Hilbert ordering), after covering the rest of the children, the UAV skips s and goes one level up in the tree and hence takes the route indicated by circles (which is a jump from depth 4 to depth 2). Note that we are always progressing in the Hilbert curves at any depth and we visit each node at most once. Given that we assume a maximum depth for the coverage tree, our algorithm terminates.

Let us assume that the number of grid cells at the finest resolution (leaves of the coverage tree) is k. The creation of the tree at the start of the mission takes $O(k^2)$. The memory space of each node in the tree is constant which leads to $O(k^2)$ total memory consumption. At each



(a) Coverage tree (b) H_1 (c) H_2 (d) H_3

Figure 4.12: Coverage tree and Hilbert curve. H_N is used to impose ordering on nodes at depth N.

waypoint, the quadrants of the sensed image are classified as being interesting or uninteresting ($O(1)$). Then, Algorithm 6 is executed, iterating over some nodes to find the next goal waypoint. At each iteration of the loop $NeedVisit(n)$ is the most expensive operation. It needs traversing to an ancestor node ($O(log(k))$) or calling $NeedVisit()$ on each child ($O(k^2)$). Note that traversing up to find an uninteresting ancestor is executed only in the first call of $NeedVisit(n)$ and not in the recursive calls. This is because $NeedVisit()$ checks all the descendants from top of the subtree to the bottom and consequently the uninteresting ancestor is found either in the first call of $NeedVisit(n)$ or in the recursive calls before its descendants. The loop runs at most for $O(log(k))$ since at each iteration we go to the parent node (either directly or after iterating over all the 4 children) or have started a sequence of descending moves which ends in an unvisited interesting leaf node. Therefore the computational cost of the planning at each waypoint is in the order of $O(k^2log(k))$ (which is $O(Klog(K^{1/2}))$ where K is the number of nodes in the tree). Note that this complexity is the worst case scenario in an abstract situation. In a concrete case, the loop runs very few times to find the next goal.

4.3.1 Experiments and Results

In the following sections we discuss the results of the simulations and field trials.

Simulations

In these experiments, a 128×128 m^2 area, free of obstacles, is considered. The simulated UAV is equipped with a sensor pointing downwards with $l(h) = h$, i.e., a field-of-view of 90°. In order to generate many different configurations of interesting patches, we introduce two parameters to represent a distribution of interesting regions: p, the percentage of the whole area that is interesting and c, the number of interesting segments. We assume that the interesting regions are rectangles (of any form) but have the same size. For instance, $(40, 3)$ refers to all environment configurations in which 40% of the whole area is interesting and in form of 3 rectangles with equal area. For each pair of (p, c), $p \in \{10, .., 90\}$ and $c \in \{1, .., 4\}$, 10 random environments are generated and the four strategies (H, DF, SH, lawnmower) are used to cover the area. For the traversal strategies we use $b = 2$, $l_{max} = 1$ and coverage tree with maximum depth of 5. For each strategy, the average of the total distance that the robot travelled is used as a performance metric. The results are shown in Figure 4.13. In all experiments the variance was below 192 m (the error bars are very small and difficult to see in the figures).

The graphs indicate that our new Hilbert based traversal of the coverage tree outperforms Depth-First and Shortcut strategies in all the tested configurations of the environment. Moreover, the cost of the Lawnmower pattern which is independent of interestingness, is almost always higher than our approach. Consequently, if the estimated distribution of interesting regions falls in the wide range of the tested configurations, using our method, one can perform the coverage task in shorter time compared to Lawnmower. Note that we have also computed the lower-bound of the trajectory length which is obtained when the interesting regions are fully known and covered by Lawnmower pattern. Comparing the Hilbert strategy with the lower-

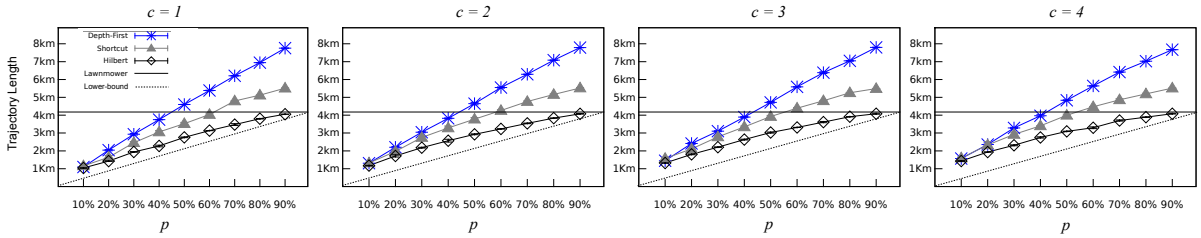


Figure 4.13: Results of the simulation experiments in different environment configurations (p : ratio of the whole area that is interesting, c : number of patches) (See section 4.3.1).

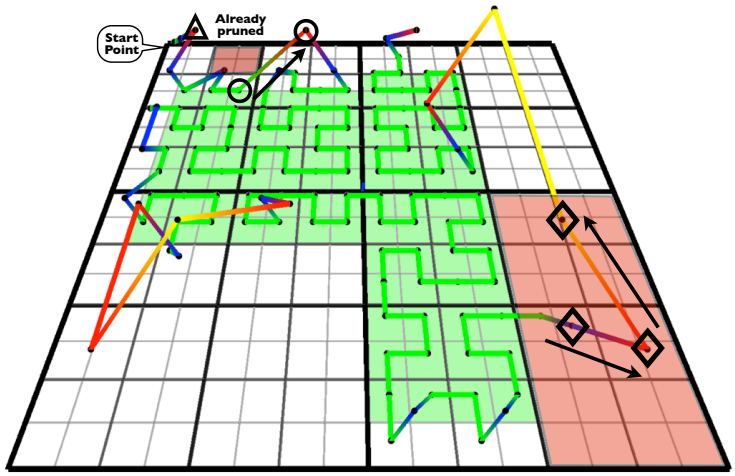


Figure 4.14: Final coverage plan in simulation. The color of the trajectory changes with height. Red squares on bottom-right are pruned when the UAV ascends to visit the nodes indicated by diamonds. (best seen on screen)

bound, we can see that our Hilbert coverage strategy is performing well considering that it is not relying on any *a priori* information.

In the Hilbert-based traversal of nodes at a specific depth, the UAV flies from one node to the other while observing grid cell neighbouring criteria, i.e. it moves from a grid cell to one of its 4 (2 or 3 if on the grid boundary) neighbours. This means that the local coverage paths at one tree level is optimal in terms of path length. Similarly, in the sequence of nodes that Algorithm 6 generates, when the UAV ascends from node n to n', n' is either the parent of n or a sibling of an ancestor of n. Therefore the neighbouring criteria is held at all the levels of the tree. On the contrary, in previously proposed strategies i.e. Depth-First and Shortcut, no such criteria was imposed and a constant ordering among the children nodes was used which was not optimal at different levels. The other property of our new approach is that it exploits the patchiness of interesting regions by travelling between nodes while preserving locality. Both Shortcut and Hilbert-based traversal opportunistically visit an unclassified neighbour of the current interesting grid cell to avoid visiting nodes at higher levels while still being in that interesting section of the environment. However, the Hilbert curve exhibits high locality and as a result the UAV exits the interesting regions very few times, compared to the greedy Shortcut. Figure 4.14 demonstrates

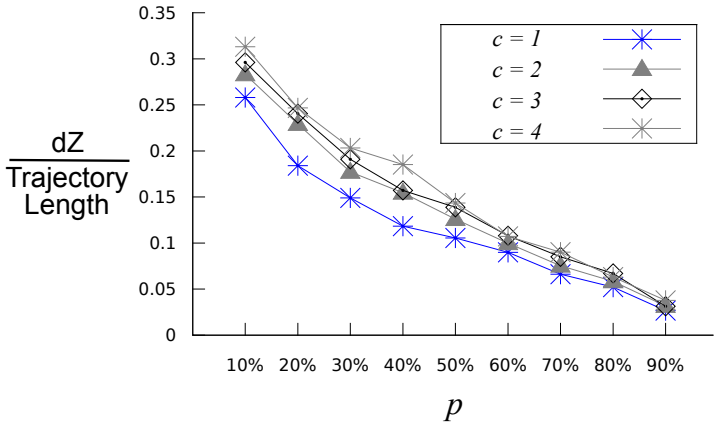


Figure 4.15: The ratio of total displacement along Z-axis to the total length of the trajectory in environments with different number of patches.

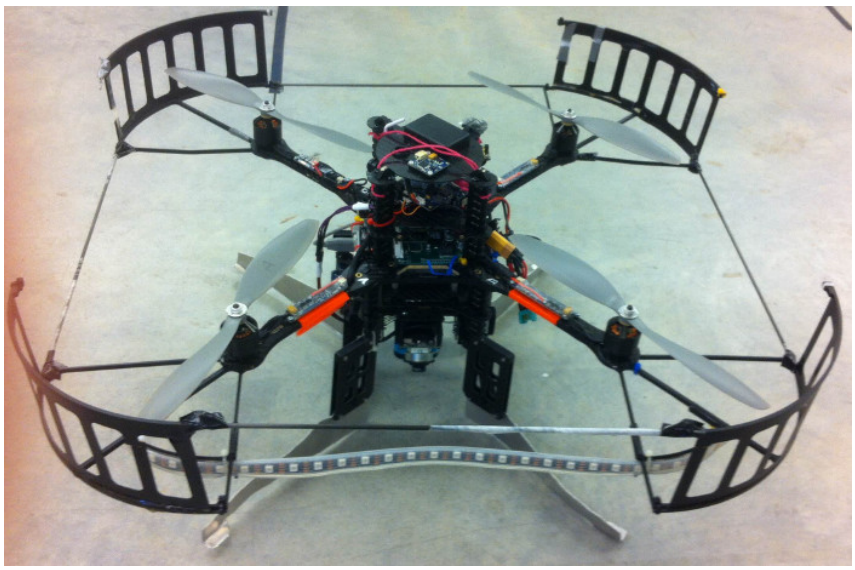


Figure 4.16: AscTech Pelican was used for the field trials.

how the Hilbert curve covers two interesting patches with few crossings of the boundary of the patches.

Figure 4.15 depicts the ratio of mean motion along the Z-axis to the mean of trajectory length. It implies that with low total interesting area, the UAV tends to visit nodes at lower depth of the tree (high altitude). This is due to the fact that uninteresting regions cause the UAV to ascend in the tree, and those sections are consequently sampled sparsely (by pruning out all their uninteresting children nodes). In cases where most of the area is interesting the vehicle stays at leaf level as is suggested by the graph. Moreover, with a single interesting patch in the area it will be more likely that a node at high altitude (low tree depth) is uninteresting compared to the situation where multiple patches (with the same total interesting area as one patch) exist. Hence, in the former situation larger uninteresting segments can be pruned out at once with a series of successive single ascending.

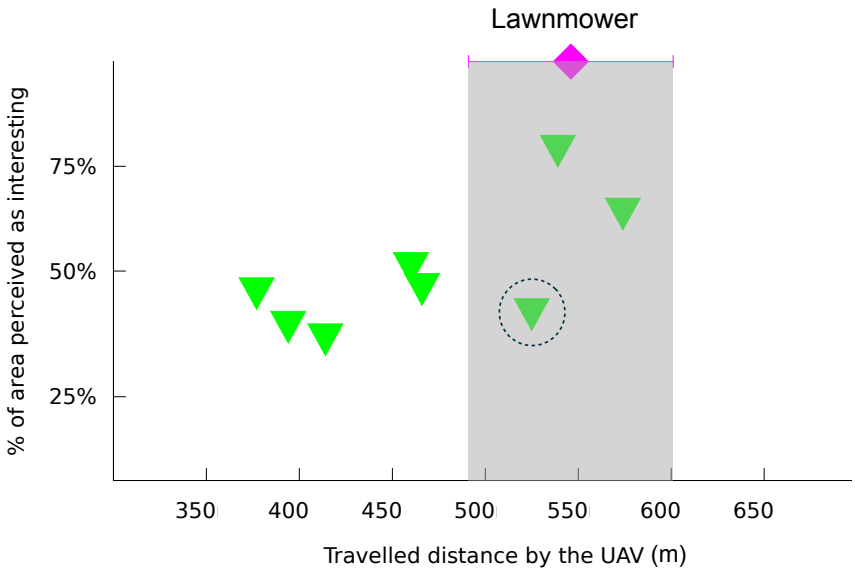


Figure 4.17: Results of the experiments with real UAV. The performance of the Lawnmower pattern is shown in terms of mean and variance of 6 trials. The performance of our method in each trial is reported individually.

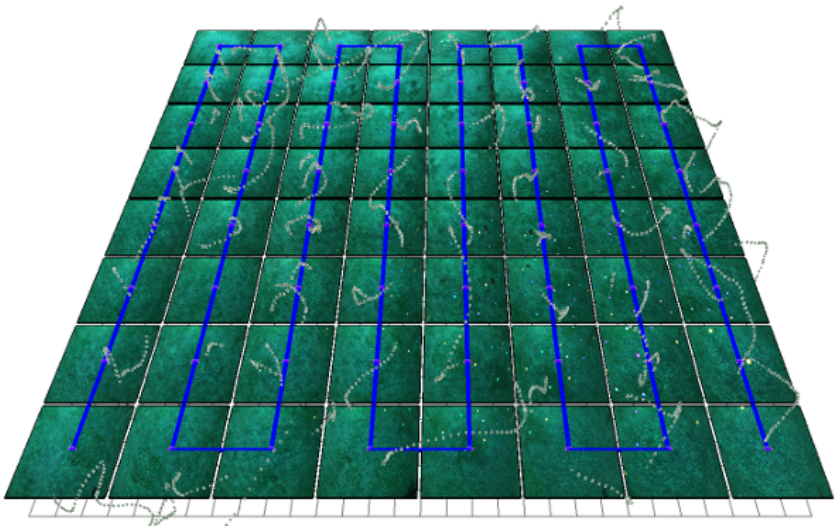


Figure 4.18: The trajectory of the UAV while performing an exhaustive coverage in one of the field trials.

Field Trials

We also performed some experiments in outdoor environment using a Pelican quadrotor from Ascending Technologies. GPS is used to estimate the robot position which is then fed into a PID controller to generate the required velocity commands. The task was to cover an area of 30×30 m^2. A number of frisbees were spread out in the area forming a single patch that occupied about 31% of the target environment as shown in Figure 4.19 with a yellow polygon. Using a gimbal mounted colored camera with 90° field-of-view, we detect frisbees from different heights (Figure

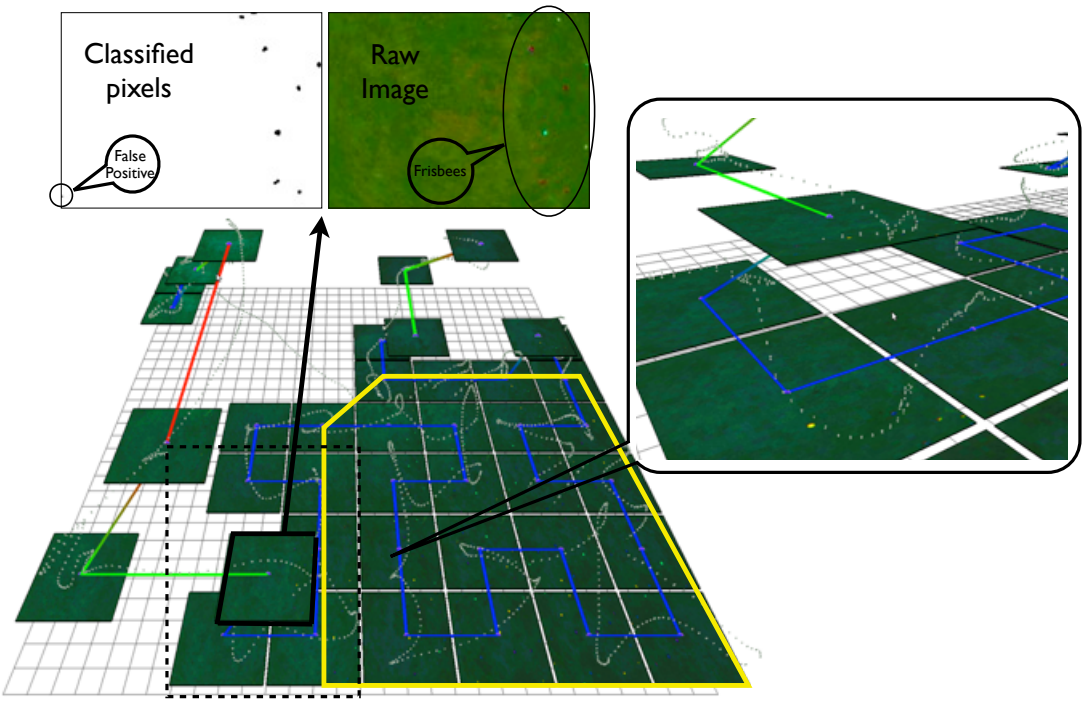
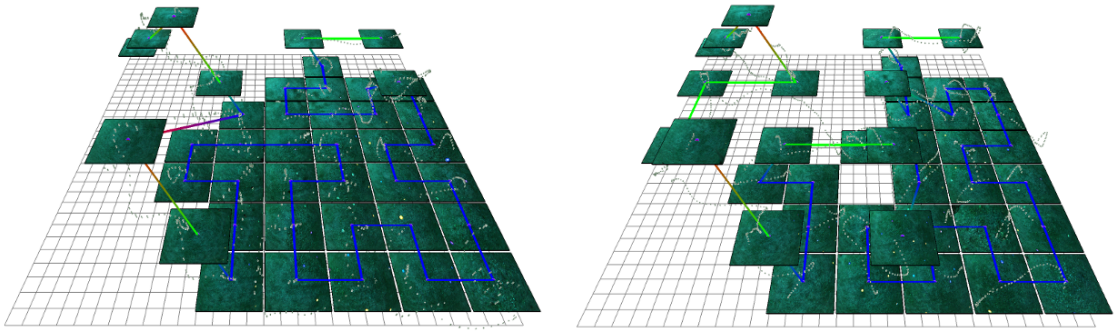


Figure 4.19: The trajectory of the UAV. The yellow line approximately shows the true interesting area (best seen on screen).

4.19). A query region in an image with non-empty intersection with a frisbee is classified as interesting. Due to light changes and other sources of noise, there are false positives in frisbee detection. The experiment was repeated 6 times performing Lawnmower-based coverage (as shown in Figure 4.18) and 8 times with our proposed method. The results are shown in Figure 4.17. In this figure, we report the length of the UAV trajectory along with the overall ratio of perceived interesting cells, i.e. false positives plus true positives.

The mean and variance of the Lawnmower performance are shown in pink. The shaded part indicates the possible performance of Lawnmower with different interestingness ratio of the environment. The variance of the Lawnmower trajectory indicates the amount of noise in the position estimation and control of the robot. Despite that, it can be observed that for situations where the perceived interestingness is below 50% our method generates a shorter trajectory, except in one trial (indicated by a dashed circle). In that trial false positive interesting middle nodes which were apart from each other led to longer trajectory.

Note that as the width m (length) of the target area increases, the length of the Lawnmower coverage pattern grows with $O(m^2)$. On the other hand with our approach, the length of a trajectory that ascends up to eliminate (classify as uninteresting) a node N will be of $O(m')$ where m' is the width (length) of $A_{parent(N)}$ (sensed area at the parent node of N). This can be seen on the bottom right of the Figure 4.14 where, traveling from the leaf node to its ancestor at depth 1, prunes out (at least) $\frac{1}{4}$ of $A_{parent(N)}$. Therefore as the size of the area grows the performance benefits of traveling to higher nodes will be larger and the difference between Lawnmower and Hilbert-based coverage becomes more clear. Due to lack of suitable flying sites,



(a) The trajectory of the MAV in one of the field trials.

(b) The false positives in the detections at internal nodes resulted in a long trajectory with relatively low high resolution coverage.

Figure 4.20: Example trajectories of the MAV in the field trials.

we could not perform the outdoor experiments in a larger environment and left it for future work.

4.4 Noisy Observations

In the previous sections we assumed that the observations made by the MAV are perfect, i.e. there is no sensor noise. Therefore, the nodes of the tree could be pruned as soon as they were sensed as being uninteresting. However, real-world sensor readings are often distorted by some noise. In our situation, the noise appears as false alarms and missed detections of the interesting cells. Consequently, when the MAV makes a binary observation, we need to consider the possibility of these distortions in the decision of whether or not the sensed region is interesting. We use the ideas from the probabilistic search literature [119, 120] to create a more robust model of the interesting regions.

4.4.1 Bayesian Estimator

Instead of deterministic classification of grid cells to interesting or uninteresting, we define a binary random variable H_c such that $H_c = 1$ if the cell c is interesting and $H_c = 0$ otherwise. Hence, for each cell we maintain the probability of that cell being interesting $Pr(H_c = 1)$. At the start of the mission, this probability is initialized to 0.5, which is equivalent to having no prior knowledge about the interesting regions. The measurements that are taken by the MAV modifies these probabilities as will be described later. Note that, for simplicity, we assume that the random variables are independent form each other.

During the traversal of the coverage tree, the MAV should decide whether to prune a node or not. For this decision, we define a threshold P_{int} such that a leaf node c with $Pr(H_c = 1) < P_{int}$ can be pruned. Also, a non-leaf node can be pruned if all its descendent leaf nodes can be pruned. Setting P_{int} to a small probability value results in less number of truly interesting cells being pruned by the MAV, whereas with less conservative P_{int}, more interesting cells are likely

to be missed. Nevertheless, a very small value for P_{int} will lead to no pruning and hence an exhaustive coverage of the leaf nodes. For our experiments, we used $P_{int} = 0.3$.

Lets use O_c to denote the observation made at c where $O_c = 0$ and $O_c = 1$ denote negative and positive interesting cell detections. In order to characterize the sensor noise, we define the observation likelihoods as in [121], where the probability of false positive and negative observations depend on the altitude of the MAV:

$$Pr(O_c = 1|H_c = 0) = \alpha_h$$
$$Pr(O_c = 0|H_c = 1) = \beta_h$$

The subscript h in α_h and β_h denote the altitude of the MAV. The false positive and negative likelihoods form the observation model and are obtained empirically for a specific sensor. After each observation O_c, the belief corresponding to the observation region is obtained using the Bayes' theorem:

$$Pr(H_c = 1|O_c) = \frac{Pr(O_c|H_c = 1)Pr(H_c = 1)}{Pr(O_c)}$$

Depending the actual observation value the posterior probability is calculated:

$$Pr(H_c = 1|O_c) = \begin{cases} \dfrac{(1-\beta_h)Pr(H_c = 1)}{(1-\beta_h)Pr(H_c = 1) + \alpha_h(1 - Pr(H_c = 1))}, & O_c = 1 \\ \\ \dfrac{\beta_h Pr(H_c = 1)}{\beta_h Pr(H_c = 1) + (1-\alpha_h)(1 - Pr(H_c = 1))}, & O_c = 0 \end{cases}$$

Note that $Pr(H_c = 1)$ is the prior probability of the observation region being interesting. For the non-leaf nodes we obtain this probability as

$$Pr(H_c = 1) = 1 - \prod_{n \in children(c)} (1 - Pr(H_n = 1))$$

After the belief of an internal node k is updated with the observation O_c, we use Chung's approach [121] to propagate the new information to the children of k:

$$Pr(H_n = 1|O_c) = 1 - Pr(H_n = 0)^{\lambda},\ \lambda = \frac{\log Pr(H_k = 0|O_c)}{\log Pr(H_k = 0)}$$

Therefore, the change in the interestingness belief of an internal node is uniformly propagated to the its children and recursively to all its descendents. Note that we use $Pr(H)$ to denote the belief prior to the current observation.

The probabilistic estimation of interesting cells, as described above, provides a more reliable way to decide whether the MAV should abandon a node or it needs to make more accurate observations. In order to use the above interestingness detection together with the traversal strategies introduced in the previous sections, we only need to replace the deterministic interestingness detection with the probabilistic one that uses the threshold P_{int}. The rest of the algorithms remain unchanged.

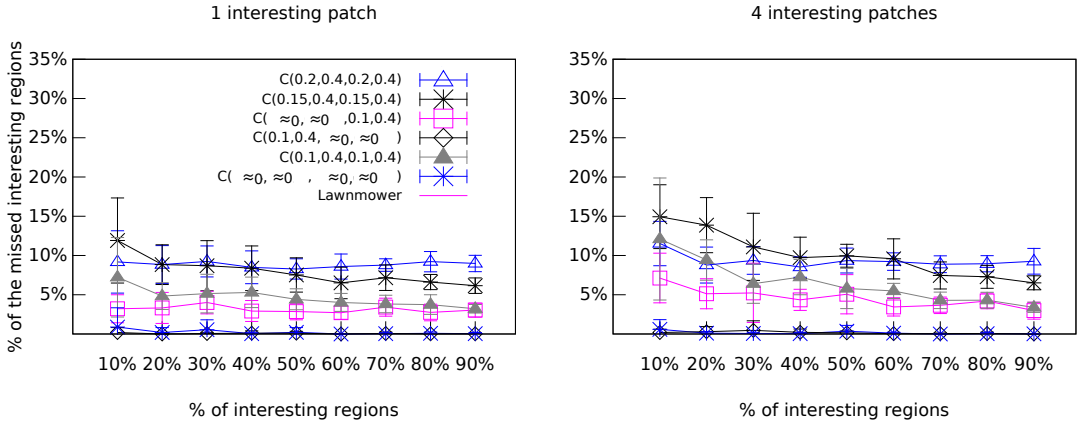


Figure 4.21: The percentage of the interesting regions that was missed by the MAV. The plots show that in all the tested configurations, less than 15% of the targets is missed by the MAV, and the rest of the interesting regions are covered with high resolution. Note that the plots for 2-patch and 3-patch configurations show a similar trend and hence not shown here.

4.4.2 Experiments and Results

We performed a series of simulation experiments to evaluate our approach. The settings of the environment is the same as the previous simulations (see section 4.3.1). However we only use the Hilbert strategy for the tree traversal. In order to investigate the effect of the sensor noise, we used various sensor configurations in the experiments. Also, we assumed that α and β change linearly across different heights. This is denoted by C(α_{h0}, α_{hm}, β_{h0}, β_{hm}) where α_{h0} and β_{h0} are false positive and negative likelihoods at the leaf nodes, and α_{hm} and β_{hm} indicate the same values at the root node. For pruning the nodes, we used $P_{int} = 0.3$, i.e. for a leaf node c, if $Pr(H_c = 1) < 0.3$, then c is pruned and not visited.

Figure 4.21 shows the ratio of the missed target cells to the total true interesting cells. The plot indicates that, as expected, when there are almost no false negative detections, i.e. $Pr(O_c = 0|H_c = 1) \approx 0$, all the interesting cells are covered with high resolution. However, as the rate of false negative observations increases, a larger portion of the targets are missed by the MAV. Using a smaller threshold P_{int} will lead to a more conservative pruning of the nodes and can help to achieve a complete coverage of interesting cells. However by doing so, more falsely interesting cells are visited by the MAV which will increase the cost. The plot demonstrates that in all the tested configurations and $P_{int} = 0.3$, less than 15% of the interesting regions are missed by the MAV.

Due to the noise in the detections, uninteresting regions can be perceived as interesting and covered by the MAV. Figure 4.22 shows the percentage of the environment that was uninteresting but covered with the MAV. With the sensor configuration C(0.2, 0.4, 0.2, 0.4), the observations do not contain enough information to enable confident pruning of the nodes and hence considerable number of truly uninteresting cells are covered with high resolution. Note that as the truly interesting area of the environment increases, less falsely interesting cells are possible. This can be seen in the slop of the plot for C(0.2, 0.4, 0.2, 0.4). It is important to note that, in this plot we

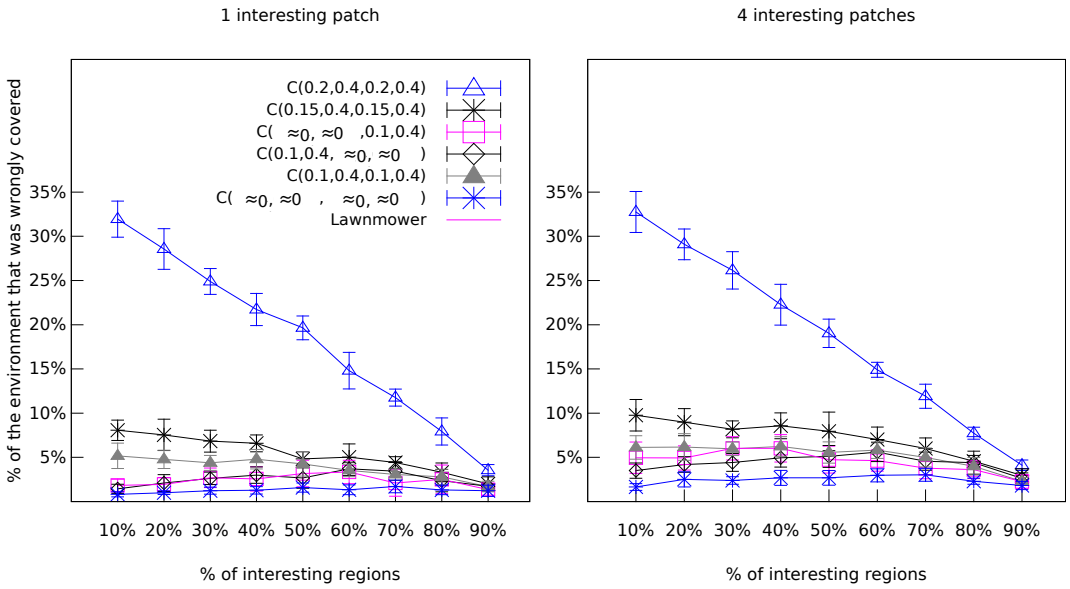


Figure 4.22: The percentage of the environment that was uninteresting and mistakenly covered by the MAV with high resolution. With sensor configuration C(0.2, 0.4, 0.2, 0.4), due to the relatively high noise in the observations, the MAV is unable to prune uninteresting nodes with high confidence and hence more falsely interesting cells are covered with high resolution. Note that the plots for 2-patch and 3-patch configurations show a similar trend and hence not shown here.

also included the cells that were intentionally visited (opportunistic behaviour in Hilbert-based traversal (see section 4.3)) but turned out to be uninteresting. Hence, in configurations where there is almost no sensor noise, some uninteresting regions are still visited (Figure 4.22).

The distance travelled by the MAV in various configurations is depicted in Figure 4.23. As expected, with C(0.2, 0.4, 0.2, 0.4), due to large number of visited uninteresting regions, the travelled distance is higher than the rest of the configurations. On the other hand, when the MAV is able to prune the middle nodes with confidence (e.g. C(≈ 0, ≈ 0, ≈ 0, ≈ 0) and C(0.1, 0.4, ≈ 0, ≈ 0)) large sections of the uninteresting regions are pruned with few observations and leads to less travelled distance. The length of the coverage path increases as the noise in the sensor increases. However, it can be seen that the Hilbert-based traversal is less costly than the exhaustive lawnmower pattern in most of the tested configurations.

In order to understand the behaviour of the MAV in the presence of the sensor noise, we have included some snapshots of the simulation (Figures 4.24-4.26) that demonstrates the final trajectory of the vehicle:

Highly accurate sensor. When the observations are very accurate, the MAV is able to prune large sections of the uninteresting regions with a few measurements (see Figure 4.24a). Therefore, the uninteresting regions are covered coarsely (pink and yellow parts of the trajectory). Also, the interesting cells are completely covered with high resolution and there is no missed cells.

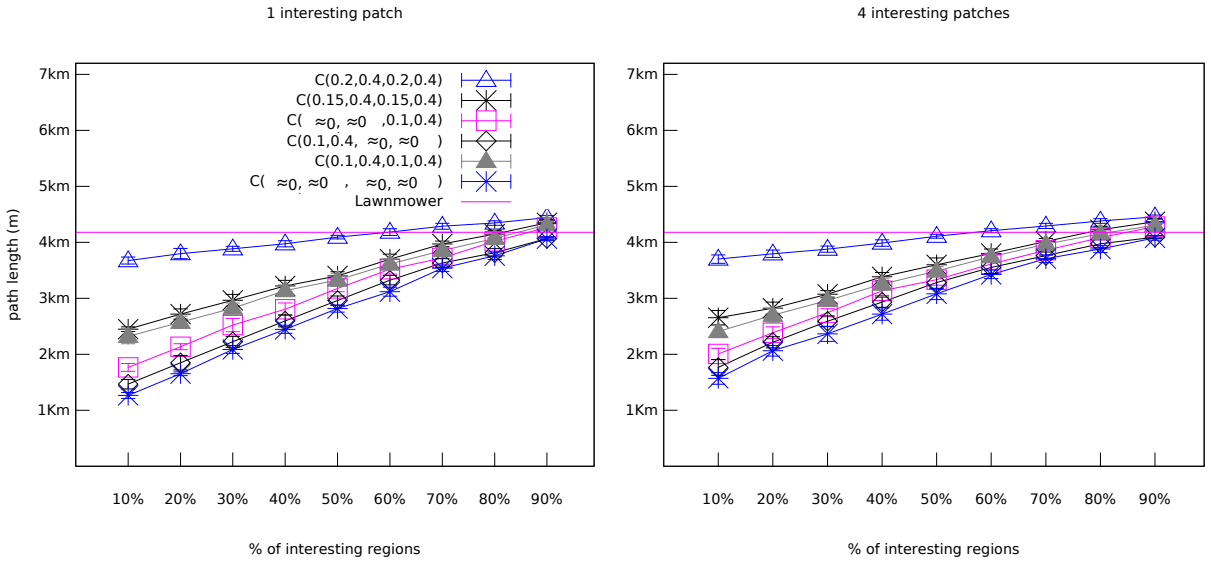


Figure 4.23: The length of the trajectory travelled by the MAV in two different environments and with various sensor configurations. The MAV flies for a larger distance with sensor configuration C(0.2, 0.4, 0.2, 0.4) due to high number of false positive detections. When the sensors are unlikely to have false positive detections (e.g. C(≈ 0, ≈ 0, ≈ 0, ≈ 0) and C(0.1, 0.4, ≈ 0, ≈ 0)), shorter coverage trajectories are required. Note that the plots for 2-patch and 3-patch configurations show a similar trend and hence not shown here.

No false negatives As shown in Figure 4.25a, when there are no false negatives but false positives in the interestingness detection, no truly interesting region is missed. Additionally, some non-target cells are also covered due to false positive detections. On the other hand, similar to the case with accurate sensors, large sections of uninteresting regions can be pruned upon negative detection events at the nodes with high altitude. These sparse samplings result in efficient trajectories as shown in Figure 4.23 (C(0.1, 0.4, ≈ 0, ≈ 0)).

No false positives With no false positives but false negatives present in the interestingness detections, the negative observations at high middle nodes are not informative enough to allow pruning the sub-trees. Consequently, as it is evident in Figure 4.25b, the MAV flies low enough over the uninteresting regions (red parts of the trajectory) such that negative detections can lead to the removal of the descendants. However, since the MAV never makes a positive observation in the uninteresting regions (no false positives), it did not fly very low at those regions as opposed to the previous case (Figure 4.25a). Nevertheless, some parts of the interesting regions are missed by the MAV due to the false negative detections.

Increasing the noise Figures 4.24b, 4.26a, 4.26b demonstrate the effect of increasing the sensor noise (false positives and negatives) on the final trajectory of the MAV. With a noisier sensor, the observations are increasingly less informative as the vehicle flies higher, i.e. the negative detections do not reduce the interestingness probability of the descendant leaf nodes below P_{int}. Accordingly, the MAV spends a significant amount of time flying at the lower levels

of the tree to make more accurate observations. As mentioned before, the threshold P_{int} is very influential on the performance of the MAV. Using a smaller threshold will reduce the area of missed interesting sections. However in that case, the MAV will require more low-altitude observations (and hence more costly trajectories) to prune uninteresting sections.

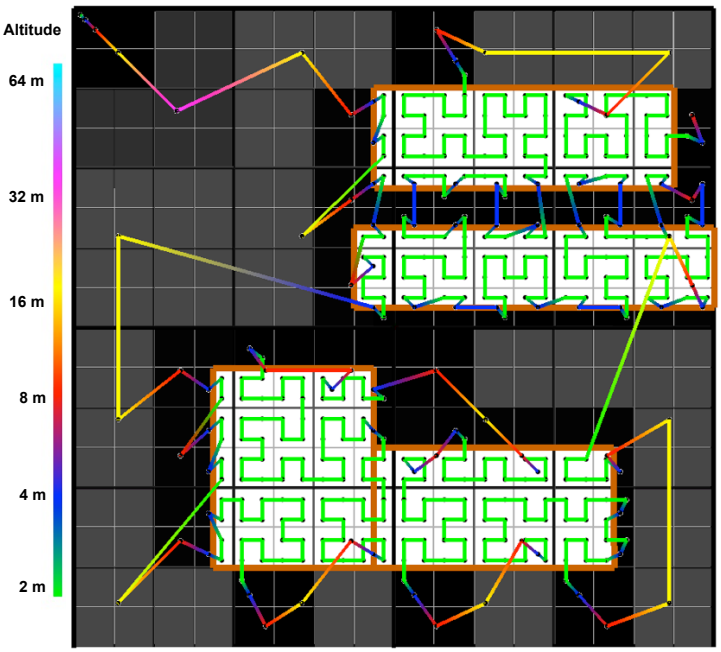
4.5 Conclusion

In this chapter, we introduced our approach to non-uniform aerial coverage and demonstrated its efficiency over the exhaustive uniform coverage method. In applications where the items of interest are in form of patches in the environment, the uniform Lawnmower pattern spends considerable amount of time/energy in uninteresting areas. Our proposed strategies adaptively cover all the initially unknown targets with high resolution. On the other hand, the uninteresting sections are pruned by making a few observations at high altitude which consequently reduces the length of the MAV trajectory.

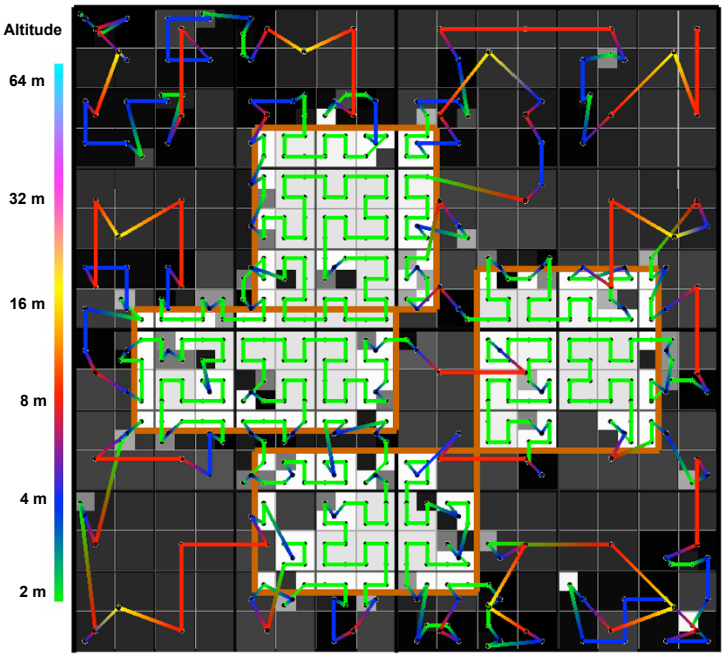
Among the proposed strategies, Hilbert-based method demonstrated the most promising results. Due to the locality of this curve, the coverage resolution can be increased upon detection of an interesting region, with the anticipation that the subsequent areas are likely to be interesting too. This behaviour leads to efficient trajectories to cover unknown targets without any planning involved, i.e. the MAV reactively changes its behaviour based on the observations.

Additionally, we studied the effect of sensor noise on the performance of the proposed method. The performance of the interestingness detection is assumed to be dependent on the height such that, as the observation altitude increases, the chances of false positive and negatives grows. It is shown that the benefits of the non-uniform approach remain as the detection noise increases. Obviously, if the observations are not informative any more due to the noisy sensor, the MAV is unable to prune the uninteresting sections of the environment. Eventually, a large portion of the high resolution coverage is in the truly uninteresting areas and hence the trajectory length increases and approaches that of the uniform Lawnmower coverage[4].

[4]The work in this chapter was published at IROS 2014 and ICRA 2015 [122, 123].

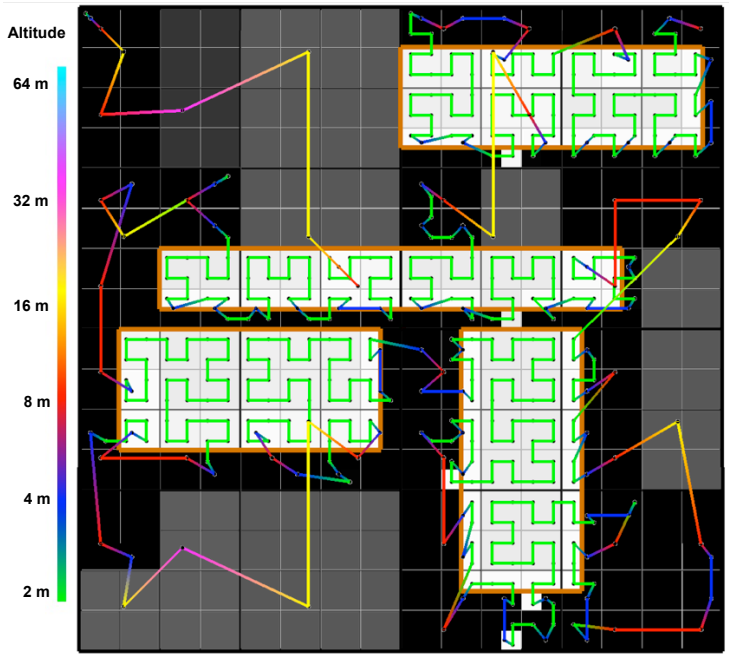


(a) (≈ 0, ≈ 0, ≈ 0, ≈ 0). When there is almost no sensor noise, negative observations made at the nodes with high altitude, suffice to prune their descendent nodes. Therefore, uninteresting regions are sampled with low resolution.

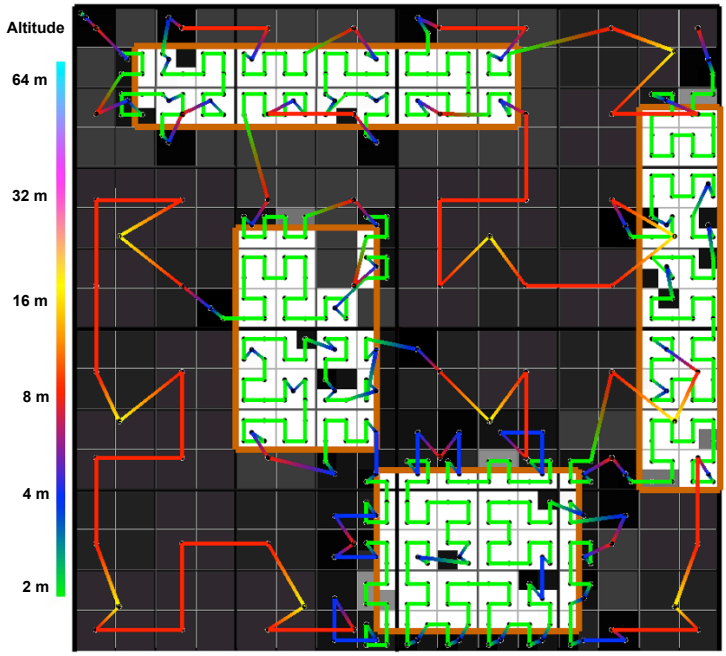


(b) C(0.1, 0.4, 0.1, 0.4). With noisy observations, the MAV flies closer to the surface to make more accurate observations. Some of the cells in the interesting regions can also be missed due to the false negative detections. Using a more conservative P_{int} makes sure that less interesting regions are missed.

Figure 4.24: Sample simulation runs.

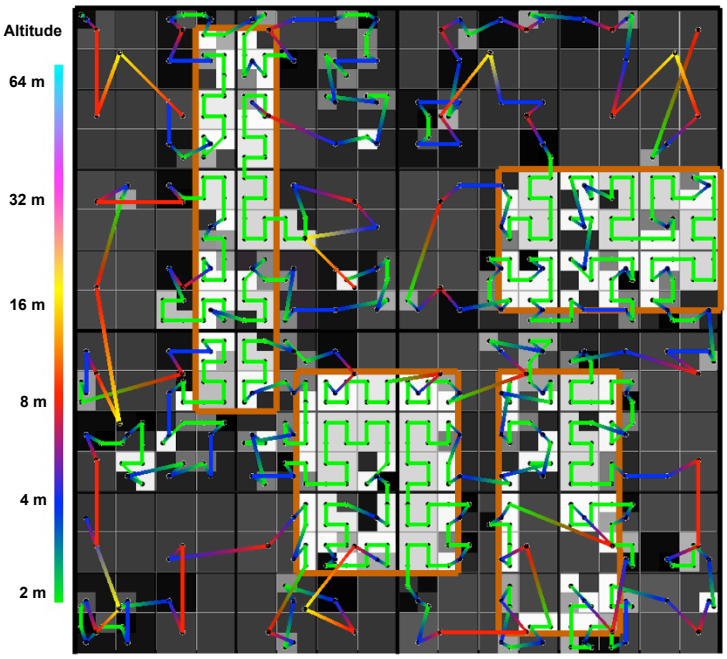


(a) C(0.1, 0.4, $\approx$ 0, $\approx$ 0). When there is no false negative but false positives in the interestingness detections, all the target cells together with some uninteresting cells (false positives) are covered with high resolution. Furthermore, in case of a negative detection at a middle node with high altitude (yellow and pink parts of the trajectory), all the descendants are pruned. However, due to occasional false positives, the MAV sometimes flies down the tree for more accurate observations.

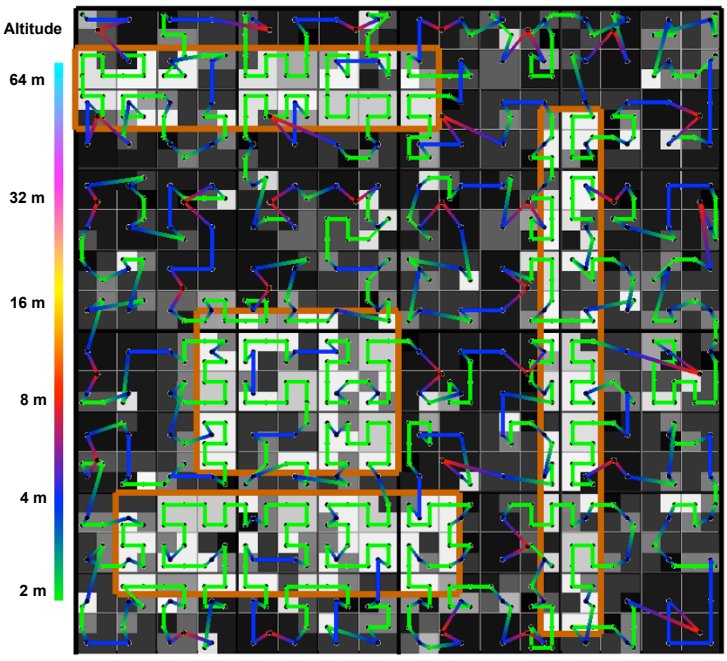


(b) C($\approx$ 0, $\approx$ 0, 0.1, 0.4). When there are false negative detections, the negative observations at high middle nodes are not informative enough to prune the sub-trees. Therefore, in the uninteresting regions, the MAV flies low enough (red parts of the trajectory) that the negative detections can lead to the removal of the descendants. Furthermore, parts of the interesting regions are missed due to false negatives detections.

Figure 4.25: Sample simulation runs.



(a) C(0.15, 0.4, 0.15, 0.4).



(b) C(0.2, 0.4, 0.2, 0.4).

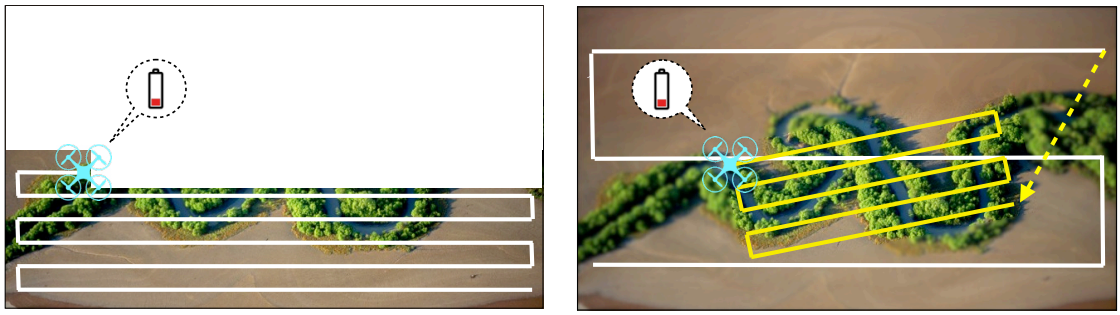
Figure 4.26: Sample simulation runs. As the accuracy of the sensor degrades, the MAV flies at a lower altitude to make more informative observations.

Chapter 5

Time-Discounted Active Aerial Survey

As mentioned before, in the task of aerial coverage, the value of the data collected in the region might not be uniform and some regions may be more interesting than others. However, one might prefer to spend UAV energy/time budget to obtain a uniform survey of a large environment at relatively low resolution (by flying higher) rather than a small region at higher resolution. For example, consider a mapping application in which the output model is used to simulate the impact of a flood on a town. The completeness of the map's coverage is essential to model water transport correctly. Figure 5.1 schematically illustrates the two different approach. In Figure 5.1a, only part of the environment is covered densely most of which is uninteresting terrain. However, as shown in Figure 5.1b, only the interesting sections can be densely covered after s coarse survey of the entire area.

In this chapter we consider the task of performing both high-resolution ROI survey and complete region coverage at the same time. We maintain a hard constraint that the complete survey must be performed, but the UAV reasons as it goes about its remaining flight time, and will survey ROIs at high resolution as much as the energy budget allows. Thus we provide both



(a) (b)

Figure 5.1: With limited flight time (a) the UAV can densely cover part of the region (b) the entire region can be coarsely surveyed and then the remaining flight time can be used to densely cover the detected interesting sub-regions.

complete coverage and a best-effort at high-value information. This is a novel problem that has not been described in the aerial survey literature.

Another aspect of data collection is that it should be achieved in a timely manner. The sooner we see the important things, the better. For example the high resolution observations of ROIs can be used for urgent decision making in emergencies. Therefore, we model the performance of the UAV using a time-discounted reward function. According to this function, the value of the collected data for a region drops down exponentially based on the time observation time. Hence, a generally acceptable behaviour will be to acquire the high resolution data as soon as the regions of interest are known.

The way we approach the above introduced task is based on two considerations. Firstly, as we mentioned before, by densely covering the ROIs immediately after they are detected, more reward is obtained for that region. Secondly, the cost of sweeping a region as a whole is (most of the times) less than when covering it part by part. Subsequently, we will propose three different policies to decide when to switch between performing the coarse survey and collecting high resolution data from ROIs.

5.1 Environment Model

We model the distribution of valuable/interesting locations in the world as a *Gaussian Process*: a probabilistic representation of a scalar field [124]. At each location $\mathbf{x}$, the value of the field, denoted by $f(\mathbf{x})$, is estimated by a Gaussian distribution with mean $m(\mathbf{x})$ and a covariance function $k(\mathbf{x}, \mathbf{x}')$:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$$

where $\mathbf{x}'$ is another location in the environment. The form of the covariance function incorporates assumptions about the smoothness of the estimated field. The input to the model is a set of noisy observations $\{(\mathbf{x}_i^*, f_i^*)|i = 1, ..., n\}$ where $f_i^* = f(\mathbf{x}_i^*) + \epsilon_i$ and $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. It is assumed that the observations are under constant Gaussian measurement noise with zero mean. The value of f at location $\mathbf{x}$ is estimated as a Gaussian distribution:

$$f'(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T (K + \sigma^2 I)^{-1} \mathbf{f}^*$$
$$V[f'] = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^T (K + \sigma^2 I)^{-1} \mathbf{k}(\mathbf{x})$$

where $\mathbf{k}(\mathbf{x}) = [k(\mathbf{x}_1, \mathbf{x}), ..., k(\mathbf{x}_n, \mathbf{x})]^T$, $K = [k(\mathbf{x}, \mathbf{x}')]_{\mathbf{x},\mathbf{x}' \in \{\mathbf{x}_i^*\}}$ and $\mathbf{f}^* = [f_1^*, ..., f_n^*]^T$.
The form of the kernel function $k(\mathbf{x}, \mathbf{x}')$ should reflect the assumptions about the scalar field. We use *squared exponential* kernel function with the form

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\frac{\| \mathbf{x} - \mathbf{x}' \|}{2l^2}).$$

The parameter l specifies how quickly the value of the scalar field becomes de-correlated with distance. In this work, we will not focus on the learning aspects and assume that the parameters of the model are learned in advance.

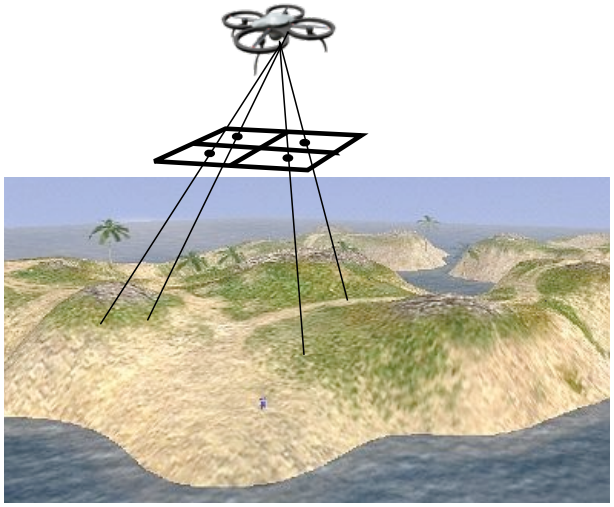


Figure 5.2: Generating observations using the images captured by the on-board camera.

Sensing

We use the on-board camera to take measurements in the environment. At each location[1] $\mathbf{p} \in \mathbf{R}^3$ the UAV can make a number of observations $O(\mathbf{p}) = \{(\mathbf{x}_i^{\mathbf{p}}, f_i^{\mathbf{p}}) | i = 1, ..., N\}$ using the rectified images captured by the camera. The observations $f_i^{\mathbf{p}}$ are generated using the image patch centered around the pixel at $(u_i^{\mathbf{p}}, v_i^{\mathbf{p}})$ in the image. The locations of these measurements $\mathbf{x}_i^p$, are calculated based on the position of the UAV $\mathbf{p}$, the image coordinates $(u_i^{\mathbf{p}}, v_i^{\mathbf{p}})$, and the field of view of the camera[2]. We use a uniform grid as the set of measurement positions $\{(u_i^{\mathbf{p}}, v_i^{\mathbf{p}})\}$ in the image (see Figure 5.2).

Target Regions

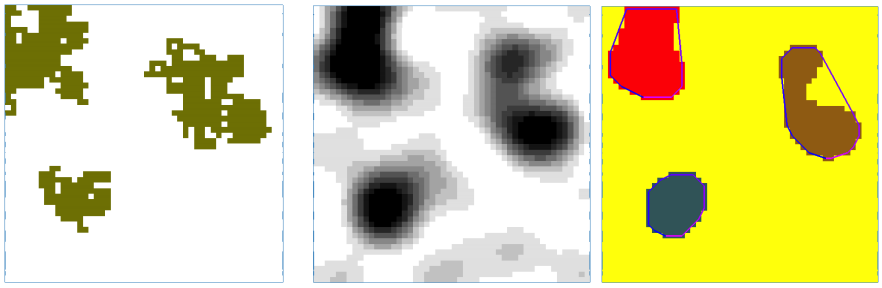
The area is discretized into a grid $\mathcal{G}$ with square cells. Using the estimated scalar field, a posterior gaussian distribution $\mathcal{N}(f(\mathbf{x}_c), v(\mathbf{x}_c)^2)$ can be obtained for each grid cell c, where $\mathbf{x}_c$ denotes the center location of the cell c. We use level-set estimation techniques ([125]) to partition the cells into two sets of target $\mathcal{T}$ and non-target $\mathcal{NT}$. For a given threshold h, all the cells that have, with high probability, values less than h are classified as non-target, i.e.

$$\mathcal{NT} = \{c \in \mathcal{G} | f(\mathbf{x}_c) + \lambda v(\mathbf{x}_c) < h\}.$$

With λ, we can control the confidence in the classification. Note that the set $\mathcal{T}$ will contain two types of cells: those that can be classified as having, with high probability, value higher than h and cells that are ambiguous and cannot be labeled with confidence. We treat both these cells as targets for high resolution sensing.

Neighbour target cells in the grid form bigger regions of interest (see Figure 5.3). These regions are obtained by performing a depth-first-search in the grid. A convex polygon that contains the cells of each ROI is estimated and used for subsequent coverage path planning.

[1] We assume that the UAV always maintains a constant rotation around the z axis, i.e. no change in yaw.
[2] We assume that when the UAV is on the ground $z = 0$.



(a) (b) (c)

Figure 5.3: Process of generating the target regions. (a) The ground truth for the underlying field. (b) The posterior gaussian process after taking measurements in the entire area. (c) The target cells form regions of interest. A bounding convex polygon is estimated for the target regions.

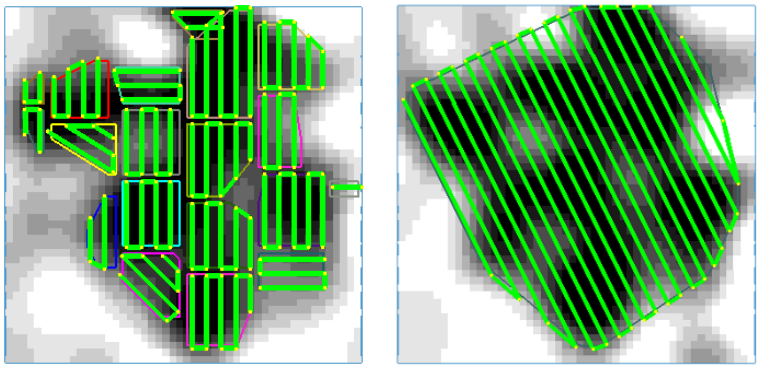
5.2 Efficient Coverage Path Planning

Small unmanned aerial vehicles are limited in payload and unable to carry high capacity batteries. Therefore, they can only fly for a limited time. The rate of power consumption varies across different manoeuvres of UAVs. Consequently, the amount of energy required to execute a trajectory depends on the velocity profile of the vehicle along the path. For example, performing a sharp 90° turn may require as much energy as flying 5 meters in a straight line with a constant speed and altitude.

Consequently, many coverage path planning methods try to generate efficient trajectories that require less acceleration, mainly by minimizing the number of turns and assuming constant speed along the path. With a more accurate energy model, one can generate trajectories that are more efficient. For example, in [126], power consumption of travelling between two waypoints is calculated by breaking up the manoeuvre into accelerating, moving with constant speed, and decelerating, and the consumed energy is calculated for each part. This allows the calculation of an optimal travelling speed as a function of the distance between the waypoints. Here we assume that the spatial sampling density remains constant along the straight lines at a same altitude. Since the sampling rate is constant, the UAV must travel at a constant speed. Therefore the power consumption for flying between two locations can be proportional to the actual travel time, plus a constant penalty to perform the acceleration and deceleration (Note that here we are only considering quadrotors and assume that it never performs a turning manoeuvre and its heading (yaw) is constant). Therefore, instead of energy, we use time as the cost of a trajectory and assume a time budget for the entire mission.

High resolution data collection from a ROI is achieved by sweeping the region according to an efficient coverage pattern. Given the convex polygon of a target region, we find the sweeping direction that yields the minimum number of turns in the coverage path [82]. The altitude h_c at which the target regions are covered is obtained using the user defined parameter τ (which indicates the desired meters per pixel of the captured images):

$$h_c = f \times \tau$$



(a) (b)

Figure 5.4: Coverage of a target polygon. (a) Coverage paths are planned for partitions of the original target which leads to high number of turns. (b) When the polygon is fully known, a single coverage trajectory is less costly compared to partitioning.

where f represents the focal length of the camera. Also, the inter-lap distance of the coverage pattern is calculated based on h_c and the field of view of the camera. When planning for coverage of a convex polygon, a single coverage path to cover the whole polygon is going to be less costly than multiple coverage paths sweeping partitions of the same polygon. The reason for this is that, ideally, the length of the coverage trajectory will remain the same in the two cases while the number of turns is higher in the latter case (see Figure 5.4).

As the UAV executes the coverage paths of the ROIs, a reward equal to the area of the swept segment is received. Note that the reward for each grid cell is only received once and multiple coverage of a cell does not yield more reward. Moreover, we consider the reward function to be additive and not subject to sub-modularity.

5.3 Active Survey

In order to accomplish a complete coarse survey, an initial lawnmower path is planned offline to cover the entire area. The purpose of this survey is to sample the entire area such that the entropy of the posterior Gaussian distribution at each grid cell is reduced to a user-defined threshold. For this, we only need to determine the altitude a_s at which the UAV flies while collecting samples. A high altitude leads to fewer lawnmower laps but increases the sensor noise and the entropy remains high. On the other hand, a low altitude enables more accurate observations but will incur more cost. We find the highest altitude at which the posterior entropy of the cells goes below the required threshold. This can be done due to the fact that the covariance of the Gaussian process does not directly depend on the actual value of the measurements but on their positions. We assume that measurements are taken at regular intervals during the coarse survey. For any candidate survey path, according to the sensing mechanism described in section 5.1, the measurement positions can be known and hence a posterior entropy is obtained (see [127] for example).

After each sensing action, the estimated Gaussian distribution for each surveyed grid cell is

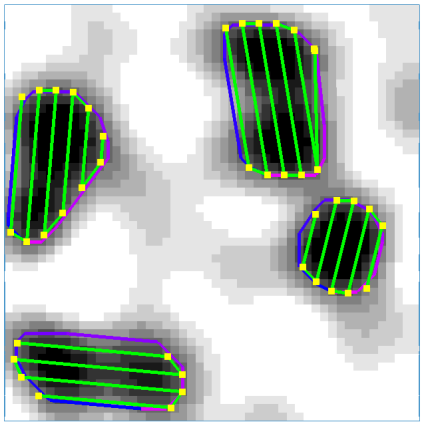


Figure 5.5: The high resolution coverage paths planned for each ROI. The sweeping direction is determined such that minimum number of turns is required to cover a target.

updated and target regions are detected as described in section 5.1. Note that only the cells that have been swept by the camera footprint A_{fp} are considered for target detection and the rest of the area remain as *unexplored*.

Reward Function

Each target T has an intrinsic reward value $r(T)$ which is equal to its area. We use $\mathcal{LM}(T)$ to denote the lawnmower coverage path for target T which is a sequence of waypoints as shown in figure 5.5. The time-discounted reward depends on the actual time that the targets are sensed. In a discretized setting, the reward is collected at each time step t_i. Let $r_{t_i}(T)$ denote the new area of target T that has been covered with high resolution in time period $(t_{i-1}, t_i]$, where $t_i = t_{i-1} + dt$ and dt is the time step for reward evaluation. The total time-discounted reward after complete coverage of target T is

$$r_{discounted}(T) = \sum_{t_i=t_T^s}^{t_T^e} \beta^{t_i} r_{t_i}(T)$$

where t_T^s is the time at which the robot reaches the first waypoint of $\mathcal{LM}(T)$ and t_T^e is the time step when the UAV abandons T. The parameter β is the discounting factor.

Target Visitation Tour

As the UAV executes the policies described in the next section, we are often presented with a target tour design problem: Given a set of targets $\mathcal{T}$, their corresponding lawnmower coverage paths $\mathcal{LM}(T), T \in \mathcal{T}$, current position of the UAV P_{UAV}, and a time budget $\mathcal{B}$, find a sequence $S_{\mathcal{T}} = \langle T_{i_1}, .., T_{i_j} \rangle$, $j \leq |\mathcal{T}|$ such that the cost of the path comprised of waypoint sequence $\langle P_{UAV}, \mathcal{LM}(T_{i_0}), .., \mathcal{LM}(T_{i_j}), P_{UAV} \rangle$ is not more than the budget, and the sum of the rewards for visited targets, i.e. $R(S_{\mathcal{T}}) = \sum_{k=0}^{j} r(T_{i_k})$, is maximized.
We take a greedy approach to solve this problem as follows: We start with $S_{\mathcal{T}} = \langle\rangle$ and $P = P_{UAV}$. At each iteration, a target T_{max} that maximizes the utility function $U(T, P) =$

$\frac{r(T)}{D(P,\mathcal{LM}(T))}$, where $D(P,\mathcal{LM}(T))$ denotes the distance from P to the nearest end of the coverage path $\mathcal{LM}(T)$. The selected target is added to the end of the tour $S_{\mathcal{T}} = \langle S_{\mathcal{T}}, T_{max}\rangle$. If the cost of the produced path $\langle P_{UAV}, \mathcal{LM}(T_{i_0}), .., \mathcal{LM}(T_{i_j}), P_{UAV}\rangle$ is more than the available budget, the last target in the sequence is partially covered to satisfy the budget constraints. We use $Tour_{\mathcal{B}}(P_{UAV}, \mathcal{T})$ to denote the solution to this problem generated by the described algorithm. Note that this problem is a version of *orienteering problem* which is NP-hard.

Sensing Policies

Since the targets are unknown in advance, high resolution data collection from ROIs is dependent on the progress in the coarse survey. As the completion of the coarse survey is a hard constraint, it is required to estimate the required time to complete the survey and only spend the extra available time to collect high resolution data. Let us use $\mathcal{B}^P$ to denote the extra time available to the UAV. Note that the cost of stopping during the lawnmower lap is also taken into account to obtain $\mathcal{B}^P$.

We propose and compare three different strategies to switch between coarse survey and high resolution sensing of ROIs.

Greedy Since the collected reward is discounted with time, a simple strategy is to perform high resolution coverage of target regions as soon as they are detected. Therefore following each sensing action at location P, there is a set of newly detected targets $\mathcal{T}_{new}$. We solve a target tour design problem as described in section 5.3 to find a solution $Tour_{\mathcal{B}^P}(P, \mathcal{T}_{new})$. Then the coarse survey is interrupted and the UAV starts performing the high resolution sensing of targets. After the targets are swept according to their lawnmower coverage paths, the UAV returns to the location where the survey was interrupted and continues the coarse sampling.

Semi-greedy When the regions of interest in the environment are in form of big patches, a single coverage path for the whole polygon can be more efficient than covering the region part by part. Hence, in the Semi-greedy strategy, we first find all the targets that are adjacent to the unexplored regions. Due to the locality exhibited by the underlying field, these targets are likely to be part of a bigger region of interest. Therefore, we delay the high resolution coverage of these targets and all the targets that are within a distance of $dist_{delay}$. The delayed targets are considered for coverage when the unexplored boundaries are coarsely sampled. However, the decision of delaying a boundary target T_b is made with probability:

$$p_{delay}(T_b) = \begin{cases} 1 - \dfrac{r(T_b)}{\gamma \|A_{fp}\|} & r(T_b) < \gamma \|A_{fp}\| \\ 0 & \text{Otherwise} \end{cases}$$

where $\|A\|$ indicates the area of the camera footprint in the coarse survey and $\gamma = 2.5$. Therefore as the boundary targets grow, the chance of postponing their high resolution coverage decreases. The rationale behind delaying boundary targets is that, there is a high chance that it will

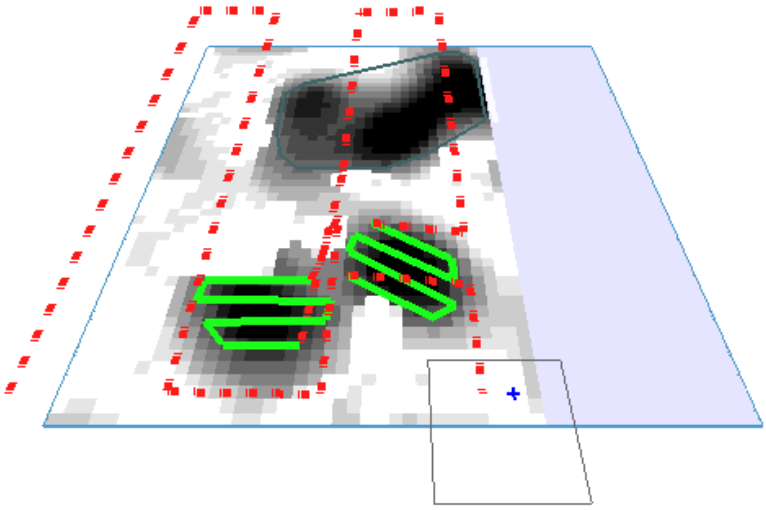


Figure 5.6: Snapshot of the simulation. The red dashed line is the coarse survey trajectory and the green lines show the coverage paths used to collect high resolution data from target regions (dark areas). The light blue sections are the regions that have not yet been covered by the coarse survey.

extended and become a larger target and hence, as discussed before (section 5.2), a less costly trajectory can achieve complete high resolution coverage. This approach is specifically in contrast to the Greedy policy where a large polygon might be covered in multiple partitions (Figure 5.4). Additionally, the UAV will come back to the unexplored neighbouring area as it is performing the coarse survey which means the energy that will be spent on coming back to a delayed target (i.e. switching cost) will not be too large.

Two-phase Often in the literature a coarse initial survey is completed such that a complete prior is available for further planning. We follow this strategy by completing the coarse survey. After the UAV reaches the last waypoint of the coarse survey, all the targets in the entire region are detected and the same target tour design problem is solved, with a small change that instead of returning to the current position, the UAV should return to the starting position.

The policies above are used to decide when the UAV needs to stop the coarse survey and perform high resolution data collection of the discovered ROIs. The Semi-greedy policy is designed to behave similar to the Greedy policy when locality is not high, i.e. they both exploit the targets as they become discovered. This can yield high performance in a time-discounted task. However, with high locality, postponing some targets (similar to the two-phase policy) can result in a more informed decision about which targets to exploit. This can be seen in Semi-greedy policy where boundary targets are postponed.

5.4 Experiments and Results

The above algorithms are implemented and tested in simulations as described in the following section. The result of the experiments is presented in section 5.4.2 along with some discussions.

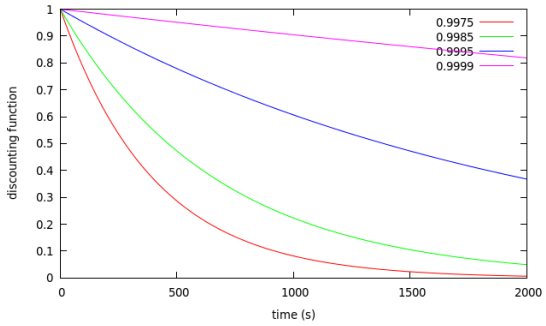


Figure 5.7: Sample discount parameters.

5.4.1 Experimental Setup

We simulated a 100×100 m^2 area with randomly generated fields of high-value targets. The area is represented by a grid of 2×2 m^2 cells. To generate a random field, 2 parameters (c, d) are used: We start with a random set of c grid cells and assign the highest value of the scalar function f to these cells and mark them as *visited*. Then iteratively, a neighbour cell of a visited cell is selected and assigned the highest value again and marked as visited. This process continues until $d\%$ of the grid cells are visited. Note that the higher the value of c, the greater the chance of generating a scattered field.

For each field configuration (c, d) we generate 20 environments and use the three policies for active aerial survey. The UAV has a budget of 30 minutes. It flies at 1 m/s and it is assumed to take $5s$ to decelerate and accelerate at turning points. Also, when a waypoint is not at the same altitude of the UAV, i.e. there is change in altitude, it flies at half the normal speed.

The discounting factors used in the experiments are comparable to the flight time of the UAV, i.e. we ignore the extreme values of β and focus on moderate values. Undoubtedly with extreme discounting in the task, the greedy collection of reward leads to superior performance. As β increases, the Two-phase policy can perform better since the planning is more informed. However, in between these two extremes, our Semi-greedy policy can perform more efficiently. A sample of the discount parameters used for the experiments are shown in Figure 5.7.

5.4.2 Results

Figure 5.8 shows the discounted reward that is collected using different policies in various fields (4 different values of d). The results show a similar trend across different values of c. The plots indicate that with low discounting factor, the Greedy policy performs better than the Two-phase strategy as the UAV covers the targets as soon as they are revealed. However, the reward of Two-phase method improves and outperforms the Greedy policy as β increases. This is because by completion of the coarse survey, all the information about the distribution and size of the interesting regions is known and a better plan can be generated, with little performance loss due to the discounting. On the other hand, the Semi-greedy policy demonstrates an comparable performance with different β values. Although the boundary targets are delayed for a short time in the Semi-greedy approach, the UAV can achieve coverage with a more efficient trajectory and

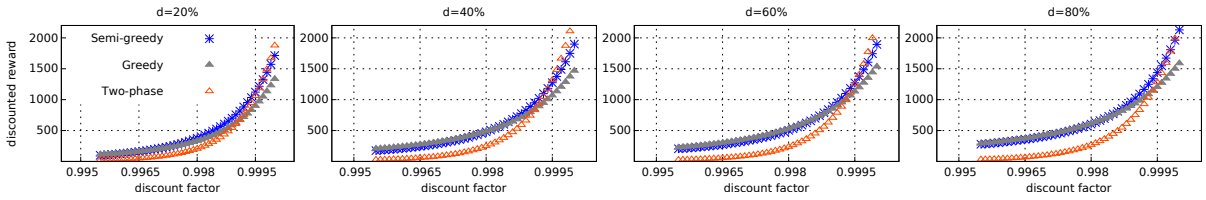


Figure 5.8: he collected reward with different discount factors. The parameter $c = 7$ in all the trials. Also, the standard deviation in all the configurations in less than 200. The total collected reward with Two-phase and Greedy policies drops down (compared to the other methods) when the discount-factor is low and high respectively. On the other hand, the Semi-greedy strategy demonstrates a more steady performance across different values of β.

hence more time will be left and used towards subsequent high resolution coverage. These results imply that when β cannot be obtained exactly for an application, the Semi-greedy policy can be a better choice for active survey.

It is worth noting that, since the Greedy approach only relies on a coarse sampling in a local area, it suffers from inaccurate estimation whereas in the other policies more samples are fed into the Gaussian process model and the estimation can be more accurate. This is shown in Figure 5.10 where the Semi-greedy (Figure 5.10c) and Two-phase (Figure 5.10d) policies have a more precise and complete knowledge of the target regions compared to the Greedy (Figure 5.10b) method.

Figure 5.9 contains the result as d changes with 4 different values of c and $\beta = 0.999$. As the value of d (ratio of the area that is interesting) increases, the reward collected by all the policies increases. With higher values of c (larger number of possible ROIs) the locality in the target regions decreases and the increased cost of switching between targets leads to lower performance of all policies (compared to when c is small). However, the Semi-greedy policy maintains a superior performance in all configurations. When the environment is almost entirely interesting (e.g. $d > \%60$), the detected targets are always on the boundaries of unexplored regions and hence likely to be postponed in the Semi-greedy policy. Due to the correlation of the probability of postponing and the size of the ROI, the UAV decides to collect high resolution data when a target region becomes large enough and there is no loss of value due to the discount, in contrast to Two-phase policy. Therefore, although the switching cost in the Two-phase strategy is reduced and more reward is collected (Figure 5.9), the Semi-greedy policy still has better performance. This can be seen in Figure 5.11c where large parts of the environment are covered without postponing for a long time (which is the case in the Two-phase method, Figure 5.11d). Figure 5.12 shows a sample simulation experiment in a larger environment ($200m \times 200m$). It indicates that with the Greedy strategy, due to the fragmentation of the ROIs and high cost of the coverage trajectories, the MAV is unable to collect high resolution data from some of the interesting regions. Whereas, with the Semi-greedy policy ROIs are swept using efficient trajectories and, in contrast to the Two-phase policy, in a timely manner.

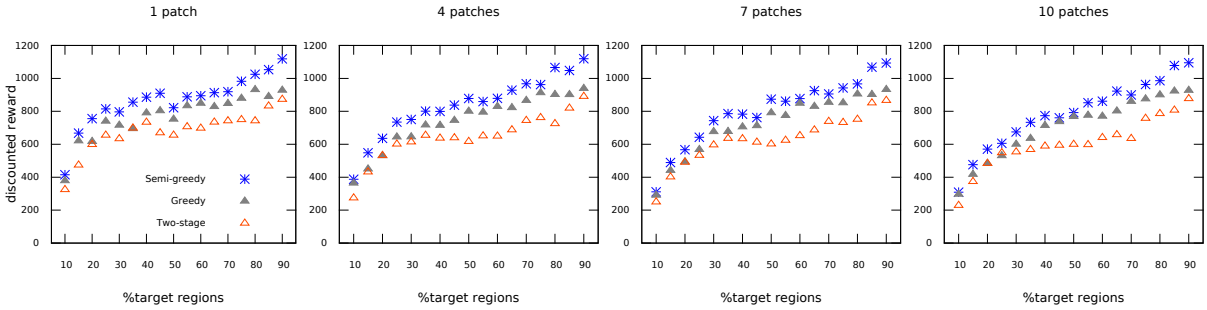
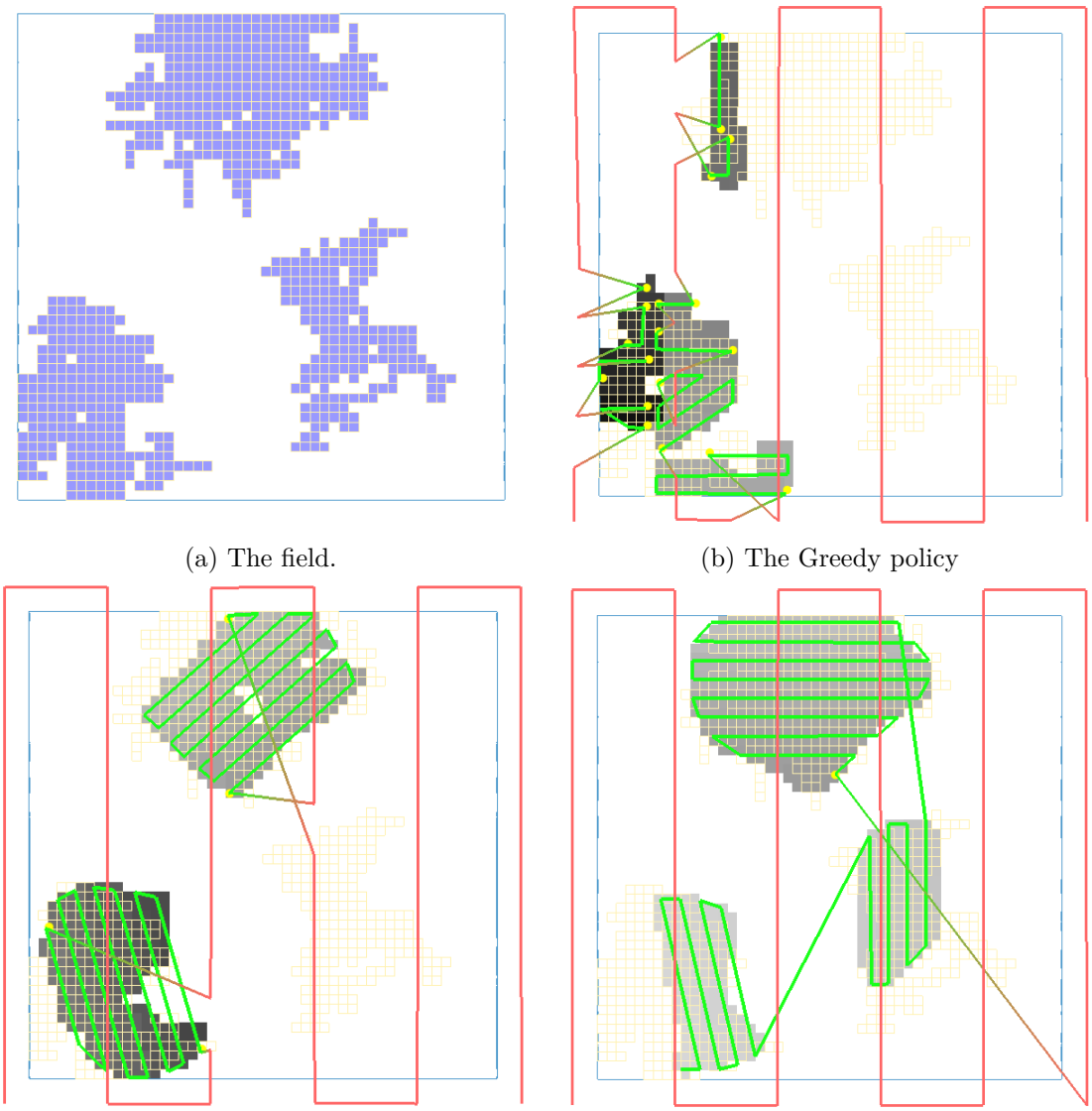


Figure 5.9: The collected reward with $\beta = 0.999$ using different policies. The standard deviation in all the configurations in less than 200 (average 100). With $d < \%50$, as the number of patches increases, the locality decreases and hence the switching costs get bigger. This can be seen in the reduction of the performance in all policies as c increases.

5.5 Conclusion

With the limited flight time of small UAVs, it is important to perform sensing in parts of the environment with high data value. The task gets more complicated when the value of high resolution data degrades over time. With no *a priori* information available, the proposed Semi-greedy policy is empirically shown to be effective in active aerial surveys with time discounted rewards. By postponing the coverage of targets for a short time, a more accurate estimation of the interesting regions is obtained and a less costly coverage trajectory is planned. On the other hand, covering targets greedily as soon as they are detected, suffers from inaccurate estimation and high cost of acceleration in the coverage trajectory. Additionally, the Two-phase method postpones the data collection to the time that a complete estimation map is available and ignores the discounting effect.



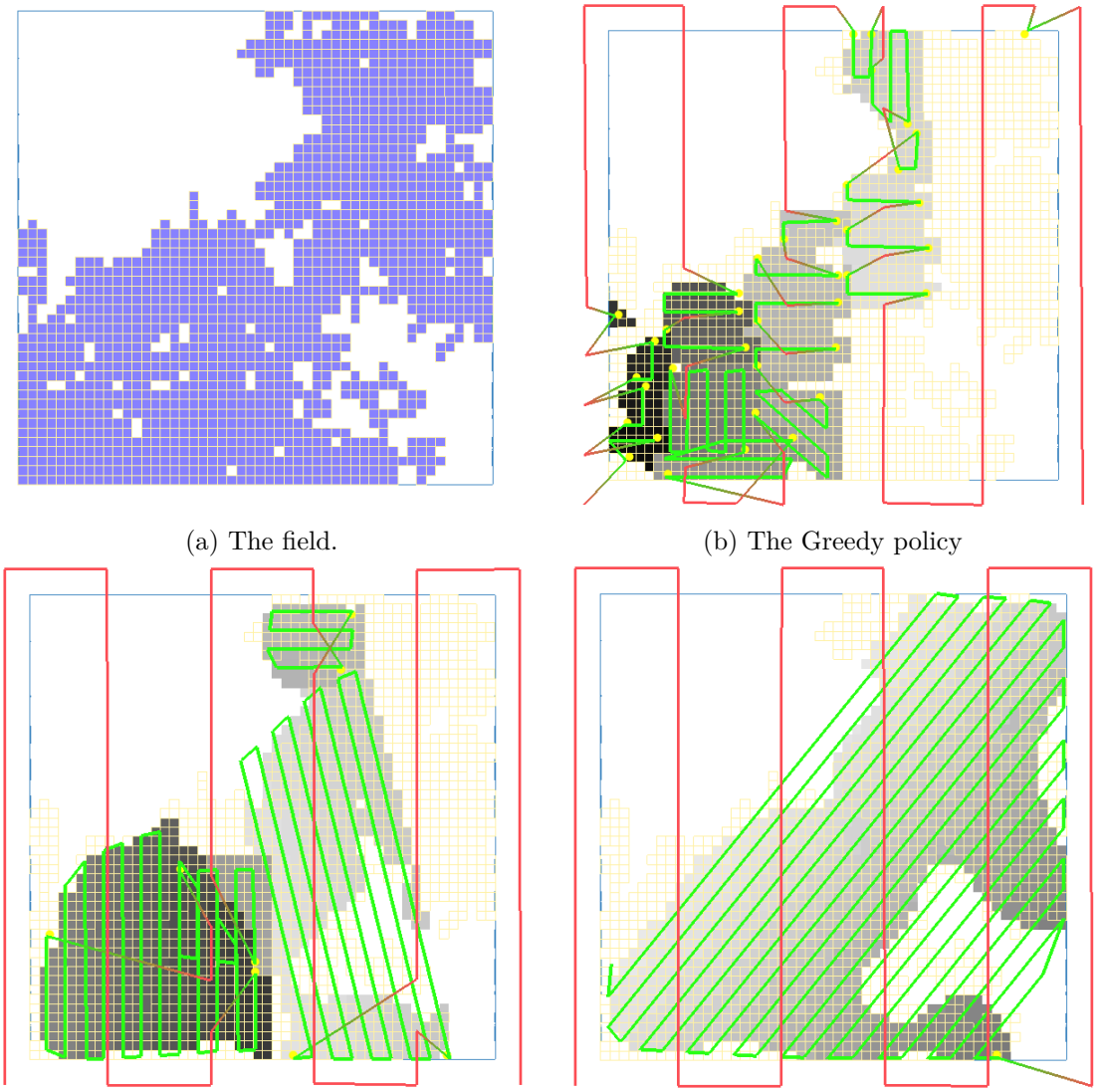
(a) The field.

(b) The Greedy policy

(c) The Semi-greedy policy

(d) The Two-phase policy

Figure 5.10: The green lines indicate the high resolution sensing trajectories and the color intensity of swept regions shows the time of coverage (the darker, the earlier). The Greedy policy has a less accurate estimation of the target regions compare to the other two policies as it relies on the local coarse samples. Moreover, the high resolution coverage paths are less costly in the Semi-greedy and Two-phase methods.



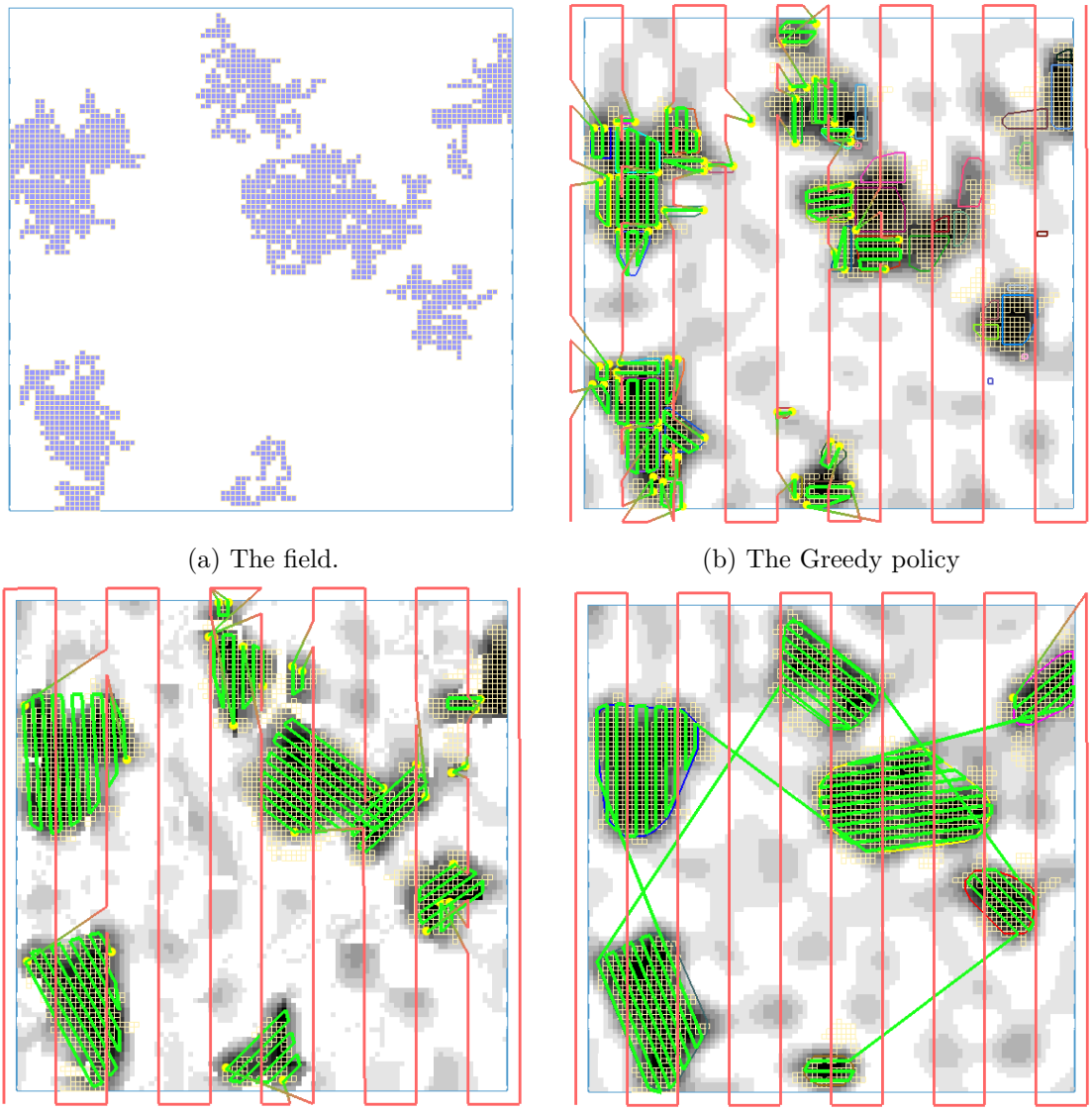
(a) The field.

(b) The Greedy policy

(c) The Semi-greedy policy

(d) The Two-phase policy

Figure 5.11: When most of the environment is interesting, the Semi-greedy policy covers the discovered ROIs as they become large enough. This results in efficient trajectories and timely collection of target data. On the contrary, the Two-phase policy postpones the high resolution coverage of targets to the end of the coarse survey.



(a) The field.

(b) The Greedy policy

(c) The Semi-greedy policy

(d) The Two-phase policy

Figure 5.12: Sample simulation run in a larger environment ($200m \times 200m$). Note that the posterior Gaussian Process is visualized in these figures. It is clear that with the Greedy policy, the MAV is unable to cover some of the detected ROIs due to lack of time.

Chapter 6

Conclusion and Future Research

We proposed adaptive planning methods for MAVs to accomplish some important tasks efficiently and robustly without any prior information about the environment. The unknown environment assumption maintains the motivations behind using UAVs as platforms that require almost no infrastructure and can fly almost anywhere to collect sensory data as opposed to humans or ground vehicles.

This proposed thesis is an effort to devise planning algorithms for better use of the short flight time of MAVs in data acquisition tasks such as aerial imagery and mapping. Also, the proposed navigation method tries to remedy the use of light-weight and low-powered sensors on-board these robots.

To summarize this thesis:

- We presented a novel method for reliable navigation in unknown environments. Existing methods assume that the localization quality is maintained at an acceptable level anywhere in the environment and hence are unable to avoid visually-poor regions. On the contrary, by estimating feature-richness and considering it during path planning, we direct the camera towards feature-rich regions. Iterative re-planning ensures that the latest information is taken into account and a safe path (collision-free and feature-rich) to the goal is found. Our approach does not use any prior knowledge about the distribution of visual features or obstacles in the environment and only relies on the information sensed by the drone as it navigates towards the goal.

- We proposed a novel terrain coverage method that considers non-uniformity in the environment. The existing methods for coverage path planning consider a uniformly interesting target area and hence all the regions are covered exhaustively with high resolution. On the contrary, in many real world applications items of interest are not uniformly distributed but form patches of locally high interest. Our proposed coverage strategies generate shorter trajectories compared to exhaustive coverage plans, and achieve sparse sampling of uninteresting sections of the environment and high resolution sampling of interesting regions.

- We also introduced a practical task of coarse coverage of an area, while identifying ROIs, and locally surveying as much of the ROIs at high resolution as the battery allows. The

existing approaches perform high resolution data collection from ROIs after obtaining an initial coarse survey, which results in weak performance when the task is time-discounted. However, our proposed policies accomplish these two subtasks simultaneously and in a timely manner. We assume a time/energy budget for the vehicle and consider specific costs for different manoeuvres of the MAV. The proposed policies decide when to switch between the coarse survey and high resolution imagery data collection.

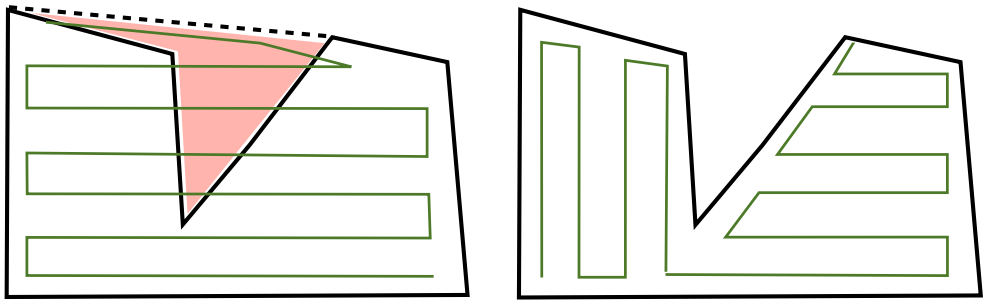
Future Research

The methods that are presented in this thesis can be extended in several ways. Here we list suggestions for further research:

Expanding Viewpoint Score The metric for localization quality that we proposed in Chapter 3, can be expanded to include the saliency of the visible feature points. Repetitive visual patterns such as grass and leaves on a tree, can result in wrong feature matching and hence the corruption of the mapping and tracking. One way to obtain the saliency of the scene is to project the reconstructed textured mesh on a virtual camera and measure the visual saliency using existing methods (see [46] for a short review). However, this might be computationally expensive considering that the viewpoint score is computed for each sample point in RRT* planner. We can approximate this process by measuring the similarity between the textures of adjacent triangles in the reconstructed mesh. This simplification can reduce the computational demand and allow real-time performance of the planner. Consequently, viewpoints with self-similar scene will have lower score and avoided during the navigation.

Furthermore, we assumed that the MAV will fly relatively close to the environment and considered a preferred distance to the scene. If the same scene is viewed from a greater distance, the features that were previously visible in the first level of the image pyramid may not be observable anymore. On the other hand, the features that were detected in the last pyramid level are likely to be detected again. Besides, some other feature points may emerge as the camera moves to the distant location. Hence, the viewpoint score should be extended to predict the feature-richness of a scene as the camera moves away.

Aerial Coverage Path Planning for Non-convex Regions Assuming an open environment where the MAV can fly anywhere (no obstacles and no-fly zones), a coverage trajectory can be planned for a non-convex target by covering the smallest convex-hull that contains that target as shown in Figure 6.1a. In Chapter 5, we used this method for all the discovered targets. However, depending on the shape of the target, non-negligible parts of the coverage path might be in uninteresting regions which is wast of the UAV flight time. Existing methods avoid coverage of areas outside the target region by decomposing the non-convex region into simpler convex (or monotone) polygons [128] and sweeping them by lawnmower pattern (Figure 6.1b). However, even though the coverage path becomes shorter, it may have more number of turns. Therefore, there will be a trade-off between avoiding useless coverage and the number of turns.



(a) A convex approximation of a non-convex polygon is used for coverage path planning. Some non-target regions are covered which is a wast of flight time, but it might reduce the number of turns.

(b) A non-convex polygon is decomposed into simpler convex polygons to plan coverage trajectories that only covers the target. This will lead to shorter path, but possible more turns may be necessary.

Figure 6.1: Aerial coverage of a non-convex target. The MAVs can fly outside the specified region which might result in fewer number of turns.

Multi-UAV Non-uniform Coverage In multi-robot coverage tasks, the target area is often decomposed into partitions and each robot is allocated to sweep one of the parts. Assuming a homogeneous group of UAVs, the partitions are equal in size to minimize the overall task completion time. However, in a non-uniform environment, the cost of adaptive coverage (as described in Chapter 4) depends on the distribution of the interesting sections. Therefore, allocating partitions of equal size does not guarantee to minimize the task completion time. An interesting future research topic can be to use online task allocation methods in order to adaptively adjust the partitions assigned to each UAV as the environment is explored.

Bibliography

[1] Mozhdeh Shahbazi, Jérôme Théau, and Patrick Ménard. Recent applications of unmanned aerial imagery in natural resource management. *GIScience & Remote Sensing*, 51(4):339–365, 2014.

[2] L Geng, YF Zhang, PF Wang, J Jay Wang, Jerry YH Fuh, and SH Teo. Uav surveillance mission planning with gimbaled sensors. In *Control & Automation (ICCA), 11th IEEE International Conference on*, pages 320–325. IEEE, 2014.

[3] Eduard Semsch, Michal Jakob, Dušan Pavlíček, and Michal Pěchouček. Autonomous uav surveillance in complex urban environments. In *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on*, volume 2, pages 82–85. IET, 2009.

[4] Janosch Nikolic, Michael Burri, Joern Rehder, Stefan Leutenegger, Christoph Huerzeler, and Roland Siegwart. A uav system for inspection of industrial facilities. In *Aerospace Conference, 2013 IEEE*, pages 1–8. IEEE, 2013.

[5] Carlos Alphonso F Ezequiel, Matthew Cua, Nathaniel C Libatique, Gregory L Tangonan, Raphael Alampay, Rollyn T Labuguen, Chrisandro M Favila, Jaime Luis E Honrado, Vinni Canos, Charles Devaney, et al. Uav aerial imaging applications for post-disaster assessment, environmental management and infrastructure development. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 274–283. IEEE, 2014.

[6] Paul EI Pounds, Daniel R Bersak, and Aaron M Dollar. Stability of small-scale uav helicopters and quadrotors with added payload mass under pid control. *Autonomous Robots*, 33(1-2):129–142, 2012.

[7] Ali Haydar Göktoan, Salah Sukkarieh, Mitch Bryson, Jeremy Randle, Todd Lupton, and Calvin Hung. A rotary-wing unmanned air vehicle for aquatic weed surveillance and management. *Journal of Intelligent & Robotic Systems*, 57(1-4):467, 2010.

[8] CW Zecha, J Link, and W Claupein. Mobile sensor platforms: Categorisation and research applications in precision farming. *Journal of Sensors and Sensor Systems*, 2(1):51–72, 2013.

[9] Esther Salamí, Cristina Barrado, and Enric Pastor. Uav flight experiments applied to the remote sensing of vegetated areas. *Remote Sensing*, 6(11):11051–11081, 2014.

[10] Dji's agriculture drone takes to the air down on the farm. `http://www.gizmag.com/dji-mg-1-agriculture-drone/40646/`. Accessed: 2016-05-31.

[11] Sebastian Siebert and Jochen Teizer. Mobile 3d mapping for surveying earthwork projects using an unmanned aerial vehicle (uav) system. *Automation in Construction*, 41:1–14, 2014.

[12] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. Probabilistic robotics. 2005, 2005.

[13] Felix Endres, Jürgen Hess, Nikolas Engelhard, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. An evaluation of the rgb-d slam system. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1691–1696. IEEE, 2012.

[14] Jakob Engel, Jorg Stuckler, and Daniel Cremers. Large-scale direct slam with stereo cameras. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 1935–1942. IEEE, 2015.

[15] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[16] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 15–22. IEEE, 2014.

[17] Raul Mur-Artal, JMM Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *Robotics, IEEE Transactions on*, 31(5):1147–1163, 2015.

[18] Abraham Bachrach, Ruijie He, and Nicholas Roy. Autonomous flight in unknown indoor environments. *International Journal of Micro Air Vehicles*, 1(4):217–228, 2009.

[19] Shaojie Shen, Yash Mulgaonkar, Nathan Michael, and Vipin Kumar. Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft mav. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 4974–4981. IEEE, 2014.

[20] J. Sturm, E. Bylow, F. Kahl, and D. Cremers. Dense tracking and mapping with a quadrocopter. In *Unmanned Aerial Vehicle in Geomatics (UAV-g)*, Rostock, Germany, September 2013.

[21] Davide Scaramuzza, Michael C Achtelik, Lefteris Doitsidis, Felice Friedrich, Elias Kosmatopoulos, Alessio Martinelli, Markus W Achtelik, Maria Chli, Savvas A Chatzichristofis, Laurent Kneip, et al. Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in gps-denied environments. *Robotics & Automation Magazine, IEEE*, 21(3):26–40, 2014.

[22] Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Monocular-slam–based navigation for autonomous micro helicopters in gps-denied environments. *Journal of Field Robotics*, 28(6):854–874, 2011.

[23] J. Engel, J. Sturm, and D. Cremers. Camera-based navigation of a low-cost quadrocopter. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[24] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[25] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustmentâĂŤa modern synthesis. In *Vision algorithms: theory and practice*, pages 298–372. Springer, 1999.

[26] Matthew Brown and David G Lowe. Unsupervised 3d object recognition and reconstruction in unordered datasets. In *3-D Digital Imaging and Modeling, 2005. 3DIM 2005. Fifth International Conference on*, pages 56–63. IEEE, 2005.

[27] Ethan Eade and Tom Drummond. Scalable monocular slam. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 469–476. IEEE, 2006.

[28] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067, 2007.

[29] Etienne Mouragnon, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser, and Patrick Sayd. Real time localization and 3d reconstruction. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 363–370. IEEE, 2006.

[30] Hauke Strasdat, José MM Montiel, and Andrew J Davison. Visual slam: why filter? *Image and Vision Computing*, 30(2):65–77, 2012.

[31] Hauke Strasdat, JMM Montiel, and Andrew J Davison. Scale drift-aware large scale monocular slam. In *Robotics: Science and Systems*, volume 2, page 5, 2010.

[32] Hauke Strasdat, Andrew J Davison, JMM Montiel, and Kurt Konolig. Double window optimisation for constant time visual slam. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2352–2359. IEEE, 2011.

[33] Hyon Lim, Jongwoo Lim, and H Jin Kim. Real-time 6-dof monocular visual slam in a large-scale environment. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 1532–1539. IEEE, 2014.

[34] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *Computer Vision–ECCV 2014*, pages 834–849. Springer, 2014.

[35] Ruzena Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):966–1005, 1988.

[36] Frédéric Bourgaul, Alexei A Makarenko, Stefan B Williams, Ben Grocholsky, and Hugh F Durrant-Whyte. Information based adaptive robotic exploration. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 540–545. IEEE, 2002.

[37] Cyrill Stachniss and Wolfram Burgard. Exploring unknown environments with mobile robots using coverage maps. In *Proceedings of the 18th international joint conference on Artificial intelligence*, pages 1127–1132. Morgan Kaufmann Publishers Inc., 2003.

[38] Alexei A Makarenko, Stefan B Williams, Frederic Bourgault, and Hugh F Durrant-Whyte. An experiment in integrated exploration. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 534–539. IEEE, 2002.

[39] Robert Sim and Nicholas Roy. Global a-optimal robot exploration in slam. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 661–666. IEEE, 2005.

[40] Stefan Wenhardt, Benjamin Deutsch, Elli Angelopoulou, and Heinrich Niemann. Active visual object reconstruction using d-, e-, and t-optimal next best views. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–7. IEEE, 2007.

[41] Enrique Dunn and Jan-Michael Frahm. Next best view planning for active model improvement. *Procedings of the British Machine Vision Conference 2009*, pages 53.1–53.11, 2009.

[42] Christof Hoppe, Andreas Wendel, Stefanie Zollmann, Katrin Pirker, Arnold Irschara, Horst Bischof, and Stefan Kluckner. Photogrammetric camera network design for micro aerial vehicles. In *Computer vision winter workshop (CVWW)*, 2012.

[43] Christof Hoppe, Manfred Klopschitz, Markus Rumpler, Andreas Wendel, Stefan Kluckner, Horst Bischof, and Gerhard Reitmayr. Online feedback for structure-from-motion image acquisition. In *BMVC*, volume 2, page 6, 2012.

[44] Inhyuk Moon, Jun Miura, and Yoshiaki Shirai. On-line viewpoint and motion planning for efficient visual navigation under uncertainty. *Robotics and Autonomous Systems*, 28(2-3):237–248, August 1999.

[45] Daniele Fontanelli, Paolo Salaris, FelipeA.W. Belo, and Antonio Bicchi. Visual appearance mapping for optimal vision based servoing. In Oussama Khatib, Vijay Kumar, and GeorgeJ. Pappas, editors, *Experimental Robotics*, volume 54 of *Springer Tracts in Advanced Robotics*, pages 353–362. Springer Berlin Heidelberg, 2009.

[46] Ayoung Kim and Ryan M Eustice. Combined visually and geometrically informative link hypothesis for pose-graph visual slam using bag-of-words. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1647–1654. IEEE, 2011.

[47] Andrew Davison and David Murray. Mobile robot localisation using active vision. *Computer VisionâĂŤECCVâĂŹ98*, pages 809–825, 1998.

[48] Giovanni Bianco and Alexander Zelinsky. Biologically-inspired visual landmark learning and navigation for mobile robots. In *Intelligent Robots and Systems, 1999. IROS'99. Proceedings. 1999 IEEE/RSJ International Conference on*, volume 2, pages 671–676. IEEE, 1999.

[49] Simone Frintrop and Patric Jensfelt. Active gaze control for attentional visual SLAM. *2008 IEEE International Conference on Robotics and Automation*, pages 3690–3697, May 2008.

[50] Christian Mostegel, Andreas Wendel, and Horst Bischof. Active monocular localization: Towards autonomous monocular exploration for multirotor mavs. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3848–3855. IEEE, 2014.

[51] F. S. Hover, R. M. Eustice, a. Kim, B. Englot, H. Johannsson, M. Kaess, and J. J. Leonard. Advanced perception, navigation and planning for autonomous in-water ship hull inspection. *The International Journal of Robotics Research*, 31(12):1445–1464, November 2012.

[52] Ayoung Kim and Ryan M. Eustice. Perception-driven navigation: Active visual slam for robotic area coverage. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 2013.

[53] Samuel Prentice and Nicholas Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *The International Journal of Robotics Research*, 2009.

[54] Blai Bonet and HÃĺctor Geffner. Planning with incomplete information as heuristic search in belief space. pages 52–61. AAAI Press, 2000.

[55] N. Roy, W. Burgard, D. Fox, and S. Thrun. Coastal navigation-mobile robot navigation with uncertainty in dynamic environments. In *IEEE International Conference on Robotics and Automation, 1999.*, volume 1, pages 35–40 vol.1, 1999.

[56] Sam Prentice and Nicholas Roy. Planning in information space for a quadrotor helicopter in a GPS-denied environment. In *2008 IEEE International Conference on Robotics and Automation*, pages 1814–1820. IEEE, May 2008.

[57] Adam Bry and Nicholas Roy. Rapidly-exploring Random Belief Trees for motion planning under uncertainty. In *2011 IEEE International Conference on Robotics and Automation*, pages 723–730. IEEE, May 2011.

[58] Markus W Achtelik, Simon Lynen, Stephan Weiss, Margarita Chli, and Roland Siegwart. Motion-and uncertainty-aware path planning for micro aerial vehicles. *Journal of Field Robotics*, 31(4):676–698, 2014.

[59] Vinay Pilania and Kamal Gupta. A localization aware sampling strategy for motion planning under uncertainty. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 6093–6099. IEEE, 2015.

[60] Mitch Bryson and Salah Sukkarieh. An information-theoretic approach to autonomous navigation and guidance of an uninhabited aerial vehicle in unknown environments. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3770–3775. IEEE, 2005.

[61] Franz S Hover, Ryan M Eustice, Ayoung Kim, Brendan Englot, Hordur Johannsson, Michael Kaess, and John J Leonard. Advanced perception, navigation and planning for autonomous in-water ship hull inspection. *The International Journal of Robotics Research*, 31(12):1445–1464, 2012.

[62] Gabriele Costante, Christian Forster, Jeffrey Delmerico, Paolo Valigi, and Davide Scaramuzza. Perception-aware path planning. *IEEE Transactions on Robotics*, Conditionally accepted, 2016.

[63] Jorge Cortes, Sonia Martinez, Timur Karatas, and Francesco Bullo. Coverage control for mobile sensing networks. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1327–1332. IEEE, 2002.

[64] M. Schwager, Brian J. Julian, and D. Rus. Optimal coverage for multiple hovering robots with downward facing cameras. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3515–3522, May 2009.

[65] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258–1276, 2013.

[66] Esther M. Arkin, SÃąndor P. Fekete, and Joseph S.B. Mitchell. Approximation algorithms for lawn mowing and milling. *Computational Geometry*, 17(1âĂŞ2):25 – 50, 2000.

[67] Howie Choset. Coverage for robotics–a survey of recent results. *Annals of mathematics and artificial intelligence*, 31(1-4):113–126, 2001.

[68] Howie M Choset. *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[69] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 2008.

[70] Timo Oksanen and Arto Visala. Coverage path planning algorithms for agricultural field machines. *Journal of Field Robotics*, 26(8):651–668, 2009.

[71] Howie Choset and Philippe Pignon. Coverage path planning: The boustrophedon cellular decomposition. In *Field and Service Robotics*, pages 203–209. Springer, 1998.

[72] K. Zhou, A. Leck Jensen, C.G. SÃÿrensen, P. Busato, and D.D. Bothtis. Agricultural operations planning in fields with multiple obstacle areas. *Computers and Electronics in Agriculture*, 109(0):12 – 22, 2014.

[73] Alexander Zelinsky, Ray A Jarvis, JC Byrne, and ShinâĂŹichi Yuta. Planning paths of complete coverage of an unstructured environment by a mobile robot. In *Proceedings of international conference on advanced robotics*, volume 13, pages 533–538, 1993.

[74] Yoav Gabriely and Elon Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):77–98, 2001.

[75] E. Galceran and M. Carreras. Planning coverage paths on bathymetric maps for in-detail inspection of the ocean floor. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4159–4164, May 2013.

[76] Andrew Howard, Maja J Matarić, and Gaurav S Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Distributed autonomous robotic systems 5*, pages 299–308. Springer, 2002.

[77] Carlos Franco, David Paesa, Gonzalo Lopez-Nicolas, C Sagues, and Sergio Llorente. Hierarchical strategy for dynamic coverage. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5341–5346. IEEE, 2012.

[78] Islam I Hussein and Dusan M Stipanovic. Effective coverage control for mobile sensor networks with guaranteed collision avoidance. *Control Systems Technology, IEEE Transactions on*, 15(4):642–657, 2007.

[79] Anqi Xu, Chatavut Viriyasuthee, and Ioannis Rekleitis. Efficient complete coverage of a known arbitrary environment with applications to aerial operations. *Autonomous Robots*, 36(4):365–381, 2014.

[80] Sang-Woo Moon and David Hyun-Chul Shim. Study on path planning algorithms for unmanned agricultural helicopters in complex environment. *International Journal of Aeronautical and Space Sicences*, 10(2):1–11, 2009.

[81] Ivan Maza and Anibal Ollero. Multiple uav cooperative searching operation using polygon area decomposition and efficient coverage algorithms. In *Distributed Autonomous Robotic Systems 6*, pages 221–230. Springer, 2007.

[82] JF Araujo, PB Sujit, and JB Sousa. Multiple uav area decomposition and coverage. In *Computational Intelligence for Security and Defense Applications (CISDA), 2013 IEEE Symposium on*, pages 30–37. IEEE, 2013.

[83] Mohsen Mehdizade. *A Decomposition Strategy for Optimal Coverage of an Area of Interest using Heterogeneous Team of UAVs*. PhD thesis, Concordia University, 2012.

[84] JoÃčo Valente, David Sanz, Jaime Del Cerro, Antonio Barrientos, and MiguelAngel de Frutos. Near-optimal coverage trajectories for image mosaicing using a mini quad-rotor over irregular-shaped fields. *Precision Agriculture*, 14(1):115–132, 2013.

[85] Antonio Barrientos JoÃčo Valente and Jaime del Cerro. Coverage path planning to survey large outdoor areas with aerial robots: A comprehensive analysis. In Daisuke Chugo and Sho Yokota, editors, *Introduction to Modern Robotics II*. iConcept Press Ltd., 2012.

[86] Antonio Barrientos, Julian Colorado, Jaime del Cerro, Alexander Martinez, Claudio Rossi, David Sanz, and João Valente. Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots. *Journal of Field Robotics*, 28(5):667–689, 2011.

[87] Peng Cheng, J Keller, and Vijay Kumar. Time-optimal uav trajectory planning for 3d urban structure coverage. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 2750–2757. IEEE, 2008.

[88] L Lin and MA Goodrich. Hierarchical heuristic search using a gaussian mixture model for uav coverage planning. *IEEE Transactions on Cybernetics*, 2014.

[89] Jarurat Ousingsawat and Matthew G Earl. Modified lawn-mower search pattern for areas comprised of weighted regions. In *American Control Conference, 2007. ACC'07*, pages 918–923. IEEE, 2007.

[90] Ghulam Murtaza, Salil Kanhere, and Sanjay Jha. Priority-based coverage path planning for aerial wireless sensor networks. In *Intelligent Sensors, Sensor Networks and Information Processing, 2013 IEEE Eighth International Conference on*, pages 219–224. IEEE, 2013.

[91] Stefano Carpin, Derek Burch, Nicola Basilico, Timothy H. Chung, and Mathias KÃűlsch. Variable resolution search with quadrotors: Theory and practice. *Journal of Field Robotics*, 30(5):685–701, 2013.

[92] S. Carpin, Derek Burch, and T.H. Chung. Searching for multiple targets using probabilistic quadtrees. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4536–4543, Sept 2011.

[93] L. Paull, C. Thibault, A. Nagaty, M. Seto, and H. Li. Sensor-driven area coverage for an autonomous fixed-wing unmanned aerial vehicle. *Cybernetics, IEEE Transactions on*, 44(9):1605–1618, Sept 2014.

[94] Liam Paull, Sajad SaeediGharahbolagh, Mae Seto, and Howard Li. Sensor driven online coverage planning for autonomous underwater vehicles. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2875–2880. IEEE, 2012.

[95] Enric Galceran and Marc Carreras. Efficient seabed coverage path planning for asvs and auvs. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 88–93. IEEE, 2012.

[96] Enric Galceran and Marc Carreras. Coverage path planning for marine habitat mapping. In *Oceans, 2012*, pages 1–8. IEEE, 2012.

[97] Geoffrey A Hollinger, Urbashi Mitra, and Gaurav S Sukhatme. Active and adaptive dive planning for dense bathymetric mapping. In *Experimental Robotics*, pages 803–817. Springer, 2013.

[98] David P Williams. On optimal auv track-spacing for underwater mine detection. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4755–4762. IEEE, 2010.

[99] David P Williams. Auv-enabled adaptive underwater surveying for optimal data collection. *Intelligent Service Robotics*, 5(1):33–54, 2012.

[100] F. Fraundorfer, L. Heng, D. Honegger, G.H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys. Vision-based autonomous mapping and exploration using a quadrotor mav. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012*, pages 4557–4564, Oct.

[101] Jakob Engel, JÂ§rgen Sturm, and Daniel Cremers. Camera-based navigation of a low-cost quadrocopter. In *IROS*, pages 2815–2821. IEEE, 2012.

[102] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *In European Conference on Computer Vision*, pages 430–443, 2006.

[103] Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. Efficient Multi-View Reconstruction of Large-Scale Scenes using Interest Points, Delaunay Triangulation and Graph Cuts. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[104] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, June 2011.

[105] Seyed Abbas Sadat, Kyle Chutskoff, Damir Jungic, Jens Wawerla, and Richard Vaughan. Feature-rich path planning for robust navigation of mavs with mono-slam. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3870–3875. IEEE, 2014.

[106] T.A. Mastor, N. Kamarulzaman, M.M. Abidin, A.M. Samad, K.A. Hashim, I. Maarof, and K. Zainuddin. The unmanned aerial imagery capturing system (uaics) flight planning calculation parameters for small scale format imagery. In *Control and System Graduate Research Colloquium (ICSGRC), 2014 IEEE 5th*, pages 120–124, Aug 2014.

[107] Haitao Xiang and Lei Tian. Development of a low-cost agricultural remote sensing system based on an autonomous unmanned aerial vehicle (uav). *Biosystems engineering*, 108(2):174–190, 2011.

[108] T.A. Mastor, N.A. Sulaiman, S. Juhari, and A.M. Samad. An unmanned aerial imagery capturing system (uaics): A review of flight mission planning. In *Signal Processing its Applications (CSPA), 2014 IEEE 10th International Colloquium on*, pages 129–133, March 2014.

[109] E. Galceran and M. Carreras. Efficient seabed coverage path planning for asvs and auvs. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 88–93, Oct 2012.

[110] Fpv-10x zoom camera (ntsc). http://www.fpvflying.com/products/FPV%252d10X-zoom-camera-%28ntsc%29.html. Accessed: 2016-05-31.

[111] Firefly mv 0.3 mp color usb 2.0 (aptina mt9v022). https://www.ptgrey.com/firefly-mv-03mp-color-usb-20-micron-mt9v022. Accessed: 2016-05-31.

[112] Stephan Weiss, Markus Achtelik, Laurent Kneip, Davide Scaramuzza, and Roland Siegwart. Intuitive 3D Maps for MAV Terrain Exploration and Obstacle Avoidance. *Journal of Intelligent & Robotic Systems*, 61(1-4):473–493, November 2010.

[113] G. a. Hollinger, B. Englot, F. S. Hover, U. Mitra, and G. S. Sukhatme. Active planning for underwater inspection and the benefit of adaptivity. *The International Journal of Robotics Research*, 32(1):3–18, November 2012.

[114] Brendan J Englot. Sampling-based coverage path planning for complex 3D structures, 2012.

[115] Christof Hoppe, Andreas Wendel, Stefanie Zollmann, Katrin Pirker, Arnold Irschara, Horst Bischof, and Stefan Kluckner. Photogrammetric camera network design for micro aerial vehicles. In *Computer Vision Winter Workshop (CVWW)*, volume 8, pages 1–3, 2012.

[116] Shannon V Spires and Steven Y Goldsmith. Exhaustive geographic search with mobile robots along space-filling curves. In *Collective robotics*, pages 1–12. Springer, 1998.

[117] Jordan J Cox, Yasuko Takezaki, Helaman RP Ferguson, Kent E Kohkonen, and Eric L Mulkay. Space-filling curves in tool-path applications. *Computer-Aided Design*, 26(3):215–224, 1994.

[118] H.K. Dai and H.C. Su. On the locality properties of space-filling curves. In Toshihide Ibaraki, Naoki Katoh, and Hirotaka Ono, editors, *Algorithms and Computation*, volume 2906 of *Lecture Notes in Computer Science*, pages 385–394. Springer Berlin Heidelberg, 2003.

[119] Timothy H Chung and Joel W Burdick. A decision-making framework for control strategies in probabilistic search. In *IEEE International Conference on Robotics and Automation*, pages 4386–4393, 2007.

[120] Andrew Symington, Sonia Waharte, Simon Julier, and Niki Trigoni. Probabilistic target detection by camera-equipped uavs. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4076–4081. IEEE, 2010.

[121] Timothy H Chung and Stefano Carpin. Multiscale search using probabilistic quadtrees. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2546–2553. IEEE, 2011.

[122] Seyed Abbas Sadat, Jens Wawerla, and Richard T Vaughan. Recursive non-uniform coverage of unknown terrains for uavs. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 1742–1747. IEEE, 2014.

[123] Seyed Abbas Sadat, Jens Wawerla, and Richard Vaughan. Fractal trajectories for online non-uniform aerial coverage. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2971–2976. IEEE, 2015.

[124] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning.* Adaptative computation and machine learning series. University Press Group Limited, 2006.

[125] Alkis Gotovos, Nathalie Casati, Gregory Hitz, and Andreas Krause. Active learning for level set estimation. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 1344–1350. AAAI Press, 2013.

[126] Carmelo Di Franco and Giorgio Buttazzo. Energy-aware coverage path planning of uavs. In *Autonomous Robot Systems and Competitions (ICARSC), 2015 IEEE International Conference on*, pages 111–117. IEEE, 2015.

[127] Jonathan Binney, Andreas Krause, and Gaurav S Sukhatme. Informative path planning for an autonomous underwater vehicle. In *Robotics and automation (icra), 2010 IEEE international conference on*, pages 4791–4796. IEEE, 2010.

[128] Marina Torres, David A Pelta, José L Verdegay, and Juan C Torres. Coverage path planning with unmanned aerial vehicles for 3d terrain reconstruction. *Expert Systems with Applications*, 55:441–451, 2016.