Rectilinear polygon partitioning for region coverage using UAVs

Amit Agarwal

2008

Amit Agarwal. (2008). Rectilinear polygon partitioning for region coverage using UAVs. Doctoral thesis, Nanyang Technological University, Singapore.

https://hdl.handle.net/10356/41780

https://doi.org/10.32657/10356/41780

Rectilinear Polygon Partitioning For Region Coverage Using UAVs

AMIT AGARWAL

School of Electrical and Electronic Engineering

A thesis submitted to Nanyang Technological University

in fulfillment of the requirement for the degree of

Doctor of Philosophy

2008

ACKNOWLEDGMENTS

I thank Professor Lim Meng-Hiot for his continued belief in my abilities and for the freedom of thought he encouraged. He encouraged a collaborative environment in which I got the opportunity to work with other members of his research group and his undergraduate students. Without this opportunity, implementing and computer testing of various ideas or algorithms I developed during the course of my candidacy would have been considerably more challenging. His constant encouragement to publish has been a major contributing factor to the number of publications my PhD work has been able to generate. I continue to be blessed by his support of my professional endeavors and his research guidance and I assume the responsibility for the resulting works.

Professor Lim's generous support of my research and of my participation in international conferences out of his own research funds facilitated my interactions with a wider research community. It additionally made Harvin and my life more comfortable during my candidacy period!

My PhD work additionally received the technical and financial support of Singapore Technologies (ST) Engineering Pte Ltd. Discussions with ST staff members, especially those with Yew Kong Leong, Irving Tjin, David Chin and Lam Chian Poh were particularly useful in identifying directions of work that would potentially also add value to the UAV program of ST Engineering. These discussions also helped in my increased understanding of systemic issues in reconnaissance using UAVs. Yew Kong Leong channelized generous funding to the project I worked on and I appreciate his advice on issues relating to potential commercialization of my research effort.

Table of Contents

Figures

Summary

Low altitude Unmanned Aerial Vehicles (UAVs) are cheap, offer flexibility in deployment and stealthier than manned aircrafts due to their smaller size. They are well-suited for missions such as damage assessment and airbase or urban reconnaissance which require low-altitude flights. Their limited altitude of operation allows the use of low-cost lightweight imaging payload and allows them to avoid occlusions due to line-of-sight or the presence of clouds.

Several UAVs can simultaneously perform region coverage in time-critical scenarios or when the endurance of a single UAV is insufficient. Parallel usage of UAVs also reduces the duration of threat exposure.

The use of multiple UAVs for low-altitude reconnaissance introduces two challenges that are absent when a single high-altitude UAV is used for reconnaissance – (i) *Workspace Division*: when multiple UAVs are used, the workspace must be divided amongst them such that the workload of each UAV is proportional to its relative capability for region coverage. (ii) *Waypoint Planning*: A high-altitude UAV has a correspondingly large footprint and it can thus complete the reconnaissance mission by photographing from a few vantage points. For low-altitude reconnaissance the waypoint planning problem is important and needs to be tackled to generate efficient paths.

This thesis is motivated by the former problem – workspace division between multiple UAVs to complete coverage in minimum time. Our problem-model makes the following assumptions:

1. η UAVs are available to cover the workspace. Each UAV carries a sensor with a well-defined footprint.

2. Rates at which UAVs can conduct coverage are known *a priori*.

3. The workspace is a rectilinear polygon $\wp$ with N edges. It may contain rectilinear holes – portions in the workspace that need not be covered.

4. The workspace is known prior to the beginning of the mission.

5. The size of the sensor footprint is much smaller than the size of the region to be covered.

6. We assume that no UAV may fail during its mission.

The time needed for each UAV to conduct coverage is lower if UAVs are assigned fractions of workspace proportional to rates at which the UAVs can perform region coverage. Therefore, we focus on the problem of polygon partitioning.

Our main contribution is in workspace partitioning. We provide formulation of algorithms which guarantee solutions in low order polynomial time. From an applications point of view, such algorithms can provide useful extension towards generating solutions in real-time or can serve as a basis for further refinements. Our novel contributions are stated below:

1. We give an algorithm to divide the workspace $\wp$ into η rectilinear, contiguous pieces whose areas are proportional to rates at which the UAVs can perform region coverage. We prove that out algorithm runs in $O(N \log N + \eta N)$ time.

2. Given workspace $\wp$ and points on its boundary where UAVs must begin coverage, we give an algorithm to divide $\wp$ into η rectilinear pieces. The pieces are contiguous with areas proportional to the rates at which the UAVs can perform region coverage and include start points of UAVs assigned to the particular pieces. We prove that our algorithm runs in $O(N \log N + \eta^2 N)$ time.

Symbols and Acronyms

N	–	Number of sides (edges) in a polygon
N_h	–	is the horizontal complexity of $\wp$. This is equal to the number of vertical edges in $\wp$
N_v	–	is the vertical complexity of $\wp$. This is equal to the number of horizontal edges in $\wp$
A	–	Total area of the workspace
C_i	–	Cost of covering a part of the workspace assigned to UAV$_i$.
η	–	Number of UAVs available for region coverage
$\wp$	–	The workspace that must be covered using multiple UAVs
ω^2	–	Area of sensor footprint
ACO	–	Ant colony optimization
DFS	–	Depth first search
GAs	–	Genetic algorithms
HALE UAV	–	High altitude long endurance UAV
LALE UAV	–	Low altitude long endurance UAV

LOS	–	Line of sight
MILP	–	Mixed integer linear programming
RAG	–	Region adjacency graph
RAT	–	Region adjacency tree
UAV	–	Unmanned aerial vehicle
UCAV	–	Unmanned combat aerial vehicle

This page has been intentionally left blank.

Chapter 1

Introduction

1.1 Unmanned Aerial Vehicles

An unmanned aerial vehicle (UAV) [1, 2] is a powered recoverable and reusable aircraft flown under the guidance of a remote operator or an onboard sensor-based navigation system without the presence of an onboard crew. UAVs have been referred to by several names in the literature – drones, pilotless aircrafts, remotely operated vehicles and unmanned combat aerial vehicles. Two examples of mini-UAVs are as shown in Figure 1.1.

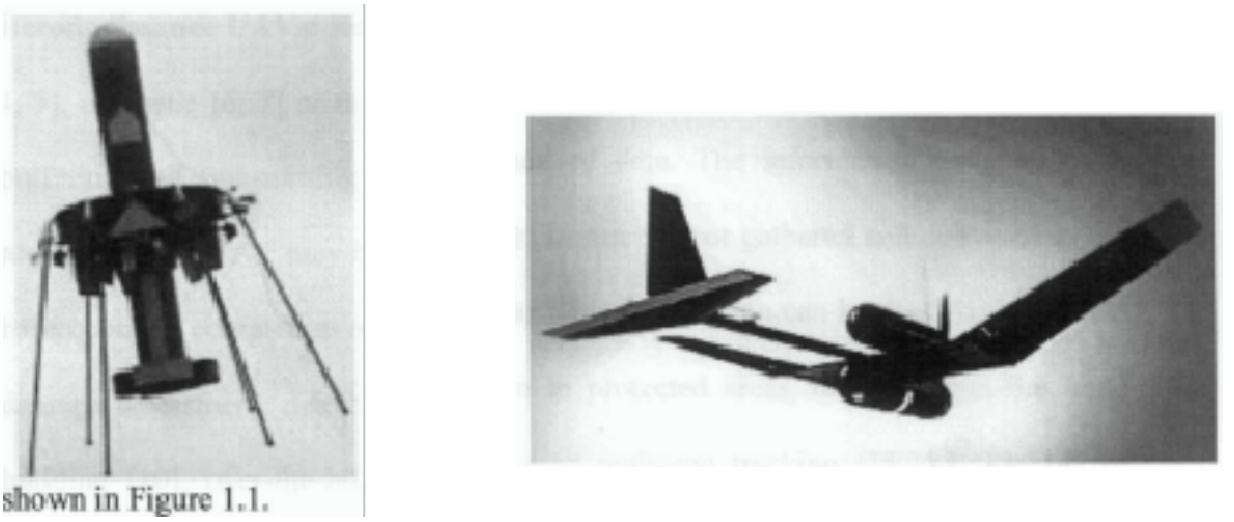


Figure 1.1: Mini UAVs.

1.1.1 Types of Unmanned Aerial Vehicles

UAVs are mainly classified by their endurance or their mission types. UAV types classified by their endurance include micro aerial vehicles, mini aerial vehicles, low altitude long endurance (LALE) UAVs and high altitude long endurance (HALE) UAVs. An alternative classification is based on the nature of the mission of the UAV. The following are the main roles performed by UAVs.

Aerial Artillery Practice UAVs: UAVs in this category are flown along a predetermined or a dynamically computed path to provide as targets in aerial gunnery practice. These UAVs are also referred to as drones. They are rugged, have fire protection systems and typically have low endurance and do not carry sophisticated sensor systems. A trailing floatation object such as a piece of cloth or a light dummy may be attached to their tail. This object constitutes the target for gunnery or artillery practice. UAVs for testing ground-to-air missiles however are usually the target themselves and thus have a typical useful lifespan of only one mission.

Reconnaissance UAVs: Reconnaissance UAVs may carry acoustic, electro-optical [3, 4, 5], magnetic [6, 7] or thermal sensors [8, 9]. They are equipped with systems for buffering and transmitting large amount of data. The information gathered by a reconnaissance UAV may be transmitted in run time or gathered and delivered to the base upon the completion of the mission. This information can be used for battlefield damage assessment, detecting intrusion in protected areas, directing gun-fire, map building [10], over-the-hill intelligence or real-time tracking [11, 12, 13, 14] of moving targets.

Search and Rescue UAVs: Search and rescue UAVs are mainly used for civilian applications to aid in disaster management efforts. The task of these UAVs include locating survivors, assessing transportation networks, dropping food and other relief supplies and directing the movement of manned rescue teams. Search and rescue UAVs typically have low stall speed, have specialized sensors and may benefit from having hovering capability. These UAVs may also double up as relay points to rapidly set up a local communications network.

Unmanned Combat Aerial Vehicles: Unmanned combat aerial vehicles (UCAVs) can operate in high risk environment and carry defensive and offensive weapons. They generally operate in group and work with dynamic information sharing amongst members of the group as well as with the command center. Key challenges in the further development of UCAVs include autonomous target recognition, waypoint planning, obstacle avoidance, reduction in electronic or acoustic signatures, fault tolerance to lossy or jammed communications link and autonomous combat maneuver capabilities.

1.1.2 Degrees of UAV Autonomy

UAV autonomy refers to the ability of a UAV to perform functions including sensing, verifying, classifying and transmitting information, computing energy efficient or time efficient paths [15, 16, 17, 18], obstacle avoidance [19, 20, 21, 22] and landing and takeoff [23, 24] operations that are pertinent to the mission. These functions must be ideally performed by the computer onboard the UAV using inputs from stored onboard data, data acquired during the mission using onboard sensors and data communicated from the base station. The simplest form of UAV autonomy is exhibited in a radio controlled (RC) aircraft. The lift generating surfaces and engine

operation parameters of the RC aircraft are controlled using radio signals generated by a transceiver operated by a human. The transceiver must be in line of sight (LOS) of the aircraft at all times. Some RC aircrafts may have additional, though scripted, navigation functions such as waypoint following along fixed types of trajectories (e.g. moving between two points along a straight line or changing velocity vector along Dubins curves). On the other extreme is current generation UCAVs [2] that are equipped with sophisticated onboard computing systems and which use advanced algorithms for processing information and distributed decision making.

1.1.3 Technologies for UAV Autonomy

Technologies for enabling onboard autonomy for UAVs fall under the following broad categories:

Cooperative Tracking: The problem of cooperative tracking [11, 25, 26] involves the use of one or more UAVs to continuously track one or more ground units for the duration of the mission or until some mission performance parameters are satisfied. Cooperative tracking (generally) improves tracking since the target(s) may not be visible from a single UAV at all times or one or more UAVs might fail during the mission. Moreover, cooperative tracking allows tracking over a wider area since not all UAVs need to be within direct communication range of the base station at all times. Problems in cooperative tracking include the problems of track initiation (which is closely related to the problem of object recognition), track maintenance, recovery of lost tracks, target behavior estimation and track fusion.

Energy Efficient Path Planning: The endurance of UAVs can be significantly enhanced, particularly if lower average speed of flight and larger variance in flight-

time can be tolerated, by careful path planning that enables the UAV to leverage the additional lift or thrust offered by conducive air currents or rising thermals [15, 16, 17, 18]. The key challenges in enabling the leverage involve computing size and velocity gradient of the air current from simple onboard measurements, integrating weather predictions with path planning, computing optimal paths through air velocity gradients, dynamic autonomous engine control and optimal aircraft fuselage design [27].

Job Allocation and Job Scheduling: The problems of job allocation and scheduling arise when multiple UAVs are used to efficiently perform spatially or temporally distributed tasks in a single mission. Traditional strategies such as mixed integer linear programming (MILP) [28] must be adapted to take into account the inherently dynamic nature of the problem and deal with mission uncertainties.

Motion Planning: The problem of motion planning [29] involves the generation of feasible waypoints and determination of an efficient trajectory between consecutive waypoints. The path planning problem may be additionally complicated by timing constraints, no-fly zone limitations and the presence of ground threats and enemy radars [30]. Intelligent motion planning is an important component for enabling a UAV to execute autonomous combat maneuvers and interoperate with manned aircrafts in shared airspaces.

Obstacle Avoidance: The problem of obstacle avoidance involves two components – avoiding collisions with manned aircraft [31, 32, 33, 34], obstacle avoidance amongst UAVs [20] and avoidance of collision with environmental obstacles such as trees, hills and manmade structures [21, 22]. Approaches to obstacle avoidance include distributed algorithms for collision avoidance, control-theoretic solutions, computing

paths from maps that are guaranteed obstacle free and game-theoretic pursuit-evader problems.

Rendezvous Planning: UAVs can improve their mission performance by acting in groups. Each UAV performs its functions in a preplanned or in an airspace that is computed online. However, they must rendezvous at certain times during their mission (for example, to share information, or to attack a site cooperatively). The problem of rendezvous planning [35] is constrained by the limits on speeds of each UAV, the minimum distance each UAV must travel to the rendezvous point and the potential presence of threats along potential feasible flight paths for rendezvous by each UAV.

Sensor Fusion: Sensor fusion [36, 37] is the process of combining information about a system received from disparate sources to obtain more complete or more accurate information about the system than would have been possible if only one of the sensors was used. It is an active topic of research in the robotics community and the problem of sensor fusion lies at the heart of distributed decision making, target recognition, target localization and collaborative target tracking.

Enabling autonomous operations for UAVs and development of miniaturized aircraft subsystems for UAVs are perhaps the two most important areas of research focus in the UAV community. Our work falls in the former class of endeavor – we give two rectilinear polygon area partitioning algorithms that allow optimal allocation of the workspace between multiple UAVs involved in the task of reconnaissance over an *a priori* defined workspace for static targets.

1.2 Motivations

Low endurance UAVs are relatively cheap and maneuverable, offer great flexibility in deployment and have small electronic signatures. They are well-suited for missions such as damage assessment and airbase or tactical urban reconnaissance which require low-altitude flights. Additionally, their limited altitude of operation allows the use of low-cost lightweight imaging payload and allows them to avoid potential occlusions due to the presence of clouds by simply flying below them. The area of the footprint of the sensor for UAVs operating at low-altitudes is typically much smaller than that of the region to be photographed. A mosaic must therefore be constructed from successive images taken during flight. As shown in Figure 1.2, the problem of determining the path of the UAV (or that of the footprint) that allows efficient capture of images is the region coverage problem. More formally,

> *The region coverage problem is the problem of finding an ordered list of waypoints and the geometry of path between consecutive waypoints for centroid of a sensor footprint so that it can efficiently trace a minimal superset of the region of interest.*

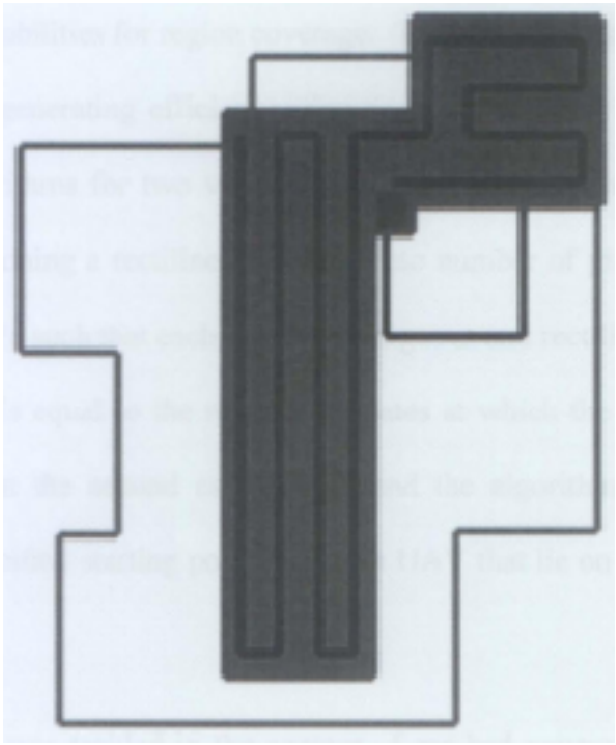


Figure. 1.2: A rectilinear polygon covered by a sensor with a rectangular footprint.

Our goal is to enable the autonomous generation of waypoints for coverage in the case when multiple UAVs with heterogeneous UAVs are used to simultaneously cover the region of interest that is contiguous and whose boundaries have been obtained in advance. Thus the map building problem need not be tackled prior to assigning the UAVs for coverage. The boundaries of the region of interest may be acquired from a high altitude aircraft or may be defined using known information regarding estimations of self-actions or adversary actions. We assume that the region of interest is rectilinear since, (i) structures of reconnaissance interests for us such as urban environments, and airbases are often rectilinear, (ii) operator fatigue is significantly reduced if the image that is being developed as a mosaic in run-time grows in a constant direction.

Even before efficient paths can be generated for the UAV, the region of interest must be divided amongst the UAVs such that the UAVs have coverage work proportional

to their relative capabilities for region coverage. This thesis focuses on this part of the larger problem of generating efficient region coverage routines for multiple UAVs. We give two algorithms for two variants of this problem. In the first case, we are interested in partitioning a rectilinear polygon into number of pieces that is equal to the number of UAVs such that each piece is contiguous and rectilinear and the ratio of area of the pieces is equal to the ratio of the rates at which the UAVs can perform region coverage. In the second case, we extend the algorithm to yield partitions anchored at prespecified starting points for each UAV that lie on the boundary of the workspace.

A similar problem was tackled in the context of sea-bed coverage using a remotely operated tethered autonomous submersible vehicle in [38, 39, 40]. Their basic approach involves partitioning a nonconvex polygon into convex pieces without the introduction of Steiner points and then using a sweep-line approach to divide the convex pieces in the required ratio [40]. The sweep-line approach is only used if the area of adjacent convex pieces is larger than the required area. The main difference in our results is that in our case, we obtain partitions with rectilinear enclosures in case the workspace is rectilinear.

1.3 Problem Significance

The problem we consider in this work has its basis in our work on usage planning of UAVs for reconnaissance. General discussions are therefore contextualized on this basis even while the polygon area partitioning algorithm we give can be extended to

solve a similar problem in robotic demining, aerial insecticide spraying and sea-bed mapping.

Low endurance (typically, an hour or less) UAVs are relatively cheap and often ideal for dangerous or tedious missions such as battlefield damage assessment, airbase reconnaissance and assessment of damage during natural disasters. In the former cases, low level flights are often desirable to help avoid electronic detection. Additionally, flying at low altitudes helps avoid occlusions due to cloud cover and enables the capture of images with acceptable signal-to-noise ratios (SNRs) despite the use of low-cost, light-weight image capturing systems. On the other hand, it restricts the field of view (FOV – its area is directly proportional to the square of the height of the sensor above the area being photographed) and increases potential for occlusions due to ground objects, especially in hilly or urban zones. One way to deal with the latter problem is to use a sensor with a FOV oriented downwards. The footprint of such a sensor can be regarded as rectangular or circular in shape (footprint of a forward looking sensor is a projection of these shapes onto an inclined plane).

The area of sensor footprint is typically much smaller than the area of the workspace. A mosaic must therefore be built using successive images taken during the flight. The question of geometry of the flight path that allows efficient capture of the images leads us to the region coverage problem.

This problem has diverse applications in areas such as demining [41], milling [42], reconnaissance [43], sea-bed mapping [38] and vacuum cleaning [44]. In case several UAVs are available for coverage, it is natural to ask how the workspace may be divided amongst them to complete the task in minimum time.

Airbase Reconnaissance

Airbases include facilities for storage of fuel, spares, weapon systems and housing of expensive communications equipment. They are heavily protected by air defense and ground support units. A functional runway or pad is essential for the operation of most (manned or unmanned) aircrafts. Therefore airbases constitute attractive targets. Following an attack, fast and accurate assessment of damage to airbase-assets is necessary. Completing the assessment in as little time as possible can enhance the utility of the information gathered and minimize the duration of risk exposure to the UAV. Critical portions of the airbase such as runways, fuel storage units, air traffic control towers and weapon storage areas may be first surveyed to minimize the duration of the mission. Additionally, determining the boundaries of air-deployed minefields and the locations and types of unexploded ordinances [45] may be necessary if a friendly airbase was attacked.

The region that must be covered by a UAV can be identified by a high-altitude aircraft or determined from the map of the terrain or asset locations and estimates of operational goals. Once the region of interest is defined, a set of waypoints that lead to an efficient flight path for the UAV must be computed. This problem is reducible to the problem of finding a Hamiltonian cycle in a uniform grid and is NP-hard in the general case. Nevertheless near-optimal solutions can be easily obtained if the altitude from which reconnaissance is conducted is 'sufficiently' high. What may be taken as a sufficiently high altitude depends on the sensor's angle of view and the size of the region of interest. For airbase reconnaissance, a value of 1000 meters above ground level (AGL) may be considered as sufficiently high. Obtaining near-optimal waypoints for a high altitude UAV involves computing the locations of interior

disjoint rectangles whose union is a minimal superset of the region of interest. High-altitude reconnaissance affords the additional benefit of reduced risk to the UAV due to shortened mission time and lowered effectiveness of ground based threats.

Despite the advantages that high-altitude airbase reconnaissance offers, low-altitude flights serve potentially several useful purposes and may even be the only option if the reconnaissance UAV has severely limited payload and endurance capabilities.

Strong Winds: Strong winds can degrade the performance of positioning and navigation unit of a UAV. This is particularly problematic for UAVs that fly unaided by GPS or carry only a crude GPS unit. Strong headwinds or crosswinds lead to higher fuel burn rate which in turn reduces the useful endurance of the UAV. Strong crosswinds can additionally cause uncontrollable and unacceptable deviations in the UAVs flight path. Finally, strong gusts can severely degrade the quality of pictures taken by unstabilized sensors that are often used with small UAVs. Low-altitude (about 100 meters AGL or less) reconnaissance is thus attractive since in a given region the wind speed is typically lower closer to the ground due to the availability of shorter mean obstacle-free path and the effects of viscosity.

Dust and Smoke: Dust and smoke adversely affect visibility. The impact is most severe for real-time reconnaissance (reconnaissance activity that is concurrent with an attack) or reconnaissance immediately following an attack. The use of radar payloads that offer dust/smoke penetration is precluded by the payload weight and space limitation of a typical tactical UAV. Thus, pictures may be taken only with cameras operating in the visible band. Figure 1.3 shows two pictures of a damaged tank taken from a rotorcraft by a wide-angle camera operating in the visible band. The first was taken from a larger distance than the second picture. The second picture illustrates the

advantage of low-altitude reconnaissance, especially in cases involving high concentration of dust in the air and soot or smoke due to the presence of several battle-damaged units.

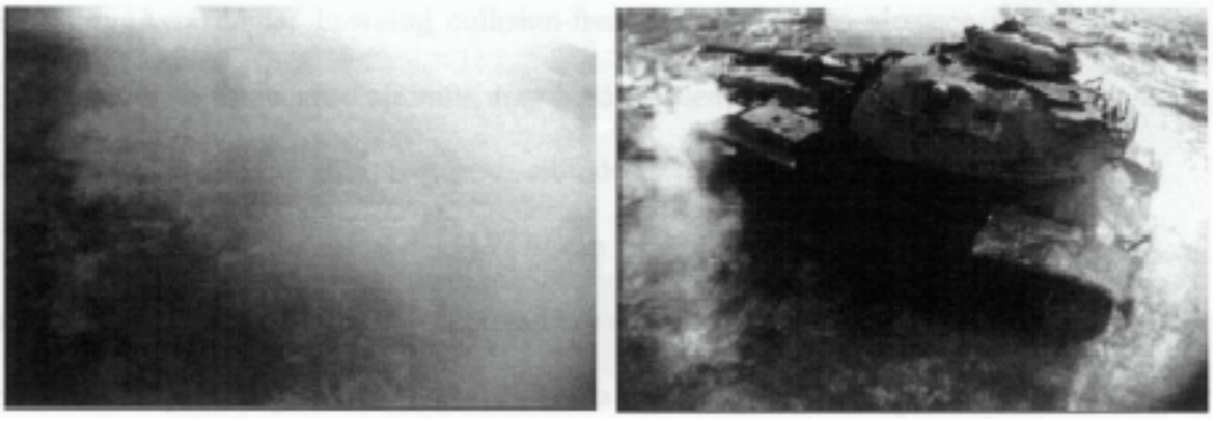


Figure 1.3: Images of a battle damaged unit – a tank taken from two different heights.

Cloud Cover: The presence of clouds between a UAV and the region of interest can severely limit the utility of a reconnaissance mission by a UAV that uses sensors operating in the visible-band of the electromagnetic spectrum. Cloud or fog penetrating radars cannot be used aboard tactical UAVs due to their weight. One way to improve 'all weather capability' is to conduct reconnaissance from under the lowest cloud cover. The resulting benefit comes at the expense of a reduction in the size of the sensor footprint In Singapore, the cloud cover belly can be at 100 meters AGL or less. This implies a maximum sensor footprint width of 82 meters for a 45° angle of view CCD array camera.

Range-limit of Ground Based Threats: Tactical UAVs are unattractive targets for seeker-assisted weapons. Their main source of threat is ground based light arms fire. The range of these threats is usually limited in the vertical direction to a few hundred meters since tactical (especially, battery powered) UAVs are difficult to detect with an

unaided eye due to their small sizes and the camouflaging color of their fuselage. They can thus fly low and obtain higher resolution pictures without running significant risks of being shot down.

Situation Awareness: Ensuring collision-free operation in an airspace shared by UAVs and friendly manned aircrafts may need frequent intervention from the remote pilot especially when the requisite altitude separation cannot be guaranteed. Uninterrupted visibility of the UAV(s) can significantly enhance the operator's situation awareness and thereby decrease the chance of collision. In reconnaissance over relatively flat regions such as an airbase a lower flight altitude generally leads to improved visibility and lesser operator fatigue.

Detection of Mines or Unexploded Ordinance: UAVs are a faster and perhaps, a safer alternative to ground mobile sensors for the determination of the boundaries of air deployed mines or the locations and types of unexploded ordinances (UXO). Magnetic sensors are particularly useful for detecting partially buried mines or UXO that are camouflaged by snow, sand or background color. A UAV with a payload capacity of about 10 kg can carry the sensors. Since magnetic field strength varies with the inverse cube of the distance and powerful sensors may weigh more, register greater noise from other metallic battlefield objects and even trigger off blasts, a UAV may need to fly only a few meters AGL to conduct magnetic reconnaissance leading to a swathe width of only a few meters [6].

The common trait among the several advantages afforded by low level reconnaissance is a reduction in the size of the sensor footprint. Our algorithmic efforts for autonomous airbase reconnaissance are guided by the view that it is accomplished via stitching of successive snapshots taken of a relatively much larger region of interest.

The time needed for completing the mission is a critical factor in most military reconnaissance scenarios. Additionally, the useful endurance of tactical UAVs can be significantly enhanced by careful flight path planning. The number and the type of turns a UAV must make to cover a given contiguous region and the total length of repeated coverage of any portion of the region of interest along with information of the flight dynamics characteristics and the flight speed of the UAV serves as a good indicator of the time and fuel needed to complete the mission.

Assessment of Damage due to Air-to-Ground Fire

Strafing is an air to ground operation in which short rounds of light ammunitions such as bullets, cannons or rockets are fired at a target. Due to the speed of the aircraft, the type of ammunitions used and the manner in which they are used, strafing damage tends to have a high spatial spread in the direction of the instantaneous path of the aircraft. In [43] we gave an ant colony optimization (ACO) based solution to the problem of conducting assessment of strafing damage using a UAV-mounted camera with a small footprint.

We assume the boundary of the region of interests is estimated using information or by a high altitude aircraft. The region of interest is approximated as a union of rectangles. In order to take advantage of the typical spatial distribution of strafing damage in computing efficient coverage paths, we divide each rectangle into a set of strips of width equal to the size of the sensor footprint. Each strip is covered by the movement of the centroid of the sensor footprint between the ends of the longitudinal axis of the strip. The problem of finding an optimal coverage path is then to decide the sequence in which the strips must be covered (covering all strips implies completion of the damage assessment task). We showed in [43] that this problem is a symmetric,

triangular non-Euclidean traveling salesman problem. We gave an ant colony optimization (ACO) based solution to the problem whose cost function accounted for nonholonomic limitations of the UAV. The goal is to find a permutation on the strips that minimizes the mission time.

Figure 1.6 comprises two diagrams. In the first, a set of strips that must be covered and the set of all possible sequence of Dubins coverage paths are shown. In the second diagram, an optimal flight path obtained using an ant colony optimization heuristic is shown.

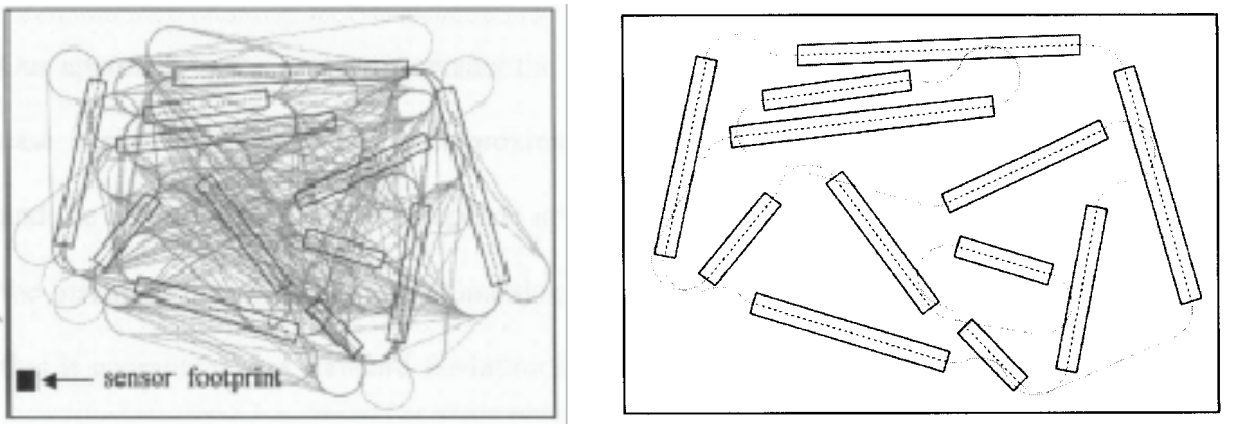


Figure 1.4: Strips and an optimal coverage path for a Dubins robot discovered using ACO.

Aerial Insecticide Spraying

Aerial dusting is the operation in which large tracts of land are sprayed with insecticide, fungicide or fertilizer from a dispensing unit mounted on an aircraft. The problem of aerial insecticide spraying involves the deposition of insecticide by a dispenser onboard a UAV. UAVs offer the fastest way to administer insecticides to crops in large farms. However, several crops may be grown on a single farm, the

crops may be in different stages of growth and water bodies and animal sheds are often located within a farm. Thus, more than one type of insecticide is needed. At any time, only one type of insecticide is mounted on the UAV. In order to achieve a constant-rate deposition, the speed of the UAV and the insecticide flow rate are kept constant. The areas over which the particular insecticide must not be sprayed are identified as rectilinear polygonal holes.

The goal in aerial insecticide spraying is essentially the same as in airbase reconnaissance. However, for insecticide spraying, minimizing length of repeated coverage path is more important than minimizing turn-costs since repeated deposition of insecticide can damage the crop.

Our approach to the problem remains the same. However, since the footprint in this case is not well-defined but an approximate, two dimensional skewed (the kurtosis and the degree of its skewness depends upon the flight vector of the UAV relative to the instantaneous wind velocity) Gaussian, the flight path of the UAV must be such that it guarantees that standard deviation in the deposition does not exceed a certain predefined limit. Results from [4] may be used to obtain optimal spacing between parallel adjacent sweeps and the optimal speed of the UAV for ensuring minimal variations in the standard deviation of the deposition density.

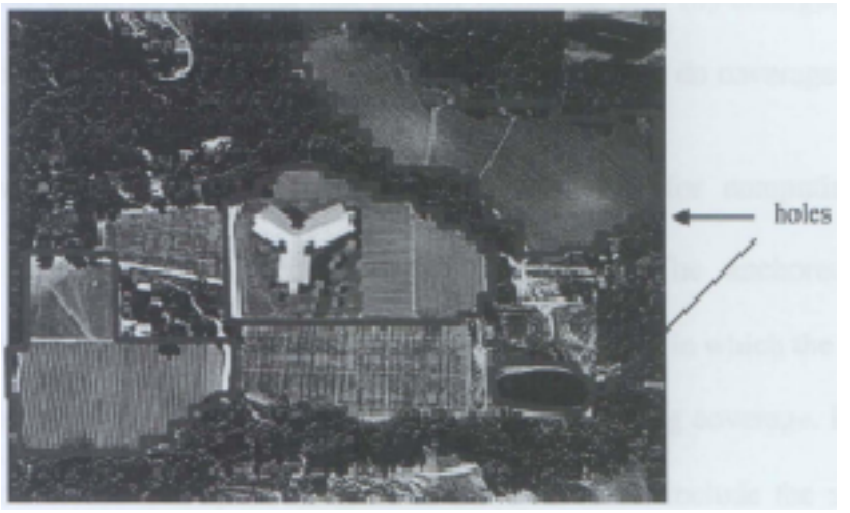


Figure 1.5: A farm showing regions that may be sprayed upon with insecticides and regions over which insecticide spray could potentially harm

1.4 Objectives and Contributions

This work has two objectives:

1. For a planar rectilinear contiguous workspace with holes, $\wp$, divide $\wp$ into η interior-disjoint pieces so that each piece can be covered by exactly one of the η available UAVs. The pieces should each be contiguous and rectilinear and the cost C_i of covering a piece using UAV_i should be proportional to the relative rate at which the UAV can do coverage.

2. For η points pre-selected on the boundary of $\wp$, give an algorithm that generates partitions that additionally satisfy the requirement that each partition includes exactly one of the η points in its closure.

Given a planar contiguous possibly holed rectilinear polygon $\wp$, we give an $O(N \log N + \eta N)$ algorithm to partition it into η interior-disjoint polygons, P_i such

that the partitions comprise polygons that are (*a*) rectilinear and (*b*) contiguous (*c*) whose area-ratios equal to the ratio of rates at which the UAVs can do coverage.

Additionally, we give an $O(N \log N + \eta^2 N)$ time algorithm for computing the anchored area partitions for a rectilinear holed polygon. The anchored area partitioning problem models workspace division for UAVs in case in which the UAVs take off from specific points (such as an airbase) prior to beginning coverage. In such cases, the portion of the workspace assigned to the UAV must include the starting point of the UAV or a point that is nearest to the starting point of the UAV so as to conserve time and fuel and time. The algorithm partitions the polygon into η interior-disjoint polygons, P_i such that the partitions comprise polygons that are (*a*) rectilinear and (*b*) contiguous (*c*) whose area-ratios equal to the ratio of rates at which the UAVs can do coverage (*d*) and whose boundaries include points at which the respective UAV begins coverage.

1.5 Organization of the Thesis

In chapter 2 we give the problem assumptions and model for solving our polygon area partitioning problem. In Chapter 3 we discuss strengths and weaknesses of related past work when viewed from the perspective of our application of interest – workspace division for maximal parallel utilization of η UAVs for reconnaissance of contiguous rectilinear workspaces. In Chapter 4 we prove that the minimum grid of a rectilinear polygon can be computed in polynomial time. We give an algorithm for computing the same. Computing the grid is helpful in path planning once the workspace division has been completed. Chapter 5 gives the polygon area partitioning algorithm. In Chapter 6, we prove our algorithm runs in $O(N \log N + \eta N)$ time. In

Chapter 7 we extend the algorithm to the anchored area partitioning problem. We conclude with a pointer for future work in Chapter 8.

Chapter 2

Problem Model

Consider η UAVs for coverage of a contiguous holed rectilinear workspace $\wp$. The workspace is assumed rectilinear since rectilinear polygons can effectively approximate regions of coverage interest such as airbases and urban zones. Additionally, flight control can be simplified if the UAV must execute only 90° or 180° turns. Finally, any polygon can be represented using convex polygons and then an efficient rectilinear approximation can be generated as shown in [49].

We assume that each UAV is equipped with the same type of sensor and can begin coverage at any point in $\wp$. During the entire coverage mission, at least a part of the instantaneous sensor footprint of each UAV lies in $\wp$. Figure 2.1 shows a minimal union of sensor footprints that covers the workspace. Note that some footprints are only partially in the interior of the workspace.

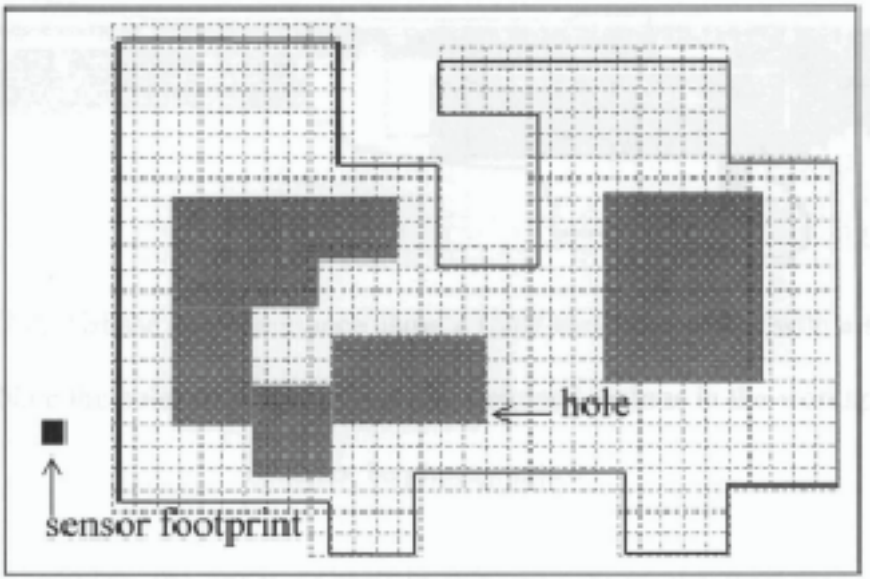


Figure 2.1: A typical workspace, $\wp$ and its minimal augmented grid.

We assume that the capability of a UAV to do region coverage, measured in terms of the rate at which it can do coverage Γ, can be precisely quantified *a priori*. This quantity may be different for each UAV and it depends on factors such as cruise speed and the noise rejection capability of onboard image capturing systems. Finally, we assume that the mission altitude is sufficient to permit the UAV to fly at constant altitude despite height variations in the terrain or that in the ground objects in the region of interest. The region $\wp$ may have an arbitrary number of holes (see Figure 2.2).

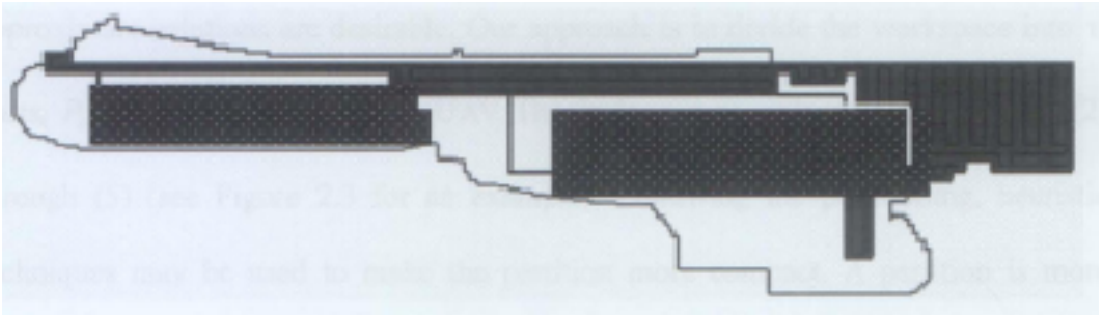


Figure 2.2: Airbase reconnaissance using a UAV-mounted sensor with a small footprint. Note the holes indicated as heavily darkened regions in the workspace that need not be photographed.

In order to complete the coverage task in minimum time, the workspace must be divided among the UAVs such that the time needed by each, while flying at optimal speed and altitude, to cover its respective part is equal. The expected time spent in making a 90° and a 180° turn can be computed in advance and, along with the total length of the straight-line segments of the flight path, l, serves as a good basis for estimating the time needed by a UAV to complete coverage. The cost of covering a part of the workspace assigned to UAV$_i$ is thus denoted as $C_i(n_{90^\circ}, n_{180^\circ}, l)$ or, C_i in short; n_{90° and n_{180° are respectively the numbers of 90° and 180° turns UAV$_i$ makes in an optimal coverage plan. Thus, if the relative capabilities of the UAVs for region coverage, as measured by the rate at which they can do region coverage, is $\Gamma_1, \Gamma_2, ..., \Gamma_\eta$ then, a partition is optimal if it satisfies

$$C_1 : C_2 : ... : C_\eta = \Gamma_1 : \Gamma_2 : ... : \Gamma_\eta \quad (1)$$

The problems of *a.* minimizing the number of turns (even when turn angles are restricted to 90° or 180°) in a holed rectilinear workspace of the workspace and, *b.* minimizing the total length of straight-line segments in a path for covering a holed rectilinear workspace are each NP hard, [50] and [51] respectively. Therefore,

approximate solutions are desirable. Our approach is to divide the workspace into η parts, P_i, and assign each part to a UAV. This is done in accordance with equations (2) through (5) (see Figure 2.3 for an example). Following the partitioning, heuristic techniques may be used to make the partition more compact. A partition is more *compact* than another if the sum of perimeters of its elements is less than the sum of perimeters of the elements of the latter.

Here we give an algorithm to generate a partition of a holed rectilinear workspace that satisfies equations (2) through (5). We prove that our algorithm runs in $O(N \log N + \eta N)$ time. To the best of our knowledge, this is the first time an algorithm that solves our constrained polygon partitioning problem has been reported.

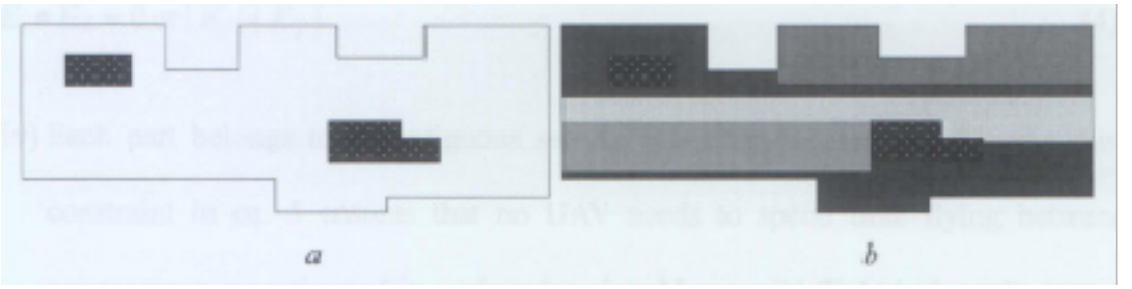


Figure 2.3: (a) A holed contiguous rectilinear workspace. (b) Partition (comprising 5 equal parts) generated by our algorithm.

The partition and the assignment satisfy the following four conditions:

(i) The parts of the workspace are interior disjoint. This ensures that (during rectilinear flight) no UAV will photograph any part of the workspace that has already been photographed by another UAV. One secondary benefit of satisfying this constraint is a lowered potential for collision between UAVs.

$$\bigcap_{i,j\in\eta,i\neq j}\left(\text{interior}(P_i), \text{interior}(P_j)\right) = \Phi \qquad (2)$$

(ii) The areas of the parts assigned to the UAVs are in the ratio of the rates at which the UAVs can do coverage. This is consistent with the intuitive notion that UAVs that can perform region coverage at a higher rate must be given a proportionally larger workload. This condition is formally stated below.

$$\text{Area}(P_1) : \ldots : \text{Area}(P_\eta) = \Gamma_1 : \ldots : \Gamma_\eta \tag{3}$$

(iii) The third constraint (eq. 4) specifies that the closure of each part is rectilinear. This can help simplify UAV control and help avoid narrow neck regions such as those encountered when boundaries of P_i are not rectilinear. For any two edges $\overrightarrow{E_i}$ and $\overrightarrow{E_j}$ that represent a boundary of any part in the partition,

$$\overrightarrow{E_i} \bullet \overrightarrow{E_j} = 0 \text{ or } |\overrightarrow{E_i}| \cdot |\overrightarrow{E_j}| \tag{4}$$

(iv) Each part belongs to a contiguous set X. Since the workspace is contiguous, constraint in eq. 5 ensures that no UAV needs to spend time flying between noncontiguous portions of its assigned region. Non-useful flight is thus eliminated.

$$P_1, P_2, \ldots, P_\eta \in \mathrm{X} \tag{5}$$

Chapter 3

Related Work

The problem of decomposing a polygon into interior disjoint regions in called the partitioning problem. It arises in image processing, computer graphics, VLSI layout design, data compression, terrain representation, robotic region coverage and preprocessing stages of several geometric computing algorithms. Here we review past works that are closely related to our own. Our partitioning problem is related to a more general class of problems called the polygon decomposition problem. Readers interested in this problem are referred to the survey work [52].

This chapter reviews past work in coverage motion planning. We focus on geometric primitives used for region representation, the cost function involved and the performance of the coverage plan generating algorithm, especially as parameterized by the number of turns in the resulting plan.

3.1 Region Coverage Using Reeb Graphs

A structured algorithm for demining unstructured nonpolygonal regions is given in [53]. The main contribution of this work is an online algorithm to efficiently detect and maintain nodes in the Reeb graph [54] of the hitherto visited region. The

algorithm is complete if achieves a full representation of the topology of the mined region in the form of the Reeb graph. Complete coverage is guaranteed by traversal of all edges of the Reeb graph via back-and-forth lapping (henceforth, referred to in short as 'lapping') motion. In practice however, the Reeb graph may not completely represent the mined region since combination of limited sensor range and a less-than-one ratio of local curvature of any convex portions of an obstacle to the curvature of the robot makes the detection of certain nodes impossible. Hence, despite the benefit of arbitrarily accurate sensing and localization information, complete coverage cannot be guaranteed. In our view, using a Reeb graph representation is not appropriate for minimizing the number of turns or the traveled distance since it captures only the topology of the coverage region. However, this shortcoming need not necessarily adversely impact the utility of Reeb graphs in demining or lawn mowing activities. Another point we note about their coverage algorithm is that it yields paths that are fairly predictable, especially in low complexity regions. 'Scripted' paths are undesirable for low-altitude unmanned reconnaissance aerial vehicles (URAVs) operating in inclement zones.

3.2 Algorithm for Minimum-Turn Coverage

In [55], an algorithm is given to find a set of sweep directions that minimizes the number of turns in a coverage plan for a holed polygonal region. The algorithm divides the coverage region into monotone areas. Each monotone area may be covered along the direction parallel to one of its side using the lapping primitive. Alternatively, two or more adjacent monotone areas may be covered using the lapping primitive perpendicular to a single sweep direction. A dynamic programming algorithm is used to check if the total turn cost of coverage can be lowered by further subdividing a

monotone area or via merging adjacent monotone areas. The algorithm is NP hard and therefore its practical applicability is limited to coverage of regions with low complexity. Also, the algorithm does not (explicitly or implicitly) minimize the sum of the lengths of the rectilinear portions of the path. It assumes the cost (the turn cost as well as the length of rectilinear path) of moving between areas is zero. An area is defined as a contiguous portion of the region of coverage interest that has a single sweep direction.

3.3 Coverage Using Weighted Cost Assignment

In [56], the free space in an orthogonal coverage region is represented using a uniform grid (Figure 3.1). The objective is to find a minimal cost coverage path between a pair of arbitrarily chosen start and goal cells. The coverage algorithm assigns a numeric value to each cell; the sum of the minimum distance of the cell from the goal cell and a weighted numeric indicator of the proximity of the cell to obstacles. More formally, each cell's value is computed as $PT(c) = \min_{p \in P} \left(\text{length}(p) + \sum_{c_i \in P} \alpha \cdot \text{obstacle}(c_i) \right)$ (see [56] for notations). The problem of finding a complete coverage path is then treated as the problem of moving from the start cell, S, to the goal cell, G, along locally maximum gradient. The algorithm's strength lies is its simplicity. However, it requires $O(\frac{lb}{\mu})$ memory even when the coverage region has low obstacle density; l, b are dimensions of the smallest bounding rectangle of the coverage region and, μ is parameterized by the controllability and size of the robot and, the width of the narrowest path between obstacles. Since the algorithm requires a cost to be assigned

to each cell and, the number of cells is $O(\frac{lb}{\mu})$ for rectangular regions. In Figure 3.1, a typical path transform (see [56] for definition) and a possible coverage path are shown. Using only the algorithm in [56], the robot cannot move beyond the cell marked with thick boundaries (the lower left portion of the coverage region is not covered). Thus, complete coverage is not guaranteed. Moreover, the number of turns in the coverage path produced by the algorithm may be unnecessarily excessive.

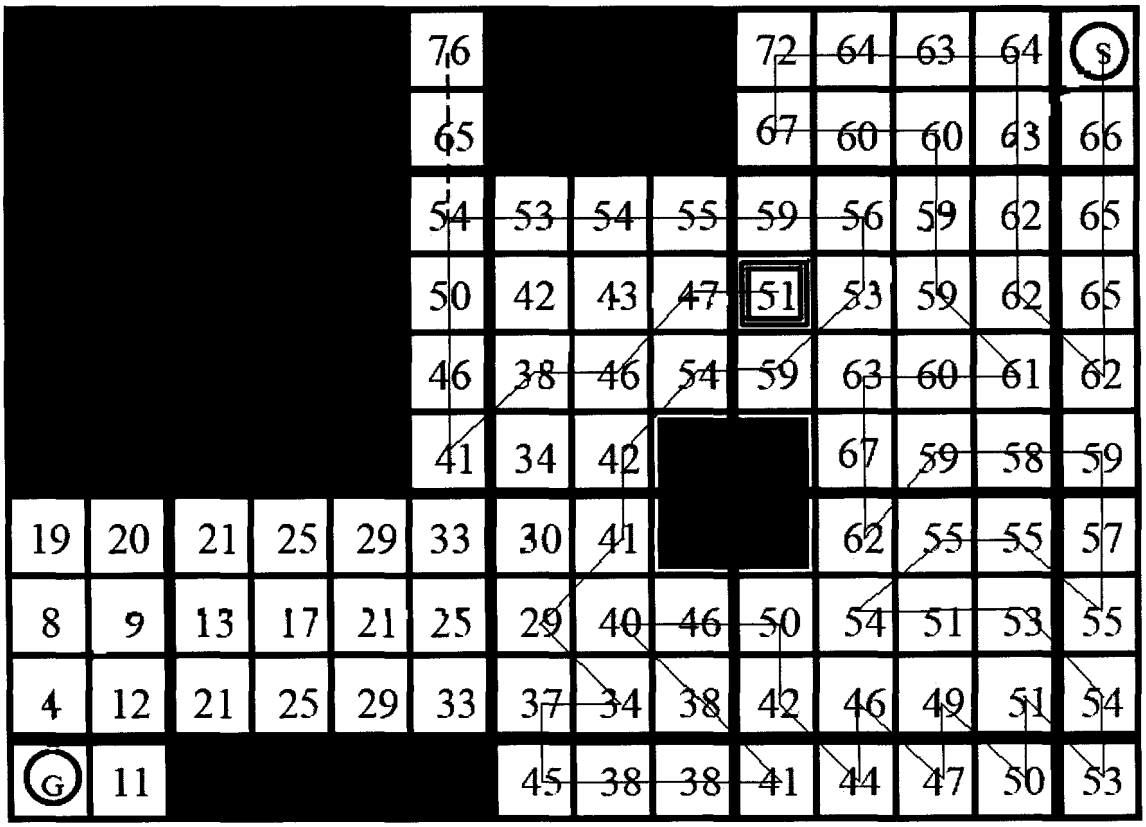


Figure 3.1: A typcial path transform matrix. S is the start cell, G is the goal cell *(adapted from [56]).*

3.4 Region Coverage Using Spanning Trees

In [48], the workspace is contiguous and its narrowest region is twice the width of the sensor-footprint (in this case, a square-shaped milling tool). The workspace is represented using a grid approximation. Three variants of a coverage algorithm are reported of which two are on-line. Since our interest is in off-line coverage planning, we discuss only the off-line variant. Each algorithm yields a resolution-complete coverage path that visits each grid cell exactly once. In the first version of the algorithm, the workspace, as in our case, is completely known in advance. The 4-neighborhood of the cells defines a spanning tree. Coverage of the spanning tree implies coverage of the region of interest. The robot navigates between cells in depth first order. The advantage of the algorithm is its speed in generating length optimal path – the off-line version computes an optimal covering path in $O(N)$ time; N is the number of vertices in the spanning tree. The user may specify weights on each 4-neighborhood edges and the algorithm may, for example find a minimum weight spanning tree. This has the effect of producing paths that reflect a desired pattern or direction of motion. Other than this parameterized control, the algorithm does not provide for minimizing the turn-cost.

3.5 A Neural Network Approach to Region Coverage

In [57], the workspace is approximated as a subset of the union of a minimal number of identical discrete cells. Each cell is represented by a unique artificial neuron (henceforth called, neuron). Neurons associated with occupied or covered cells emit

inhibitory signals. Other neurons emit excitatory signals. The robot's coverage path is incrementally computed. The next cell to visit is an 8-neighbor of the cell the robot currently occupies. It is computed as a function of the activity landscape of all neurons and the current kinematic state of the robot. The neural network approach in this work is similar to an approach based on range-limited potential fields. The robot is assumed to be holonomic and point-reducible. At any step in the incremental computation, only the cost of local move is minimized. This approach can lead to a coverage path with significantly suboptimal cost in cases involving coverage of high complexity (measured by the complexity of polygons that define the coverage region) regions. The number of neurons used is $O\left(\left\lceil \frac{l \cdot b}{c^2} \right\rceil - \sum_i \left\lceil \frac{Area(\text{hole}_i)}{c^2} \right\rceil\right)$; l, b are the dimensions of a minimal bounding rectangle of the coverage region and c is the dimension of a square-shaped sensor. These many neurons may be more than necessary, especially if the coverage region has low complexity or it comprises several widely separated contiguous subregions.

3.6 A Related Problem – Grid Graph Partitioning

As we discuss shortly, our problem is related to the grid graph partitioning problem. The graph partitioning problem [58] has been extensively studied and its formal statement is as follows:

> Given a graph $G(V,E)$ and a number n, divide V into n sets such that the number of edges connecting vertices belonging to different sets is minimized.

The connection between polygon area partitioning problem and the problem of partitioning a discrete set – a grid graph, can be seen as follows. A minimal augmented grid of a polygon $\wp$ is a grid in which at least one side of each cell lies in the interior of $\wp$ and has (at least) a pair of horizontal and vertical sides that overlap with (at least) a pair of horizontal and vertical sides of $\wp$. This grid can be computed in $O\left(N\left(\log N + I_h \log I_h + L_x/\omega_x + L_y/\omega_y\right)\right)$ time [47]; ω_x x ω_y is the dimension of cells in the grid, I_h is the average number of intersections of a horizontal sweep line with the vertical edges of $\wp$, L_x is the maximum of the distances between all pairs of vertical edges of $\wp$, etc. We have proved in [47] that the minimum augmented grid of $\wp$ can be computed in an additional polynomial number of computations.

With appropriate choice of the grid cell size, the problem of generating η compact, rectilinear partitions of the area of $\wp$ becomes equivalent to the problem of partitioning a grid graph into η parts with the minimum augmented grid of $\wp$ as the input. The grid graph bisection problem for holed polygons was however conjectured [59] and later proved [60] to be NP complete.

An $O(n^{5+2h})$ algorithm, n and h are the numbers of grid cells and holes in $\wp$ respectively, was given in [59] for the optimum grid graph bisection problem. The algorithm can in fact divide the graph into two parts with any feasible vertex ratio. Moreover, it can be applied recursively to generate η partitions. The algorithm can thus generate partitions as illustrated in Figure 3.2. This figure has been adapted from the figure in [59]. Three conditions to be satisfied are:

a. the areas being proportional to a pre-specified ratio;

b. the boundaries are ;

c. comprising pieces whose sum of perimeters is minimum.

However, its run time is not manageable for most practical problems. Moreover, the parts (subgraphs) it generates may comprise more than one connected component. This entails each UAV may have to do non-useful work while flying amongst various noncontiguous portions of the part of the workspace assigned to it.

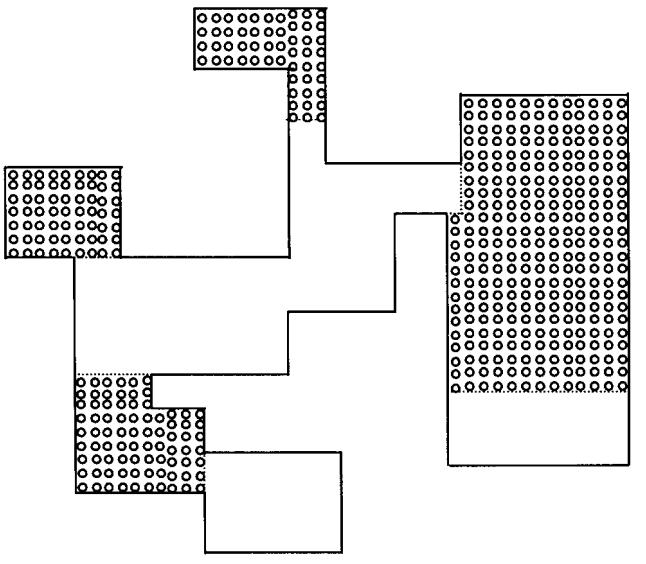
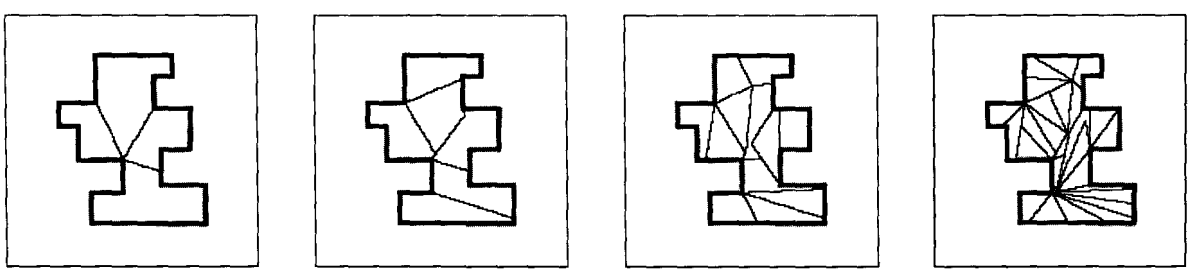


Figure 3.2: An optimum (compact) rectilinear bisection of a rectilinear polygon using the algorithm in [59]. Each part comprises 3 disjoint pieces.

3.7 Polygon Partitioning for Region Coverage Using Tethered Robots

In [40], a planar polygon partitioning algorithm is presented to enable parallel under-sea coverage using tethered submersible robots. The algorithm generates partitions

whose areas are in a pre-specified ratio. The algorithm works by first preprocessing the workspace into convex subsets. Next, a set of pairwise edge-connected convex regions with a total area not exceeding the area that must be apportioned to the first robot is assigned. The assignment procedure ensures that the union of hitherto assigned convex sets increases the face number of the planar region in which the workspace is embedded by one. Often, upon the execution of these steps, a partial area of some adjacent convex face must be added to the union of convex sets that has already been apportioned to the first robot. A rotational sweep achieves this task. Iteratively, all robots are assigned their areas in accordance with eq. 3 and eq. 5. The algorithm runs in $O(N \log N + \eta^2 N)$ time for holed polygonal workspaces. Each part in the partition is contiguous and, it satisfies an additional property of containing a specified point, for example, the point in the workspace that is closest to the robot's instantaneous location at the time it receives instructions to do coverage. By relaxing the point inclusion criteria, an $O(\eta)$ speedup is reported in [39]. Both algorithms, even for orthogonal workspaces, generate partitions whose elements have non-rectilinear closures (see Figure 3.3).



A rectilinear polygon with $n = 20$ vertices cut into $p = 3$, 6, 12, and 24 equal-area pieces.

Figure 3.3: Divisions of workspaces generated by algorithm in [39] are non-rectilinear (Figure reproduced from [39]).

3.8 A Related Problem – Polygon Area Bisection

In [61], a linear-time algorithm is given for bisecting a simple polygon in a pre-specified direction. An example of a typical partition generated by this algorithm is in Figure 3.4. An iterated version of the algorithm that generates η partitions (instead of only two) of a holed polygon of specified area ratios can be developed. It runs in $\Omega(N \log N + \eta N)$ time (same as in [39]) and satisfies the conditions spelled out eq. 4. However no extensions that enable it to satisfy eq. 5 have been reported.

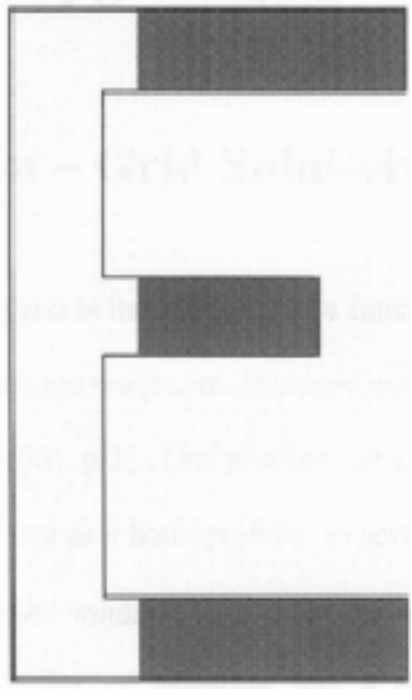


Figure 3.4: Area partition of a rectilinear polygon in 2:1 ratio using the algorithm in [61]. Note that the second part comprises 3 disjoint pieces.

Chapter 4

Computing the Minimum Grid of a Rectilinear Polygon

4. 1 Introduction – Grid Subdivisions

A *grid subdivision* of a polygon is its division into a finite number of interior disjoint cells. It is uniform if the cells are congruent. Quadtree and Delaunay triangulation are examples of nonuniform grids [62]. The problem of computing a minimum grid subdivision of a polygon arises as a basic problem in several areas. For example, one possible approach to compute minimal risk paths for an unmanned aerial vehicle (UAV) is to divide the region of operation into a finite number of uniformly distributed square cells [63]. The risk to the UAV due to external threats is computed at each cell. This risk is assumed constant at all points in a cell. The problem of finding a minimal risk path is thereby translated into a graph search problem in a 4-connected graph whose nodes are the cells and whose edges represent adjacency relation between the cells. As another example, consider the problem of computing minimum turn, minimum length path for a tool in surface milling of planar surfaces [64] or, for an unmanned reconnaissance aerial vehicle that carries an onboard

downward-looking sensor that has a fixed footprint [46]. Even the discrete version of this problem is NP-hard [65]. One technique to solve this problem is as follows:

STEP1. Compute a grid subdivision of the area approximated as an orthogonal polygon. The size of each cell is equal to the size of the working tip of the tool or the size of the sensor-footprint.

STEP2. Compute the minimal number of pairwise independent rectangles whose union is the union of the cells.

STEP3. Find a minimum turn or minimum length path or a cycle in each rectangle.

STEP4. Optimally merge the paths or cycles in adjacent rectangles.

END

Probably, a wider use of computing grid subdivisions is in developing efficient data structures for geometric computing. In [66], the authors have shown how grids may be used as building blocks for critical data structures for problems including, segment intersection, point location, map overlay and visibility computation. While the authors do not make explicit reference to the problem of computing the grid itself efficiently, it is clear that efficient algorithms for computing the grid can help increase the efficiency of higher-level data structures that use grids.

While a grid subdivision always exists for any orthogonal polygon, a grid subdivision comprising cells with pre-specified size does not necessarily exist (see Figure 4.1*a*). In this case, the alternative notion of *augmented grid* is useful. An augmented grid comprises interior disjoint cells that lie partially or entirely in the interior of the polygon as shown in Figure 4.1*b*.

A *minimal augmented grid* is an augmented grid that has at least one horizontal boundary segment that overlaps with a horizontal edge of the polygon and at least one vertical boundary segment that overlaps with a vertical edge of the polygon. We use the plane sweep technique [62] to compute a minimal augmented grid for an arbitrary orthogonal polygon whose cell-size has been specified *a priori* and whose sides are either parallel or perpendicular to the edges of the polygon. Fundamental to the plane sweep technique is an arbitrarily long sweep line that moves in discrete steps. At each stop, the intersections of the line with the geometric elements in the plane of its motion are recorded in an efficient data structure. In our case, a set of maximal intersections represents the instantaneous 'status of the sweep line'. In fact, consecutive intersections define segments of which those that lie within the closure of the polygon collectively define the grid we wish to compute. We give two algorithms that differ in the prescription for the discrete motion of the sweep line. As can be seen in Figure 4.1, a uniform grid subdivision of a polygon exists iff there exists a frame of reference in which the division of the numerical value of ordinate (abscissa) of each horizontal (vertical) edge of the polygon by the width of its cell is a whole number.

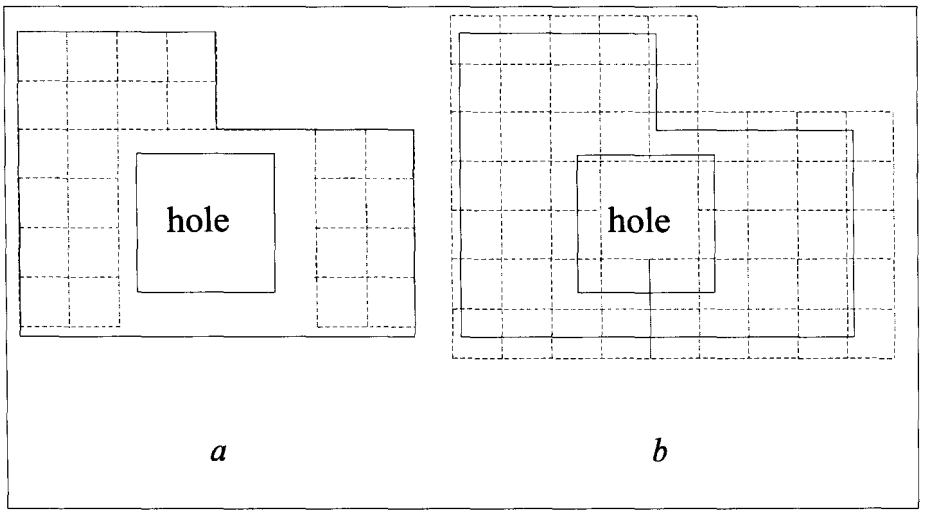


Figure 4.1: Figure 4.1*b* shows an augmented grid of the polygon.

The next section introduces the reader to the terms and symbols used in this work. The pre-processing steps of both the algorithms are common. They are presented in Chapter 4.3. The main body of the algorithms is in Chapter 4.4. The time-complexity of our algorithms is analyzed in Chapter 4.5. While neither algorithm dominates the other, depending on the complexity of the polygon for which an augmented grid must be computed, the ratio of the size of cells to the size of the polygon and the ratio of the area of the polygon to the area of the polygon plus the area of holes, one of the algorithms is likely to be more time efficient than the other. Proof of the proposition that the minimum augmented uniform rectangular grid of any orthogonal polygon can be computed in polynomial time is also given in Chapter 4.5.

4. 2 Symbols and Definitions

For convenience, we list below the symbols used in our subsequent discussion in this chapter and their definitions

$\wp$ is an arbitrary orthogonal polygon with holes and has N vertices.

An e*dge* is a boundary segment of a polygon.

A *side* is a boundary segment of a cell of a polygon.

The *interior of a set* is the difference between the set and its closure.

N_v is the vertical complexity of $\wp$. *Vertical complexity* (of an isothetic polygon) is the number of its horizontal edges. $\wp$ is minimally rotated in counter-clockwise direction whenever necessary to make its edges axis-parallel before to computing N_v.

N_h is the horizontal complexity of $\wp$. *Horizontal complexity* of $\wp$ is the number of its vertical edges.

g is an augmented uniform grid of $\wp$ with rectangular cells whose sides are parallel or perpendicular to the edges of $\wp$. G is the set of all augmented uniform grids of $\wp$.

g' is a minimal augmented uniform grid of $\wp$. G' is the set of all minimal augmented uniform grids of $\wp$. The readers may note that $\mathrm{G}' \subset \mathrm{G}$.

$\omega_x \cdot \omega_y$ is the size of a cell of a grid in G and is specified *a priori*.

$\mathrm{g}^* = \arg\min_i(g'_i)$ is the minimum augmented uniform grid of $\wp$. The term 'minimum' refers to the number of cells in the grid.

The ordered pairs $(x_k^{\wp}, y_k^{\wp})$ and (x_j, y_j) represent the coordinates of a vertex of $\wp$ and the coordinates of a vertex of the closure of an augmented grid of $\wp$ respectively.

$$x_{\min} = \arg\min_{x_j}(x_j, y_j),\ x_{\max} = \arg\max_{x_j}(x_j, y_j)$$

$$y_{\min} = \arg\min_{y_j}(x_j, y_j),\ y_{\max} = \arg\max_{y_j}(x_j, y_j)$$

$$\mathrm{X} = x_{\max} - x_{\min},\ \mathrm{Y} = y_{\max} - y_{\min}$$

$$\overline{\mathrm{X}} = \left\lceil \frac{x_{\max} - x_{\min}}{\omega_x} \right\rceil,\ \overline{\mathrm{Y}} = \left\lceil \frac{y_{\max} - y_{\min}}{\omega_y} \right\rceil$$

$cl(\cdot)$ is the closure of the set $(\cdot)$ (Figure 4.2).

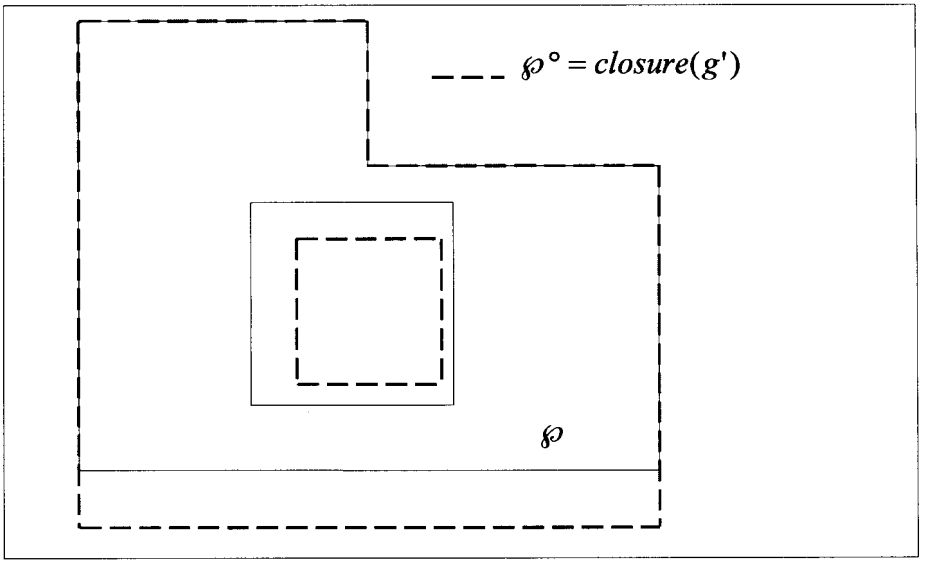


Figure 4.2. An illustration of closure $\wp^\circ$

4. 3 Preprocessing

By definition, at least one vertical segment of a minimal augmented uniform grid (a union of any maximal number of collinear vertical sides) and a vertical edge of $\wp$ must overlap and similarly, at least one horizontal segment of the augmented grid and a horizontal edge of $\wp$ must overlap. In order to compute the grid we first arbitrarily select the edges of $\wp$ represented by line-equations, $x = x_j^{\wp}; j = \arg\min_k \left(x_k^{\wp}, y_k^{\wp}\right)$ (henceforth referred to as *y-support line*) and $y = y_j^{\wp}; j = \arg\max_k \left(x_k^{\wp}, y_k^{\wp}\right)$ (*x-support line*), to be the leftmost and uppermost boundary of the grid. However, the remaining edges of $\wp$ may not overlap with segments of the grid. This can cause errors in the registration of sweep line events. Therefore, prior to computing g' (that has the aforementioned support lines), we compute the transformation $\Gamma(\wp) = \wp^\circ$ such that $cl(g') = \wp^\circ$ $cl(g') = \wp^\circ$. This causes the boundary of $\wp^\circ$ to be equal to the boundary of the grid g'.

LEMMA 1: The number of cells in any grid g' of $\wp$ is less than or equal to the number of cells in any grid $g \in G - G'$.

PROOF: The grids in $G - G'$ comprise cells of which no side overlaps with any edge of $\wp$. Thus a translation of a grid in $G - G'$ that preserves the inclusion relationship between the closure of the grid and $\wp$ can cause the interior of some cells in the grid that border its closure to not overlap with the polygon. These cells are made redundant by the translation. See Figures 4.3 and 4.4 for an illustration.

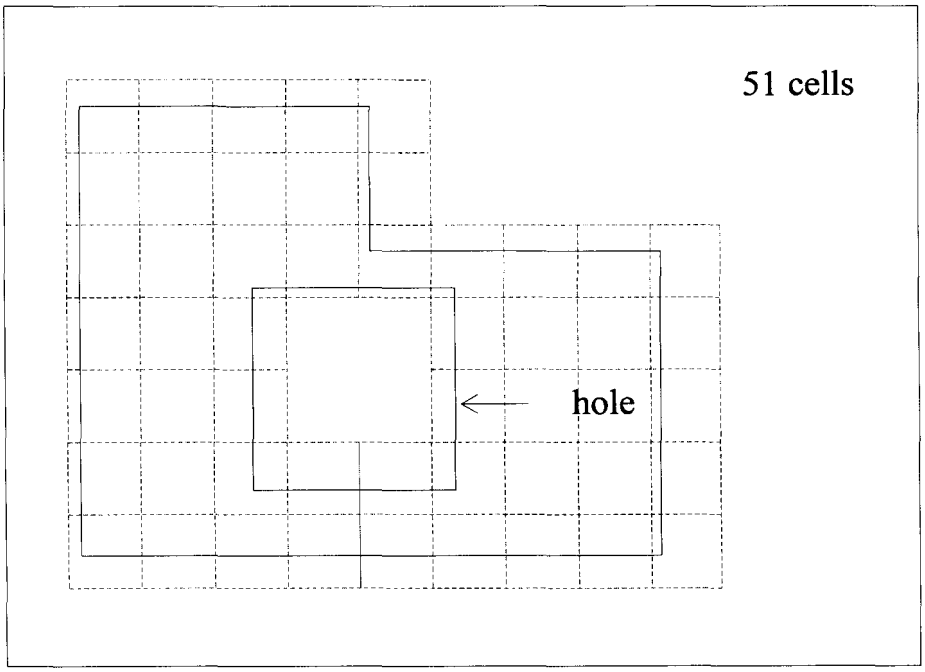


Figure 4.3: An augmented grid that belongs to the set $G - G'$.

LEMMA 2: Given a doubly connected edge list (DCEL) [62] or an equivalent representation of $\wp$, $\Gamma(\wp)$ can be computed in $O(N)$.

PROOF: Assume a DCEL representation of $\wp$. The transformation (by axis-parallel translation) of any edge of $\wp$ to the corresponding edge in $\wp^\circ$ will require that only its own coordinates and the coordinates and lengths of edges incident on its vertices be

recomputed. This results in $O(1)$ computations per transformation, each of which takes $O(1)$ time. The complexity of $\wp$ is $O(N)$. Thus follows LEMMA 2.

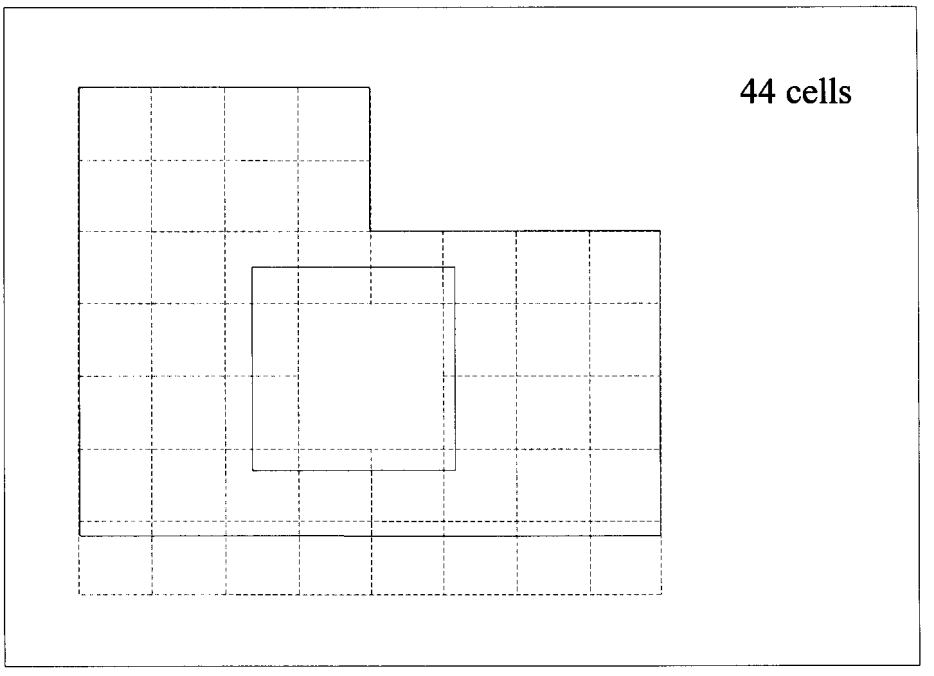


Figure 4.4: A minimal augmented grid obtained by translating the augmented grid in Figure 4.3 until its top and leftmost boundary overlaps with the top and leftmost edges of the polygon.

4. 4 Computing Grid Subdivision

We give an overview of two algorithms to compute a minimal grid subdivision of $\wp^\circ$ (a minimal augmented grid of $\wp$). Further information about the algorithms and their complexity analysis is presented in Chapter 4.5.

Both algorithms use the routine GENERATE_GRID. The two algorithms differ in that the first algorithm uses ALGORITHM1 to compute STEPS 3 and 4 (see below) whereas the second uses ALGORITHM2 to compute STEPS 3 and 4. Additionally, if ALGORITHM2 is

used, only a rectangle partition of $\wp^\circ$ is obtained. Further computations are necessary to get g'.

GENERATE_GRID

STEP1. Construct the interval tree of horizontal segments

STEP2. Construct the interval tree of vertical segments

STEP3. Sweep the plane of the polygon with a 'long' line beginning at $y = y_{\max}^{\wp^\circ}$. The motion vector of the line is in the negative y direction. The line moves in discrete steps, stopping at only site events. A site event refers to an event in which the sweep-line is collinear with a horizontal edge of the polygon. [62].

STEP4. Sweep the plane of the polygon with a long line beginning at $x = x_{\min}^{\wp^\circ}$. The motion vector of the line is in the positive x direction. The line moves in discrete steps, stopping at only site events.

STEP5. RETURN g' or data from which g' can be computed

END

4.4.1 Algorithm1

In ALGORITHM1 a site event is trivially defined – For the x-parallel (y-parallel) sweep line it is any point on $\wp^\circ$ at a distance of whole number multiples of ω_y (ω_x) from $y = y_{\max}^{\wp^\circ}$ ($x = x_{\min}^{\wp^\circ}$).

4.4.2 Algorithm2

An alternative approach to compute g' is to change the manner in which STEPS 3 and 4 are implemented in GENERATE_GRID. Instead of computing g' directly, we first compute a rectangle partition of $\wp^\circ$. Following this, the segments that belong to the grid can be easily computed for each rectangle. In order that GENERATE_GRID routine that uses ALGORITHM2 be competitive with (i.e., not be dominated by) the routine when ALGORITHM1 is used, the cardinality of the rectangle partition should not exceed $O(N)$. The reason for this will be clear by a comparative look at the time-complexity of the two algorithms in the following chapter. To compute the rectangle partition of $\wp^\circ$ that meets this condition, we move the x-parallel sweep line, stopping at only horizontal edges of $\wp^\circ$ as illustrated in Figure 4.5. These correspond to the site events for ALGORITHM2. Alternatively, one may use the y-parallel sweep line and stop only at the vertical edges of $\wp^\circ$. In this case, vertical edges of $\wp^\circ$ correspond to the site events. Once the rectangle partition has been generated, the grid subdivision of $\wp^\circ$ is computed. Without further processing, the grid subdivision is in the form of a collection of segments of the grid in each rectangle in the partition.

Figure 4.5: Site events for ALGORITHM 2. Status of sweep line moving along $-\vec{j}$.

LEMMA 3: The algorithms record all combinatorial changes of the status of the sweep line.

PROOF: The status of the sweep line can change only at horizontal (vertical) edges of $\wp^\circ$. But, all horizontal (vertical) edges of $\wp^\circ$ are at a distance of whole number multiples of ω_y (ω_x) from the support line $y = y_{\max}^{\wp^\circ}$ ($x = x_{\min}^{\wp^\circ}$). The sweep lines in both algorithms stop at all edges of $\wp^\circ$. Thus, the algorithms record all combinatorial changes of the status of the sweep line.

4. 5 Complexity Analysis

STEPS 1 and 2 take $O(N_h \log N_h)$ time and $O(N_v \log N_v)$ time respectively. Intuitively, this can be seen since the problem of computing interval trees reduces to the problem of sorting numbers for which an optimum algorithm takes $O(N \log N)$ time [62]. In ALGORITHM1, the x-parallel sweep line moves along $-\vec{j}$ in discrete steps of size ω_y. Thus, it makes $O(\overline{Y})$ stops and in, each stop, its status is checked. Each set of checks takes $O(\log N_h + I_h + I_h \log I_h)$ time where I_h is the average number of intersections of the horizontal sweep line with the vertical edges of $\wp^\circ$.

To see how each set of checks takes $O(\log N_h + I_h + I_h \log I_h)$ time, consider the following argument: since the query for the status of the sweep line is made on an interval tree, $O(\log N_h + I_h)$ time is needed to check and report all intersections. However, an interval tree returns the intersection points in arbitrary order. To construct segments that belong to the grid from the intersection points, the result of the query must be sorted. This takes additional $O(I_h \log I_h)$ time. Thus, total time that is needed is $O(\log N_h + I_h) + O(I_h \log I_h)$, i.e., $O(\log N_h + I_h + I_h \log I_h)$. Similar arguments can be made for the sweep line moving along vector $\vec{i}$. The readers may note that no further processing is necessary in this case; the output is g' itself.

In ALGORITHM2, the x-parallel sweep line stops only at horizontal edges of $\wp^\circ$. This results in $O(N_v)$ stops. The time spent at each stop remains the same as discussed earlier (in the previous paragraph). Thus, from similar analysis, our second algorithm takes $O(N_v(\log N_h + I_h + I_h \log I_h))$ time to compute the rectangle partition. Alternatively, computing the rectangle partition using the y-parallel sweep line can be done in $O(N_h(\log N_v + I_v + I_v \log I_v))$ time. Computing the collection of segments that comprise the grid subdivision of $\wp^\circ$ can be done in additional $O\left(N_v(\bar{L}_x/\omega_x + \bar{L}_y/\omega_y)\right)$ time (see LEMMA 4); $\bar{L}_x$ and $\bar{L}_y$ are the average lengths and breadths respectively of the perimeter of the rectangles in the rectangle partition of $\wp^\circ$. To compute $\bar{L}_x$, we sum the lengths of all rectangles in the rectangle partition of $\wp^\circ$ and divide it by the number of rectangles in the partition. Similarly, $\bar{L}_y$ can be computed.

LEMMA 4: Using the second algorithm the grid can be computed in $O\left(N_v(\bar{L}_x/\omega_x + \bar{L}_y/\omega_y)\right)$ time following the execution of the STEP4 in GENERATE_GRID.

PROOF: The motion of the sweep line along $-\vec{j}$ in the second algorithm gives $O(N_v)$ interior disjoint rectangles. Each rectangle comprises a whole number of only complete grid cells. Thus, in this case, computing the grid means computing the segments (union of maximal number of collinear sides of grid cells) in each rectangle. The average number of segments in each rectangle is $O(\frac{\bar{L}_x}{\omega_x} + \frac{\bar{L}_y}{\omega_y})$. Thus follows LEMMA 4.

THEOREM 1: The routine GENERATE_GRID that uses ALGORITHM1 computes the grid subdivision of $\wp^\circ$ (minimal augmented grid of $\wp$, i.e. g') in $O\left(\bar{Y}(\log N_h + I_h + I_h \log I_h) + \bar{X}(\log N_v + I_v + I_v \log I_v)\right)$time.

PROOF: This follows from the complexity analysis of the algorithm.

THEOREM 2: The routine GENERATE_GRID that uses ALGORITHM2 computes the rectangle partition of $\wp^\circ$ in $O\left(N_v(\log N_h + I_h + I_h \log I_h)\right)$time. g' can be computed in an additional $O\left(N_v(\bar{L}_x/\omega_x + \bar{L}_y/\omega_y)\right)$time.

PROOF: The proof follows from the above analysis.

Note that the first algorithm is faster than the second when the ratio of the size of the polygon to that of the grid is small. The second algorithm is faster if the grid size is

smaller. Its efficiency is even better if the ratio of the area of the free interior of the polygon whose grid subdivision must be computed to that of its total area is small.

THEOREM 3: g^* can be computed in polynomial time.

PROOF: By definition, at least one x *-parallel* segment of g' overlaps with one x *-parallel* edge of $\wp$. There are only $O(N_v)$ possible horizontal alignments that satisfy this condition. Moreover, changing the horizontal edge of $\wp$ with which a horizontal segment of g' is aligned potentially changes only the number of cells in a row. Similar argument applies to vertical segments of g'. Thus, the cardinality of the set G' is less than or equal to $(N_h + N_v)$. Also, $g^* \in G'$. These observations and THEOREM 1 (or THEOREM 2) imply that g^* can be computed in polynomial time.

Chapter 5

Polygon Partitioning Algorithm

The algorithm involves a set of preprocessing steps and a recursive area assignment stage. As we shall see, the preprocessing takes $O(N \log N)$ time. The assignment of part of the workspace in accordance with equations 2 through 5 takes $O(N)$ time for each UAV. A preliminary time-complexity analysis is in the next chapter.

5.1 Preprocessing

The polygon $\wp$ comprises N_v horizontal and N_h vertical edges. It is rotated counter clockwise to make it isothetic with the coordinate axes. In the remaining portion of this document, the symbol $\wp$ represents the resulting isothetic polygon. The horizontal and vertical edges of $\wp$ are then sorted in nondecreasing orders of their ordinate and abscissa values respectively. This step takes $O(N \log N)$ time. Further preprocessing involves the following main stages:

1) Compute Rectangle Partition of $\wp$: In many problems with complex-shaped polygons as input, a simpler representation using triangles or rectangles is first generated. We partition $\wp$ into a set of x-maximal y-maximal rectangles (a constructive definition is given below, illustration is in Figure 5.1). The area assignment algorithm assigns rectangles or parts thereof to a UAV until eq. 3 is satisfied.

In a *x-maximal y-maximal rectangle partition* of $\wp$, the upper horizontal edge of a rectangle is a contiguous segment $\subset \wp$ that is the minimum superset of a maximal union of horizontal edges that have the same ordinate value, the lower horizontal edge of a rectangle is a contiguous segment $\subset \wp$ that is the minimum superset of a maximal union of horizontal edges that have the same ordinate value (unless it defines a degenerate rectangle) or a maximal contiguous segment $\subset \wp$ that is a superset of a horizontal edge of $\wp$. By definition, any vertical edge of any rectangle in the *x*-maximal *y*-maximal rectangle partition is a subset of a vertical edge of $\wp$.

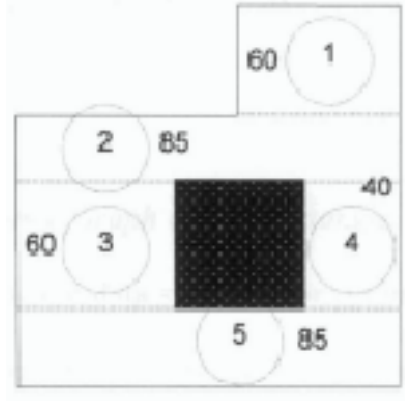


Figure 5.1: A workspace and its *x*-maximal *y*-minimal rectangle partition. The numbering of the rectangles is arbitrary. The total area of the workspace is 330 sq. units. The area of each rectangle is also shown.

LEMMA 1: The complexity of *x*-maximal *y*-maximal rectangle partition of $\wp$ is $O(N_v)$.

PROOF: This lemma follows from the observation that in the *x*-maximal *y*-maximal rectangle partition, each horizontal edge in $\wp$ is shared by at most 2 rectangles and that the horizontal side of any rectangle must be a superset of at least one horizontal edge in $\wp$.

The complexity of our partition is optimal. For example, see the polygon in Figure 5.2 where N_v is 10 and the number of rectangles in the *x*-maximal *y*-maximal rectangle partition is 5 ($N_v / 2$). By changing the procedure for assigning a rectangle or its part (see Chapter 4.2), the workspace division algorithm can possibly also work with other (for example, see [67]) rectangle partitions.

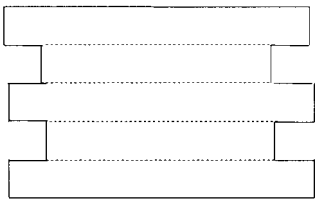


Figure 5.2: A workspace and its optimal *x*-maximal *y*-minimal rectangle partition.

2) Compute Region Adjacency Graph of the Rectangle Partition: Region adjacency graph (RAG) [68] is an efficient data structure for storing the adjacency relationship between the rectangles in the *x*-maximal *y*-maximal rectangle partition. This information is accessed frequently in the area assignment stage. The complexity of the RAG is $O(N)$ since it is a planar graph with $\Theta(N_v)$ vertices.

3) Compute Region Adjacency Tree: The assignment algorithm exploits the natural parent, child relationships of a rooted tree data structure as rectangles are sequentially processed.

We use depth first search (DFS) [69] to compute a region adjacency tree (see Figure 5.3) (RAT) from the RAG – a RAG without any cycles. Next, we arbitrarily select a node of the RAT to be its root node.

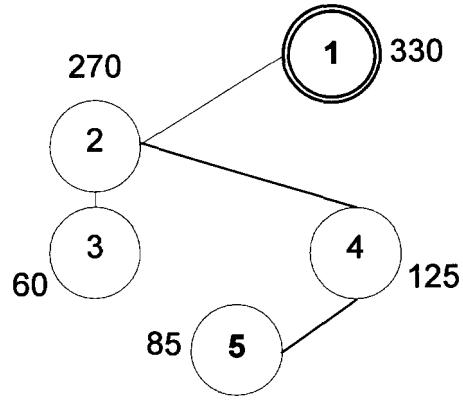


Figure 5.3: RAT of the workspace in Figure 5.1. $Node_1$ is the root node and $Node_5$ is the candidate node.

4) Assign Weights to Nodes of Rooted RAT: The nodes in the rooted RAT are assigned weights as follows:

$$wt(Node_j) = Area(\text{Rectangle represented by } Node_j) + \sum_{i \neq j} Area(\text{Descendant}_i(Node_j)) \quad (6)$$

Equation (6) means that the weight assigned to a node is equal to the area of the rectangle represented by the node plus the sum of areas of all rectangles that are represented by the descendant nodes. For example, referring to Figure 5.1, the weight assigned to node 2 = 85 + (60 + 40 + 85) = 270. This is shown in Figure 5.3. The weight assignment procedure in eq. (6) implies that the weight of the root node is equal to the area of $\wp$ itself.

5) Picking the Candidate Node: For any UAV, the *candidate node* (see Figure 5.3) is a node in the RAT corresponding to the first rectangle that will be assigned (in whole or part) to it. The candidate node for the first UAV is picked as follows:

For the UAV that is the first one to be assigned its part of the workspace, the candidate node is the deepest node in the rooted RAT.

The procedure for picking candidate nodes for the remaining UAVs is given next in Chapter 5.2.

5.2 Area Assignment to UAVs

UAVs are assigned their parts sequentially. Any UAV can be taken up for area assignment first. Without loss of generality we assume that the UAVs are assigned their parts of the workspace in nonincreasing order of their capabilities. Since the algorithm is recursive, we give details for assignment to the second UAV only. The candidate node (always, a leaf node) is selected and assigned. If the area to be assigned to the UAV is larger than the area of the candidate node then we move to the adjacent node. The weight of candidate node is now zero and it is thus effectively removed from the RAT. Under this assumption if the node adjacent to the candidate node is a non-articulation node then, it is assigned in the same manner. We continue to do this as long as the area assignment for the particular UAV is not complete and we have not encountered an articulation node. Once we reach an articulation node, special care needs to be taken else its assignment can cause the unassigned part of the graph to be split into two or more disconnected components. This will in turn force the part assigned to at least one subsequent UAV to comprise at least two noncontiguous portions. The main steps in the algorithm are as follows:

1. SELECT_CANDIDATE_NODE()

2. HANDLE_NON-ARTICULATION_NODES()

3. HANDLE_ARTICULATION_NODES()

4. UPDATE_TOPOLOGY_OF_RAT()

5. UPDATE_WEIGHTS_OF_RAT()

6. END

Candidate nodes for assignment to all but the first UAV is done as follows:

The candidate node is the deepest node in the subtree rooted at the node that represents the deepest rectangle that is partially assigned in the remaining part of the rooted tree.

An articulation node in a connected graph is a node whose removal yields a subgraph with two or more connected components. In any tree, all nodes with degree two are articulation nodes. A rectangle represented by a non-articulation node is assigned as follows:

Case A. The rectangle is a candidate node or its ancestor:

If the area of the rectangle is less than or equal to the area that must be assigned to the UAV minus the area that has been already assigned to it then, the rectangle is assigned in its entirety. Else, the rectangle is assigned by moving a horizontal sweep line along the *y*-axis, beginning from the side of the rectangle opposite to the side it shares with an adjacent rectangle. The sweep line stops when the product of its distance of travel with the length of the horizontal side of the rectangle is equal to the area that must be assigned to the first UAV.

Case B. The rectangle is a (non-candidate) leaf node:

If the area of the rectangle is less than or equal to the area that must be assigned to the UAV minus the area that has been already assigned to it then, the rectangle is assigned in its entirety. Otherwise a part of the rectangle which is connected to the

assigned portion of an adjacent rectangle is assigned to the UAV such that the unassigned portion of the rectangle is a single connected piece. This can be done in $O(1)$ time.

The procedure HANDLE_NON-ARTICULATION_NODES() works as follows. We assume that intersections with all vertical boundary segments of all rectangles in the rectangle partition of all horizontal lines whose ordinate values are equal to the ordinate value of any horizontal segment of a rectangle represented by an articulation node has been computed. This can be done in $O(N \log N)$ time. Consider Figure 5.4 which shows a workspace updated upon completion of assignment to the first UAV. Note that the part assigned to the first UAV is not shown. A fraction of the total assignment for the second UAV has been completed via assignment of rectangles corresponding to nodes 7, 8 and 9. Node 4 an articulation node, is a parent of nodes 5 and 9. Assume that the assignment for the second UAV can be completed by additionally assigning a portion of the workspace with an area equal to 275 sq. units. At least a part of the rectangle 4 must be assigned to the UAV in order to maintain contiguity. However, if it is completely assigned then, it will disconnect a portion with a total area of 40 sq. units (90+150+75-275) belonging to rectangles 5 or 6. To do a partial assignment for rectangle 4, we consider two vertical sweep lines (Figure 5.4) positioned initially collinear with the closest pair of vertical edges in rectangles 4 and 9. We check if the rectangle that shares one of its horizontal edges with the edge shared by rectangles 4 and 9, is bounded by the vertical sweep lines and is of height equal to half of the vertical side of rectangle 4 has an area that is larger than or equal to what is necessary to complete the assignment for the second UAV. If it is then, it is assigned as discussed earlier and we move on the assignment for the third UAV. Otherwise we move the right vertical sweep line (without loss of generality) towards the right

closest vertical edge in the rectangle partition. If this edge is an edge (or part of an edge) of $\wp$ itself but not the rightmost vertical edge in rectangles 4 and 9 then, we move through only half the distance and do similar checks for the rectangle as defined earlier by the new position of the vertical sweep lines. Note that it is not necessary to move by exactly half the distance, any fractional distance would do. The left vertical sweep line is similarly moved and similar checks are made. In the case of the example in Figure 5.4, we stop moving the vertical sweep lines once they reach the positions marked by '+'. Next we repeat this procedure for node 5 (all children nodes of node 4 must be assigned before any ancestor node of 4 is assigned) since it is an articulation node. Finally, node 6, a non-articulation node is reached. It is assigned just as any other non-articulation node while making sure that its assigned portion is contiguous with the assigned portion of its parent node.

In the description above we have left details on how a rectangle might be assigned partially such that its assigned area is contiguous with the assigned area of its parent node while simultaneously, its unassigned part is not disconnected from the other rectangles. This can be done in $O(1)$ time and the process can lead to an increase of up to three nodes corresponding to unassigned rectangles in the RAT.

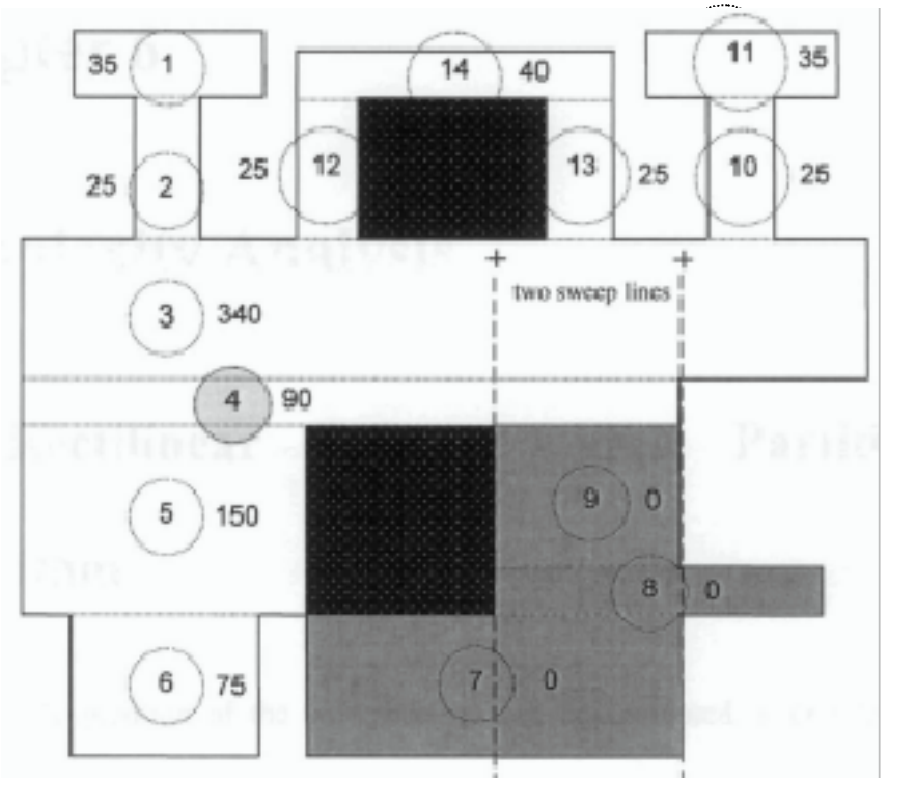


Figure 5.4: An updated workspace after the assignment to the first UAV has been completed.

Once a UAV is assigned its part of the workspace, the RAT is updated – nodes corresponding to rectangles that have been completely assigned are deleted. The (possibly partial) assignment of rectangles requires weights of all nodes that are ancestors of the youngest node corresponding to a partially assigned rectangle to be updated. Both, topology and weight update steps take $O(N)$ time.

Chapter 6

Complexity Analysis

6.1 Rectilinear Polygon Area Partitioning Algorithm

The rectangle partition of the workspace $\wp$ can be computed in $O(N \log N)$ time using a range-tree data structure.

LEMMA 2: The RAG of the rectangles in the x-maximal y-maximal rectangle partition can be computed in $O(N \log N)$ time.

PROOF: Since the RAG is a planar graph with $O(N)$ nodes, the amortized degree of nodes in the RAG is $O(1)$. Therefore if the horizontal segments of the rectangles in the partition (which the nodes represent) are stored in an interval tree (computable in $O(N \log N)$ time) then by property of interval trees, the amortized complexity of determining all nodes that are adjacent to a node is $O(\log N)$.

The DFS for computing the RAT runs in $O(N)$ time. The weight assignment procedure runs in $O(N)$ time. The candidate node must be picked for each UAV

except for the one that is assigned its part of the workspace last. Each pick of the candidate node requires $O(N)$ time.

LEMMA 3: The preprocessing stages in our workspace division algorithm take $O(N \log N)$ time.

LEMMA 4: The candidate node selection and the area assignment procedures ensure that each rectangle is considered for assignment a maximum of 2 times.

THEOREM 1: The area assignment algorithm runs in $O(\eta N)$ time.

PROOF: Step 1 (Chapter 5.2) runs in $O(N)$ time. Step 2 for each non-articulation node takes $O(1)$ time and, each UAV may be assigned $O(N)$ rectangles. Steps 4 and 5 take $O(N)$ time each for each UAV since the size of the RAT is never asymptotically larger than $O(N)$. The first positioning of the vertical sweep lines can be done in $O(\log N)$ time. Each repositioning of the vertical sweep line(s) takes $O(1)$ time since the abscissa values of the points at which the vertical sweep lines stops are pre-sorted. Thus, the positions of the pair of vertical sweep lines for assignment of an articulation node can take $O(N+\log N)$ ($\Omega(1+\log 1)$) time. $\Omega(f)$ stands for the lower bound of the complexity of the function f. This is analogous to $O(f)$ which means the upper bound for the complexity of the function f. The readers may note that since the product of the number of articulation nodes and the number of non-articulation nodes in the RAT is $O(N)$, the total time for positioning all pairs of vertical sweep lines for assigning rectangles corresponding to all articulation nodes for any UAV is also $O(N+\log N)$. Steps 1 through 5 are

repeated for all but the last UAV. Therefore, the total run time of the assignment algorithm is $(\eta-1)\times O(N+\log N)$, i.e. $O(\eta N)$ time.

THEOREM 2: The algorithm for partitioning $\wp$ among η UAVs in accordance with equations (2) through (5) takes $O(N\log N+\eta N)$ time.

PROOF: The proof follows from LEMMA 3 and THEOREM 1 since $O(N\log N)$ time is needed for preprocessing and an additional $O(\eta N)$ time is needed to partition the polygon into $O(\eta)$ pieces.

6.2 Anchored Area Partitioning Algorithm

The term anchored area partitioning was first used in [40] in which the authors gave an algorithm for the problem of partitioning an arbitrary contiguous polygon into subpolygons that are contiguous, that each include in their boundary a prespecified point on the boundary of the polygon and whose areas are proportional to a fixed ratio. Their algorithm however yields partitions (subpolygons) that are in general non-rectilinear even when the polygon is rectilinear. An example of anchored area partition is given in Figure 6.1. In this case, 7 points are pre-specified on the boundary of the polygon. Each partition must include exactly one pre-specified point on its boundary.

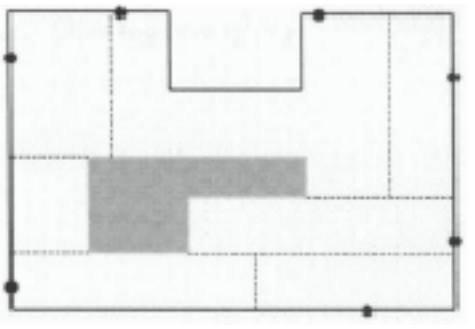


Figure 6.1: An example of anchored area partition. The polygon is divided into 7 rectilinear regions such that in addition to conditions in eqn. (2) through eqn. (5), each division also encloses one pre-specified point.

We extend our algorithm for solving the anchored area partitioning problem to yield rectilinear subpolygons in the case in which the workspace is rectilinear.

The area assignment in the case in which a fixed point on the boundary of $\wp$ must be included in the region assigned to each UAV can be done by modifying the manner in which rectangles in the RAT or parts thereof are assigned. A rectangle is not completely assigned unless it or none of its descendants have any anchors for which partitions have not yet been generated. Due to this restriction, the area assignment for the first UAV may create a hole of complexity $O(N)$. Assignment to any successive UAV can either create an additional hole of complexity $O(N)$ or cause two holes to merge or leave the topology of the hitherto assigned part of the workspace unchanged. Thus, the complexity of the workspace during the assignment procedure can grow to $O(\eta N)$. Thus, from the perspective of time-complexity analysis, anchored area partitioning of workspace with initial complexity $O(N)$ is equivalent to partitioning a workspace with complexity $O(\eta N)$ without the condition of anchors imposed. The preprocessing time is not affected by the introduction of the anchors. This leads to the anchored area partitioning algorithm having a time complexity of $O(N \log N + \eta(\eta N))$, i.e., $O(N \log N + \eta^2 N)$.

Chapter 7

Conclusions and Future Directions

UAVs are being increasingly used for reconnaissance especially in missions that are typically dull, dangerous and 'dirty'. Several missions such as battle damage assessment, urban and airbase reconnaissance and magnetic detection of mines by an onboard sensor require UAVs to operate from low altitudes. Flying low can also help prevent occlusion due to cloud cover and permit the use of cheaper and lighter unstabilized imaging sensors. This advantage however comes with an inherent limitation in the sense that it puts severe restrictions on the size of the footprint of the sensor.

It is highly desirable in time critical reconnaissance operations that the mission time be minimized. Additionally, planning efficient paths can significantly increase the operational endurance of low endurance UAVs. These may be achieved in a hierarchical manner by first computing optimal workspace division amongst multiple

UAVs and then computing optimal waypoints for each UAV within its assigned portion of the workspace. The idea of optimality with regard to workspace division is closely linked to the notion that a UAV that can perform region coverage at a higher rate should be given a correspondingly larger proportion of the total workspace than the other UAVs.

We presented an algorithm for partitioning an arbitrary contiguous rectilinear workspace into η contiguous rectilinear parts whose areas are in a pre-specified ratio. To the best of our knowledge, this is the first time that an algorithm for solving this problem has been reported. The algorithm was motivated by a need to partition a rectilinear workspace amongst η UAVs in proportion of their relative capabilities in order to maximally parallelize their usage. We proved that our algorithm runs in $O(N \log N + \eta N)$ time.

We gave an $O(N \log N + \eta^2 N)$ time algorithm to solve the problem of partitioning a holed contiguous rectilinear polygon into η pieces that are each contiguous rectilinear polygons whose area-ratios are proportional to a pre-specified ratio and, each of whose shared boundary with the workspace contains a pre-specified point. This algorithm has direct application to the problem of maximally parallelizing the task of region coverage using η UAVs. The last constraint on the elements of the partition ensures that UAVs that are outside the workspace when the instruction for coverage is received can begin coverage upon flying only a minimum distance from their current locations to the region of interest.

In addition to contiguity, rectilinearity and area proportionality of the parts that comprise the partition, compactness is desirable. The reasons for this are *a*) it lowers

the potential for collision amongst the UAVs, *b*) it potentially lowers the standard deviation in the time for mission completion and, *c*) it eases the total bandwidth requirement by allowing better spatial reuse. Our algorithm does not address the issue of compactness of partition. The problem of generating compact partitions that satisfy all constraints on the partitions listed in this work is NP-hard. In the near future we plan to use a genetic-algorithm [70, 71, 72] based heuristic to increase the average compactness of the polygons in the partition generated by our algorithm. Figure 7.1 shows preliminary results of compaction obtained using genetic algorithms.

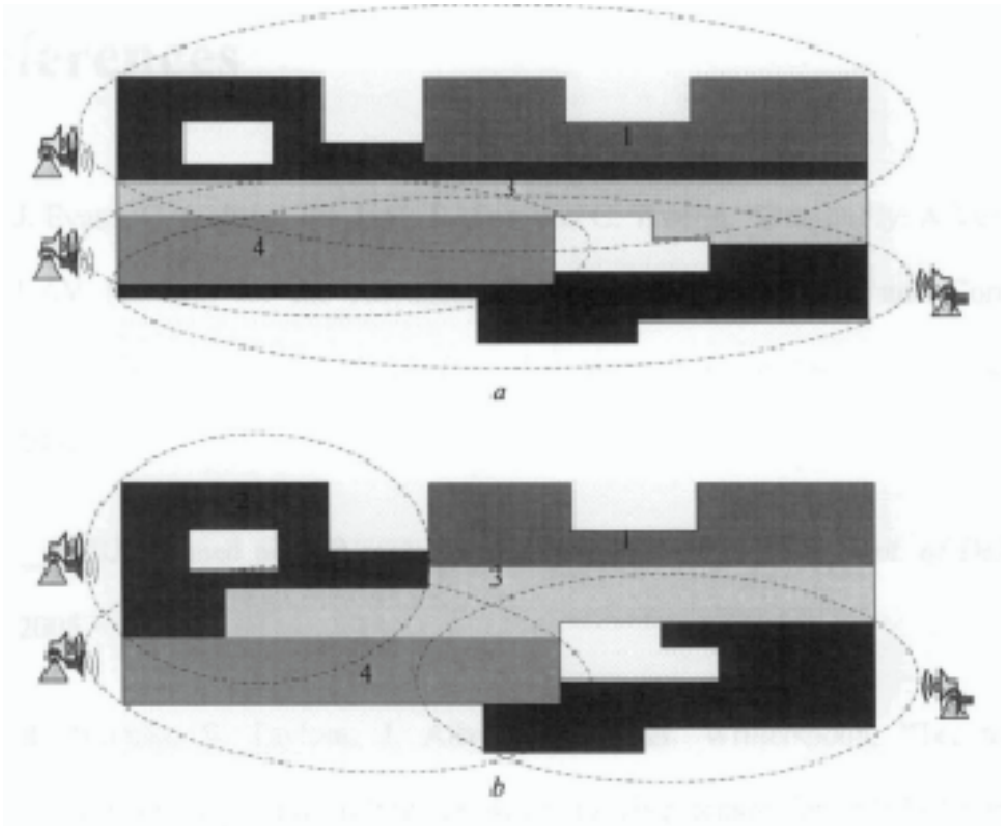


Figure 7.1: (*a*) Partition generated by our algorithm and minimal area footprints for assigning bandwidth for UAV_2, UAV_4 and UAV_5. (*b*) Preliminary results using a GA-based approach for partition compaction.

References

[1] J. Evans, G. Inalhan, J.S. Jang, R. Teo, and C. Tomlin, "DragonFly: A Versatile UAV Platform for the Advancement of Aircraft Navigation and Control," *Proceedings of the 20th IEEE Digital Avionics Systems Conference*, October 2001.

[2] ___, "Unmanned aircraft systems roadmap 2005-2030," *US Dept. of Defense*, 2005.

[3] B. Blumea, S. Taylora, J. Albersa and N.H. Witherspoon, "Technology assessment of passive millimeter wave imaging sensor for standoff airborne mine detection," *Proceedings of SPIE*, vol. 3079, 1997.

[4] R. Innocenti, "RF sensor solutions for small, lightweight unmanned aerial vehicles," *Radar Sensor Technology IX, Proceedings of SPIE*, vol. 5788, edited by Robert N. Trebits, James L. Kurtz, 2005, Bellingham, USA.

[5] G. Koh, "Ultra-wideband FMCW radar for detection of anti-personnel, detection and remediation technologies for mines and minelike targets V," *Proceedings of SPIE*, vol. 4038, edited by A.C. Dubey, J.F. Harvey, J.T. Broach and R.E. Dugan, 2000.

[6] H.S. Trammell III, A.R. Perry, S. Kumar, P.V. Czipott, B.R. Whitecotton, T.J. McManus and D.O. Walsh, "Using unmanned aerial vehicle-borne magnetic sensors to detect and locate improvised explosive devices and unexploded ordnance," *Sensors, and C3I Technologies for Homeland Security and Homeland Defense IV*, *Proceedings of SPIE*, vol. 5778, edited by Edward M. Carapezza, 2005, Bellingham, USA.

[7] C.W. Lum, R.T. Rysdyk and A. Pongpunwattana, "Autonomous airborne geomagnetic surveying and target identification," *AIAA Infotech@Aerospace Conference*, AIAA-2005-7039, 2005, Arlington, USA.

[8] D.M.T. Cochrane, P.A. Manning and T.A. Wyllie, "Uncooled thermal imaging sensor for UAV applications," *Infrared Technology and Applications XXVII*, *Proceedings of SPIE*, vol. 4369, edited by B.F. Andresen, G.F. Fulop, M. Strojnik, 2001.

[9] J.S. Zeleka, M. Holbeina, K. Hajebia, D.C. Asmara and D. Cheng, "IR depth from stereo for autonomous navigation," *Infrared Imaging Systems: Design, Analysis, Modeling, and Testing XVI*, *Proceedings of SPIE*, vol. 5784, edited by G.C. Holst, 2005, Bellingham, USA.

[10] J.H. Kim, S. Sukkarieh, "Airborne simultaneous localisation and map building," *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, pp. 406-411, 2003.

[11] M. Scheutz, P. Schermerhorn, and P. Bauer, "The Utility of Heterogeneous Swarms of Simple UAVs with Limited Sensory Capacity in Detection and Tracking Tasks," *IEEE Swarm Intelligence Symposium*, pp. 257-264, 2005, Pasadena, USA.

[12] C. Kreucher, K. Kastella and A. Hero, "A Bayesian method for integrated multi target tracking and sensor management," *IEEE International Conference on Information Fusion*, vol. 1, 2003, pp. 704-711, 2003.

[13] B.S.Y. Rao, H.F. Durrant-Whyte and J.A. Sheen, "A fully decentralized multi-sensor system for tracking and surveillance," *International Journal of Robotics Research*, vol. 12, no. 1, pp. 20-44, 1993.

[14] T. Zhang and D. Freedman, "Improving performance of distribution tracking through background mismatch," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 2, pp. 282-287, 2005.

[15] G. Sachs and O. da Costa, "Optimization of dynamic soaring at ridges," *AIAA Atmospheric Flight Mechanics Conference and Exhibit, AIAA 2003-5303*, 2003, Austin, USA.

[16] Y.J. Zhao and Y.C. Qi, "Minimum fuel powered dynamic soaring of unmanned aerial vehicles utilizing wind gradients," *Journal of Optimal Control Applications and Methods*, vol. 25, pp. 211-233, 2004.

[17] Y.C. Qi and Y.J. Zhao, "Energy-Efficient Trajectories of Unmanned Aerial Vehicles Flying through Thermals," *Journal of Aerospace Engineering*, vol. 18, no. 2, pp. 84-92, 2005.

[18] J.C.R. Torroella, *Long Range Evolution-based Path Planning for UAVs through Realistic Weather Environments*, Masters Thesis, University of Washington – Seattle, USA, 2004.

[19] A. Chakravarthy and D. Ghose, "Obstacle avoidance in a dynamic environment: a collision cone approach," *IEEE Transactions on Systems and Cybernetics - Part A: Systems and Humans*, vol. 28, no. 5, pp. 562-574, 1998.

[20] K. Sigurd and J. P. How, "UAV trajectory design using total field collision avoidance," *AIAA Guidance, Navigation, and Control Conference*, AIAA-2003-5728, 2003, Austin, USA.

[21] A. Agarwal, M.H. Lim and M.J. Er, "A VoMVi complex for supporting optimal path queries for UAVs," *International Symposium on Voronoi Diagrams in Science and Engineering*, 2005, Seoul, S. Korea.

[22] H.I. Yang and Y.J. Zhao, "Trajectory planning for autonomous aerospace vehicles amid known obstacles and conflicts," *Journal of Guidance, Control and Dynamics*, vol. 27, no. 6, pp. 997-1008, 2004.

[23] F. Ruffier and N. Franceschini, "Visually guided micro-aerial vehicle: automatic take off, terrain following, landing and wind reaction," *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, pp. 2339-2346, 2004.

[24] P. Riseborough, "Automatic take-off and landing control for small UAVs," *Proceedings of the 5th Asian Control Conference*, vol. 2, pp. 754-762, 2004.

[25] T. Matsuyama and N. Ukita, "Real-time multitarget tracking by a cooperative distributed vision system," *Proceedings of the IEEE*, vol. 90, no. 7, pp. 1136-1150, 2002.

[26] T. Berhard and J.P. Le Carde, "Distributed target tracking for nonlinear systems: application to bearings-only tracking," *Proceedings of the 8th International Conference on Information Fusion*, vol. 1, 2005.

[27] I. Kroo, "Design and Development of the SWIFT: A foot-launched seaplane," *AIAA-00-4336*, 2000.

[28] J.S. Bellingham, *Coordination and Control of UAV Fleets Using Mixed-Integer Linear Programming*, Ph.D. Thesis, Massachusetts of Technology, Cambridge, USA, 2002.

[29] J.C. Latombe, *Robot Motion Planning*, ISBN: 079239206X, Springer, 1991.

[30] M. Norsell, "Aircraft trajectories considering radar range constraints," *Aerospace Science and Technology*, vol. 6, pp. 83-89, 2002.

[31] Y.J. Chiang, J.T. Klosowski, C. Lee and J.S.B. Mitchell, "Geometric algorithms for conflict detection/resolution in airtraffic management," *Proceedings of the 36th IEEE Conference on Decision and Control*, vol. 2, pp. 1835-1840, 1997, San Diego, USA.

[32] R. Teo, Computing Danger Zones for Provably Safe Closely Spaced Parallel Approaches: Theory and Experiment, Stanford University, USA, 2003.

[33] V. Isler, D. Sun and S. Sastry, Roadmap based pursuit-evasion and collision avoidance, Proceedings of Robotics: Science and Systems, Cambridge, USA, 2005.

[34] D.M. Stipanovic, G. Inalhan, R. Teo and C.J. Tomlin, "Decentralized overlapping control of a formation of unmanned aerial vehicles," *Automatica*, vol. 40, no. 8, pp. 1285-1296, 2004.

[35] M. Mehrandezh and K. Gupta, "Time-optimal rendezvous planning for pick-and-place task sharing," *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, pp. 2703-2708, 2000.

[36] A. Huster, *Relative Position Sensing by Fusing Monocular Vision and Inertial Rate Sensors*, PhD Thesis, Stanford University, USA, 2003.

[37] S. Sukkarieh, E. Nettleton, J.H. Kim M. Ridley, A. Goktogan and H- Durrant-Whyte, "The ANSER project: data fusion across multiple uninhabited air vehicles," *The International Journal of Robotics Research*, vol. 22, nos. 7-8, pp. 505-539, 2003.

[38] S. Hert, S. Tiwari and V. Lumelsky, "A terrain-covering algorithm for an AUV," *Autonomous Robots*, vol. 3, no. 2, pp. 91-119, 1996.

[39] H. Bast and S. Hert, "The area partitioning problem," *Proceedings of the 12th Canadian Conference on Computational Geometry*, pp. 163-171, 2000, Fredericton, Canada.

[40] S. Hert and V. Lumelsky, "Polygon area decomposition for multiple-robot workspace decomposition," *International Journal of Computational Geometry and Applications*, vol. 8, no. 2, pp. 437-466, 1998.

[41] H. Choset, Y. Zhang and M.J. Schervish, "Path planning for robotic demining: robust sensor-based coverage of unstructured environments and probabilistic methods," *International Journal of Robotics Research*, vol. 22, no. 7, pp. 441-466, 2003.

[42] M.B. Bieterman and D.R. Sandstrom, "A curvilinear tool path method for pocket machining," *SME Journal of Manufacturing Science and Engineering*, **125**, pp. 709-715, 2003.

[43] A. Agarwal, M.H. Lim, M.J. Er and C.Y. Chew, "ACO for a new TSP in region coverage," *IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*), pp. 3176-3181, 2005, Edmonton, Canada.

[44] S.O. Joon, H.C. Yoon, B.P. Jin and Y.F. Zheng, "Complete coverage navigation of cleaning robots using triangular-cell-based map," *IEEE Transactions on Industrial Electronics*, vol. 51, no. 3, pp. 718-726, 2004.

[45] M. Mohd, "ATR technology holds the key to a better, faster, and a safer way to perform postattack air base damage assessment," *Proceedings of SPIE - Surveillance Technologies and Imaging Components*, edited by Sankaran Gowrinathan, C. Bruce Johnson and James F. Shanley, vol. 1952, pp. 109-118, 1993.

[46] A. Agarwal, M.H. Lim and L.C. Woon, "A divide and conquer algorithm for rectilinear region coverage," *IEEE Conference on Robotics, Automation & Mechatronics*, 2004, Singapore.

[47] A. Agarwal, M.H. Lim and N.T. Nguyen, "Computing the minimum grid of an orthogon," *under review*.

[48] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," *Annals of Mathematics and Artificial Intelligence*, vol. 31, nos. 1-4, pp. 77–98. 2001.

[49] A. Bemporad, C. Filippi and F.D. Torrisi, "Inner and outer approximations of polytopes using boxes," *Computational Geometry: Theory and Applications*, vol. 27, no. 2, pp. 151-178, 2004.

[50] E.M. Arkin et al., "Optimal covering tours with turn costs," *ACM-SIAM Symposium on Discrete Algorithms*, pp. 138-147, 2001, Washington D.C., USA.

[51] A. Itai, C. H. Papadimitriou and J. L. Szwarcfiter, "Hamilton paths in grid graphs," *SIAM Journal on Computing*, **11**, pp. 676-686, 1982.

[52] J.M. Keil, Polygon decomposition, in *Handbook of Computational Geometry*, J.-R. Sack, J. Urrutia (Eds.), Elsevier, pp. 491-518, 2000.

[53] E. Acar et al., "Path planning for robotic demining: robust sensor-based coverage of unstructured environments and probabilistic methods," *Intl. J. of Robotics Research*, vol. 22, no. 7, pp. 441-466, July 2003.

[54] G. Reeb, "Sur les points singuliers d'une forme de Pfaff completement integrable ou d'un fonction numerique," *Comptes Rendus Acad. Sciences Paris*, vol. 222, pp.847-849, 1946.

[55] W.H. Huang, "Optimal line-sweep-based decompositions for coverage algorithms," *IEEE Intl. Conf. on Robotics & Automation*, vol. 1, pp. 27-32, May, 2001.

[56] A. Zelinsky, R. Jarvis, J. Byrne and S. Yuta, "Planning paths of complete coverage of an unstructured environment by a mobile robots," *Int. Conf. on Advanced Robotics*, pp.533-538, November 1993.

[57] S.X. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Trans. on Systems, Man & Cybernetics, Part B*, vol. 34, issue 1, pp. 718-724, 2004.

[58] U. Elsner, "Graph partitioning - a survey," *Preprint SFB393/97-27, Technische Universitat Chemnitz*, 1997.

[59] C.H. Papadimitriou and M. Sideri, "The bisection width of grid graphs," *ACM-SIAM Symposium on Discrete Algorithms*, pp. 405-410, 1990, San Francisco, USA.

[60] J. Yackel, "Minimum Perimeter Tiling in Parallel Computation," *PhD Thesis*, University of Wisconsin - Madison, USA, 1993.

[61] T.C. Shermer, "A linear algorithm for bisecting a polygon," *Information Processing Letters*, **41**, pp. 135-140, 1992.

[62] M. De Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer-Verlag, 2nd edition, 2000.

[63] V.S. Dobbs, H.W. Davis and C. Lizza, "An application of heuristic search techniques to the problem of flight path generation in a military hostile environment," *International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Sys*tems, vol. 1, pp. 273-280, 1988.

[64] E.M. Arkin, S.P. Fekete and J.S.B. Mitchell, "Approximation algorithms for lawn mowing and milling," *International Journal of Computational Geometry and Applications*, vol. 17, pp. 25-50, 2000.

[65] E.M. Arkin, S.P. Fekete and J.S.B. Mitchell, "The lawnmower problem," *Proceedings of the 5th Canadian Conference on Computational Geometry*, pp. 461-466, 1993.

[66] V. Akman, W.R. Franklin, M. Kankanhalli, and C. Narayanaswami, "Geometric computing and uniform grid technique," *Computer-Aided Design*, vol. 21, no. 7, pp. 410-420, 1989.

[67] H. Imai and T. Asano, "Efficient algorithms for geometric graph search problems," *SIAM Journal on Computing*, vol. 15, no. 2, pp. 478-494, 1986.

[68] R. Klette and A. Rosenfeld, *Digital Geometry: Geometric Methods for Digital Image Analysis*, 1st Edition, Morgan Kaufmann, 2004.

[69] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd Edition, MIT Press, 2001.

[70] C.P. Erick, A.F. James, K. Deb, D. Lawrence and R. Roy (eds.): *Proceedings Genetic and Evolutionary Computation Conference – GECCO*. Springer Verlag Pub., July 2003, Chicago, USA.

[71] M. Mitchell, *An Introduction to Genetic Algorithms*, Reprint Edn. MIT Press, 1998.

[72] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Pub. Co., 1989.

Chapter 8

Author's Publications

8.1 Journals

1. F. Harary, M.H. Lim, A. Agarwal and D. Wunsch, "Algorithms for Derivation of Structurally Stable Hamiltonian Signed Graphs," *International Journal of Computer Mathematics*, vol. 81, No. 11, pp. 1349–1356, November 2004.

2. A. Agarwal, M.H. Lim, M.J. Er and T.N. Nguyen, "Rectilinear Workspace Partitioning for Parallel Coverage Using Multiple UAVs," *to appear*, *Advanced Robotics: International Journal of Robotics Society of Japan.*

3. C.W. Yeu, M.H. Lim, G.B. Huang, A. Agarwal and Y.S. Ong, "A new machine learning paradigm for terrain reconstruction," *to appear*, *IEEE Geoscience & Remote Sensing Letters.*

8.2 Conference Papers

1. A. Agarwal, M.H. Lim, M.J. Er and T.N. Nguyen, "Anchored partitioning of a rectilinear polygon for region coverage using multiple UAVs," *IEEE/AIAA Aerospace Conf.*, March 2006, Big Sky, USA.

2. N.T. Nhat, M.H. Lim, A. Balasuriya and A. Agarwal, "Collision avoidance algorithm for an underwater robot," *3rd International Conference on Computational Intelligence, Robotics & Automation*, December 2005, Singapore.

3. A. Agarwal, M.H. Lim and M.J. Er, "A VoMVi complex for supporting optimal path queries for UAVs," *International Symposium on Voronoi Diagrams in Science & Engineering*, October 2005, Seoul, S. Korea.

4. A. Agarwal, M.H. Lim and M.J. Er, "ACO for a New TSP in Region Coverage," *IEEE/RSJ International Conference on Intelligent Robots & Systems* (IROS), August 2005, Edmonton, Canada.

5. A. Agarwal and M.H. Lim, "Fast ACO for a non-Euclidean Δ-STSP," *RIUPEEEC*, July 2005, Hong Kong.

6. A. Agarwal, M.H. Lim and M.J. Er, "Proportional Partition of Holed Rectilinear Region amongst Multiple URAVs," *IEEE Conference on Robotics & Automation* (ICRA), April 2005, Barcelona, Spain.

7. A. Agarwal, M.H. Lim and L.C. Woon, "A Divide and Conquer Algorithm for Rectilinear Region Coverage," *IEEE Conference on Robotics, Automation and Mechatronics*, December 2004, Singapore.

8. A. Agarwal, M.H. Lim, M.J. Er & Y. W.K. Maung, "Inflight Rerouting for an Unmanned Aerial Vehicle," *Genetic & Evolutionary Computation Conference* (GECCO), June 2004, Seattle, USA.

9. A. Agarwal, M.H. Lim, C.Y. Chew, T.K. Poo, M.J. Er and Y.K. Leong, "Solution to the Fixed Airbase Problem for Autonomous URAV Site Visitation Sequencing," *Genetic & Evolutionary Computation Conference* (GECCO), June 2004, Seattle, USA.

10. A. Agarwal, M.H. Lim, Y. Xu and Y.S. Ong, "Evolutionary Graph Mining for the Discovery of Site Visitation Sequences for a Single URAV," *2nd International Conference on Computational Intelligence, Robotics & Automation*, December 2003, Singapore.

11. A. Agarwal, Y. Xu and M.H. Lim, "Graph Compression via Compaction of Degree-2 Nodes for URAV Visitation Sequencing," *Unmanned Vehicle Systems Technology*, December 2003, Brussels, Belgium.

12. A. Agarwal, M.H. Lim and M.J. Er, "Model-Solution Framework for Minimal Risk Planning for URAVs," *Genetic & Evolutionary Computation Conference* (GECCO), June 2004, Seattle, USA.

13. A. Agarwal, "Algorithms for rectilinear polygon partitioning for workspace division amongst k UAVs," *Workshop on Autonomous MAV*, Center for

Aerospace Systems Design & Engineering., IIT Bombay, June 2005, Mumbai, India.

14. A. Agarwal, M.H. Lim and M.J. Er, "Case based modeling of regions of coverage interest," *Aerospace Technology Seminar*, Singapore, February 2005.

15. A. Agarwal and M.H. Lim, "Path planning for low flying vehicles for region coverage: challenges and algorithms," *Shephard's Unmanned Vehicles Asia-Pacific Conference & Exhibition*, February 2005, Singapore.

16. A. Agarwal, M.H. Lim and M.J. Er, "Risk minimal flight planning for aerial reconnaissance vehicles," *Intelligent Systems: Principles and Approaches, NTU CoE Technology Week Focus Seminar*, March 2004, Singapore.

17. M. Insall, A. Agarwal and D. Wunsch, "Representation of uncertainty in a lattice theoretic framework," *AMS Special Session on Many Lives of Lattice Theory and the Theory of Ordered Sets with Connections to Combinatorics*, January 2002, San Diego, USA.