# Path Planning Algorithms For Single And Multiple Mobile Robot Systems

A THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

**Doctor of Philosophy**

BY

PARIKSHIT MAINI

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY DELHI

NEW DELHI– 110020

SEPTEMBER, 2020

# Acknowledgements

First and foremost, I would like to thank my dissertation supervisor Dr. P. B. Sujit for his support and unwavering patience. His truthful and timely assessments have helped me correct my course and bring this journey to fruition. I have had a lot to learn from him over the course of my dissertation in terms of self-belief and the value of hard work. I would also like to acknowledge Dr. S. Rathinam (Texas A&M University), Dr. P. Tokekar (University of Maryland) and Dr. K. Sundar (Los Alamos National Laboratory) for their contribution in this thesis. The work in Chapters 2 and 5 was done in collaboration with Dr. Rathinam. His positivity and non-judgemental attitude allowed me to make free conversations with him and seek his advice on both technical and personal matters. The work in Chapters 3 and 4 was done in collaboration with Dr. Tokekar. In addition to his technical knowledge I also learnt a lot from his focused approach to research. Dr. Sundar has been a friend and colleague. He also contributed on the work done in Chapter 2. I could always reach out to him and ask for advice. I would also like to acknowledge Prof. V. Isler (University of Minnesota), with whom I am currently working. He has been very understanding and supportive during the last phase of my Ph.D.

I would like to thank IIIT-Delhi and all the people that make it a closely knit family. It is hard to imagine this journey anywhere else. I would like to acknowledge and thank Prof. Pankaj Jalote, who is the founding director of IIIT-Delhi and was the director during my time here. I have had the chance to interact with him on multiple occasions. His encouragement and positivity are contagious, and always helped me get inspiration for my work. I would also like to thank Prof. Sumit Roy (University of Washington). I took the Research Methods course with him at IIIT-Delhi, that he taught as a visiting faculty. He gave me the motto 'know thyself' and it has remained with me to this day. I would like to thank my thesis review committee members, Dr. S. Anand and Dr. S. Butail, who gave me helpful feedback on my work and Dr. R. Raman (IIIT-Delhi) for many useful technical discussions. I would like to thank my friends at IIIT-Delhi, foremost Alvika and Milan, who always had my back. I would like to thank Mandeep, Anupriya, Anil, Garvita, Lokender, Anjli, Dheryta, Monalisa, Gautam and Kevin, for their help in various capacities during the course of my Ph.D. This thesis would not have been possible without the support of the administrative staff, especially Mrs. Priti Patel who

was always available for any assistance. There are many other people I would like to thank for helping me complete my dissertation and making this journey worthwhile, including members of the administrative staff, finance, purchasing and IT departments at IIIT-Delhi for their help at different points in time.

I am grateful to the TCS Foundation who generously sponsored my Ph.D. under the TCS Ph.D. fellowship program. They have been very supportive and understanding.

I humbly acknowledge and cannot thank enough, the support of my family: my mother, Mrs. Sangeeta Maini and my father, Dr. Ravinder Kumar Maini, for the life-values they gave me and the times they brought me up from my rock-bottom; my sisters, Dr. Poornima Modi and Kamal Maini for teaching me to strive for excellence in whatever I do. My brother-in-law Dr. Nikhil Modi, who stood by me as my elder brother; my parents-in-law Mrs. Anuradha Vohra and Mr. Ritu-Raman Vohra, for their love and understanding; dadima, papaji, nanima and nanaji for their love; Gayatri and Yatharth for their love and Aarav for being the little bundle of joy that he is.

Lastly but never the least, I would like to thank my wife, Dr. Jayati Vohra. It is because of her that I have completed my dissertation. Her faith, her love, support and motivation were unwavering. She has been with me from the beginning through to the finish line. This is our thesis, Jaya.

-Parikshit

# Certificate

This is to certify that the thesis titled "Path Planning Algorithms For Single And Multiple Mobile Robot Systems" being submitted by Parikshit Maini to INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY DELHI, for the award of the DOCTOR OF PHILOSOPHY, is an original research work carried out under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

Dr. P. B. Sujit
DEPARTMENT OF ELECTRONICS AND COMMUNICATIONS ENGINEERING
INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY DELHI
September, 2020

# Abstract

Mobile robots have become ubiquitous today and are used widely in sensor networks. Their use in applications involving long-term operations, remote sensing and complex environments is a widely studied area of research due to their high precision capabilities over extended periods of time and low cost operation. This thesis develops methods for path planning of mobile robot systems for use in novel applications under various application and system specific constraints. First, path planning algorithms for visual surveillance applications are developed. The problems addressed therein relate to long-term autonomy, visual monitoring on terrains, cooperative operations and large-scale coverage. The proposed solutions consider environment and robot limitations such as visibility obstructions, terrain restrictions and refueling considerations. Second, path planning for robots with a curvature-constrained motion model is considered. This motion model is widely used to model mobile robot kinematics and poses computationally hard problems in path planning. This thesis addresses two interesting problems in this space. The results focus on practical considerations and implementability of the solution approach.

**Cooperative Air-Ground Route Planning for Large-Scale Coverage Applications**

This problem considers the use of an aerial robot for large-scale visual coverage in two dimensional planar environments. The use of aerial robots for surveillance is limited by their fuel capacity. The approach discussed in this work uses a ground-based mobile refueling vehicle, constrained to travel on the road, to increase operational range of the aerial robot in both space and time. The planning algorithm determines paths for both aerial and ground robots for visual coverage and coordinated rendezvous visits to refuel the aerial robot. The solution techniques include a branch-and-cut based exact method; and a construction heuristic for computationally efficient solutions. The techniques are validated in field demonstrations using real robots.

**Persistent Monitoring of Piece-Wise Linear Features on Terrains using Multiple Aerial and Ground Robots**

Persistent monitoring on terrains using multiple mobile robotic sensors requires coordinated planning. Ter-

rain features add visibility obstacles and limited fuel capacity of aerial robots leads to range restrictions, that make the problem challenging. This work addresses the visual-monitoring problem on piece-wise linear features within a terrain using multiple mobile robots for persistent operations. The planner must account for visual coverage, refueling aerial robots during the mission and placement of refueling depots while also utilizing the available sensor diversity to minimize overall costs for the monitoring mission. Building on existing literature on visibility in specific classes of polygons and fuel-constrained routing, this thesis develops a discrete representation of the problem that allows the design and application of discrete optimization techniques to find optimal solutions. It develops a Mixed Integer Linear Programming formulation and a branch-and-cut implementation to speed-up computation. It also includes the design of a construction heuristic based on competitive construction of robot paths using a step-increment strategy. The thesis reports results from computational simulations and illustrates proof-of-concept using experiments on real robots.

**Multi-Point Visual Monitoring on a Terrain using an Aerial Robot**

The multi-point monitoring problem on terrains using an aerial robot extends the results of visual monitoring on planar environments and 1.5 D terrains. The points-of-interest may be located anywhere within the terrain. A solution strategy to the problem involves addressing the visibility constraints due to topographical features and camera field-of-view limitations. This work develops a visibility computation strategy based on existing techniques on terrain visibility, to compute visibility regions fully contained in the flight plane. The route planning problem is posed as an instance of TSP with neighbourhoods (TSPN). A constant-factor approximation algorithm is designed and construction of the proof for the approximation bounds is discussed in detail. The bounds on solution are computed based on properties of the visibility regions and the terrain. The problem is further reduced to an instance of generalized TSP (GTSP) and a practically useful approach that uses a standard GTSP solver to solve the reduced problem is developed. A branch-and-cut method based on a new ILP formulation is also developed to solve and benchmark the solutions found using the GTSP solver. Proof-of-concept experiments were performed using an aerial robot for paths computed by the planner.

**Curvature Constrained Trajectory Planning for a Mobile Robot through a sequence of points: A Perturbation - based Approach**

Curvature constrained motion modeling is a popular modeling technique for a majority of wheeled ground mobile robots as also altitude-constrained flight operations of fixed-wing aerial robots. In this work, Dubins' model for a forward moving car-like vehicle is used to account for the curvature constraint. Path planning for the model is challenging for any sequence of more than two points and is known to be NP-hard. Hence, one cannot expect to design polynomial time exact algorithmic solutions. This thesis develops a perturbation-based

method to design approximate paths through a sequence of points. The problem extends Dubins' original results on optimal paths between two locations, to a sequence of $n$ locations. It is shown that bounded perturbations of the location coordinates at precisely computed parameter values that correspond to discontinuity in the path length function, lead to computationally efficient solutions with path length close to known lower bounds.

**Path Planning for a UAV with Kinematic Constraints in the Presence of Polygonal Obstacles**

Building on the work on path planning through a sequence of points, this work develops a path planner for a curvature constrained vehicle in the presence of polygonal obstacles. It uses a visibility graph representation for the environment and develops a modified Dijkstra's shortest path algorithm as a first step in the two-step path planning approach. The second step performs a reverse search on the graph to find feasible paths and uses results of the first step as priors to speed up the search. Simulation results are used to substantiate the claims.

In summary, this thesis develops novel path planning algorithms for mobile robots that account for application and system-level constraints. Optimal planning algorithms and efficient heuristics are developed for air-ground robots operating individually and in cooperation, to perform visual coverage and persistent monitoring. The methods developed in the thesis are validated through computational simulations and field demonstrations. Further, practically useful path planners are developed for robots with a curvature constrained motion model for visiting a sequence of points and in the presence of obstacles.

# List of publications

## Journals

1. Parikshit Maini, Kaarthik Sundar, Mandeep Singh, Sivakumar Rathinam, and P. B. Sujit. "Cooperative aerial-ground vehicle route planning with fuel constraints for coverage applications." IEEE Transactions on Aerospace and Electronic Systems, Vol. 55, No. 6 , Dec. 2019, pp. 3016–3028.
2. Parikshit Maini, Pratap Tokekar and P. B. Sujit, "Visibility-based persistent monitoring of piece-wise linear features on a terrain using multiple aerial and ground robots", IEEE Transactions on Automation Science and Engineering, August 2020. [Early Access]
3. Parikshit Maini, P. B. Sujit and Pratap Tokekar, "Visual Monitoring of Points of Interest on a 2.5D Terrain using a UAV with Limited Field-of-View Constraint", submitted to IEEE Transactions on Aerospace and Electronic Systems, May 2020.

## Conferences

1. Parikshit Maini and P. B. Sujit. "On cooperation between a fuel constrained UAV and a refueling UGV for large scale mapping applications." In International Conference on Unmanned Aircraft Systems (ICUAS), Denver, CO, USA, June 2015, pp. 1370-1377.
2. Parikshit Maini and P. B. Sujit. "Path planning for a uav with kinematic constraints in the presence of polygonal obstacles." In International Conference on Unmanned Aircraft Systems (ICUAS), Arlington, VA, USA, June 2016, pp. 62-67.
3. Parikshit Maini, Sivakumar Rathinam, and P. B. Sujit. "Curvature constrained trajectory planning for a UAV through a sequence of points: A perturbation approach." In Asian Control Conference (ASCC), Gold Coast, QLD, Australia, December 2017, pp. 1276-1281.
4. Parikshit Maini. "Cooperative routing with refueling for aerial and ground vehicles for large scale surveillance: student research abstract." In ACM Symposium on Applied Computing, Pau, France, April 2018, pp. 847-848.
5. Parikshit Maini, Gautam Gupta, Pratap Tokekar, and P. B. Sujit. "Visibility-based monitoring of a path using a heterogeneous robot team." In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, October 2018, pp. 3765-3770.
6. Parikshit Maini, Kevin Yu, P. B. Sujit, and Pratap Tokekar. "Persistent monitoring with refueling on a terrain using a team of aerial and ground robots." In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, October 2018, pp. 8493-8498.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Mobile robots are ubiquitous today. Their types and variety have expanded to include micro-scaled robots that fit in the palm of the hand to large-scale defense-grade robots. They may operate on the ground, in the air and/or in water. With such large penetration, the horizon of use of mobile robots has widened significantly. Their popularity is attributed to many factors, including increased reliability of robotic hardware, robust software systems, low production costs, and the availability of accurate sensing and actuation mechanisms. In addition, the ability to operate autonomously in uncontrolled, natural and possibly cluttered environments, makes them well suited for many diverse tasks. They find application in field operations, like harvesting, fruit picking and weed mowing, and in structural monitoring for bridges, pipelines, industrial plants and the like. They are also used in surveillance, mapping, long-term monitoring and reconnaissance operations. Many applications have also benefited from the use of multiple robots operating in a cooperative setting. Low production and operating costs for the robots coupled with ease of operation admit the use of multiple robots with complementary capabilities. Multiple robots operating cooperatively help achieve better performance and lower the costs. A common component in the realization of these robotic solutions that involve autonomous mobile robots, is path planning (also called planning) for the robots.

Path planning is integral to autonomous operation of mobile robots and is required at all scales of operation. Ground robots, flying robots, manipulator arms, micro-scale mobile robots: each of these require path planning to perform useful tasks. A planning algorithm or a planner, computes a path for a mobile robot or a robotic component, to traverse from an initial state to a terminal state. The path must ensure feasibility. Feasibility may be defined in terms of problem-specific criterion such as a curvature constraint or the requirement to visit a set of intermediate states. Additionally, a planner aims to minimize the cost and maximize the reward. Traversing a path usually incurs a cost to the robot, that may quantify the distance traveled, time taken or the energy

consumed on the path. The reward on the other hand, may quantify information gain, number of successful visits to intermediate states or some measure of appropriateness of the path. A path planning problem usually involves at least one of the two, a cost function or a reward function. It may also involve both, or multiple instances of each function. The cost and the reward functions, along with the feasibility constraints in a path planning problem, are determined by the application and system specifications. In terms of implementation, path planning logic may operate in a higher abstraction module or may be embedded in low-level control layers. It may be done offline or in real time or as a combination of the two. Based on the application specific requirements, path planning may be performed individually for each robot or jointly for multiple robots. Given these large set of variations in the problem and solution space for path planning, it is forthright to appreciate the complexity and challenges involved. This thesis addresses a few important problems in path planning for single and multiple robot systems. The problems relate to the use of mobile robots that operate on or above the ground.

## 1.1 Overview

This thesis develops path planning solutions for single and multiple robot systems operating in a variety of environments. The problems addressed in this thesis are classified in two broad themes: 1). Path planning for visual monitoring and coverage using autonomous robots and 2). Path planning for nonholonomic robots.

Visual monitoring is one of the most important and popular application areas for mobile robots. This thesis addresses a set of problems related to visibility-based monitoring and coverage in outdoor environments. These environments are inherently dynamic and uncontrolled, and any large scale or long term operations also need to account for limited fuel capacity of the robots. Chapter 2 develops solutions for large-scale aerial coverage in two dimensional planar environments. It develops methods to utilize mobile ground robots as refueling stations to increase the operational range of aerial robots. Chapters 3 and 4 extend these results to visual monitoring within altitude varying terrains. Chapter 3 develops methods to the use of a heterogeneous set of ground and aerial robotic sensors for persistent monitoring along piece-wise linear features on terrains. Chapter 4 addresses the path planning problem for a single aerial robot to perform visibility-based monitoring for multiple points of interest that lie within a terrain. The results for each of these problems are augmented with field deployments using real robots.

Path planning for a robot with curvature constraints (nonholonomic) is a well studied area of research. This is because nearly all automobiles and a majority of wheeled ground mobile robots follow a nonholonomic motion model. Dubins [1] developed a constant-time algorithm to compute a point to point path for a car-like

vehicle in a two dimensional obstacle-free environment. The extension of these results to any sequence of more than two points and to environments with obstacles is hard and there exist important gaps in the current literature. Chapters 5 and 6 identify these problems and develop novel solution methods for planning paths for curvature constrained robots. A detailed review of existing literature highlighting the contributions of this thesis is also included within these chapters.

## 1.2 Contributions

### 1.2.1 Cooperative Air-Ground Route Planning for Large-Scale Coverage Applications

This problem considers the use of an aerial robot for large-scale visual coverage in two dimensional planar environments. The use of aerial robots for surveillance is limited by their fuel capacity. The approach discussed in this work uses a ground-based mobile refueling vehicle, constrained to travel on the road, to increase the operational range of an unmanned aerial vehicle (UAV) in both space and time. The aerial robot must rendezvous with the ground refueling vehicle to refuel as needed, to complete the coverage mission. A solution method for the problem must account for fuel constraints of the aerial robot, terrain restrictions of the ground vehicle and speed differential of the two vehicles. The planning algorithm determines paths for both aerial and ground robots for visual coverage and coordinated rendezvous visits to refuel the aerial robot. The solution techniques include a branch-and-cut based exact method; and a construction heuristic based on a repair mechanism for a TSP (Traveling Salesman Problem) tour of a reduced version of the problem. Comparative computational simulations are presented to discuss the efficacy of the developed approaches. Field deployments of the proposed approach using aerial and ground robots operating cooperatively are also presented.

### 1.2.2 Persistent Monitoring of Linear Features on Terrains using Multiple Aerial and Ground Robots

Visual monitoring on terrains using multiple heterogeneous robotic sensors is challenging due to obstructions to visibility and the need for coordinated planning. This thesis designs methods to perform long term persistent monitoring of piece-wise linear features within a terrain. The requirement for persistent monitoring adds additional constraints for refueling the robots. The placement of refueling depots within the environment is also considered within the planning problem. The terrain feature is modeled as an *x-monotone* curve in 1.5 dimensions and permits the computation of an efficient discrete representation of visibility regions to cover the terrain. A mixed-integer linear program (MILP) is formulated using this representation that allows the compu-

tation of exact solutions. A branch-and-cut implementation of the MILP is used to speed-up the computation. A construction heuristic based on competitive construction of robot paths is also designed. The planner optimizes over a min-max function for cost-balancing over all robots. Finally, the methods are evaluated in simulation and also demonstrated experimentally.

### 1.2.3 Multi-Point Visual Monitoring on a Terrain using an Aerial Robot

As extension to the results on visual monitoring in planar environments and piece-wise linear features on a terrain, this work addresses the multi-point monitoring problem 2.5 dimensional terrains. The points-of-interest may be located anywhere within the terrain and must be visited by a aerial robot equipped with a downward-facing camera with a limited field of view. A solution strategy to the problem involves addressing the visibility constraints due to topographical features and camera field-of-view limitations. This work develops a visibility computation strategy based on existing techniques on terrain visibility, to compute visibility regions fully contained in the flight plane. The route planning problem is posed as an instance of TSP with neighbourhoods (TSPN) and a constant-factor approximation algorithm is designed based on a characterization of the neighbourhoods within the TSPN version and the terrain. For practical purposes, an approximation algorithm may not be very useful and hence the problem is further reduced to an instance of generalized TSP (GTSP) and solved using a standardised GTSP solver. A branch-and-cut method for a new ILP formulation for the GTSP is also developed to benchmark the GLNS based approach. Experimental deployment were carried out using an aerial robot. The experiments were used to demonstrate the applicability and usability of paths generated by the planner for multi-point monitoring within a visibility-constrained environment.

### 1.2.4 Curvature Constrained Trajectory Planning for a Nonholonomic Mobile Robot through a sequence of points: A Perturbation - based Approach

Path planning for a robot with a curvature constrained motion model is challenging and most commonly occurring versions are known to be NP-hard. This model finds relevance due to its applicability to wheeled ground mobile robots, as also altitude-constrained flight operations of fixed-wing aerial robots. The problem addressed in this thesis builds on Dubins' [1] classical model for a nonholonomic vehicle and extends the results on path planning between two locations in an unobstructed two dimensional environment to a sequence of $n$ locations. The problem has been shown to be NP-hard and one cannot expect to design polynomial time exact algorithmic solutions due to the inherent hardness of the problem. This thesis develops a perturbation-based method to design approximate paths through a sequence of points. It is shown that bounded perturbations of

the location coordinates at precisely computed parameter values that correspond to discontinuity in the path length function, lead to computationally efficient solutions with path length close to known lower bounds. This approach is straight forward to implement and useful to practitioners, as mission objectives within many real world applications are not affected by small deviations in location coordinates.

### 1.2.5 Path Planning for a UAV with Kinematic Constraints in the Presence of Polygonal Obstacles

Path planning for an unmanned aerial vehicle (UAV) with kinematic constraints in the presence of polygonal obstacles is computationally hard and occurs commonly in many operations. Dubins' model [1] for a forward moving car-like vehicle is used to model the kinematic/curvature constraints of a UAV restricted to operate in constant-altitude setting. A visibility graph representation is used to capture the information about the environment and the obstacles. A modified Dijkstra's shortest path algorithm is designed as the first step in the two-step path planning approach. The algorithm in step one takes polynomial time in the number of obstacle vertices in the visibility graph. The second step performs a reverse search on the graph to find feasible paths and uses results of the first step as priors to speed up the search. Simulation results are presented to substantiate the claims.

## 1.3 Thesis Organisation

The rest of this thesis is organized in 6 chapters. Each of the five contributions listed above is written as an individual chapter. Each chapter is self-contained and can be read independent of the others. Chapter 2 addresses the fuel constrained UAV routing problem using a mobile refueling station. Chapter 3 develops solutions to the problem on persistent monitoring of piece-wise linear features on terrains using aerial and ground robots operating cooperatively. Chapter 4 addresses the multi-point aerial monitoring problem on terrains using a single aerial robot. Chapters 5 and 6 develop solutions to path planning problems for a curvature constrained mobile robot through a sequence of points and in the presence of obstacles, respectively. Chapter 7 makes concluding remarks and identifies future research directions.

# Chapter 2

# Cooperative Air-Ground Route Planning for Large-Scale Coverage Applications

Mapping and surveillance are important exercises undertaken in a variety of applications like precision agriculture[2], remote sensing [3], biodiversity surveys [4] and intelligence gathering missions [5]. Manual operations can be strenuous and time consuming, ranging anywhere from a few days to weeks [6]. Furthermore, some regions may not be reachable by land due to terrain restrictions, leading to incomplete surveys. Hence, small UAVs with on-board cameras have emerged as an economical alternative for mapping applications [4, 7].

For any reasonable-sized mission, a small, low-cost UAV may not be able to visit all points of interest or *targets* in a single sortie, thus requiring multiple refuels to visit all targets. An alternative is to make use of stationary refueling station(s) to refuel the UAV [8, 9]. The placement of refueling stations affects coverage area size and incurs additional cost for each refueling station. This work investigates the use of a mobile Ground

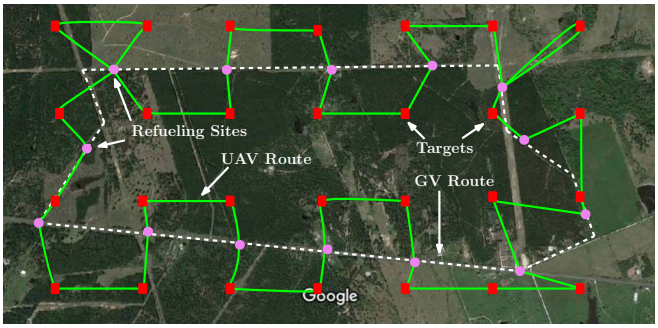



Figure 2.1: A sample scenario of a coverage application using a UAV. The UAV must visit all targets (red squares) to complete the mission. Due to limited fuel capacity, it needs to refuel to visit all targets. It may rendezvous with a GV, constrained to travel on the road-network, to refuel as needed. The figure depicts possible routes for the UAV and GV.

Vehicle (GV) as a refueling station to increase the operational range of a UAV in space and time. Figure 2.1 illustrates this scenario for a coverage application. The UAV must visit all the targets (red squares) to complete the coverage mission. However, limited fuel capacity makes it incapable of visiting all targets in a single flight. The UAV must rendezvous with the GV, restricted to traverse on the road-network, to refuel and extend its range of operation.

The cost of the mission, in terms of fuel consumed by the UAV, depends on the identification of an appropriate set of rendezvous locations for the UAV and the GV, referred to as *refueling sites*. The placement of refueling sites must ensure reachability of the targets and address fuel limitations of the UAV. In addition, as the GV is restricted to traverse on the road network, the refueling sites are restricted to be located on the road network. Furthermore, consecutive refueling sites must lie within the road-distance that the GV can travel in one flight time of the UAV, to ensure timely refueling. The speed differential of the two vehicles and the fuel and terrain restrictions described above, build a strong coupling in the UAV and GV routes. In this scenario, the following fuel-constrained UAV routing problem with a mobile refueling station (FCURP-MRS) arises naturally:

*Given a set of targets, a road network, a UAV and a GV; plan routes for the UAV and the GV such that 1) each target is visited by the UAV, 2) the UAV and the GV rendezvous at suitable locations on the road network to refuel so that the UAV never runs out of fuel, and 3) the total distance traveled by the UAV is a minimum.*

This work develops off-line planning techniques to compute solutions to FCURP-MRS and does not assume communication between the two vehicles. A two-stage strategy is designed for joint route planning for the UAV and GV to complete the mapping operation and rendezvous as needed. Contributions of this work are as follows:

- A two-stage approach to solve FCURP-MRS; the first stage computes a set of refueling sites on the road network while accounting for fuel constraints of the UAV and the speed differential of the two vehicles. The second stage solves the joint routing problem for the UAV and GV with refueling constraints and terrain restrictions.

- Mixed Integer Linear Programming (MILP) based formulations to optimally solve the coupled UAV-GV routing problem. A branch-and-cut implementation framework for the MILP formulations is also described.

- FCURP-MRS is an NP-hard problem, and hence a computationally efficient construction heuristic is also designed to generate feasible solutions that are benchmarked using lower bounds to the optimal solution

generated using the MILP formulations.

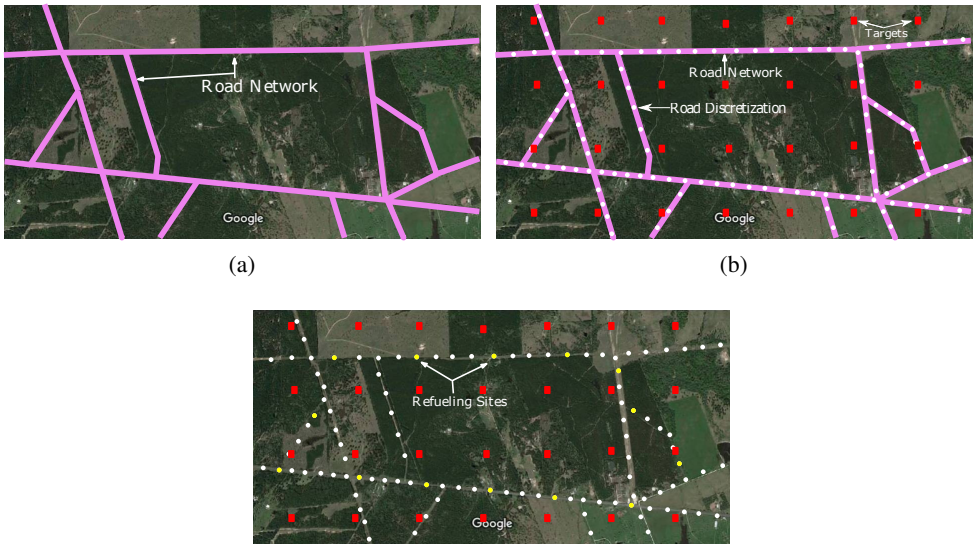
- Performance of all approaches is corroborated using extensive simulations based on solution quality and computation time.
- On-field experiments were conducted to verify the efficacy of the developed approaches.

## 2.1 Related work

FCURP-MRS is a generalization of the multiple-vehicle, multiple-depot, fuel constrained vehicle routing problem [10], that aims at computing minimum distance routes through a given set of targets for multiple vehicles while satisfying the fuel constraints for each vehicle using known, static fuel stations. Our problem generalizes the problem in [10] by enabling the fuel stations to be mobile and is NP-Hard. The notion of a mobile refueling station creates a strong coupling between the routes for the UAV and GV, and makes the problem challenging.

Khuller et al. [11] proposed the fuel-constrained routing problem in the context of a ground vehicle for a given set of stationary refueling stations; they developed a constant factor approximation algorithm for the problem. Fuel-constrained routing for a UAV, assuming that location of refueling stations are fixed and known a priori, has been addressed in [8, 9]. They develop MILP-based approaches and heuristics to determine UAV routes. Similar techniques have also been employed by authors in [12, 13] for parcel delivery and information gathering applications using multiple UAVs. In [14], a multi-objective evolutionary algorithmic approach is proposed to solve a persistent coverage problem for multiple UAVs. Alternate strategies for the multiple-vehicle, multiple-depot variants of the problem are proposed in [10, 15, 16]. Heuristics for the multiple-vehicle versions have been developed in [17]. The problem addressed in this paper, FCURP-MRS, differs from aforementioned work in the sense that the refueling station is mobile.

Erdogan and Miller-Hooks [18] develop construction heuristics for alternative fuel vehicle routes and validate the performance using numerical experiments. The complementary problem of optimal placement of recharging stations for electric vehicles to guarantee energy supply on any shortest path has been studied in [19]. Persistent monitoring and data delivery using UAVs have been addressed in [20, 21, 22]. Mathew et al. [22] also use mobile refueling stations to refuel the UAVs but they assume a priori knowledge of UAV trajectories and develop strategies on the use of multiple ground-based refueling vehicles to aid the UAVs. Recently, Yu et al. [23] have addressed a relaxed version of FCURP-MRS, wherein they assume zero speed differential between the aerial and ground vehicles, allow the ground vehicle to transport the UAV between two points and discretize the fuel consumption. This work, however, models fuel consumption as a continuous state variable




(a) (b)

(c)

Figure 2.2: (a) A characteristic environment with an interior road network. Refueling must happen along the road network as the RV is restricted to traverse on the road network. (b) The discretized locations on the road network that form the initial set of candidate refueling sites $\mathcal{S}$. (c) The set of refueling sites obtained after the first stage using the algorithm described in Section 2.3.

and explicitly addresses the speed differential of the two vehicles. It also considers terrain restrictions of the GV in the joint path planning problem. The speed differential mandates that consecutive refueling visits on the UAV path must be reachable by GV within a single flight time of the UAV to ensure that the UAV lands on the GV. This develops a strong coupling in the paths for the two robots and makes the problem very challenging. The current state of the art on joint-route planning of a UAV and a ground-based refueling vehicle is inadequate and there does not exist mathematical formulations to estimate the optimum. This work aims to bridge this gap in the literature by addressing the FCURP-MRS applied to a coverage application.

The remainder of the chapter is organized as follows: Section 2.2 gives an overview of the problem scenario and the underlying assumptions. Section 2.3 presents the site selection algorithm that forms the first stage of the solution approach. Sections 2.4 and 2.5 develop algorithms for the second stage of the solution approach. In particular, the Section 2.4 develops MILP formulations for the routing problem and presents a branch-and-cut algorithm to solve the MILP formulations and Section 2.5 details a computationally efficient heuristic algorithm. Section 2.6 compares the various algorithms and presents simulation results. On-field experimental setup and accompanying results are documented in Section 2.7. Concluding remarks are discussed in Section 2.8.

## 2.2 Application scenario

Consider an environment ($\mathcal{E}$) with an interior road network as shown in Fig. 2.2(a), a UAV with a fixed down-facing camera on board and a GV that traverses on the given road network. To perform a mapping mission, $\mathcal{E}$ is discretized into cells of equal size. The size of a cell is equal to the camera foot print of the UAV at a given altitude. The grid centers are denoted as targets or data points ($T$) in $\mathcal{E}$.

**UAV Model:** The UAV is assumed to be multi-rotor or VTOL (Vertical Take-Off and Landing) type of vehicle flying under ideal conditions without wind. It travels at a constant altitude and a constant air speed, $V_u$. UAV autopilot may use autonomous landing techniques (see [24, 25, 26] and references within), to land on the GV. The refueling process on the GV may be automated using battery swap systems [27, 28] or performed manually by a human operator.

Further, the following assumptions are made on the UAV model. Fuel consumed by the UAV to travel from location $i$ to $j$ is proportional to euclidean distance (in 2D) between $i$ and $j$ and is independent of flight maneuvers. For ease of exposition, the proportionality constant is assumed to be unity. Given constant speed for the UAV, distance traveled by the UAV is also proportional to flight duration. Hence, distance traveled, time of flight and fuel consumed are used interchangeably through the rest of the text, unless otherwise indicated. Let $U$ be the fuel capacity of the UAV available after compensating for a take-off and landing sequence; due to the previous assumption, $U$ is also the maximum distance the UAV can travel when starting at full fuel capacity. It may hover at the refueling site if the GV has not reached, but the sum of traveling time and hovering time must not exceed $U$.

**Ground vehicle model:** GV travels with a constant speed, $V_r$, between refueling sites and remains stationary during rendezvous operations with the UAV. Let $t_u = U/V_u$ denote the maximum flight time of the UAV in one sortie, then $R = t_u V_r$ is the maximum distance traveled by the GV, on road, in one flight time of the UAV. The road-distance traveled by the GV between two points in the environment is at least as large as the euclidean distance between the two points. The GV has a sufficient supply of UAV batteries and never runs out of fuel itself. Since the aim of this work is to plan routes for the UAV and GV, vehicle dynamics are not considered.

## 2.3 Refueling Site Selection

Refueling site selection problem is a precursor to route planning for both UAV and GV. A greedy algorithm (Algorithm 1) is presented to compute a set of refueling sites so that the joint optimization problem in the

second stage is feasible. To begin with, the road network is discretized at a resolution $\triangle << R$. The set of candidate refueling sites, denoted by $\mathcal{S}$, comprises of the discretized locations on the road network (Fig. 2.2b). Let $T$ denote the set of targets. For a subset, $S \subset \mathcal{S}$, of candidate refueling sites, to be deemed as a valid set of refueling sites, it must satisfy the following two conditions:

**1) Coverage condition:** Each target $t \in T$ has a refueling site $s \in S$ such that the distance between $s$ and $t$ is at most $U/2$ units. This distance also corresponds to the fuel consumed by the UAV to travel from $s$ to $t$.

**2) Connectedness condition:** Refueling sites in $S$ form a connected component in the graph $G_r$, where $G_r \equiv (S, E)$; an edge $(i, j)$ where $i, j \in S$ is contained in the set $E$ if and only if the distance between $i$ and $j$ via the road network (road distance) is at most $R$ units.

**Lemma 1.** *The above conditions ensure that if there exists a solution to the site selection problem, then there always exist a feasible solution to the joint routing problem of a UAV and GV in the presence of refueling constraints for the UAV.*

*Proof.* A trivial feasible solution may be computed as follows [29]: assuming that the GV and the UAV are initially stationed at $s \in S$, the UAV visits every unvisited target that is reachable from $s$ with refueling visits to $s$ after each target visit. Once all targets reachable from $s$ are visited, $s$ is marked *processed* and the GV and UAV proceed to the nearest unprocessed refueling site from $s$ (feasible due to connectedness condition). This sequence is repeated until all targets are visited. □

The solution computed as described above may not be optimal but provides a baseline to benchmark the results. This method was also designed as part of this work and is used in the Section 2.6 for comparison with other solution methods.

**Algorithm 1** Refueling Site Selection

1: $S = \{s_0\}$ ▷ $s_0$ : initial refueling site (starting point)
2: $T = T \setminus H(S)$
3: **while** $T \neq \phi$ **do**
4: $\quad s_{max} = \underset{s \in N(S)}{\operatorname{argmax}}(|H(s) \setminus H(S)|)$
5: $\quad S = S \bigcup \{s_{max}\}$
6: $\quad T = T \setminus H(s_{max})$
7: **end while**
8: **return** $S$

The objective of the refueling site selection problem is to compute a subset $S \subseteq \mathcal{S}$ of low cardinality that satisfies *coverage* and *connectedness* conditions. To that end, let $H(s)$ and $N(s)$ be two sets associated with every candidate refueling site $s \in \mathcal{S}$ . The *reachable set*, $H(s)$, comprises of all targets reachable from $s$, while

the *neighbor set*, $N(s)$, comprises of neighboring candidate refueling sites of $s$ within $R$ road distance. Also, $H(S) = \bigcup_{s \in S} H(s)$ and $N(S) = \bigcup_{s \in S} N(s)$. A target $t \in T$ is said to be reachable from $s$ if the distance between $t$ and $s$ is at most $U/2$. To compute such a subset $S$ of candidate refueling sites $\mathcal{S}$, a greedy algorithm as detailed in Algorithm 1 is used. The algorithm works as follows. It starts from a predefined initial refueling site (starting point), $s_0$, where the two vehicles are stationed. Thereafter, it iteratively selects a refueling site $s_{max} \in \mathcal{S}$, that covers the maximum number of uncovered targets and lies in the neighborhood of the current set of refueling sites (line 4). $s_{max}$ is then added to $S$ (line 5). This process is repeated until all targets are reachable from $S$, i.e. $H(S) = T$. An illustration of the solution obtained using the greedy refueling site selection algorithm (Algorithm 1) is shown in Fig. 2.2c. The worst case time complexity of the algorithm is $\mathcal{O}(|T||\mathcal{S}|)$.

## 2.4 Routing problem

Given the minimal set of refueling sites, stage two must design the routes for the UAV-GV team. The UAV and GV are initially stationed at a common refueling site and return to the same location after the mission. Let $\mathcal{P}$ represent a feasible UAV route for the routing problem. Let $\mathcal{P}_i$ be the $i^{\text{th}}$ subpath of $\mathcal{P}$, traversed by the UAV between two consecutive refueling sites, $s_j$ and $s_k$. Also, let $F_i$ be the fuel consumed and $\tau_i$ be the set of targets visited by UAV on $\mathcal{P}_i$ and $r_{jk}$ be the road distance between $s_j$ and $s_k$. Then $\mathcal{P}$ satisfies the following constraints: (i) $\bigcup_i \tau_i = T$ *i.e.*, all the targets are visited by the UAV, (ii) $\tau_i \bigcap \tau_j = \emptyset, \forall i, j,\ i \neq j$ *i.e.*, each target is visited exactly once by the UAV, (iii) $F_i \leq U, \forall i$ *i.e.*, the UAV never runs out of fuel, and (iv) $r_{jk} \leq R, \forall i$ *i.e,* the refueling sites $s_j$ and $s_k$ at the start and end of each subpath $\mathcal{P}_i$ are within road distance $R$ to ensure that GV reaches $s_k$ before UAV runs out of fuel. Since the goal of the problem is aerial coverage using a UAV, the objective function is formulated to minimize the total fuel consumed (proportional to distance traveled) by the UAV,

$$\min \sum_{i:\mathcal{P}_i \in \mathcal{P}} F_i. \tag{2.1}$$

The fuel constrained UAV routing problem with mobile refueling station (FCURP-MRS) problem can be formulated in the Mixed Integer Linear Programming (MILP) framework. The FCURP-MRS is formulated on a complete directed graph $G = (V, E)$, with vertex set $V = T \bigcup S$ and edge set $E$. [1] UAV and GV are initially stationed at a refueling site $s_0 \in S$. Associated with the edge set are two weight functions: $f : E \rightarrow \mathbb{R}^+$, where $f_{ij}$ denotes the fuel consumed by UAV when it travels along the directed edge $(i, j)$ and $r : (S \times S) \rightarrow \mathbb{R}^+$,

[1] A common constraint in UAV path planning is no-fly-zone. If any target or candidate refueling sites are in no-fly-zone, then they may be removed before the site selection stage. If a travel edge passes through the no-fly-zone, then a large cost is assigned to the edge. This mechanism will ensure no route is allowed inside the no-flying-zone.

that represents the road distance between two refueling sites. Let $N : S \to \wp(S)$, where $\wp(S)$ is the power set of $S$, denote a neighborhood function defined as $N(s_i) := \{s_j : r_{ij} \leq R, s_j \in S\}$. Refueling constraints for UAV can be formulated using node-labeling or edge-labeling approach.

### 2.4.1 Node-based formulation

In this formulation, decision variables on each vertex of the graph $G$ are used to formulate FCURP-MRS. The set of decision variables used in the formulation are defined as follows. $x_{ij}$, defined for each edge $(i, j) \in E$, are binary variables that represent whether or not the UAV traverses the edge $(i, j)$. Associated with each target $t_i \in T$ is a variable $u_i$ that represents the amount of fuel left in the UAV when it reaches $t_i$. $y_{ij}$ are binary decision variables, defined for every pair $< t_i, s_j >$: $t_i \in T$ and $s_j \in S$, and take value 1 if $s_j$ is the most recently visited refueling site when the UAV is at target $t_i$. Also, for any subset of vertices $P \subseteq V$, define $\delta^+(P) := \{(i, j) : (i, j) \in E, i \in P, j \notin P\}$. The objective and the constraints of FCUPR-MRS are wriiten as follows:

*Objective*:

$$\mathcal{F}_1 : \ \min \sum_{i \in V} \sum_{j \in V} f_{ij} x_{ij}. \tag{2.2}$$

*Degree constraints*:

$$\sum_{i \in V \setminus \{j\}} x_{ij} = \sum_{i \in V \setminus \{j\}} x_{ji}, \quad \forall j \in V \text{ and} \tag{2.3}$$

$$\sum_{i \in V \setminus \{t\}} x_{it} = 1, \quad \forall t \in T. \tag{2.4}$$

The degree constraints in Eq. (2.3) enforce in-degree to be equal to out-degree $\forall j \in V$. Constraint (2.4) ensures that each target is visited exactly once by the UAV. By not limiting the in/out-degree for refueling sites, UAV is permitted to make multiple visits to the refueling sites.

*Sub-tour elimination constraints*:

$$\sum_{(i,j) \in \delta^+(P)} x_{ij} \geq 1, \quad \forall P \subseteq V \setminus \{s_0\}, P \cap T \neq \phi. \tag{2.5}$$

Constraint set (2.5) eliminates sub-tours in the UAV route by enforcing a path to exist from the initial refu-

eling site to every target in the set $T$. It may be observed that the number of such constraints in the formulation is exponential; a dynamic cut-generation procedure is detailed in Section 2.4.4 to add these constraints into the problem as needed, without having to enumerate all of them.

*Fuel constraints*:

$$u_t - u_j + f_{jt} \leq M(1 - x_{jt}), \quad \forall t, j \in T, \tag{2.6}$$

$$u_t - u_j + f_{jt} \geq -M(1 - x_{jt}), \quad \forall t, j \in T, \tag{2.7}$$

$$u_t - U + f_{kt} \leq M(1 - x_{kt}), \quad \forall t \in T, \forall k \in S, \tag{2.8}$$

$$u_t - U + f_{kt} \geq -M(1 - x_{kt}), \quad \forall t \in T, \forall k \in S, \tag{2.9}$$

$$-u_t + f_{tk} \leq M(1 - x_{tk}), \quad \forall t \in T, \forall k \in S, \tag{2.10}$$

$$f_{ij} \cdot x_{ij} \leq U \quad \forall i, j \in S, \tag{2.11}$$

$$\sum_{i \in V} \sum_{j \in V} f_{ij} x_{ij} \leq U \sum_{k \in S} \sum_{i \in V} x_{ki}. \tag{2.12}$$

The fuel constraints (2.6)–(2.12) ensure that the UAV does not run out of fuel as it traverses its route. In particular, constraints (2.6) and (2.7) ensure fuel conservation when the UAV travels between two targets. Constraints (2.8)–(2.10) enforce similar restrictions on the UAV when it travels between a refueling site and a target. In all of these constraints, $M$ represents a large constant $M = U + \max_{(i,j)} f_{ij}$. It may be noted that UAV can reach a refueling site with some fuel left in the vehicle. Constraint (2.11) restricts direct paths between refueling sites to exist only between sites at most $U$ distance away. Constraint (2.12) states that the total fuel consumed by the UAV must be at most equal to $U$ times the total number of refueling visits.

*Refueling site constraints*:

$$y_{ts} - x_{st} \geq 0, \quad \forall t \in T, s \in S, \tag{2.13}$$

$$y_{t_2 s} - y_{t_1 s} \leq (1 - x_{t_1 t_2}), \quad \forall s \in S, \forall t_1, t_2 \in T, \tag{2.14}$$

$$y_{t_2 s} - y_{t_1 s} \geq -(1 - x_{t_1 t_2}), \quad \forall s \in S, \forall t_1, t_2 \in T, \tag{2.15}$$

$$\sum_{k \in S \setminus N(s)} x_{tk} \leq (1 - y_{ts}), \quad \forall t \in T, s \in S, \tag{2.16}$$

$$\sum_{k \in S \setminus N(s)} x_{sk} = 0, \quad \forall s \in S, \tag{2.17}$$

$$\sum_{s \in S} y_{ts} = 1, \quad \forall t \in T. \tag{2.18}$$

Constraints (2.13)–(2.18) limit the road distance between consecutive refueling sites to be at most $R$. Constraints (2.13)–(2.15) set the value of $y_{ts}$ variables to appropriate value using UAV route decision variables. Constraints (2.16) and (2.17) restrict refueling visits on the UAV path to neighbors of the most recently visited refueling site as computed by the neighborhood function $N(s)$. Constraint (2.18) imposes the restriction of marking exactly one refueling visit as the most recently visited refueling site for each target.

*Variable restrictions*:

$$x_{ij} \in \{0,1\}, \quad \forall (i,j) \in E, \text{either } i \text{ or } j \in T \tag{2.19}$$

$$u_i \in [0,U], \quad \forall i \in T, \tag{2.20}$$

$$y_{ts} \in \{0,1\}, \quad \forall t \in T, s \in S. \tag{2.21}$$

Finally, constraints (2.19), (2.20), and (2.21) enforce domain restrictions and bounds on $x_{tk}$, $u_i$, and $y_{ts}$ decision variables.

### 2.4.2 Edge-based formulation

In this formulation, decision variables on each edge of $G$ are used to formulate fuel constraints for the UAV. Binary decision variables $x_{ij}$ and $y_{ij}$ introduced for node-based formulation in the previous section are retained in the edge-based formulation while continuous decision variables $u_i$ are replaced with $z_{ij}$ variables, defined for each $(i,j) \in E$, to formulate fuel constraints. The decision variable $z_{ij}$, represents the amount of fuel used by the UAV to reach the $j^{th}$ vertex from a (most recent) refueling site when traveling along the incoming edge $(i,j)$. The incoming edge included in the UAV tour is uniquely defined for each target. Hence, the differentiating factor between node-based and the edge-based formulations for FCURP-MRS is in the way fuel constraints for the UAV are formulated.

*Fuel constraints*:

$$\sum_{i \in V} z_{ti} - \sum_{i \in V} z_{it} = \sum_{i \in V} f_{ti} x_{ti}, \quad \forall t \in T, \tag{2.22}$$

$$z_{ki} = f_{ki} x_{ki}, \quad \forall k \in S, \forall i \in V, \tag{2.23}$$

$$0 \leq z_{ij} \leq U x_{ij}, \quad \forall i, j \in V, \tag{2.24}$$

$$z_{ij} \in \mathbb{R}^{+}, \quad \forall i, j \in V. \tag{2.25}$$

The new formulation $\mathcal{F}_2$ replaces the fuel constraints (2.6)-(2.12) and variable restrictions on the $u_i$ variables (2.20), with constraints (2.22)-(2.24) and (2.25), respectively. Constraint (2.22) conserves the fuel at each target and (2.23) addresses the terminal cases. Constraints (2.24) and (2.25) specify the upper and lower bounds on $z_{ij}$ variables.

### 2.4.3 Ground vehicle route

The route for the GV is computed from the solution to the MILP formulation. The formulation ensures that consecutive visits to refueling sites on the UAV tour are within $R$ road distance. The tour comprising refueling site visits in the sequence as they occur on the UAV tour is a valid tour for the GV and ensures feasibility of the UAV tour. The tour so computed, ensures that the GV, when traveling at the constant speed $V_r$, always reaches the refueling site before the UAV runs out of fuel. The sum of traveling time and hovering time for the UAV is thus always constrained to be within $U$, allowing the UAV to land on the GV.

**Algorithm 2** Separation Algorithm

```
Build graph G(directed) ≡ (V, E)
Add edge (i, j) to E, for each x_ij = 1
P = strongly connected components in G
for all P ∈ P do
    if (|P| > 1)&&(P ⊆ V \ {s_0})&&(P ∩ T ≠ φ) then
        Add violated constraint (Eq. (2.26))
    end if
end for
```

### 2.4.4 Branch-and-cut algorithm

To optimally solve the two formulations presented in the Sec. 2.4.1 and 2.4.2, a branch-and-cut implementation is employed. Due to the presence of sub-tour elimination constraints (2.5), it is not computationally efficient

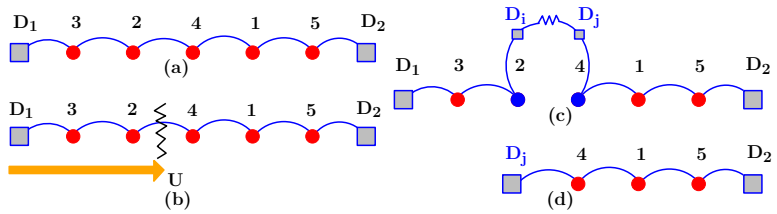



Figure 2.3: Tour repair algorithm for the TSP-based heuristic to generate feasible solutions to FCURP-MRS from TSP tours of the targets. Circles represent targets and squares mark refueling sites. Targets shown in blue color constitute a fuel violation. (a) A TSP tour with exactly two refueling sites, one at the start and other at the end. ($D_1 == D_2$, in case of original TSP sequence) (b) Detection of fuel violation. (c) Compute indirect path to fix the fuel violation. (d) Repair algorithm resumes from the second vertex in the violation.

to enumerate all constraints in Eq. (2.5) and provide them to a MILP solver. To address this issue sub-tour elimination constraints in the formulation are relaxed, and whenever the solver obtains an integer solution feasible to this relaxed problem, a check is made to find if any of the relaxed constraints are violated by the feasible solution. If so, the violated constraints are added to the formulation and the problem is given back to the solver. This process of adding violated constraints to the problem at runtime has been observed to be computationally efficient for many variants of the traveling salesman problem [30] and also the fuel constrained vehicle routing problems [15]. The algorithms used to identify violated constraints, also called *valid inequalities* are referred to as separation algorithms. Algorithm 2 presents the pseudo-code for a separation algorithm used to dynamically identify violated constraints (2.5) given an integer feasible solution for FCURP-MRS. The algorithm computes the strongly connected components (line 3) in the graph defined by the integer feasible solution. Each component $P$ that satisfies the condition $P \subseteq V \setminus \{s_0\}$, and $P \cap T \neq \phi$ (line 5) violates the corresponding constraints as given in Eq. (2.26). The violated constraints are then added to the formulation and the solver is allowed to optimize the problem with the new constraints.

$$\sum_{(i,j)\in\delta^+(P)} x_{ij} \geq 1, \quad P \cap T \neq \phi. \tag{2.26}$$

## 2.5 Heuristics

The MILP formulation and the branch-and-cut algorithm compute an optimal tour for FCURP-MRS, expectedly the computation time grows exponentially as the number of targets in the environment increases. In such cases, it is useful to find a *good* solution (feasible solutions) quickly by developing a fast heuristic for FCURP-MRS. The heuristic computes a TSP (Traveling Salesman Problem) tour of the targets and the initial refueling site

for the UAV using Lin-Kernighan-Helsgaun heuristic [31]. If the UAV does not run out of fuel as it traverses the tour then a feasible solution to FCURP-MRS has been found, else a repairing algorithm (Algorithm 3) is used to convert the tour into a feasible solution satisfying fuel and road distance constraints. The input to the repairing algorithm (Algorithm 3) is a sequence of targets and exactly two refueling sites (same or different), one at the beginning and the other at the end (Figure 2.3(a)). The algorithm traverses the UAV route starting at the initial refueling site while keeping track of fuel consumption. When it finds a fuel capacity violation, if any, it calls the $indirectPath$ routine (Algorithm 4) to fix the violation (see Figure 2.3(b)). It is marked by a pair of consecutive vertices on the UAV route, as shown in Figure 2.3(c). The *indirectPath* routine attempts to fix the violation by a computing a feasible subpath that only visits refueling sites. Once a feasible path is computed, it is inserted between the two vertices that constitute the fuel violation. The algorithm then resumes from the latter of the two vertices (target 4 in the figure) and continues to search for fuel violations (Figure 2.3(d)). In case a feasible path is not found, the algorithm backtracks to one previous vertex on the tour. It keeps backtracking until it finds a feasible path or reaches a refueling site .

**Algorithm 3** Repair Algorithm

```
1:  curr_vertex = start_of_tour
2:  while curr_vertex ≠ end_of_tour do
3:      check for fuel capacity violation between curr_vertex and next_vertex
4:      if fuel_capacity_violation == true then
5:          while feasible_path == not_found do
6:              compute indirectPath between curr_vertex and next_vertex
7:              if feasible_path == found then
8:                  insert indirectPath after curr_vertex
9:                  break
10:             else
11:                 curr_vertex = prev_vertex
12:             end if
13:         end while
14:     end if
15:     freeze curr_vertex
16:     curr_vertex = next_vertex
17: end while
```

The $indirectPath$ routine takes as input, $v_i$ and $v_j$, the two consecutive vertices on the UAV route that constitute the violation; $s_{mrv}$, the most recently visited refueling site; $U_{rem}$, amount of fuel remaining at $v_i$ and $S$, the set of all refueling sites. It computes a feasible *indirect path* comprising only of refueling sites between the two vertices as shown in Figure 2.3(c). To compute the feasible path, it builds a graph, $G_i$, with its vertex set as the union of the set of refueling sites and the two vertices on the UAV route. Edges in the graph exist between neighboring refueling sites. If $v_i$ is a target then edges connecting $v_i$ to refueling sites that lie in the intersection set of the sites reachable from $v_i$ and neighbors of $s_{mrv}$, are also added to $G_i$. If $v_j$ is a target,

it is connected to all of its neighboring refueling sites in $G_i$. The algorithm then computes the shortest path between $v_i$ and $v_j$ in $G_i$ and returns this as the feasible path to fix the violation. If a path from $v_i$ to $v_j$ cannot be found in $G_i$, it implies the graph is disconnected. As mentioned in Section 2.3, the set of refueling sites form a connected component and each target must be reachable from at least one refueling sites. Hence, if $G_i$ is a disconnected graph then $v_i$ is the disconnected vertex within $G_i$, implying there is not enough fuel available to reach a refueling site from $v_i$. In this case, Algorithm 3 backtracks to the previous vertex on the UAV route (line 11). Now, $v_i$ becomes $v_j$ and the previous vertex becomes $v_i$. It keeps backtracking until it finds a pair of vertices $v_i$ and $v_j$ for which it can compute an indirect path.

The heuristic will compute a feasible solution in at most $|T|$ iterations of the repairing algorithm and each iteration can make at most $|T|$ calls to the *indirectPath* routine. Hence, the computation time of the heuristic is polynomial in the number of targets.

**Algorithm 4** indirectPath

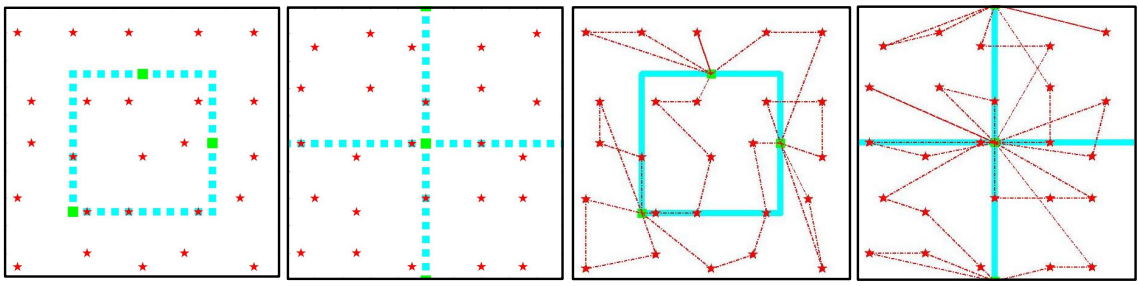
1: Build graph $G_i \equiv (V, E)$
2: $V \equiv S \cup \{v_i, v_j\}$
3: add edges between all neighboring refueling sites
4: **if** $v_i$ is a target **then**
5: add edges from $v_i$ to all reachable refueling sites that are also neighbors of $s_{mrv}$
6: **end if**
7: **if** $v_j$ is a target **then**
8: add edges from $v_j$ to all refueling sites within $U/2$ distance
9: **end if**
10: **return** $\mathcal{P} \equiv$ shortest path from $v_i$ to $v_j$ in $G_i$

## 2.6 Simulation Results

The performance of proposed solution approaches for FCURP-MRS are evaluated using simulations. MILP formulations implemented using the branch-and-cut algorithm presented in Sec. 2.4 are compared with TSP-based heuristic algorithm (Sec. 2.5) and a baseline greedy algorithm [29].

### 2.6.1 Simulation Setup

Two different computing systems were used for the simulations. System 1, used for small computation activity, is an HP Spectre 360 laptop with Intel i7 7500U processor with 2 cores and 16 GB RAM running Windows 10. System 2, used for computationally intensive simulations, is a High Performance Cluster node with 20 cores and 96 GB RAM running Cent OS 6.5.



(a) (b) (c) (d)

Figure 2.4: The figure shows sample instances used in the simulations. Cyan colored lines represent the road network and cyan squares represent road discretization points (candidate refueling sites), red dashed lines represent the UAV route, red stars represent data points and green squares represent refueling sites. The two road networks used are shown in (a) and (b). Figures (a) and (b) also show the candidate refueling sites (cyan) and refueling sites (green) selected from amongst those. Figures (c) and (d) show UAV routes computed by the MILP solver for the same instances for simulation parameter values, $U = 20$; $R = 15$ (road-network 1) and $R = 10$ (road-network 2).

| $n$ | Edge MILP | | Node MILP | |
|---|---|---|---|---|
| | road-nw 1 | road-nw 2 | road-nw 1 | road-nw 2 |
| 3 | 100 | 86.67(13.33) | 100 | 86.67(13.33) |
| 4 | 95.83 | 79.16(20) | 99.16 | 80(20) |
| 5 | 13.33 | 0(33.33) | 35 | 0(33.33) |
| 6 | 0 | 0(33.33) | 0 | 0(33.33) |

Table 2.1: Percentage of instances solved to optimality within 7200 seconds. Numbers in braces represent the percentage of infeasible instances, if any.

The site selection algorithm (Algorithm 1) was implemented in MATLAB and executed on System 1. MILP formulations were implemented in C++, using the traditional branch-and-cut framework and solver callback functionality of IBM ILOG CPLEX library version 12.7 and executed on System 2. A computational time limit of 7200 seconds was imposed on every run of the branch-and-cut algorithm. The heuristic was also implemented in C++ and executed on System 1. The baseline greedy algorithm [29] was implemented in MATLAB and executed on System 1. The solutions generated by the TSP-based heuristic were also input to the MILP solver as warm start for both formulations and the program was allowed to run for 7200 seconds. The solver was set to stop the optimization process at $1\%$ relative gap from the lower bound.

The performance of all the algorithms was tested with randomly generated test instances as described in the following section. Computation time and relative gap of the solution, using lower bound computed by CPLEX, are reported in Section 2.6.3.

| $n$ | Edge MILP | | Node MILP | |
|---|---|---|---|---|
| | road-nw 1 | road-nw 2 | road-nw 1 | road-nw 2 |
| 3 | 100 | 86.67 (13.33) | 100 | 86.67 (13.33) |
| 4 | 100 | 80 (20) | 100 | 80 (20) |
| 5 | 100 | 66.67 (33.33) | 100 | 66.67 (33.33) |
| 6 | 100 | 66.67 (33.33) | 100 | 66.67 (33.33) |

Table 2.2: Percentage of instances for which the MILP solver could compute a feasible solution when given a warm start using the solutions generated by the TSP-based Heuristic. Numbers in braces represent the percentage of infeasible instances, if any. Feasible solutions were computed for all feasible instances by the heuristic.

### 2.6.2 Instance Generation

A grid of size $20 \times 20$ square kilometers (km) is used for all simulations. The round trip distance between the two farthest points in the environment is approximately $57$ km; this distance is greater than the maximum distance a small UAV can travel in one flight time [32]. The value of $U$ is varied in the range of $15$ to $25$ km and $R$ is varied in the range of $10$ to $15$ km, both in steps of $5$. Two different road networks, as shown in Figure 2.4, are used. To achieve area coverage, an $n \times n$ grid is placed in the environment. The grid size $n$ is varied from $3$ to $10$, to denote different camera resolutions. Targets are placed at the grid centers and their number is varied in the range $9$ to $100$. $20$ instances are created for each configuration by randomly selecting target locations from within a uniform distribution around the grid centers.

The two different road networks used, are shown in Figure 2.4. Road Network 1 is very expansive within the environment and the mean shortest distance of any point in the environment from the road network is small. While in the case of Road Network 2, the mean shortest distance is much higher, specifically the points in each of the four corners of the environment are far off from the road network. The choice of particular road networks was made to include two representative extremes of road network coverage. In each case, the road network is discretized at a resolution of $1$ km to generate a set of candidate refueling sites (Figures 2.4(a) and 2.4(b)). Then, the greedy algorithm presented in Section 2.3 is used to obtain a reduced set of refueling sites. The set of refueling sites along with target locations are given as input to the second stage to compute routes for UAV and GV.

### 2.6.3 Results

Simulation instances generated as described in Section 2.6.2 were solved using six strategies namely (i) Edge-based MILP (ii) Node-based MILP (iii) Edge-based MILP with warm start (iv) Node based MILP with warm start (v) TSP-based heuristic (vi) Baseline greedy algorithm. The MILP solvers were used to generate optimal

| $n$ | Edge MILP | | Node MILP | |
|---|---|---|---|---|
| | road-nw 1 | road-nw 2 | road-nw 1 | road-nw 2 |
| 3 | 0.94 | 0.90 | 0.83 | 0.61 |
| 4 | 1.00 | 1.00 | 0.98 | 0.99 |
| 5 | 10.85 | 21.84 | 10.10 | 21.18 |
| 6 | 18.06 | 28.84 | 21.69 | 34.34 |

Table 2.3: Gap percentage from lower bound as computed by CPLEX (computed only for feasible instances) for feasible solutions computed by using the TSP-based heuristic solution as warm start to the formulations.

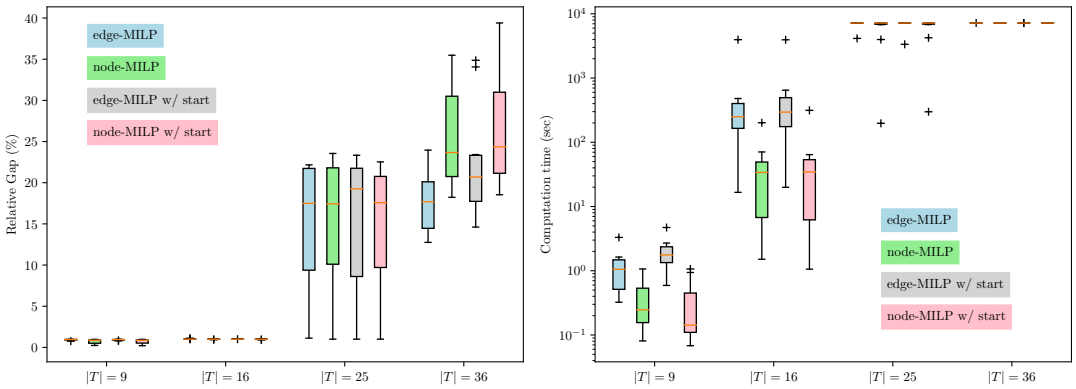



Figure 2.5: (a).Relative gap plot for MILP-based approaches for different instance sizes. (b). Computation time plot for MILP-based approaches for different instance sizes. The values are plotted over all values of the simulation parameters $U$ and $R$.

solutions for benchmarking purposes. In the case, when the optimal solution was not found within stipulated time limit, lower bounds generated by the solver were used as a proxy for the optimal objective. Table 2.1 shows the percentage of instances that were solved to optimality for different values of $n$. The solver was not able to compute optimal solutions for any instances with more than 25 targets ($n = 5$), hence the table only shows the numbers for up to 36 targets. The fraction of instances that were solved to *optimality* in the stipulated time period decreases drastically with increasing values of $n$. The numbers were slightly better for the node-based formulation. However, both formulations show a clear trend and do not scale well for larger instances. In case of road-network 2, some of the instances were infeasible due to the limited road network. The percentage of such instances is indicated in braces in the Table 2.1. The site selection algorithm in stage one of the strategy was not able to compute a feasible set of refueling sites for any of these instance. Tables 2.2 and 2.3 give computation results for the MILP formulations when the solver was given a warm start using solutions generated by the TSP-based heuristic. Table 2.2 reports the percentage of instances solved while Table 2.3 tabulates the relative gap of the solutions. The relative gap is computed as the percentage difference between the feasible solution and the lower bound as computed by the MILP solver. It is seen, that while the solver was able to compute feasible solutions for all instances, the solution quality degrades rapidly with increase in instance size. It may also be observed that the gap is higher for road-network 2. This is attributed to characteristics of the respective environments: large value of mean shortest distance of targets from the road in case of road network 2.

Figures 2.5a and 2.5b show box plots for relative gap of the MILP solutions from the optimal (or solver generated lower bound) and computation time taken by the MILP solver, respectively, for different instance sizes. The figures give insights on the performance of the two different MILP paradigms, namely edge-based and node-based. The edge-based formulation has lower relative gap than the node-based formulation, as may be observed from Figure 2.5a for large instance sizes. At the same time, the node-based formulations is computationally more efficient, for the instances where it is able to compute the optimal, as seen in Figure 2.5b. This is because of the lower dimensionality of the node-based formulations. Figures 2.5a and 2.5b also report comparative results for MILP solvers using TSP-based heuristic solutions as warm start. A counter intuitive insight from these figures is that the warm start does not improve the solution quality nor does it help the formulations to converge faster. This is not consistent with results for similar approaches for other combinatorial problems. The reason for these results is manifold. One, the heuristic solutions have significant gaps (Figure 2.6a) and do not give much improved starting points for the solver. Two, this problem has a very large number of constraints and has a strong coupling between the routes for the two vehicles. For these reasons, the heuristic solutions do not improve the MILP formulation.

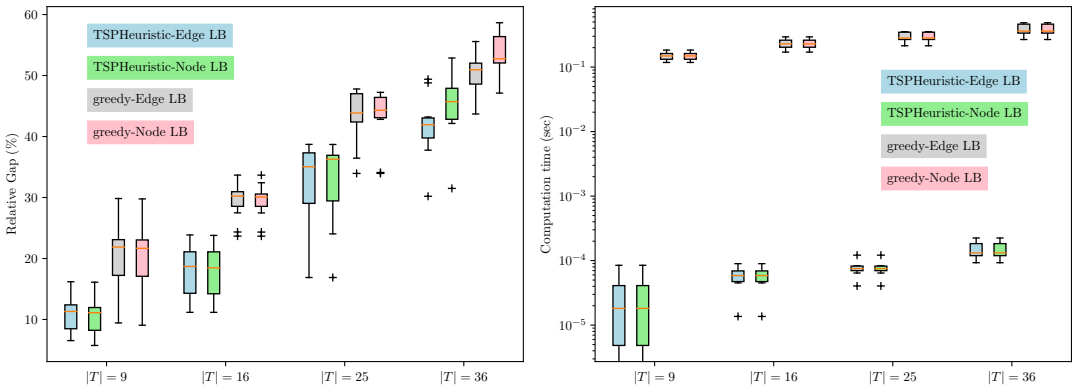



Figure 2.6: (a). Relative gap plot for heuristic approaches for different instance sizes. (b). Computation time plot for heuristic approaches for different instance sizes. The values are plotted over all values of the simulation parameters $U$ and $R$.

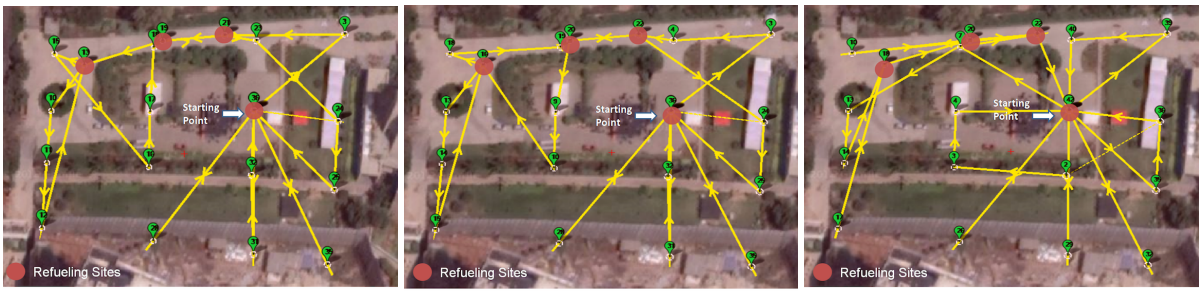
The TSP-based heuristic, however, does have its own merits. As may be observed from Figure 2.6b, the heuristic is extremely fast and completes execution in milliseconds. It consistently performs better than the greedy algorithm over all instance sizes. This is very useful in missions that need computationally efficient solutions and can afford a higher mission cost. Figure 2.6a reports the relative gap for both the TSP-based heuristic and the greedy algorithm using lower bounds generated by the MILP formulations. Even though the mission cost is same, the relative gap is persistently lower for the edge based formulation. This admits the conclusion that the edge based formulation generates tighter lower bounds than the node based formulation. The results shown in Figure 2.5a, when inspected with this knowledge on lower bounds, may be understood with a different perspective. The node based formulation, does not necessarily generate worse solutions. The higher value of the relative gap is also attributed to relatively loose lower bounds. The major takeaway from the simulations results may be summarized as follows: MILP-based approach, scales well for small- and medium-sized test instances and the TSP-based heuristic provides feasible solutions in a computationally efficient manner for small-, medium-, and large-sized test instances. One cannot expect to find optimal solutions for all instance sizes due to the hardness of the problem.

## 2.7 Field Experiments

The proposed solution techniques are demonstrated experimentally on a sample scenario using a quadrotor and a ground rover. The experiments were conducted at IIIT-Delhi campus in Delhi, India in an area of size approximately 110m $\times$ 90m. The environment and the contained road-network used in the experiments is shown in Figure 2.7a. 3D Robotics IRIS+ (Figure 2.7b) quadrotor is used as the UAV and was operated at an altitude of 15m, with a speed of 1 $m/s$. The refueling vehicle (Figure 2.7c) is a custom built RC rover with

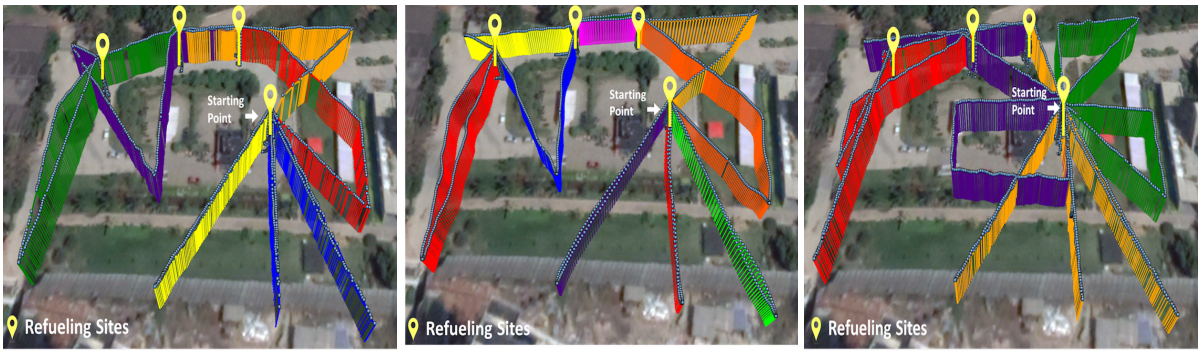

(a) (b) (c)

Figure 2.7: Experimental setup: (a) The environment showing data points (blue squares) and the road network (yellow line). Red line forms the area boundary. (b) UAV, 3DR IRIS+, used in the experiments. (c) Refueling Vehicle used in the experiments.




(a) (b) (c)

Figure 2.8: Ground Station view of UAV mission computed by (a) Edge-based MILP formulation within a time limit of 10 minutes (b) Node-based MILP formulation within a time limit of 10 minutes (c) TSP-based Heuristic Algorithm




(a) (b) (c)

Figure 2.9: Flight Altitude profile for the UAV to traverse the path. Each visit of the UAV to a refueling site simulates a refueling operation using a landing and take-off sequence. (a) UAV path for edge-based MILP formulation (b) UAV path for node-based MILP formulation and (c) UAV path for TSP-based heuristic algorithm

| | Edge MILP | Node MILP | Heuristic |
|---|---|---|---|
| Objective Value | 819.5 | 817.7 | 1078.3 |
| Lower Bound | 669.8 | 598.3 | - |
| Relative Gap (%) | 18.2 | 26.8 | - |
| Comp. Time (s) | 600 | 600 | 0.13 |

Table 2.4: Performance comparison of Edge-based MILP, Node-based and TSP-based Heuristic for the field experiment. The heuristic does not compute an explicit lower bound. Lower bounds generated by MILP formulations are used for performance evaluation.

Pixhawk autopilot running ArduRover firmware. GV was operated with a speed of 0.5 $m/s$. Both the vehicles started simultaneously from the first refueling site and traversed their respective missions. Maximum flight distance, $U$, for the UAV was fixed to 120m and the value of $R$, maximum distance traveled by GV in a single flight time of the UAV, was fixed as 40m. The refueling process was simulated by a rendezvous operation at each refueling site. The rendezvous operation involves a landing and take-off sequence of the UAV and the GV staying stationary at the same location during the operation. The video of the experiment may be seen in [33].

The number of targets within the environment were set to be 16. The road network was discretized at a resolution of 5m. UAV and GV routes were generated for edge-based MILP (Figure 2.8a), node-based MILP (Figure 2.8b) and TSP-based heuristic (Figure 2.8c). For each strategy, the actual altitude profile of the UAV while doing the experiment are shown in Figures 2.9a, 2.9b and 2.9c respectively. The routes generated for both vehicles were successfully traversed by the respective vehicles for each of the three experiments. The mission cost for the three strategies from the field experiments is given in Table 2.4. The table shows the objective function value, relative gap and computation time. As may be observed the edge-based formulation generates solutions with the lowest relative gap. The corresponding mission cost for the UAV in terms of distance traveled is minimum for the edge-based MILP formulation as a direct consequence of the same.

## 2.8 Conclusion

This work addresses the cooperative routing problem for a fuel constrained UAV and a terrain constrained ground based refueling vehicle in the context of a coverage application. A two stage solution strategy is designed to find efficient solutions to the problem, wherein the first stage computes a feasible set of refueling sites for UAV-GV rendezvous and the second stage performs joint route planning that satisfies the fuel and speed limitations of the two vehicles, respectively. Alternate MILP formulations and a computationally efficient heuristic algorithm are presented to solve the routing problem in the second stage. Extensive simulations and field experiments were carried out to verify the approaches on hardware implementations. The results show

that edge based MILP formulation computes tighter lower bounds than node based formulation. The heuristic scales well for large instances and gives promising results on time efficiency of the planner with trade off on solution quality. Field experiments are used to corroborate simulation results.

# Chapter 3

# Visibility-based persistent monitoring of piece-wise linear features on a terrain using multiple aerial and ground robots

Visual monitoring using autonomous mobile robots has been transformative in applications like Intelligence, Surveillance and Reconnaissance operations (ISR) [34, 35], disaster management [36, 37] and structural monitoring [38, 39]. In this context, the utility of cooperative aerial and ground robot systems for monitoring tasks is an active area of research and has garnered a lot of interest from the research community over the last decade. This chapter discusses the use of a cooperative heterogeneous robot system for persistent monitoring on a terrain. Applications like, rail-track monitoring [38, 40], power-line inspection [41], border & highway patrol [42] and river monitoring [43], involve extended features that may span several kilometers in length and often witness terrain variability in terms of altitude. This restricts visibility of robotic sensors and requires intricate planning algorithms for successful mission completion. This work models terrain features like roads, borders and pipelines as piece-wise linear ([43] and [44] also use similar models) and addresses the problem of persistent monitoring of such features using a cooperative system of aerial and ground robots (Figure 3.1). This class of problems are combinatorial in nature and are computationally intensive.

There exists literature that addresses cooperative routing for mobile robots [45, 46, 23, 47], persistent monitoring [48, 49, 34, 50, 51, 46, 23, 52] and other related problems. However, this work fills a gap in the literature and addresses the visibility-based persistent monitoring problem for piece-wise linear features on a terrain using autonomous aerial and ground robots. It includes negotiating visibility and fuel restrictions, joint

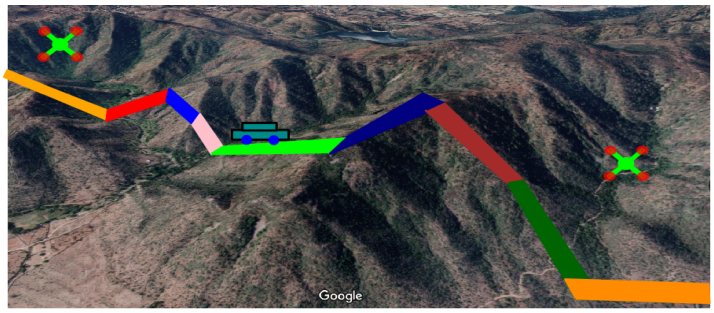



Figure 3.1: An example scenario of a piece-wise linear feature (colored segments) on a terrain where a team of robots consisting of two aerial vehicles and one ground vehicle are monitoring.

planning for a cooperative system of heterogeneous robots and the placement of refueling depots that comes into significance due to the requirements of a persistent mission. Mission cost is modeled to include one-time setup costs for refueling depots and operating costs for the robotic agents. The solution involves geometric modeling for visibility computation within the terrain, selection of a guard set, fuel-constrained route planning for robotic agents and placement of refueling depots on the terrain to ensure mission feasibility while minimizing the total cost. The terms tour, route and sequence are used interchangeably through the rest of the text.

## 3.1 Related Work

Determining tours for a robot to fully observe a given space is popularly known as the watchman routing problem (WRP) [53, 54]. WRP is a well-studied problem. The simple version of WRP in a polygon without holes may be solved in polynomial time[54]; for polygon with holes, Mitchell [55] devised an approximation algorithm with a $\log^2 n$-approximation factor and showed that the approximation factor cannot be improved beyond $c \log n$ for some constant $c > 0$. In the multiple watchmen ($n$-WRP) version [56], the goal is to compute tours for $n$ robots (watchmen) to cover the environment. Carlsson et. al. [56] show that $n$-WRP is NP-hard. Due to inherent complexity of the problem, practical solution approaches including decoupled viewpoint selection followed by robot routing [57] and self-organizing map heuristics [58] have been developed. There have also been efforts to design algorithms for restricted polygon domains: spiral polygons [59], histograms [60], and more recently street polygons[49, 56]. Exact[59, 60] and approximate [49] solution approaches for the $n$-WRP have been designed for these special polygonal domains by exploiting their structural characteristics. All of these approaches address the homogeneous robots case. This work addresses a persistent version of $n$-WRP to monitor a piece-wise linear feature on a terrain and extends the current state of the art by admitting the use of heterogenous robotic sensors.

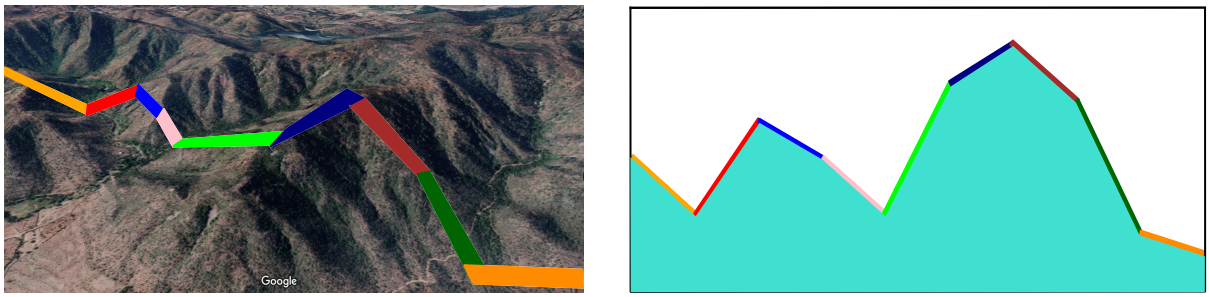
Persistent monitoring using mobile robots is also a well-studied problem in the literature. Variations of the

persistent monitoring problem for a single robot have been addressed in [50, 61, 8, 62]. Alamdari et. al. [50] introduce the idea of the kernel of an infinite walk as a closed walk that may be repeated to find an infinite walk for a mobile robot. Manyam et. al. [34] address the homogeneous multi robot persistent monitoring problem in the context of a data delivery application with constraints on data freshness. Mathew et. al. [63] solve a scheduling problem for mobile recharging stations to rendezvous with aerial robots for a persistent monitoring application. Ghazzai et. al. [37] develop a scheduling framework to optimize battery capacity of aerial robots to visit spatially and temporally distributed targets. The use of mobile ground robots as mobile refueling nodes has been explored in [51, 46, 23, 52]. Lasla et. al. [39] develop an approach to utilize public transport to extend the operational range of aerial robots and optimize energy consumption in a smart city scenario. The problem of placement of refueling depots is complementary to persistent monitoring and significantly affects the mission cost. [46] [23] and [63] address different versions of the fuel-constrained routing problem for aerial robots and use mobile refueling stations operating on the ground. They develop methods to determine rendezvous sites and time intervals to refuel the aerial robots and extend their range of operation. The work of Funke et. al. [19] addresses the problem of placement of recharging depots for electric automobiles ensuring they can travel long distances with minimal detours for recharging.

This work addresses both, route planning and placement of refueling depots, for persistent monitoring using a diverse set of robotic agents. Contributions of this work are as follows:

- It addresses the persistent monitoring problem for piece-wise linear features on altitude varying terrains using a heterogeneous set of ground and aerial mobile robotic sensors.
- The solution approach considers robot path planning and refueling depot placement as a coupled problem and determines solutions to both problems while optimizing on the overall mission cost.
- A Mixed Integer Linear Programming (MILP) formulation is developed for the persistent monitoring problem. A branch-and-cut based implementation to speed-up computation is also designed.
- A construction heuristic, called $\delta$-search, is developed based on competitive construction of robot paths using a step-increment strategy.
- The solution methods are validated using large scale computational simulations.
- Outdoor experiments performed using multiple aerial and multiple ground robots are used to establish proof-of-concept.

The rest of the chapter is organized as follows. Section 3.2 discusses preliminaries on terrain model, robot models, persistent monitoring and the placement of refueling depots.. Section 3.3 describes the application




(a) (b)

Figure 3.2: A piece-wise linear feature within a terrain may be *straightened out* to build a 1.5D model. The individual edges of the feature are color coded in the two figures.

scenario and establishes the problem statement. Section 3.4 develops the MILP and heuristic solution methods. Section 3.5 describes the computational simulations including setup, instance generation, examples and results. Section 3.6 describes the experiments used to illustrate the proof-of-concept. Section 3.7 concludes the chapter and identifies future research directions.

## 3.2 Preliminaries

This section discusses some preliminary ideas needed to develop the solution methods. Specifically, it formalizees the visibility-based model of the terrain and establishes the definitions used in the rest of the chapter. It also describes the chain visibility property and the robot model used.

### 3.2.1 Terrain Model

The environmental feature to be monitored is modeled as a 1.5 dimensional structure. As shown in Figure 3.2, a piece-wise linear feature within a terraineous environment may be straightened out in one dimension to build a 1.5D model. The 1.5D representation of a feature is $x$-monotone and is characterized by terrain points and reflex points (Figure 3.3a).

**Definition 1** (Terrain point)**.** *A terrain point is any point on the surface of the terrain that observes a change in slope.*

Consider a 1.5D terrain as shown in Figure 3.3a. The figure shows 12 terrain points marked using an asterisk symbol.

**Definition 2** (Reflex point)**.** *A reflex point is a terrain point where the slope decreases while going in the left to right direction on the terrain.*

The set of reflex points is a subset of the set of terrain points. Figure 3.3a shows 7 reflex points on the terrain, marked using a red asterisk symbol.

**Definition 3** (Chain Visibility)**.** *A curve $C$ and a set of points $X$ in 2D are said to be chain visible if for each point $x \in X$, the intersection of the visibility polygon, i.e. space that has an unrestricted view of $x$, and $C$ is either an empty set or a connected chain [49].*

*Connected chain* here refers to a connected and continuous curve. Figure 3.3b shows the visibility polygon for a point $x$ on the terrain. Examples of chain visible pairs includes street polygons and collapsed watchman routes [64], points on a 1.5D terrain and fixed altitude paths [49]. In the case of this problem, the fixed altitude path (line $C$ in Figure 3.3b) corresponds to the aerial robot flight altitude and the set of points $X$ comprises points that lie on the terrain.

**Definition 4** (Visibility segment)**.** *The surface of a 1.5D terrain between two consecutive reflex points forms a convex polyline and is called a visibility segment.*

Figure 3.3c shows a visibility segment marked as a blue colored line on the terrain. When the context is clear from the text, the term 'segment' is also used to refer to a 'visibility segment'. Each visibility segment is marked on the terrain by a left and a right reflex point.

**Definition 5** (Visibility region)**.** *The region on the constant-altitude flight path between extended projections of the right edge of the left reflex point and left edge of the right reflex point of the visibility segment, is defined as the visibility region of the given visibility segment.*[1]

Visibility region of a segment is a continuous curve and any point in the visibility region has an unobstructed view of the corresponding segment. Figure 3.3c shows the visibility region of a segment on the terrain on the constant altitude flight path. It also shows the extended projections from the left and right reflex points of the segment that intersect with the chain visible curve to mark the visibility region. The continuity of the visibility region is attributed to the property of chain visibility between the constant altitude flight path and the terrain, and convexity of the visibility segment.

[1] In case of visibility obstructions, the projections are suitably adjusted to pass through the obstructing reflex point to compute the visibility region.

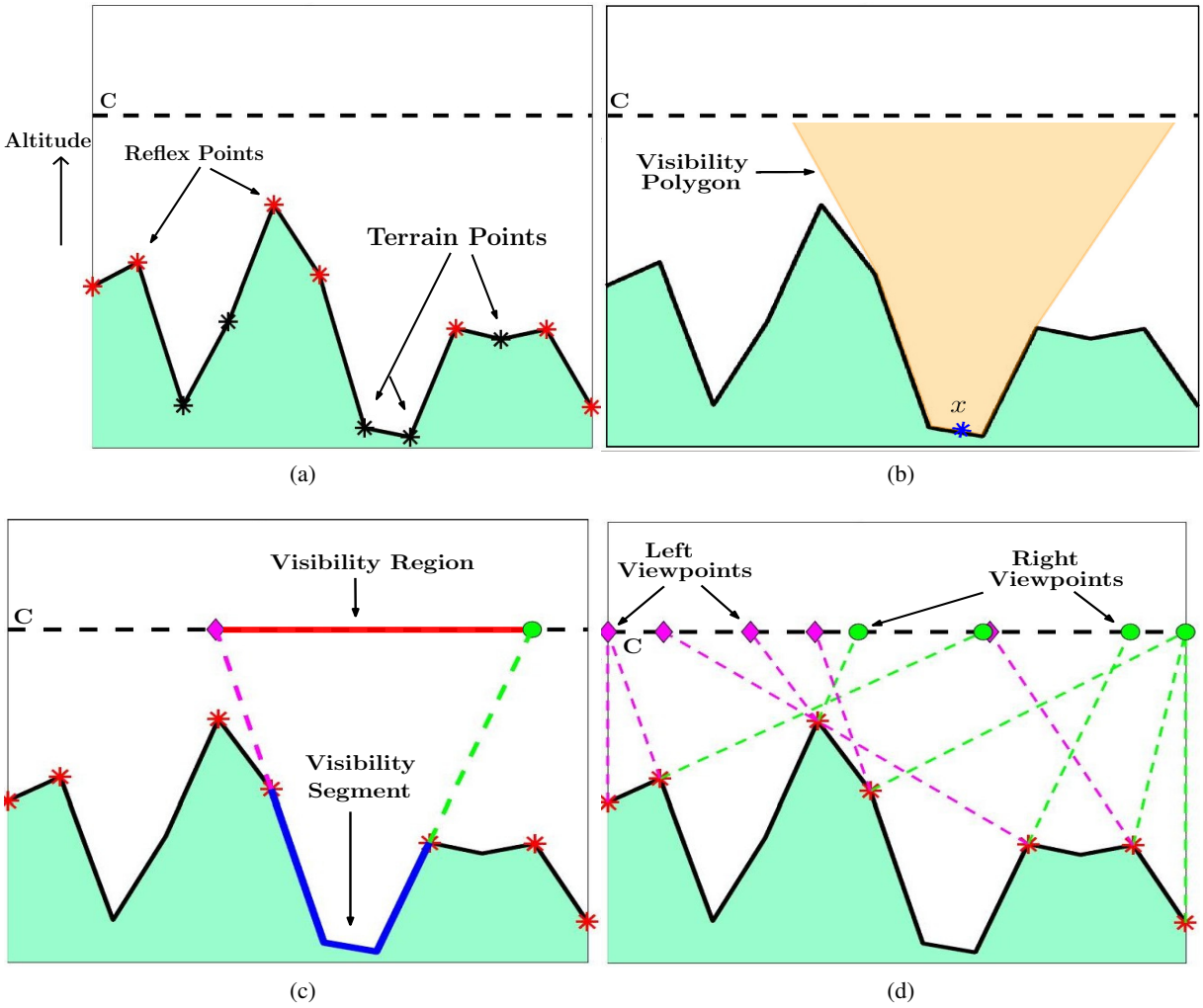



Figure 3.3: (a) Terrain points, reflex points and a fixed altitude path, C, chain visible with the terrain. (b) Visibility polygon corresponding to a point $x$ on the terrain. (c) A visibility segment and its corresponding visibility region. (d) Left and right viewpoints for all visibility segments.

In the case of ground robots, the visibility region for a segment is explicitly defined to be the segment itself. The visibility regions on ground robot paths also satisfy the condition that any point in the visibility region has an unobstructed view of the corresponding segment. This is true since each visibility segment is convex.

**Definition 6** (Viewpoints)**.** *Left and right end-points of a visibility region are referred to as the left and right viewpoints, respectively, of the corresponding visibility segment.*

The intersection of the extended projections from the left and right reflex points of a segment, with the chain visible curve as shown in Figure 3.3c mark the two viewpoints for the segment. Figure 3.3d shows the left and right view points for all visibility segments on the terrain.

### 3.2.2 Robot Model and System Assumptions

This work is concerned with developing high level route planning algorithms and hence does not consider robot kinematics and dynamics. Further, the techniques developed here find application in offline planning and do not consider communication between any of the robots during the monitoring task. The following system assumptions are made for ease of exposition:

- There exists a robust point-to-point navigation and obstacle avoidance system on each robot, including the availability of localization estimates.
- Ideal operating conditions are assumed and the effects of weather, wind or material properties of the terrain is not considered. Further, the terrain material properties are considered to be uniform throughout the operational environment.

These assumptions are not restrictive and allow the modular development of high level planning and coordination algorithms for mobile robots. Point-to-point navigation is usually performed by low level controllers that directly interface with on-board sensors and actuators. The effect of environmental disturbances and variations are independent topics of research and outside but complementary to the scope of this study.

The aerial robot model used in this work is a multi-rotor type vertical take-off and landing (VTOL) model with a given maximum fuel capacity. Aerial robots may have different fuel capacities and operate at different fixed altitudes with a constant individual air speed. Assuming a constant rate of fuel consumption, regardless of the maneuver, and a constant speed for each robot; flight time, distance traveled, fuel consumed and cost are proportional quantities and are used interchangeably. Further, the operating cost of aerial robots is directly proportional to the euclidean distance traversed and flight altitude. The ground robots are assumed to have

infinite fuel availability and do not run out of fuel. The cost function for ground robot operation is proportional to the euclidean distance traversed and slope of the terrain. The fuel limitations are considered only for aerial robots since they usually operate under strong payload restrictions and each refueling operation entails a landing and take-off sequence leading to significant overhead costs.

### 3.2.3 Persistent Monitoring

To conduct persistent monitoring, a planner must compute an infinitely long monitoring route for each robot. Such an infinite sequence may exist in one of the following two forms: (1) a repetitive sequence, and (2) a non-repeating sequence. In this work, this search is restricted to repetitive sequences, wherein a certain finite sequence is repeated infinitely many times to build an infinite sequence. Alamdari et. al. [50] define the repeating sequence as the *kernel* of an infinite sequence. The length of the kernel, i.e. the time taken to complete one execution of the kernel determines the frequency of monitoring the terrain. The maximum time between two consecutive observations of any point on the terrain is upper bounded by the length of the kernel.

In the multi-robot persistent monitoring problem, the kernel would comprise of a route for each robot such that each visibility segment on the terrain is covered by at least one robot. Further, a robot may visit certain viewpoints multiple times within the kernel. Let the maximum number of times a robot visits a viewpoint within the kernel be $\kappa$. In this setup, the monitoring takes place along an $x$-monotone curve, and any point on a robot's path that does not lie on the left or the right extreme would need to be visited twice - once going forward and once returning back. A viewpoint on one of the extremes would be visited only once. Hence, the smallest feasible value of $\kappa$ is two. Except in the case where one of the viewpoints lies directly above a take-off point (starting location or a refueling depot), it is straight forward to show that for any path that visits a viewpoint more than twice, there exists a shorter equivalent path that visits the viewpoint at most two times. For this reason, the value $\kappa = 2$ is used in this setup.

In the scenario, when a viewpoint exists directly above a take-off location, a robot may visit the viewpoint more than two times. While this does not pose any problems in field operations, even a small increase in the value of $\kappa$ significantly increases the size of the search space (additional details on $\kappa$ are discussed in Section 3.4.1). To address this, a preprocessing step is added, that takes linear time to check if a viewpoint lies directly above the starting point or any of the candidate refueling depot opening sites. If such a viewpoint is found, a small noise may be added to the take-off location. Such a small deviation does not affect field operations since take-off platform space requirements are relatively insignificant due to the size of the aerial robots in consideration.

### 3.2.4 Refueling Depot Placement

A solution to the persistent monitoring problem must also decide on the placement of refueling depots in the environment. Both, automated and manually operated depots for refueling or battery replacement in aerial robots incur additional resources and hence their number and placement needs to be optimized. The placement of refueling depots and planning of robot paths are complementary problems. While the refueling depot placement ensures reachability of viewpoints by aerial robots, the selection of viewpoints on a robot's path determines the subset of viewpoints that affect the placement of refueling depots. Too many refueling depots would result in wastage of resources while too few would result in higher cost robot paths or potentially an incomplete mission, in a scenario where only aerial robots are available. Existing works in the literature either assume that refueling depot locations are given [8], decouple route planning and placement of refueling depots/sites [51, 63, 46, 52] or setup refueling depots/sites at a subset of points visited by the aerial robot [23]. This work significantly extends the state-of-the-art by jointly optimizing route planning for robots and the placement of refueling depots. The placement problem is modeled on the lines of the classic facility location problem [65] where refueling depots must be opened at a subset of potential depot opening sites to ensure reachability of viewpoints. A combined mathematical formulation is devised to determine both robot paths and refueling depot placement while also accounting for associated costs in the optimization criteria.

## 3.3 Problem Formulation

Consider an environment $\mathcal{E}$ as shown in Figure 3.2 for a persistent monitoring application. The piece-wise linear feature is represented as a 1.5D terrain. Two types of robots are available: aerial robots with given fuel capacities and flight altitudes (greater than the highest point on the terrain), and ground robots that traverse on the terrain. Each individual flight altitude corresponds to a chain visible curve to the 1.5D terrain. Let $\mathcal{C}$ be a set that comprises constant altitude flight paths and the 1.5D terrain representation that corresponds to the paths for ground vehicles. Let $\mathcal{A}_c$ be the set of autonomous robots, ground or aerial, on the curve $c \in \mathcal{C}$. Then, $\mathcal{A} = \bigcup_{c \in \mathcal{C}} \mathcal{A}_c$, is the set of all robots. Also, let $U_k^c$ be the fuel capacity and $s_c^k$ be the starting location for the robot $k \in \mathcal{A}_c$. The fuel capacity for ground robots is assumed to be infinite. The starting location of an aerial robot is also considered an a refueling depot and is also available to other aerial robots, if the aerial robot is used in the monitoring mission.

A planner must compute paths for robots in the set $\mathcal{A}$ to monitor, in combination, each of the visibility segments on the terrain. There also exist, a set of refueling depot opening sites, $\mathbb{D}$, located on the terrain.

A solution to the persistent monitoring problem must also determine the placement of refueling depots from amongst the sites in $\mathbb{D}$, i.e. it must determine a subset of sites in $\mathbb{D}$ to open refueling depots. Each site $d \in \mathbb{D}$, where a refueling depot is opened, incurs a one time depot opening cost. The aerial robots may use any of the opened refueling depots as computed by the path planner, to refuel during the mission.

Cost function comprises of two components: (a) a maximum robot path cost, and (b) sum of refueling depot opening costs for each opened depot. The first component adds the maximum operational cost (proportional to distance traveled/travel time) of a single robot, to the cost function. This component rewards simultaneous robot operation and prioritizes the length of the kernel of the persistent mission over individual robot operating costs. The second component corresponds to the one time setup costs incurred in opening refueling depots. This component optimizes on the number and placement of refueling depots setup in the environment. In such a scenario, the Heterogeneous Watchmen Persistent Routing Problem on a Terrain (HWPRPT) may be defined as follows.

**Definition 7** (HWPRPT)**.** *Given a 1.5D terrain and a set of ground and aerial robots with optical sensors, fuel limitations, and given operating altitudes; determine a placement of refueling depots in the environment and plan routes for the robots such that all points in the terrain are observed by at least one robot and the aerial robots never run out of fuel. The cost function, as given in Eq. 3.1, minimizes the sum of maximum robot path cost and refueling depot opening costs .*

$$\text{minimize} \left\{ \max_{i \in \mathcal{A}} cost(\Pi_i) + \sum_{d \in \mathbb{D}} \beta(w_d) \right\}, \tag{3.1}$$

where, $cost(\Pi_i)$ is the cost of the path $\Pi_i$ for robot *i* and $\beta(w_d)$ is the cost of opening a refueling depot at site $d \in \mathbb{D}$.

Note to practitioners: By design, this problem does not minimize the total operating cost of the robots, and hence the solutions may include overlapping paths for the robots operating at the same altitude. This is a direct repercussion of the the min-max component of the objective function. In the context of this problem, if needed, the robot path overlaps can be removed in a simple post-processing step by considering the paths pairwise and snipping at the point of first intersection of the paths when going in the left to right direction. The paths of both robots on the same side of the snipping point may be combined and assigned to one robot and paths of both robots on the other side of the snipping point be combined and assigned to the other robot. This process does not increase the value of the objective function as defined by Eq 3.1. This post-processing is not included in this work.

## 3.4 Solution Approach

This work develops two solution methods - a) an exact approach that builds on a Mixed Integer Linear Programming formulation for the problem and is implemented within a branch-and-cut framework and b) a computationally efficient construction heuristic, named as $\delta$-search algorithm. The core idea of the heuristic is competitive construction of routes for all robots while minimizing the step increase in objective function value. This section, establishes the notation used in the solution methods and then describes the two solution methods.

Let $\nu$ be the set of visibility segments on the terrain. By virtue of the chain visibility property and the convexity of the segments, each segment is visible from every curve in $\mathcal{C}$ and has at most two distinct endpoints of the visibility region on each curve, referred to as the left and right viewpoints. Let $\mathcal{V}_c^L$ and $\mathcal{V}_c^R$ be the set of left and right viewpoints on the curve $c$, respectively (see Figure 3.3). Let $\mathcal{V}_c = \mathcal{V}_c^L \bigcup \mathcal{V}_c^R$, be the set of all viewpoints on the curve $c \in \mathcal{C}$ and $\mathcal{V} = \bigcup_{c\in\mathcal{C}} \mathcal{V}_c$ be the set of all viewpoints over all curves in the set $\mathcal{C}$. $\mathcal{V}^L = \bigcup_{c\in\mathcal{C}} \mathcal{V}_c^L$ and $\mathcal{V}^R = \bigcup_{c\in\mathcal{C}} \mathcal{V}_c^R$ are defined similarly. $\mathcal{V}_c(v)$, $\mathcal{V}_c^L(v)$, $\mathcal{V}_c^R(v)$, $\mathcal{V}^L(v)$ and $\mathcal{V}^R(v)$ are subsets of viewpoints corresponding to the visibility segment $v$ contained in the respective set. To quantify coverage and path costs, a visibility function and two cost functions, as follows:

**Definition 8** (Visibility function)**.** *$\gamma_c(v) : \nu \rightarrow \wp(\mathcal{V}_c)$ where $\wp(x)$ is the power set of $x$, is defined for each curve $c \in \mathcal{C}$. $\gamma_c(v)$ is the set of viewpoints on curve $c$ that lie between the left and right end points of the visibility region for the segment $v$, thus $\gamma_c(v) \subseteq \mathcal{V}_c$. Also, $\gamma(v) = \bigcup_{c\in\mathcal{C}} \gamma_c(v)$.*

**Definition 9** (Cost function - 1)**.** *$\alpha_{ij}^c : V_c \times V_c \rightarrow \mathbb{R}^+$, returns the cost to travel from $i$ to $j$ on the curve $c \in \mathcal{C}$, where $i$ and $j$ belong to the set $V_c = \mathcal{V}_c \bigcup \mathbb{D}$.*

**Definition 10** (Cost function - 2)**.** *$\beta_d : \mathbb{D} \rightarrow \mathbb{R}^+$, defines the cost of opening a refueling depot at the site $d \in \mathbb{D}$.*

### 3.4.1 MILP Formulation

The following set of variables are used in the MILP formulation:

- $x_{ij}^{ckl_1l_2}$, defined for each $c \in \mathcal{C}, k \in \mathcal{A}_c, i, j \in V_c$ and $l_1, l_2 \in \{1 \dots \kappa\}$. $x_{ij}^{ckl_1l_2} = 1$, if the $k^{th}$ vehicle on the $c^{th}$ curve visits the $j^{th}$ vertex along the edge $(i, j)$ on $c$ and 0 otherwise. $l_1$ and $l_2$ refer to the serial number of the current visit by the $k^{th}$ vehicle to $i$ and $j$, respectively.

- $z_{ij}^{ckl_1l_2}$, defined for each $c \in \mathcal{C}, k \in \mathcal{A}_c, i, j \in V_c$ and $l_1, l_2 \in \{1 \dots \kappa\}$. $z_{ij}^{ckl_1l_2}$ are real valued decision variables that represents the amount of fuel consumed by the $k^{th}$ vehicle on the $c^{th}$ curve to visit the $j^{th}$

vertex along the edge $(i, j)$ on $c$, from the most recently visited depot. $l_1$, $l_2$ refer to the serial number of the current visit on the two vertices.

- $y_i^{ck}$ are binary variables defined for each $c \in \mathcal{C}, k \in \mathcal{A}_c, i \in V_c$. They take value 1 if the $k^{th}$ vehicle on the $c^{th}$ curve visits the $i^{th}$ vertex, and 0 otherwise.

- $w_d$ are binary decision variables defined for each $d \in \mathbb{D}$. $w_d$ mark the opened refueling depots at sites in $\mathbb{D}$.

The serial numbers refer to the time ordering of the robot visits to viewpoints. For each $c \in \mathcal{C}$ and $\mathcal{P} \subseteq V_c$, let $\delta_c^+(\mathcal{P}) \equiv \{(i, j) : i \in \mathcal{P}, j \in V_c \setminus \mathcal{P}\}$, be the set of all outgoing edges from set $\mathcal{P}$ to $V_c \setminus \mathcal{P}$. Expanded form of the objective function described in Eq. 3.1 to be used in the MILP formulation, is given next.

$$\text{minimize} \left\{ \left( \max_{c \in \mathcal{C},\, k \in \mathcal{A}_c} \sum_{l_1 \in \{1 \ldots \kappa\}} \sum_{l_2 \in \{1 \ldots \kappa\}} \sum_{i \in V_c} \sum_{j \in V_c} \alpha_{ij}^c \cdot x_{ij}^{ckl_1l_2} \right) + \sum_{d \in \mathbb{D}} \beta_d w_d \right\} \tag{3.2}$$

The constraints used in the MILP are as follows:

*Degree Constraints*:

$$\sum_{l_2 \in \{1 \ldots \kappa\}} \sum_{j \in V_c \setminus i} x_{ji}^{ckl_2l_1} - \sum_{l_2 \in \{1 \ldots \kappa\}} \sum_{j \in V_c \setminus i} x_{ij}^{ckl_1l_2} = 0, \quad \forall c \in \mathcal{C}, k \in \mathcal{A}_c, i \in V_c, l_1 \in \{1 \ldots \kappa\} \tag{3.3}$$

$$\sum_{l_2 \in \{1 \ldots \kappa\}} \sum_{j \in V_c \setminus i} x_{ji}^{ckl_2l_1} \leq 1, \quad \forall c \in \mathcal{C}, k \in \mathcal{A}_c, l_1 \in \{1 \ldots \kappa\}, i \in \mathcal{V}_c \tag{3.4}$$

$$\sum_{l_3 \in \{1 \ldots \kappa\}} \sum_{j \in V_c \setminus i} x_{ji}^{ckl_3l_1} - \sum_{l_3 \in \{1 \ldots \kappa\}} \sum_{j \in V_c \setminus i} x_{ji}^{ckl_3l_2} \geq 0, \quad \forall c \in \mathcal{C}, k \in \mathcal{A}_c, l_1, l_2 \in \{1 \ldots \kappa\}, l_2 = l_1 + 1, i \in V_c \tag{3.5}$$

*Visit Constraints*:

$$y_i^{ck} - \sum_{l_2 \in \{1 \dots \kappa\}} \sum_{j \in V_c \setminus i} x_{ji}^{ckl_2 1} = 0, \quad \forall c \in \mathcal{C}, k \in \mathcal{A}_c, i \in \mathcal{V}_c \tag{3.6}$$

$$y_d^{ck} - x_{jd}^{ckl_2 1} \geq 0, \quad \forall c \in \mathcal{C}, k \in \mathcal{A}_c, d \in \mathbb{D}, j \in V_c \setminus d, l_2 \in \{1 \dots \kappa\} \tag{3.7}$$

$$y_d^{ck} - \sum_{l_2 \in \{1 \dots \kappa\}} \sum_{j \in V_c \setminus d} x_{jd}^{ckl_2 1} \leq 0, \quad \forall c \in \mathcal{C}, k \in \mathcal{A}_c, d \in \mathbb{D} \tag{3.8}$$

$$w_d - y_d^{ck} \geq 0, \quad \forall c \in \mathcal{C}, k \in \mathcal{A}_c, d \in \mathbb{D} \tag{3.9}$$

$$w_d - \sum_{c \in \mathcal{C}} \sum_{k \in \mathcal{A}_c} y_d^{ck} \leq 0, \quad \forall d \in \mathbb{D} \tag{3.10}$$

*Sub-tour Elimination Constraints*:

$$\sum_{l_1 \in \{1 \dots \kappa\}} \sum_{l_2 \in \{1 \dots \kappa\}} \sum_{(i,j) \in \delta_c^+(\mathcal{P})} x_{ij}^{ckl_1 l_2} \geq 1 + \sum_{i \in \mathcal{P}} (y_i^{ck} - 1), \quad \forall c \in \mathcal{C}, k \in \mathcal{A}_c, \mathcal{P} \subseteq V_c \setminus s_c^k \tag{3.11}$$

*Coverage Constraints*:

$$\sum_{c \in \mathcal{C}} \sum_{k \in \mathcal{A}_c} \sum_{i \in \gamma_c(v)} y_i^{ck} \geq 1, \quad \forall v \in \nu \tag{3.12}$$

*Refueling Constraints*:

$$\sum_{l_2 \in \{1 \dots \kappa\}} \sum_{j \in V_c} z_{ij}^{ckl_1 l_2} - \sum_{l_2 \in \{1 \dots \kappa\}} \sum_{j \in V_c} z_{ji}^{ckl_2 l_1} = \sum_{l_2 \in \{1 \dots \kappa\}} \sum_{j \in V_c} \alpha_{ij}^c \cdot x_{ij}^{ckl_1 l_2}, \forall c \in \mathcal{C} \setminus c_0, k \in \mathcal{A}_c, i \in \mathcal{V}_c, l_1 \in \{1 \dots \kappa\} \tag{3.13}$$

$$z_{di}^{ckl_1 l_2} = \alpha_{di}^c \cdot x_{di}^{ckl_1, l_2}, \quad \forall c \in \mathcal{C} \setminus c_0, k \in \mathcal{A}_c, l_1, l_2 \in \{1 \dots \kappa\}, d \in \mathbb{D}, i \in V_c \tag{3.14}$$

$$z_{ij}^{ckl_1 l_2} \leq (U - \min_{d \in \mathbb{D}} \alpha_{jd}^c) \cdot x_{ij}^{ckl_1 l_2}, \quad \forall c \in C \setminus c_0, k \in \mathcal{A}_c, l_1, l_2 \in \{1 \dots \kappa\}, i, j \in V_c \tag{3.15}$$

*Variable Domain Constraints*:

$$x_{ij}^{ckl_1 l_2} \in \{0, 1\}, \quad \forall c \in \mathcal{C}, k \in \mathcal{A}_c, l_1, l_2 \in \{1 \dots \kappa\}, i, j \in V_c \tag{3.16}$$

**Algorithm 5** Separation Algorithm

**for all** $c \in \mathcal{C}$ **do**
  **for all** $k \in \mathcal{A}_c$ **do**
    Build graph $G$(directed) $\equiv (V = V_c, E)$
    Add edge $(i, j)$ to E, if $\exists_{l_1,l_2 \in \{1 \dots \kappa\}}\ x_{ij}^{ckl_1l_2} = 1$
    Find connected components in $G$
    **for all** connected components $\mathcal{G} \in G$ **do**
      **if** $(|\mathcal{G}| > 1)\&\&(\mathcal{G} \subseteq V_c \setminus s_c^k)$ **then**
        Add violation constraint (Eq. (3.11))
      **end if**
    **end for**
  **end for**
**end for**

$$y_i^{ck} \in \{0, 1\}, \quad \forall c \in \mathcal{C}, k \in \mathcal{A}_c, i \in V_c \tag{3.17}$$

$$w_d \in \{0, 1\}, \quad \forall d \in \mathbb{D} \tag{3.18}$$

$$z_{ij}^{ckl_1,l_2} \in [0, U], \quad \forall c \in \mathcal{C} \setminus c_0, k \in \mathcal{A}_c, l_1, l_2 \in \{1 \dots \kappa\}, i, j \in V_c \tag{3.19}$$

The objective function (3.2) combines the length of the kernel of the persistent mission and the setup costs for opening refueling depots in the environment. This helps to jointly optimise the overall cost of the mission. Constraint (3.3) is a set of degree constraints that ensures that the result is a closed walk. Constraints (3.4) and (3.5) together ensure that no viewpoint is visited more than $\kappa$ number of times. It may be noted that, the number of variables and consequently the size of the search space increases significantly, with a small increase in the value of $\kappa$. Hence, it is advantageous to restrict its value ($\kappa = 2$) as discussed in Section (3.2.3). Constraints (3.6)-(3.8) populate visit variables for viewpoints and depots. Constraint (3.6) populates the '$y$' variable when a robot visits a viewpoint for the first time. Constraints (3.7) and (3.8) together ensure that a depot visit by a robot is tracked without capping the total number of visits. Constraints (3.9) and (3.10) restrict the placement and number of refueling depots opened in the environment. They populate placement variables '$w$' based on aerial robot visits to refueling depots. Constraint (3.11) ensures that there does not exist a disconnected subtour unreachable from the starting point ($s_c^k$). Constraint (3.12) ensures that for each visibility segment $v$ at least one viewpoint in the set $\gamma(v)$ is visited by one of the vehicles, thus ensuring coverage. Constraints (3.13) and (3.14) ensure fuel conservation at the viewpoints and when traveling out of a depot, respectively. Constraint 3.15 makes sure the aerial robot has enough fuel to reach the nearest depot. Constraints (3.16)-(3.19) specify the domain for decision variables used in the formulation.

### Branch-and-Cut Algorithm

The number of sub-tour elimination constraints is exponential in the number of viewpoints. It is not computationally efficient to enumerate them all and add to the solver. To address this issue, a branch-and-cut framework is used that works as follows: initially a relaxed version of the problem is solved that does not include the sub-tour elimination constraints. When the solver obtains an integer feasible solution to this relaxed problem, it checks if the feasible solution violates any of these constraints. If so, the violated constraints are added to the formulation and the solver then continues to solve the problem. This process of adding constraints to the problem sequentially has been observed to be computationally efficient for many variants of TSP [30] and fuel constrained vehicle routing problems [46, 8]. The algorithm to identify violated constraints is called a separation algorithm. Psuedocode for the separation algorithm is presented in Algorithm 5. The procedure computes the connected components in the graph defined by the integer solution for each vehicle. Each component $P$ of cardinality greater than one that satisfies the condition $P \subseteq \mathcal{V}_c \setminus \{s_c^k\}$, where $c$ is the curve on which the $k^{th}$ vehicle traverses and $s_c^k$ is the starting point of the vehicle, violates the corresponding constraint (Constraint 3.11). The violated constraints are then added to the formulation and the solver is allowed to optimize the problem with the new set of constraints.

### 3.4.2 Construction Heuristic

**Algorithm 6** $\delta$-Search

```
initialize robot paths to starting locations
initialize segment coverage vector
while all segments are not covered do
    for all robots do
        compute-Δ-path
    end for
    δ-increment = minimum cost Δ-path
    update robot paths
    update coverage data
end while
return robot paths
```

Owing to the computational hardness of HWPRPT, one cannot hope for an exact method like the MILP based branch-and-cut algorithm to scale well. In this light, this work includes the design of a computationally efficient heuristic that may be used to compute feasible solutions quickly. The construction heuristic, named as $\delta$-search algorithm is based on competitive construction of robot paths using a step increment strategy. Algorithm 6 gives pseudo-code for the $\delta$-search algorithm. It has an iterative construction and incrementally

builds robot paths until all segments are covered. The paths are competitively allowed a $\delta$-increment in every iteration based on the value of a cost function. This cost function accounts for distance traveled (fuel consumed), new visibility segments covered on the $\delta$-increment, and refueling cost in the case of aerial robots.

**Lemma 2.** *The set of viewpoints on a ground robot's path (tour) that begins and ends at a given starting location on the 1.5D terrain can be ordered in non-decreasing sequence based on their x-coordinate values.*

*Proof.* The viewpoints visited by a ground robot lie on the 1.5 D terrain. A 1.5D terrain representation is a $x$-monotone curve. Hence, the set of all viewpoints on a ground robot's path can be ordered in a non-decreasing sequence. □

**Lemma 3.** *The set of viewpoints on an aerial robot's path (tour) that begins and ends at a given starting location on the terrain can be ordered in non-decreasing sequence based on their x-coordinate values.*

*Proof.* The viewpoints on an aerial robot's path lies on the constant-altitude flight path that is parallel to the x-axis and is $x$-monotone. Hence, the set of all viewpoints on an aerial robot's path can be ordered in a non-decreasing sequence. □

The $\delta$-search algorithm (Algorithm 6) works as follows. Each robot path is initialized to the given starting location. Segment coverage vector is initialized and updated to include segments covered by the ground robots at their starting locations. A while loop is used (lines 3-10) to incrementally build robot paths until all visibility segments are covered in combination by the robots. Within each iteration of the while loop, a $\Delta$-path is computed for each robot using the algorithmic routine given in Algorithm 7. The $\Delta$-path with the minimum cost is then marked as the $\delta$-increment. $\Delta$-paths are compared based on the following cost function:

$$cost(\Delta_i) = d_{cost}(i) + r_{cost}(i) + \upsilon_{rd}(i), \tag{3.20}$$

where, $\Delta_i$ refers to the $\Delta$-path for robot *i*, *cost($\Delta_i$)* is the value of cost function for $\Delta_i$, $d_{cost}(i)$ corresponds to length of $\Delta_i$ (fuel consumed), $r_{cost}(i)$ corresponds to the cost of refueling visits on $\Delta_i$ and $\upsilon_{rd}(i)$ is the visibility score for $\Delta_i$. $\upsilon_{rd}(i)$ is inversely proportional to the number of previously uncovered visibility segments covered by $\Delta_i$. If a valid $\Delta$-path for an aerial robot is not found, its $d_{cost}$ value is set to $\infty$. In the case of ground robots, there always exists a valid $\Delta$-path.

The $\delta$-increment is then added to the corresponding robot's path. Segment coverage vector is updated to include new segments covered by the $\delta$-increment. The while loop continues until all segments are covered. This

ensures that $\delta$-search algorithm terminates with a valid set of paths for each robot and ensures each visibility segment is covered.

The $\Delta$-path computation is performed using the algorithmic routine given in Algorithm 7. It follows from the results in Lemmas 2 and 3, that there always exists a left-most viewpoint (smallest x-coordinate) and a right-most viewpoint (largest x-coordinate) on a robot's path at any stage of the algorithm. These viewpoints are referred to as *end-viewpoints*. To compute the $\Delta$-path, end-viewpoints on the robot's path, $\Pi$, are identified. Next, a neighbor (adjacent) viewpoint not currently on the robot's path, is computed for each end-viewpoint. A $\Delta$-path is defined as the path that starts at an end-viewpoint, visits the neighbor viewpoint and then ends at the same end-viewpoint, while also satisfying the fuel constraint of the robot at every point on the path.

**Algorithm 7** compute-$\Delta$-path

1: $\Pi =$ current robot path
2: $E_\Pi$ = end-viewpoints of $\Pi$
3: $N_\Pi = \{n_i$ : neighbor viewpoint of $e_i \in E_\Pi, n_i \notin \Pi\}$
4: **for all** $e_i \in E_\Pi$ **do**
5: $\quad \Delta_i = \{\phi\}$
6: $\quad \upsilon_i = \{\phi\}$
7: $\quad$ *length*($\Delta_i$) = $\infty$
8: $\quad \Delta_i^{ab}$ = path from $e_i$ to $n_i$
9: $\quad \Delta_i^{ba}$ = path from $n_i$ to $e_i$
10: $\quad$ **if** ($\Delta_i^{ab}$ exists) && ($\Delta_i^{ba}$ exists) **then**
11: $\quad\quad \Delta_i = \Delta_i^{ab} + \Delta_i^{ba}$
12: $\quad\quad \upsilon_i$ = new segments covered by $\Delta_i$
13: $\quad\quad$ Compute *length*($\Delta_i$)
14: $\quad$ **end if**
15: **end for**
16: maxID = $\underset{i:e_i \in E_\Pi}{\operatorname{argmax}} \left\{|\upsilon_i| / \textit{length}(\Delta_i)\right\}$
17: **return** $\Delta_{\text{maxID}}$

$\Delta$-path construction is done in two steps: 1) from end-viewpoint to the neighbor viewpoint 2) from neighbor viewpoint back to end-viewpoint. The computation is shown in the for loop (lines 4-15) in Algorithm 7 and works as follows. For ground robots, the $\Delta$-path computation is straight-forward and is computed as point to point traversals. In case of aerial robots, the path computation in step 1 must account for amount of fuel left in the robot when it reaches the end-viewpoint and the fuel-required to reach the neighbor viewpoint. For path computation in step 2, the amount of fuel left when the robot completes the return journey to the end-viewpoint must be sufficient to reach the next refueling depot or return to the starting depot, as the case may be, on the original path. In both cases, if point to point paths satisfy fuel constraints the paths are computed as direct traversals. Otherwise, the minimum cost traversal that includes a refueling visit to the nearest refueling depot is computed. Any $\Delta$-paths and the resultant robot paths, computed in this way satisfy fuel constraints. The search

for $\Delta$-paths is restricted to include at most one refueling visit on each step of the path between an end-viewpoint and its neighbor. This choice was observed to perform well while still being computationally tractable. Utility value for a $\Delta$-path is computed as the number of new visibility segments covered per unit length (line 16 in Algorithm 7). The path that maximizes the utility function is returned to the $\delta$-search algorithm as the robot's $\Delta$-path.

## 3.5 Computational Simulations

A synthetic simulation platform was built to study and analyse the computational efficiency of the developed approaches. The following sections discuss the simulation setup, instance generation, a few sample scenarios and results of computational simulations.

### 3.5.1 Simulation Setup

The MILP formulation requires significant computational resources. Therefore, simulations were performed on a High Performance Cluster running Cent OS 6.5. Two blade servers within the cluster, each with 20 cores and 96 GB RAM, were used. The number of cores assigned to each process was dynamically decided by the HPC scheduler and was not fixed. The MILP formulation was implemented in C++ using IBM ILOG CPLEX library version 12.7. The branch-and-cut framework was implemented using solver callback (lazy callbacks) functionality of the CPLEX library and LEMON graph library. For each instance, the solver was allowed to execute for 3600 seconds or up to 1% relative gap, whatever was reached first. Relative gap is defined as $\frac{x-lb}{lb} * 100$, where $x$ is the cost of the computed solution and '$lb$' is the best lower bound found so far.

### 3.5.2 Instance Generation

The simulations were executed for a total of 2000 randomly generated instances. The values of various simulation parameter are summarized in Table 3.1. Size of the environment, in terms of terrain points was varied from 5 to 20 in steps of 5. For each environment size, 20 random 1.5 D terrains were generated. A total of 6 robotic sensor nodes were made available for each run; 2 ground robots and 4 aerial robots. For each environment, 5 different starting locations were generated randomly for each robot. Some sample instances used in the simulation are shown in Figure 3.4 along with the routes computed by the MILP solver. The starting locations were chosen randomly from amongst the reflex points on the terrain. Refueling depot opening sites were placed at each of the reflex points. The total width of the terrain was fixed at 1000 units. The terrain point

| S.No. | Parameter Name | # values used | Range of Values |
|---|---|---|---|
| 1 | size (# terrain points) | 4 | $\{5, 10, 15, 20\}$ |
| 2 | 1.5D terrain | 20 | random |
| 3 | starting location | 5 | random |
| 4 | fuel capacity | 5 | see Table 3.2 |

Table 3.1: Simulation parameters used for instance generation.

| S.No. | UAV 1 | UAV 2 | UAV 3 | UAV 4 |
|---|---|---|---|---|
| 1 | 2000 | 2000 | 2500 | 2500 |
| 2 | 2000 | 4000 | 2500 | 5000 |
| 3 | 4000 | 4000 | 5000 | 5000 |
| 4 | 4000 | 6000 | 5000 | 7500 |
| 5 | 6000 | 6000 | 7500 | 7500 |

Table 3.2: Fuel capacities ($U_k^c$) for aerial robots (UAV) used in simulation. UAV 1 and 2 were operated at altitude 1500 units. UAV 3 and 4 were operated at altitude 1800 units.

altitudes were generated within the range 0 to 1300 units. The two aerial robots were operated at 1500 units altitude (A1) and 1800 units altitude (A2). The fuel capacity values ($U_k^c$) for the aerial robots were selected from five pre-generated set of values as given in Table 3.2. The costs associated with ground robot operation are given in Table 3.3. Note that in the case of ground robots, the cost is proportional to both, the distance traveled and the slope of the terrain, as mentioned in Section 3.2.2. The ground robots incurs an additional cost while traveling uphill. To include this aspect, we define an uphill travel coefficient, $\bar{\gamma} = 0.35$, and a downhill travel coefficient, $\underline{\gamma} = -0.15$. The cost computation for ground robots is shown in Table 3.3. The same table also mentions the costs associated with aerial robot operations which include takeoff/landing and traveling on a path segment. Aerial robot cost computation uses the following parameters, $A_i, \ \forall i \in \{1 \ldots (|\mathcal{C}| - 1)\}$ and $\bar{A}$ = 2000 units, where $A_i$ is the flight altitude and $\bar{A}$ represents the maximum permissible altitude. The cost for opening a refueling depot is fixed at 1000 units.

| | |
|---|---|
| Ground Robot Path Cost | distance * costFactor<br>where, costFactor (uphill) = 1 + $\frac{\theta}{\pi/2} * \bar{\gamma}$<br>costFactor (downhill) = 1 + $\frac{\theta}{\pi/2} * \underline{\gamma}$ |
| Aerial Robot Path Cost | takeoff/landing cost = 2*distance<br>horizontal travel cost = distance*costFactor<br>where, costFactor = 2 + ($A_i/\bar{A}$) |

Table 3.3: Costs for different vehicles

| Instance Size | # instances solved optimally |
|---|---|
| 5 | 500(0) |
| 10 | 491(9) |
| 15 | 334(166) |
| 20 | 73(420) |

Table 3.4: Number of instances solved optimally by the MILP solver. Numbers in brackets represent the instances for which the solver could find at least one feasible solution but not optimal.

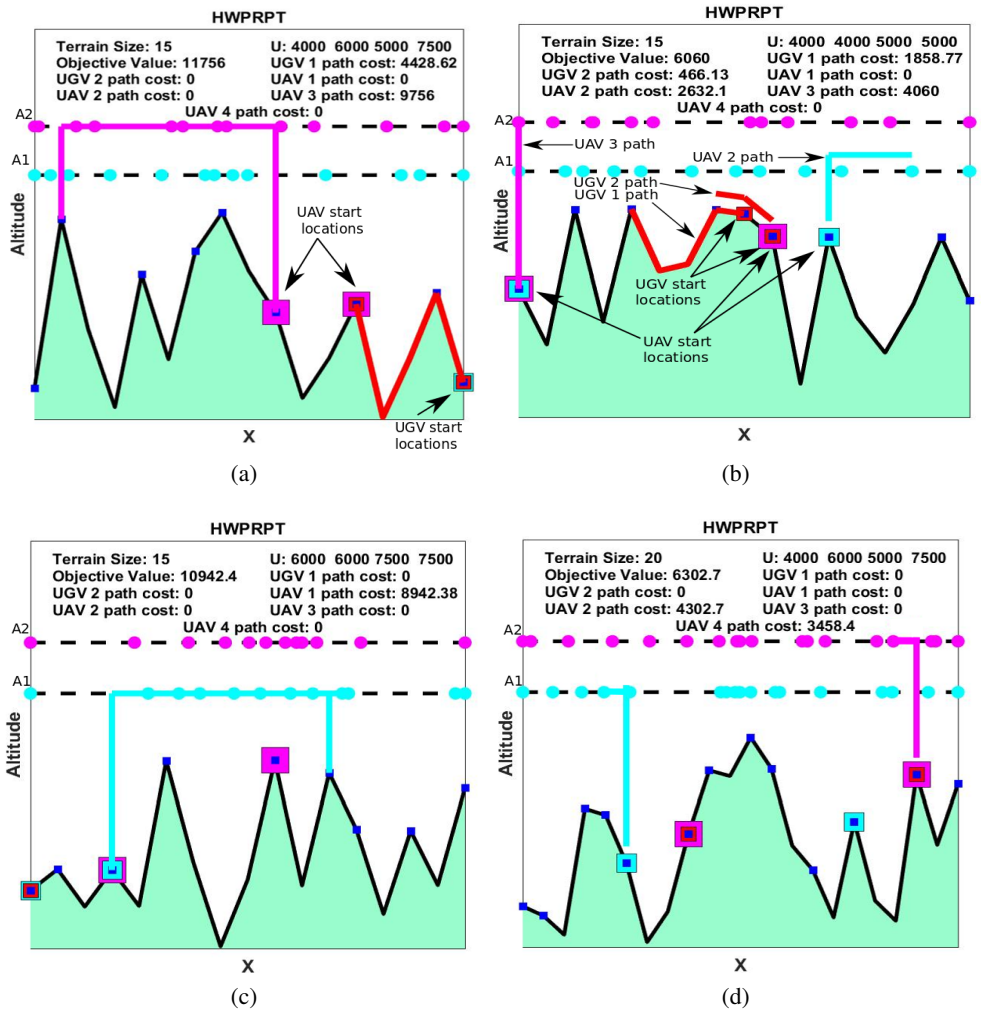
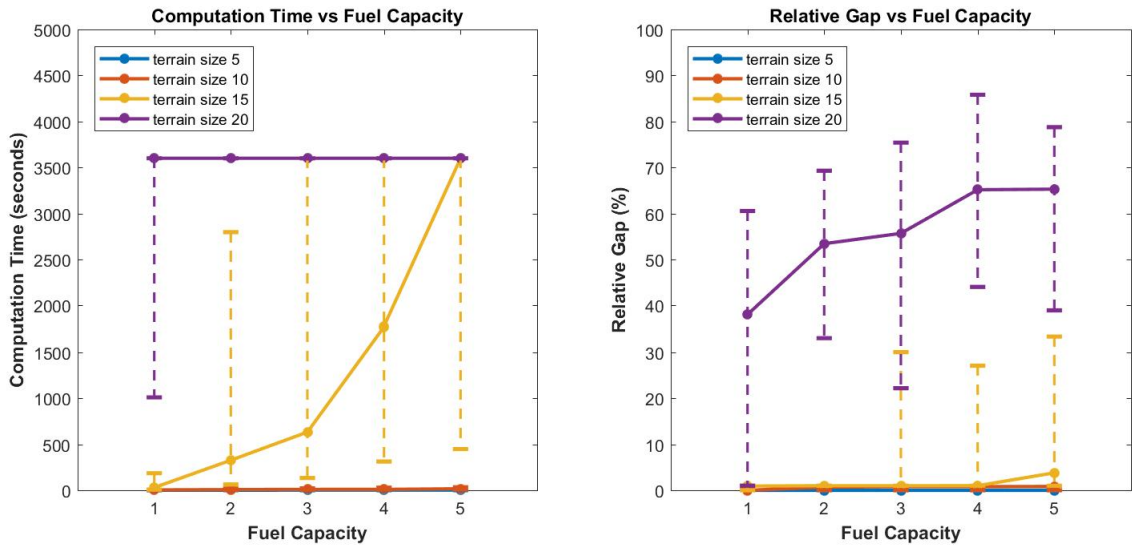
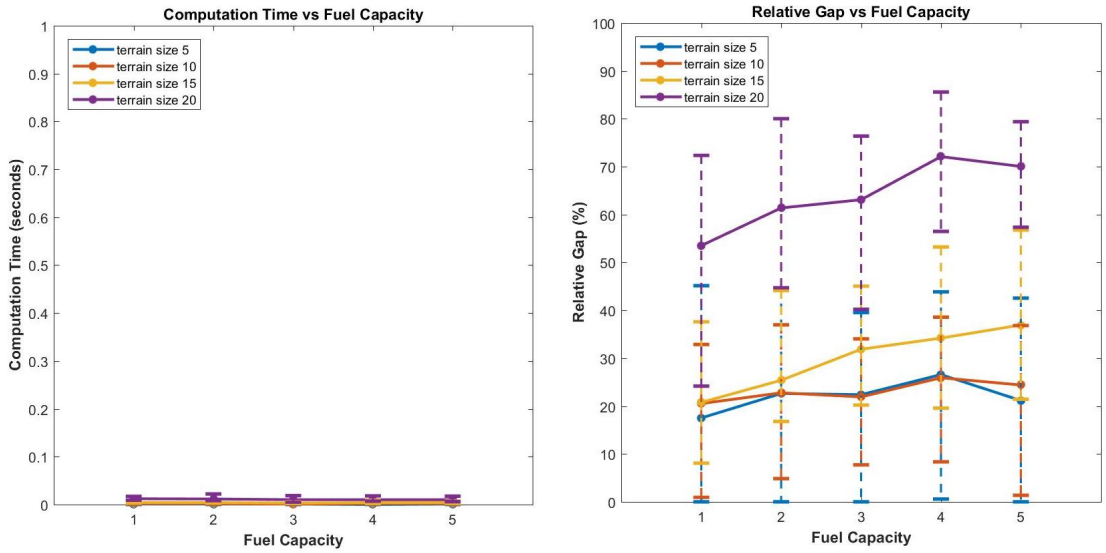



Figure 3.4: Sample instances from the simulation set. The paths are color coded. Red color represents ground robots, cyan color represents robots flying at altitude 1 and magenta color represents aerial robots flying at altitude 2. Blue squares represent refueling depot opening sites. Starting location for the robots are shown in concentric hollow squares on the reflex points in corresponding color. The size of these squares increases with increase in the robot's operating altitude.



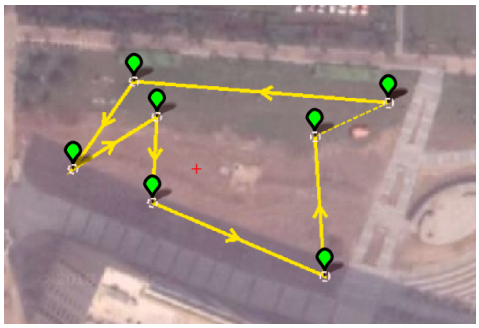

(a) (b)

Figure 3.5: Result plots for branch-and-cut based approach. Median values for (a) computation time and (b) relative gap of the best solution computed by the solver within the stipulated time limit are plotted on Y-axis against fuel capacity serial IDs on the X-axis. The values on x-axis identify the fuel capacities as represented by the corresponding serial numbers in Table 3.2. Whiskers represent first and third quartile values.

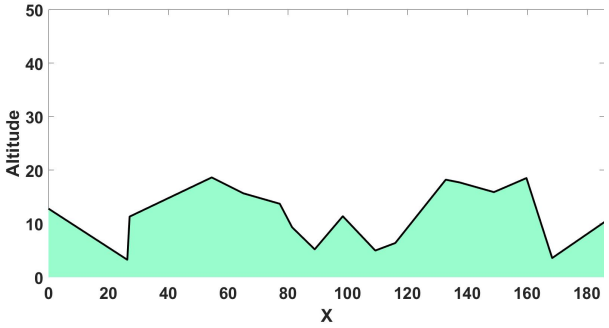



(a) (b)

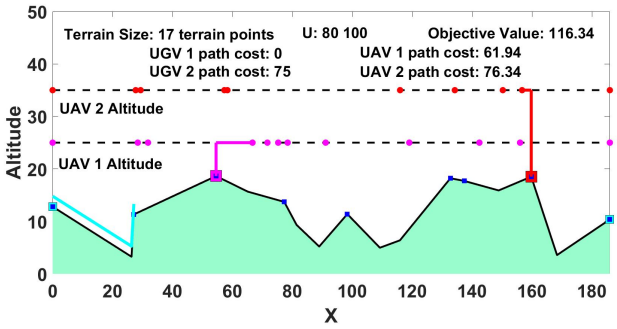
Figure 3.6: Result plots for the heuristic algorithm. Median values for (a) computation time and (b) relative gap using lower bounds, computed by the MILP solver, are plotted on Y-axis against fuel capacity serial IDs on the X-axis. The values on x-axis identify the fuel capacities as represented by the corresponding serial numbers in Table 3.2. Whiskers represent first and third quartile values.



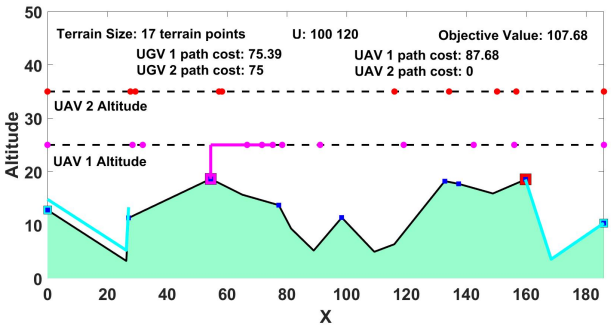
(a)




(b)




(c)




(d)

Figure 3.7: (a) Simulated path-like feature used in the experiments. (b) 1.5 D terrain superimposed on the feature. (c) and (d) Robot paths generated by the MILP solver for experiment 1 and 2, respectively. Ground robot paths are shown in cyan color lines. Aerial robot #1 and #2 paths are shown in magenta and red color lines respectively. The starting locations for the robots are shown as large hollow squares of the corresponding color. Refueling station opening sites are shown in small blue squares.

### 3.5.3 Illustrative Scenarios

This section discusses a few illustrative scenarios and shows how paths for the robotic sensors are generated by our planner to perform monitoring on a terrain. Figure 3.4 shows a few sample instances and the robot paths computed by the MILP solver. The robots available for monitoring are: 2 ground robots (UGV 1 and UGV 2), 2 aerial robots (UAV 1 and UAV 2) at altitude A1 and 2 aerial robots (UAV 3 and UAV 4) at altitude A2. The ground robot paths are shown in red lines. Paths for aerial robots at altitude A1 are shown in cyan color, while magenta colored paths represent UAV 3 and UAV 4 at altitude A2. To distinguish the paths of UGV 2, UAV 2 and UAV 4 from other robots operating at the same altitude namely, UGV 1, UAV 1 and UAV 3, respectively, a small vertical offset is added in the visualization. Refueling depot opening sites are shown in small blue squares. Starting location for the robots are shown in concentric hollow squares on the reflex points in corresponding color. The size of these squares increases with increase in the robot's operating altitude.

Consider the 1.5D terrain shown in Figure 3.4a. The terrain consists of 9 reflex points. The fuel capacities $U_k^c$ for the UAVs are mentioned within the figure following the letter $U$. The robot operating costs were computed as given in Table 3.3. The route determined by the proposed MILP solution is shown in the Figure 3.4a. In this scenario, only UGV 1 and UAV 3 at altitude A2 are used for monitoring. The cost incurred by the individual robots are shown in the same figure. These two robots persistently monitor the terrain by operating on the assigned routes. The solution makes use of 2 refueling depots whose opening costs equal $2 * 1000$ units. The objective value mentioned in the figure reports the total cost of the operation, as given by Eq 3.2. In this instance, it include the opening cost of two refueling depots, i.e. 2000 units, and the maximum operating cost of a robot, i.e. 9756 units, thus making a total of 11756 units.

Consider another scenario as shown in Figure 3.4b. The environment consists of 17 terrain points, and 9 reflex points. In this case, 2 aerial robots (UAV 2 at A1 and UAV 3 at A2 altitudes) and 2 UGVs were used to monitor the terrain. UAV 3 must hover above its take-off location at altitude A2, so that the combined visibility of the agents covers the terrain.

The next example, shown in Figure 3.4c uses only one aerial robot (UAV 1) for monitoring. The scenario consists of 16 terrain points, and 9 reflex points. In the scenario shown in Figure 3.4d, two aerial robots (UAV 2 and UAV 4) are used at different altitudes to monitor. The example has 22 terrain points and 13 reflex points. These example shown how the paths for different vehicles are determined for monitoring. The generated paths depend on the terrain profile, robot operating and travel costs, fuel levels, operating altitudes and the placement of refueling depots. When these parameters are modified the optimal cost and routes for monitoring will also change.

### 3.5.4 Results

This section analyzes the effect of fuel capacity and number of terrain points when using the MILP formulation. In Section 3.4.1, the simulation instance generation is described and after solving these instances for different fuel capacities and terrain points, the results are shown in Figure 3.5. Figure 3.5a shows the computation time taken by the solver as median values along with first and third quartiles. Figure 3.5b shows the relative gap for the best solution computed by the MILP solver within the time limit (3600 seconds). The plots show a distinctive correlation. The values for large environment size (terrain size with 20 terrain points) have high time consumption and relative gap. This is a consequence of exponential increase in the size of the MILP formulation with increase in instance size and hence the solver is not able to compute the optimal solution for most instances.

Results of computational simulations for the construction heuristic are shown in Figure 3.6. Figure 3.6a shows the time taken to compute the solution against fuel capacity for the various instance sizes. It is observed that computation time taken to compute solutions for instances of all sizes and fuel capacity values are less than one second. The heuristic has very low computational time requirement. However, as visible in Figure 3.6b, that shows the median relative gap for solution computed by the heuristic algorithm, the trade-off results in a drop in solution quality. The heuristic is thus useful in cases where computation time is of importance. It also can be used as an initial feasible solution to the MILP solver.

The number of instances solved optimally by the MILP solver are reported in Table 3.4. It also reports the number of instances for which the solver found a feasible solution, but was unable to prove optimality within the given time limit. Adding up the numbers for each instance size, we observe that for instances with 20 terrain points, the solver could not find any solution for 7 instances within the stipulated time. An increase in computation time (for instances solved optimally) and declination in solution quality within the given time limit, is also observed with increase in fuel capacity (Figures 3.5 and 3.6). This is attributed to the increase in search space with increasing values of fuel capacity.

## 3.6 Proof-of-Concept Experiments

The solution methods proposed for coverage of a piece-wise linear feature with load-balancing and placement of refueling depots, were also demonstrated in proof-of-concept experiments. Due to limited resources, the experiments were conducted within the university campus at IIIT-Delhi using 4 robots: two aerial robots and two ground robots. A simulated piece-wise linear feature as shown in Figure 3.7a was superimposed on the

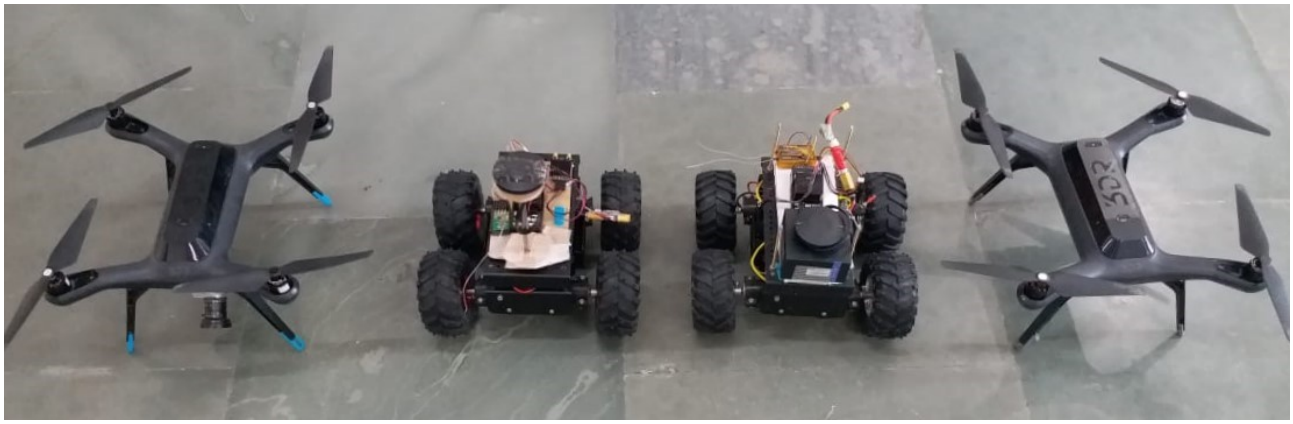
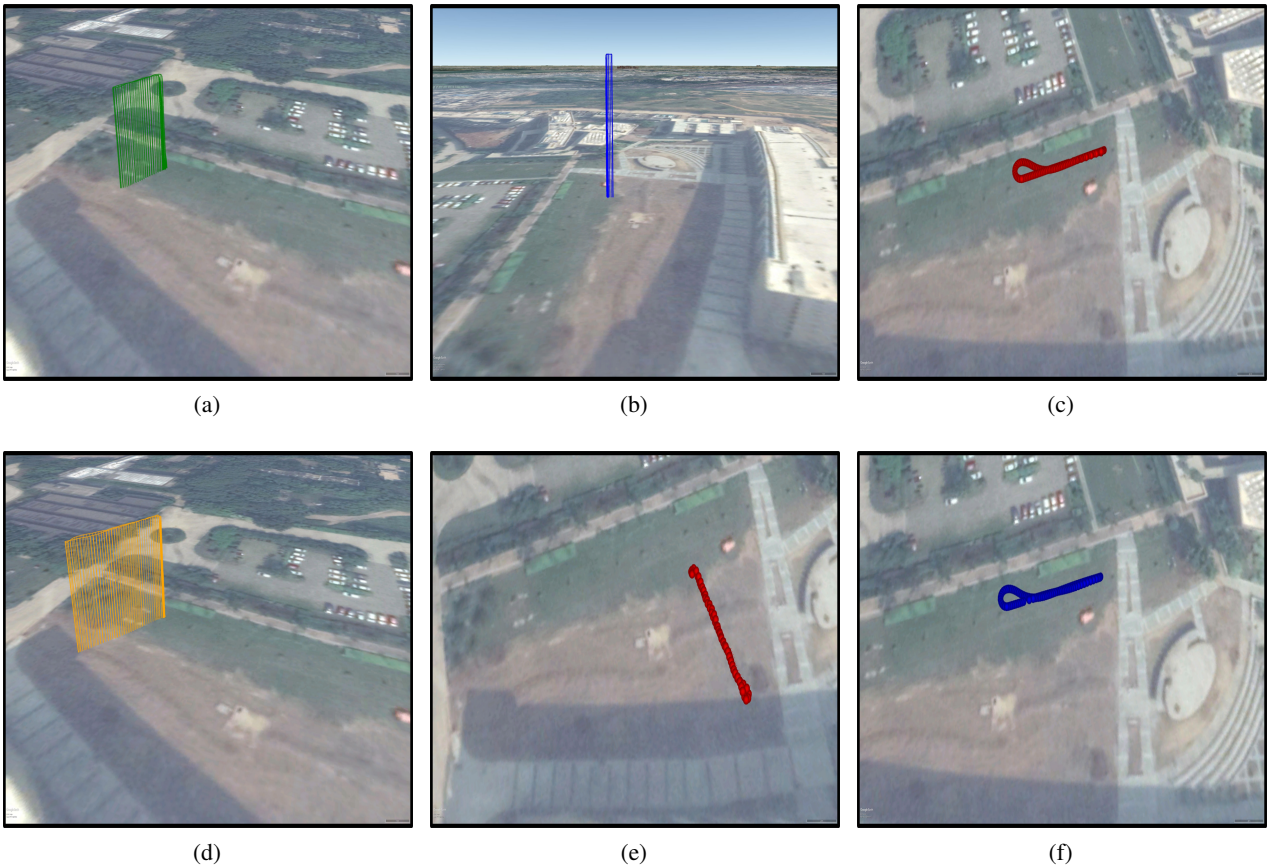


Figure 3.8: Robots used in the experimental validations. 2 3DR Solo quadrotor and 2 custom build ground rovers.



(a) (b) (c)

(d) (e) (f)

Figure 3.9: Paths traversed by individual robots in the experiments. The paths were traced out using telemetry data collected during the robot mission. Visulizations were done in Google Earth Pro software. The paths also show the altitude profile for the aerial robots. Figures 3.9a, 3.9b and 3.9c show the paths traversed by UAV #1, UAV #2 and UGV #2 during experiment 1. Figures 3.9d, 3.9e and 3.9f show the paths traversed by UAV #1, UGV #1 and UGV #2 during experiment 2.

terrain shown in Figure 3.7b to build the experimental setup. The operational region was of size $\sim 80 \times 60$ sq. meters and the length of the simulated feature was $\sim 188$ meters. Paths for the robots were computed using the MILP based solver (Figures 3.7d and 3.7c). The proof-of-concept experiments holds importance, since it shows the feasibility of the kernel computed by the solver for the persistent mission. The optimal placement of refueling depots and the planned routes ensure that the aerial robots never run out of fuel and are able to sustain the persistent mission. The experiments illustrate that even with purely offline planning based on geometry of the terrain and cost models for the robots, the methods developed herein can be used in real implementations.

3DR Solo quad-copters were used as aerial robots in the experiments (Figure 3.8). They were operated using the ArduCopter firmware and employed at 25 and 35 meters altitude respectively. We assign id $\#1$ to the aerial robot operated at altitude 25 meters and id $\#2$ to the one operated at 35 meters. Both the aerial robots were set to a cruise speed of 1 m/s and vertical speed (during take-off and landing) of 1 m/s. Aerial robot operating cost was directly proportional to the distance traveled and the constant of proportionality was based on flight altitude. The ground robots were custom built using Pixhawk cube flight controllers and used the ArduRover firmware. They were also operated at a speed of 1 m/s. The ground robot operating cost was proportional to distance traveled and slope of the terrain. All costs were computed in terms of the fuel consumption.

Two set of experiments were conducted. The simulator visualization of robot paths generated for the experiment are shown in Figures 3.7c and 3.7d. The flight range of aerial robots, $U$, was varied over the two experiments. The set of values used was (80,100) in experiment 1 and (100,120) in experiment 2. The refueling depot opening cost was fixed at 30 units. First experiment utilized both the aerial robots and one ground robot. The exact paths traversed by the robots as traced using the telemetry logs from the robots and visualized in Google Earth are shown in Figures 3.9a, 3.9b and 3.9c. In the second experiment, one aerial robot and both ground robots were used. The robot paths captured using telemetry data and visualized in Google Earth are shown in Figures 3.9d, 3.9e and 3.9f. The experiment video showing complete mission operation for both the experiments may be viewed in [66].

## 3.7 Conclusion

In this work, path planning algorithms are developed for the persistent monitoring problem on piece-wise linear features on terrains. These methods admit the use of a heterogeneous set of aerial and ground robotic sensors. The MILP-formulation based implementation allows the computation of exact solutions for smaller instances of the problem. For larger instances, an extremely fast construction heuristic that can be used to compute feasible solutions, is designed. The solution methods are evaluated and analysed in simulation. Field experiments

were also conducted using real robots, albeit on a virtually simulated terrain. The experiments demonstrate proof-of-concept for persistent monitoring operations in a limited outdoor setting.

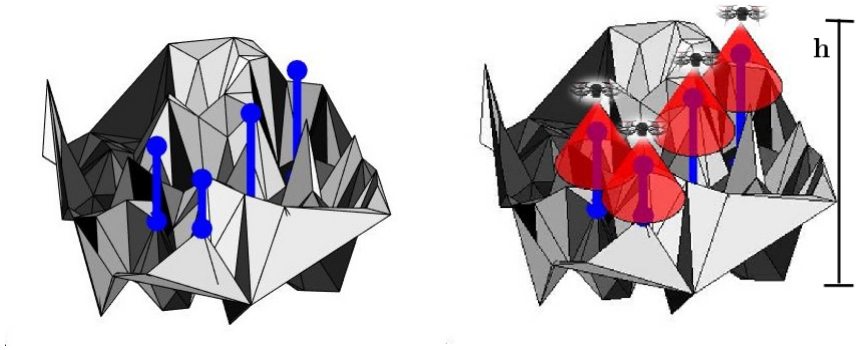
This work opens up avenues for future extension. There is a need to develop approximation algorithms that have bounds on the quality of the solution. It would also be interesting to evaluate the performance of evolutionary heuristics that speed up MILP computations for this problem, since the search space size increases very quickly with large instances. The exact solution to 2.5D version of the problem for coverage on terrains is also an open problem and is very challenging.

# Chapter 4

# Multi-Point Visual Monitoring on a Terrain using an Aerial Robot

Visual surveillance and monitoring is an important application area for Unmanned Aerial Vehicles (UAVs). Crop management [67], area coverage [68, 69], terrain mapping [70], structural inspection [71], and disaster management [72, 73] are some applications where UAVs present a promising solution, essentially acting as "eyes in the sky." When planning paths in such missions, it is important to take visibility obstructions into account. Landscape features such as mountains, gorges, buildings, and bridges limit the line-of-sight of the UAVs. Besides, operative limitations such as camera Field-of-View (FOV) and maximum flight altitude corresponding to the image resolution and/or regulatory requirements also restrict visibility. Such restrictions must be accounted for when planning for monitoring missions. This chapter addresses the visual monitoring problem on 2.5D terrains using a UAV while accounting for camera FOV and terrain imposed visibility restrictions (Figure 4.1). Specifically, the input to the problem is a set of points of interest on the terrain that must be monitored by a UAV with a downward-facing camera. The goal is to find a tour such that every point of interest is seen by the UAV and minimize the tour length. The UAV is restricted to fly at a certain fixed altitude above the ground plane. This fixed altitude flight plane may or may not be above the highest point in the terrain, further complicating the matter.

A naive strategy is to visit a point directly above each point of interest. This is equivalent to solving the Traveling Salesperson Problem (TSP) which is NP-Hard [74]. However, this strategy does not take into account the FOV of the camera. The UAV may be able to view a point of interest from further away and may be able to view multiple points of interest from the same location. Therefore, the problem requires solving for the locations from where to monitor the points of interest and to find a minimum length tour that visits them.




(a) (b)

Figure 4.1: (a) A example 2.5 D terrain showing the set of points of interest marked with blue spikes. (b) These points must be monitored by a UAV with a downwards-facing camera flying at altitude $h$. Camera FOV shown in red conical regions.

## 4.1 Related Work

The visual monitoring problem broadly falls in the category of aerial coverage using UAVs. Applications include target monitoring [75], surveillance [76, 77], and persistent monitoring [78]. There has been relatively limited work on aerial coverage with UAVs that take into account constraints introduced by terrains. Terrain visibility, however, is a classic problem in computational geometry [79] and graphics [80]. In particular, terrain guarding [79] and watchtower problems [81] ask for a stationary camera placement either on the terrain or at some fixed height above the terrain to ensure line-of-sight area coverage of all points on the terrain. Most versions of these camera placement problems, especially when considering a 2.5D terrain, turn out to be NP-Hard. The mobile version of the problem is known as the Watchman Routing Problem (WRP) [54]. In the typical formulation, the environment is a 2D polygon and the robot has an omnidirectional vision with no FOV constraints [56, 60]. In Chapter 3 a variant of the WRP for monitoring points on a 1.5D terrain is discussed. In a 1.5D terrain, the left-to-right ordering can be exploited to yield efficient solutions. In this chapter, the focus is on the 2.5D version of the problem, where no such ordering is available. Efrat et al. [82] studied a restricted version of the visual monitoring problem for a 2.5D terrain with some specific conditions; in particular, they assume that the robots move in straight lines and the camera is a vertical plane that can sweep a terrain.

A closely related work is that of Choi et al. [83] whose goal was to plan a trajectory for the UAV at a fixed distance from the terrain while also taking the camera viewing angle into account. However, their method does not model a finite FOV camera and no theoretical guarantees are presented. Plonski et al. [84, 85] do take the camera FOV into account and provide approximation guarantees. They consider the problem of viewing a set of points using an aerial robot with a conical FOV. However, they assume that all the points to be observed are

on a plane and there are no obstructions to the visibility cones due to the terrain. This work explicitly considers issues that arise due to obstructed visibility by the terrain and limited FOV.

**Contributions.** This work formalizes the visual monitoring problem on terrains for a UAV with limited FOV. The main theoretical contribution is a constant-factor approximation algorithm for solving the planning problem. The constant-factor depends on some terrain-related geometric parameters but is otherwise independent of the size of the input including the number of points of interest. This algorithm does not require any discretization and runs in polynomial time. While this algorithm is satisfactory from a theoretical point-of-view, the constant-factor may be too large in practice. Therefore, a practical algorithm that uses a Generalized Traveling Salesperson Problem (GTSP) solver as a subroutine is also presented. This formulation is general enough that it can be applied to a broader set of scenarios than those considered in this paper. Empirical evaluation of the performance of the algorithm using a new Integer Linear Programming (ILP) formulation with branch-and-cut implementation as a baseline is also presented. Proof-of-concept field deployments were carried out using a UAV to monitor points of interest in a campus environment and are discussed in Section 4.6.

## 4.2 Problem Formulation and Preliminaries

This section describes the notation and formally defines the problem studied. Background information on how visibility polyhedron can be computed for points on a terrain is also presented.

### 4.2.1 Problem Formulation

Consider an environment $\mathcal{E}$ and a set of points of interest, $\mathcal{P} = \{p_1, \ldots, p_m\}$, as shown in Figure 4.1a. $\mathcal{E}$ is modeled as a 2.5D polyhedral terrain ([86], pg 352). The terrain is represented as a triangular irregular network (TIN). The points of interest are located on the surface of $\mathcal{E}$. Without loss of generality, assume that the lowest point on the terrain is at $z = 0$ and the highest point of interest is at $z = l$. The UAV is equipped with a fixed downward-facing camera having a constant focal length and a circular FOV. The FOV may then be represented as a fixed view angle, $\delta$, in each direction. The camera casts a conical FOV on the terrain, as shown in Figure 4.1b.

**Problem 1** (UAV Routing Problem on Terrains (URPT))**.** *Given an environment with a polyhedral terrain, a set of points $\mathcal{P}$ located on the terrain, a UAV equipped with a fixed downward-facing camera with a view angle of $\delta$ operating at a fixed altitude $h$ above ground level, find a minimum length tour for the UAV to visually*

*monitor all points in the set $\mathcal{P}$.*

Two strategies are presented for solving this problem: (1) A polynomial-time constant-factor approximation algorithm that does not require any discretization. (2) A practical algorithm that reduces the problem to GTSP through careful discretization. The latter uses an efficient GTSP solver called as GLNS [87] as a subroutine. This approach is benchmarked using branch-and-cut ILP solutions. In the theoretical analysis, the UAV is assumed to fly at a fixed altitude of $z = h$ that must be above the highest point of interest to be monitored (i.e., $h > l$). However, this plane need not be above the highest point on the terrain. As such, parts of the terrain may act as obstacles along the flight path for the UAV. The analysis takes such obstacles into account.

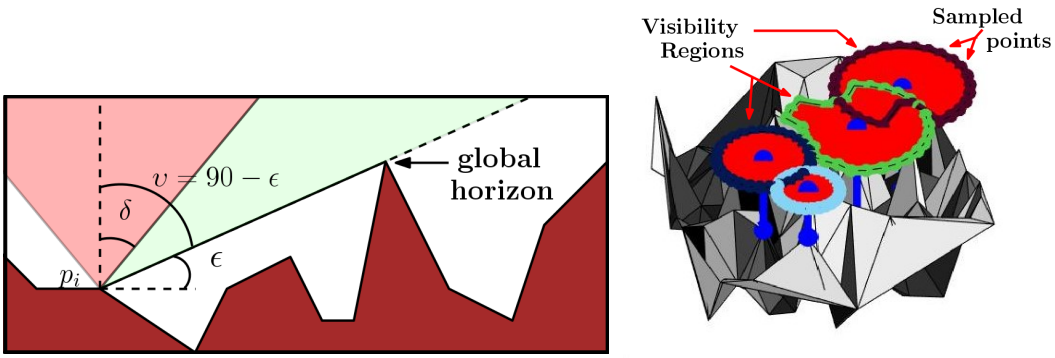
While the theoretical results assume that the UAV flies at a fixed height, the practical GTSP-based strategies can be extended to the case of varying flight altitude, e.g., if the points of interest need to be viewed from a fixed distance. The extension to constant resolution (varying altitude) flight operations is discussed briefly later in the text (Section 4.4). The theoretical analysis is also based on a parameter $\gamma > 0$ which is the minimum angle made by any line joining two points on the terrain with respect to the $Z$ axis.

The strategy for solving URPT works in two phases. In the first phase the 2D visibility region, $V_i$, for each point of interest, $p_i$, in the fixed altitude flight plane is computed. Visibility region of a terrain point is a closed connected space in this plane. Due to occlusions with the terrain, these visibility regions may have complex geometric shapes (Figure 4.2b). If the robot visits any point within $V_i$, it can visually monitor $p_i$. Once the visibility regions for each point of interest are computed, the problem reduces to that of finding a tour for the robot that visits at least one point in each visibility region.

In the next subsection, a method that is employed to compute the visibility region for each point of interest is reviewed. The two planning algorithms are presented in Sections 4.3 and 4.4.

### 4.2.2 Visibility Computation

Consider a point, $p_i \in \mathcal{P}$, that needs to be monitored as shown in Figure 4.2a. To compute the visibility region for $p_i$ on the terrain, a viewpoint is placed at $p_i$ and terrain visibility is computed from the viewpoint. The farthest visible point in a given radial direction, that obstructs all points beyond itself when viewed from a specific viewpoint is called *global horizon point* [80] and is expressed in terms of the elevation angle $\epsilon$. The locus of all global horizon points forms the global horizon. *Visibility angle* $\upsilon$ in a radial direction is computed as the complement of the elevation angle of the global horizon point. The smaller of the visibility angle and the camera view angle defines the boundary of the visibility region in a radial direction.




(a) (b)

Figure 4.2: (a) Global horizon point is the farthest visible point in a radial direction. $\epsilon$ is the elevation of the horizon point, $\upsilon$ is the visibility angle and $\delta$ is the camera view angle. The minimum of $\upsilon$ and $\delta$ determines the boundary of the visibility region in a given radial direction. (b) Visibility regions (red closed shapes) on the constant altitude plane for a set of points on the terrain. Visibility region boundary for each point is marked in a different color for ease of distinguishing. The boundary for a visibility region is computed as the linear interpolation of the projections in $d$ radial directions.

Horizon computation is a well-studied problem in the graphics literature ([80] and references within). This work employs approximate horizon computation method developed by Stewart [88]. However, other methods in the literature may also be used since the solution approach developed in this work is independent of the horizon computation algorithm used.

The selection of Stewart's algorithm was motivated due to its ease of implementation and computational tractability. The algorithm is used to compute the horizon at each point of interest on the terrain, as shown in Figure 4.1a. The radial space is sampled in a discrete number of values, $d$[1] and the minimum of $\upsilon$ and $\delta$ is computed in each direction. Linear interpolation of the extended projections in each direction on the constant altitude plane at height $h$ is then used to compute the boundary of the visibility region as shown in Figure 4.2b.

## 4.3 Polynomial-Time Approximation Algorithm for URPT

In this section, the main theoretical algorithm for solving URPT is presented. The input to the algorithm is the set of visibility regions that are computed using the method described in Section 4.2.2. The problem of finding the shortest tour that visits a set of 2D regions is known as the TSP with neighborhoods (TSPN). The neighborhoods correspond to the visibility regions in this case. TSPN is NP-hard. However, there exist polynomial-time approximation algorithms for many special cases such as when the neighborhoods are all disks of the same radii [89] and non-overlapping convex polygons [90].

[1] Sampling resolution, $d$, is a user-input parameter discussed in Section 4.4

The visibility regions in this case may not necessarily be polygonal (for example, they may contain circular arcs) or convex. In general, they can be overlapping. Furthermore, the fixed flight plane may be at an altitude below the highest point on the terrain. Therefore there may be obstacles within this plane. Nevertheless, it is shown how to approximate the visibility regions by possibly-overlapping disks of the same radius. It is also shown that this approximation still yields a tour whose length is bounded with respect to the optimal.

### 4.3.1 Algorithm Description

In the following, let $V = \{V_i : i \in [1, m]\}$ be the set of input visibility regions corresponding to the points of interest $\mathcal{P}$ that the robot must monitor, computed using the method presented in Section 4.2.2. Each $V_i$ is replaced by an inner disk, say $d_i$, such that it is completely contained within $V_i$, i.e., $d_i \subseteq V_i$. The inner disk $d_i$ lies in the fixed flight altitude plane and is centered at the same coordinates, $\mathbf{x}_i = (x_i, y_i)$, as $p_i$. Let $d^I$ be the set of all inner disks. All inner disks have the same radius of $r_d \triangleq \min\{(h - l)\tan\gamma, (h - l)\tan\delta\}$, where $\gamma = \upsilon$.

Next a tour is computed that visits at least one point in each inner disk, $d_i$. A modified approach from the one presented by Dumitrescu and Mitchell [89] is developed. (1) Find the maximum independent set of non-overlapping inner disks, say $d^I$. (2) Find a $3/2$–approximation to the optimal TSP tour that visits the center of all disks in $d^I$. (3) Follow the tour found in the second step. Every time the tour enters a new disk, take a detour to follow the circumference till the same point is reached again and then move towards the center. Note that this step adds a detour of length at most $2\pi r_d$ to the TSP tour.

In step (2), a TSP tour is computed that visits the centers of all disks in the fixed altitude plane at height $h$. If the fixed altitude plane is above the highest point on the terrain, then this amounts to solving a Euclidean TSP on the plane. This is NP-complete but there exists a $(1 + \epsilon)$–approximation algorithm [74]. However, if the fixed altitude plane is below the highest point on the terrain, then the fixed altitude plane will contain holes (i.e., obstacles that the robot cannot pass through). In such a case, the problem is solved as a metric TSP instance by creating a complete graph where edge costs are the shortest distance between the pair of points (considering the obstacles). Metric TSP has a simple $3/2$–approximation algorithm using the Christofides heuristic [91]. In the analysis, the worse $3/2$–approximation for step (2) is assumed. This is also the reason why a modification is added to the approach by Dumitrescu and Mitchell [89] wherein they assume that there are no holes in the environment. The analysis describes how this algorithm returns a valid tour for URPT that is within a constant-factor of the optimal tour.

### 4.3.2 Theoretical Analysis

The analysis begins with showing that visiting all the inner disks is sufficient to visit all $V_i$, i.e., to monitor each point of interest.

**Lemma 4.** *The inner disk $d_i$ is completely contained in $V_i$.*

*Proof.* Consider any point on the boundary of $V_i$. If it can be shown that the distance between that point and the center of $d_i$, say $x$, is no less than $r_d$, then the proof is complete. There are two possibilities: the line joining that point and $p_i$ does not pass through any other point on the terrain or the line does pass through some other point on the terrain. For example, in Figure 4.3 point $z$ corresponds to the former and $y$ corresponds to the latter. Therefore, $\angle z p_i x = \delta$ and $\angle y p_i x \geq \gamma$. Furthermore, distance between $x$ and $p_i$ is no less than $h - l$. Therefore, the distance between $z$ and the center of the disk is no less than $(h-l)\tan\delta$ and that between $y$ and the center of the disk is no less than $(h-l)\tan\gamma$. Since, $r_d = \min\{(h-l)\tan\gamma, (h-l)\tan\delta\}$, both $y$ and $z$ are no less than $r_d$ away from the center of the disk. □

Let $L_d$ be the length of the tour found by the algorithm. First an upper bound to the length of this tour is presented. Let $L_V^*$ be the length of the optimal tour that visits at least one point in each visibility region, $V_i \in V$, i.e., the optimal solution to URPT. $L_d$ and $L_V^*$ are related as functions of the maximum number of non-overlapping inner disks. Specifically, let $d^I \subseteq D$ be the largest set of inner disks, $d_i$, such that no two disks overlap with each other. This can be found out greedily by constructing the maximum independent set of the disks, as shown in [89]. Let $m_d$ be the number of disks in $d^I$.

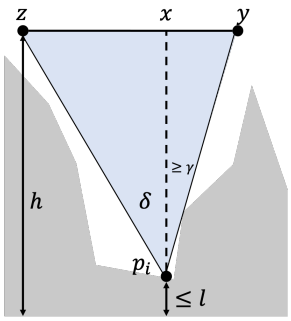



Figure 4.3: A 1D slice of the terrain.

**Lemma 5.** *Let $L_d$ be the length of the tour found using the proposed algorithm. Then $L_d \leq \frac{3}{2}\left(L_V^* + 2m_d h \tan\delta\right) + 2m_d \pi r_d$, where $m_d$ is the maximum number of non-overlapping inner disks, $d^I$.*

*Proof.* The length of the tour, $L_d$, is equal to the distance traveled to visit the centers of the disk in $d^I$ (Step 2) and the detours added every time the center is visited (Step 3). Let $L_{TSP}^*$ be the length of the optimal TSP tour that visits the center of the disks in $d^I$. Although finding $L_{TSP}^*$ is NP-hard, there exists polynomial time

approximation algorithms that find a tour whose length is at most $3/2L^*_{TSP}$ as described earlier. Therefore,

$$L_d \leq \frac{3}{2}L^*_{TSP} + 2m_d\pi r_d \tag{4.1}$$

$$L_d \leq \frac{3}{2}\left(L^*_V + 2m_d r_d\right) + 2m_d\pi r_d. \tag{4.2}$$

The second inequality follows from the fact that a tour can always be constructed such that it visits the center of the disks in $d^I$ by first finding the optimal tour that visits at least one point in each disk in $d^I$ (of length $L^*_d$) and then adding a detour of at most $2r_d$ to visit the center. That is, $L^*_{TSP} \leq L^*_d + 2m_d r_d$ and $L^*_d \leq L^*_V$. □

Next, a lower bound on the length of the optimal tour is developed. Recall that $h$ is the height of the fixed-altitude plane on which the robot is allowed to fly and $\delta$ is the FOV angle. A lower-bounding approximation tour for the optimal solution may be computed as follows. Replace each $V_i$ by an outer disk, say $D_i$, whose radius is equal to $r_D \triangleq h\tan\delta$. Let $D$ denotes the collection of all the disks $D_i$. The disk, $D_i$, lies in the constant altitude flight plane at the height $h$ and is centered at the same $x$ and $y$ coordinates as that of $p_i$.

**Lemma 6.** *The visibility region $V_i$ is completely contained within the disk $D_i$.*

*Proof.* Recall that $V_i$ is obtained by projecting a reverse cone whose apex is at $p_i$ on the fixed altitude plane at height $h$. Let the coordinates of $p_i$ be $(x_i, y_i, z_i)$. Consider a reverse cone drawn centered at $(x_i, y_i, 0)$. Further assume that this cone is not obstructed by any point on the terrain. It is clear that this cone completely contains the cone drawn at $p_i$. The intersection of the larger cone with the fixed altitude plane at height $h$ yields the disk $D_i$. Therefore, $V_i$ is completely contained within $D_i$. (In the extreme case, $V_i$ is the same as $D_i$.) □

**Lemma 7.** *Let $L^*_D$ be the length of the optimal tour that visits at least one point in each disk, $D_i \in D$. Then $L^*_D \leq L^*_V$.*

*Proof.* From Lemma 6, it is known that $V_i \subseteq D_i$. Therefore, any tour that visits at least one point in each $V_i$ is also a tour that visits at least one point in each $D_i$. Thus, the optimal tour (of length $L^*_V$) that visits at least one point in each $V_i$ is also a tour that visits at least one point in each $D_i$. However, $L^*_D$ is the length of the optimal tour that visits at least one point in each $D_i$ giving $L^*_D \leq L^*_V$. □

Finally, lower bound on the length of the optimal tour that visits at least one point in each $V_i$ is computed. The lower bound is computed as a function of the maximum number of non-overlapping outer disks. Let $D^I \subseteq D$ be the largest set such that no two outer disks overlap with each other and $m_D$ be the number of disks in $D^I$.

**Lemma 8.** *Let $L_V^*$ be the length of the optimal tour that visits at least one point in each $V_i$. Then $L_V^* \geq \frac{m_D}{2}\alpha r_D$ where $\alpha = 0.4786$ and when $m_D \geq 3$.*

*Proof.* From Theorem 1 in [92], it is known that any tour of length $L$ that visits at least one point in $m_D$ disjoint disks of radius $r_D$ satisfies: $L \geq \frac{m_D}{2}\alpha r_D$. Therefore, the optimal tour of length $L_{D_I}^*$ that visits all the $m_D$ disks in $D^I$ must satisfy: $L_{D_I}^* \geq \frac{m_D}{2}\alpha r_D$. Since $D^I \subseteq D$, the optimal tour that visits at least one point in each disk in $D$ will have a length: $L_D^* \geq L_{D_I}^* \geq \frac{m_D}{2}\alpha r_D$. From the above equation and Lemma 7, the desired inequality is derived. □

What remains to be shown is the relationship between $m_D$ and $m_d$. It is easy to see that $m_d \geq m_D$, which is the number of non-overlapping outer disks (in $D$) cannot be more than the number of non-overlapping inner disks (in $d$). Also, $m_d$ cannot be arbitrarily larger than $m_D$.

**Lemma 9.** *Let $m_D$ and $m_d$ be the maximum number of non-overlapping outer disks, $D_i$ and inner disks, $d_i$. Then $m_d \leq \left(\frac{2r_D}{r_d}\right)^2 m_D$.*

*Proof.* Consider an outer disk, $D_i$, whose radius is equal to $r_D$. Draw another disk, say $D_i'$, whose radius is equal to $2r_D$ with the same center. Any inner disk that intersects with $D_i$ is completely contained within $D_i'$. The maximum number of inner disks that can be packed within $D_i'$ without any two overlapping can now be bound. One inner disk has an area of $\pi r_d^2$. Therefore, at most $\left(\frac{2r_D}{r_d}\right)^2$ non-overlapping inner disks are contained within $D_i'$. Since there are $m_D$ non-overlapping outer disks, the desired result is found. □

The main result of this section is then stated as:

**Theorem 10.** *Let $L_d$ be the length of the tour found using the proposed algorithm. Let $L_V^*$ be the length of the optimal tour that visits at least one point in each visibility region, $V_i$. Then:*

$$L_d \leq \left(\frac{3}{2} + 50.15c^2 + 105.02c\right) L_V^*, \tag{4.3}$$

*where $c \triangleq \frac{h}{(h-l)} \frac{\tan\delta}{\min\{\tan\gamma, \tan\delta\}}$ and $h$ is the height of the fixed-altitude plane, $l$ is the height of the tallest point of interest, $\delta$ is the FOV angle, and $\gamma > 0$ is the minimum angle between any two points on surface on the terrain with respect to the $Z$ axis.*

*Proof.* It is known from Lemma 5 that,

$$\begin{aligned}
L_d &\leq \frac{3}{2}\left(L_V^* + 2m_d r_d\right) + 2m_d \pi r_d, \\
&\leq \frac{3}{2}\left(L_V^* + 2\left(\frac{2r_D}{r_d}\right)^2 m_D r_d\right) + 2\left(\frac{2r_D}{r_d}\right)^2 m_D \pi r_d, \\
&\leq \frac{3}{2}\left(L_V^* + 2\left(\frac{2r_D}{r_d}\right)^2 \frac{2}{\alpha r_D} L_V^* r_d)\right) + 2\left(\frac{2r_D}{r_d}\right)^2 \frac{2}{\alpha r_D} L_V^* \pi r_d, \\
&\leq \left(\frac{3}{2}\left(1 + 16\frac{h}{\alpha(h-l)}\frac{\tan\delta}{\min\{\tan\gamma, \tan\delta\}}\right) + 16\pi\frac{h}{\alpha(h-l)}\frac{\tan\delta}{\min\{\tan\gamma, \tan\delta\}}\right) L_V^* \\
&\leq O(1) L_V^*.
\end{aligned}$$

□

This shows that the proposed algorithm yields a constant-factor approximation. The constant depends on three parameters, maximum height of a point of interest, the height of the fixed-altitude flight plane, and the maximum slope angle of the terrain, but is otherwise independent of the input (e.g., $|\mathcal{P}|$, the width of the terrain, etc.). It may be remarked that, the same approximation algorithm can also be used in the case of constant resolution imagery (variable flight altitude) missions to compute UAV tours within a constant-factor of the optimal. It is easy to see that the enclosing outer disks ($D_i$) and enclosed inner disks ($d_i$), used to compute the lower and upper bounds on the UAV tour respectively, are still valid and may be used to compute the same constant approximation factor.

## 4.4 A Practical Algorithm for Route Planning

In the previous section, a polynomial-time algorithm for solving URPT is presented. Given fixed parameters for the terrain, it yields a constant-factor approximation which is appealing from a theoretical standpoint. However, from a practical standpoint, the constant factor may be large resulting in longer paths. Therefore, in this section a practical approach for finding the routes is presented.

This approach is based on reducing the route planning problem to the GTSP. The input is a graph, as in TSP, as well as a set of clusters of vertices. The goal is to find the minimum-cost tour that visits at least one vertex in each cluster. GTSP can be thought of as the discrete version of TSPN — the clusters are analogous to neighborhoods. A naive approach to convert URPT to GTSP is to discretize each visibility region, $V_i$. All the discrete sample points within $V_i$ can then be clustered together. Visiting any one of these sample points will

be sufficient to monitor $p_i$. This approach has been used in existing literature [93]. While this is sufficient, it is not necessary to discretize all of $V_i$. Recall that $V_i$ is a 2D region. Instead of discretizing 2D regions, it is sufficient to discretize only the boundaries of all visibility regions as is shown next. Similar ideas have been used by Obermeyer et al. [70] for path planning for a non-holonomic robot through a set of polygonal spaces.

A tour that visits a visibility region, say $V_i$, must cross or touch the boundary of that visibility region as long as there is at least one other visibility region, say $V_j$, that is not completely contained in $V_i$. However, if $V_i$ completely contains all other visibility regions, then considering $V_i$ is redundant. It can be ignored and the problem can be solved by considering only the remaining regions since any tour that visits a point in $V_i$'s interior can cover $V_i$. This preprocessing can be continued until all non-redundant regions are found. For ease of exposition, in the following it is assumed that none of $m$ regions are redundant. Therefore, it is sufficient to discretize and consider points only on the boundary of all of the (non-redundant) visibility regions to compute a tour for the aerial robot. This significantly reduces the size of the GTSP instance created.

### 4.4.1 GTSP-Based Algorithm

The algorithm starts with computing the visibility regions for all points of interest, as discussed in Section 4.2.2. Points on the boundary of each visibility region are sampled uniformly. Let $S_i$ be the set of sample points on the boundary of $V_i$. Some of these points may also lie in the interior (or boundary) of other visibility region(s). In this case, copies of such points are created and added it to the other sets (Figure 4.2b).

Each point corresponds to a unique vertex in the input to the GTSP instance, $\mathcal{V} = \bigcup_{i=1}^{m} S_i$. An edge is added between every pair of vertices. The cost function, $c_{ij} : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}^+$, defines the length of path for the aerial robot to go from $v_i$ to $v_j$, where $v_i, v_j \in \mathcal{V}$. Note that when $v_i$ and $v_j$ are copies of the same sample point, then the cost to travel between them is zero. In addition to defining the set of vertices and edges, the input to GTSP also requires defining the clusters. $m$ clusters, one corresponding to each point of interest, are created. The vertices corresponding to the sample points from the same $S_i$ belong to the same cluster. Note that because of the copies created due to overlapping visibility regions, the clusters will be non-overlapping.

In this form, the problem reduces to an instance of the GTSP. The solution to GTSP is a tour that visits at least one point in each cluster. Therefore, this tour corresponds to one from which each point of interest is visible. Although GTSP is NP-Hard, there are specialized solvers such as GLNS [87], which can find efficiently the optimal solution for large instances. In the next section, empirical results are presented in support of this approach. An alternative strategy is to solve the GTSP instance directly using an Integer Linear Programming (ILP) formulation with branch-and-cut, discussed in the next subsection.

### 4.4.2 Branch-and-Cut ILP Formulation

To formulate the problem as an Integer Linear Program, binary decision variables, $y_{ij}$, are defined for each pair of vertices $v_i$ and $v_j$ in the set $\mathcal{V}$. Here, $y_{ij} = 1$ if the aerial robot visits $v_i$ and $v_j$ vertices in order. Let $\delta^+(X)$ denote the set of pairs $(i, j)$ such that $v_i \in X$ and $v_j \in \mathcal{V} \setminus X$ and $\mathbb{P}(X)$ denote the power set of $X$. The objective function and constraints of the ILP formulation are defined as

*Objective:*

$$\min \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{V}} c_{ij} y_{ij} \tag{4.4}$$

*Degree Constraints:*

$$\sum_{v_j \in \mathcal{V} \setminus v_i} y_{ji} - \sum_{v_j \in \mathcal{V} \setminus v_i} y_{ij} = 0, \quad \forall v_i \in \mathcal{V} \tag{4.5}$$

$$\sum_{v_i \in S_k} \sum_{v_j \in \mathcal{V} \setminus S_k} y_{ji} = 1, \quad \forall k \in [1 \dots m] \tag{4.6}$$

*Sub-tour Elimination Constraints:*

$$\sum_{(i,j) \in \delta^+(s)} y_{ij} = 1, \quad \forall s \in \mathbb{P}(\mathcal{S}) \setminus \{\mathcal{S}, \phi\} \tag{4.7}$$

*Variable Domain:*

$$y_{ij} \in \{0, 1\} \quad \forall v_i, v_j \in \mathcal{V} \tag{4.8}$$

Equations (4.5) and (4.6) represent tour constraints and ensure each visibility region is visited. Equation (4.7) represents the set of sub-tour elimination constraints. The number of sub-tour elimination constraints grows exponentially with increase in the number of visibility regions. Therefore, this work develops a branch-and-cut strategy to solve the ILP formulation. A relaxed formulation, minus the sub-tour elimination constraints is given as input to the solver. A separation algorithm (Algorithm 8) computes valid inequalities $\sum_{(i,j) \in \delta^+(\kappa)} y_{ij} = 1$ at runtime and adds them to the formulation to ensure feasibility of the final solution.

**Algorithm 8** Separation Algorithm

Build graph $G$(directed) $\equiv (\mathcal{P}, E)$
Add edge $(i, j)$ to E, if $\exists\, v_k \in S_i, v_l \in S_j$ and $y_{kl} = 1$
Find connected components $\mathcal{G}$ in $G$
**if** $(|\mathcal{G}| > 1)$ **then**
  **for all** connected components $\kappa \in \mathcal{G}$ **do**
    Add valid inequality
  **end for**
**end if**

### 4.4.3 Extensions

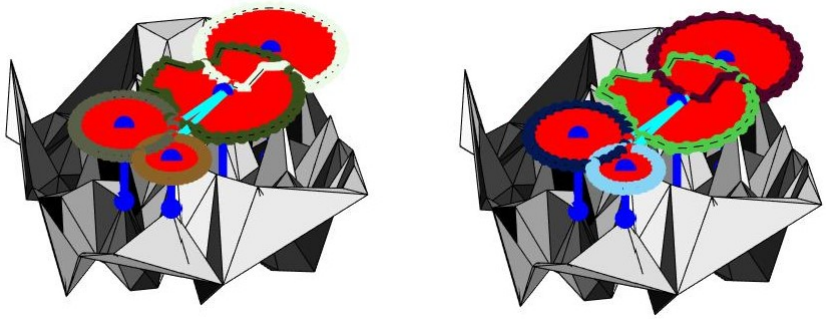
Note that the presented approach is general and can incorporate more scenarios than that described in URPT. It is assumed that the aerial robot must fly at a fixed altitude relative to the ground. In some cases, it may be required for the robot to take images of the points of interest from a fixed height relative to the point of interest. In this case, the sampled points on the boundary of the visibility will be at different heights and the edges connecting them may require the robot to change altitudes. This only affects the way the edge costs are computed. In fact, if there is a higher cost associated with changing altitudes, it can easily be taken into account. If the field of view of the camera is not a cone but instead say rectangular, it can also be incorporated in the methods developed in this thesis as long as the visibility regions can be computed. It is implicitly assumed that the robot is a point robot and the plan only considers the position of the robot. However, there may be more degrees of freedom that need to be taken into account in practice. For example, if the orientation of the camera can be controlled (e.g., with the aid of a gimbal), the output should be a path for the robot along with the camera orientation at each sample point. This can be incorporated by sampling the 3D pose at each sample point in $S_i$, each of which becomes a separate vertex in the GTSP input graph.

## 4.5 Simulation Results

The performance of the two stage strategy is evaluated using IBM ILOG CPLEX library (version 12.7) in C++11 and GLNS solver in Julia [87]. For visualization, a TIN based modeling of the terrain is built using MATLAB R2017a.

### 4.5.1 Instance Generation

To generate the simulation instances an environment of size 200 $\times$200 units is used. A 10$\times$10 grid is placed on the environment and terrain altitude at each grid point is sampled randomly between 0 and 100 units. A



(a) (b)

Figure 4.4: Aerial robot tours generated by (a) GLNS solver and (b) ILP solver. The tour is shown in cyan color. In both the cases, the tour is the same for this particular instance.

Number of instances optimally solved by the ILP solver.

| $m$ | $\delta = 20°$ | | | $\delta = 30°$ | | | $\delta = 40°$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | d = 20 | d = 30 | d = 40 | d = 20 | d = 30 | d = 40 | d = 20 | d = 30 | d = 40 |
| 4 | 20 (20) | 20 (20) | 9 (20) | 20 (20) | 19 (20) | 7 (20) | 20 (20) | 18 (20) | 13 (20) |
| 6 | 14 (20) | 0 (20) | 0 (20) | 10 (20) | 0 (20) | 0 (19) | 7 (20) | 0 (20) | 0 (19) |
| 8 | 0 (20) | 0 (20) | 0 (18) | 0 (19) | 0 (17) | 0 (8) | 0 (18) | 0 (13) | 0 (8) |

Table 4.1: The table shows the number of instances solved optimally by the ILP solver. Numbers in bracket represent the number of instances for which the solver could compute at least a feasible solution within 900 seconds.

TIN representation of the terrain is then generated by a piecewise triangular interpolation between neighboring grid points. The aerial robot flight altitudeiss fixed at 125 units (clear of all terrain features). Size of the set $\mathcal{P}$ of points to be monitored on the terrain is varied from $m = 4$ to 15. The camera FOV view angle $\delta$ is varied from 20° to 40° in steps of 10. The value of sampling resolution parameter $d$, that determines the number of points sampled on the boundary of the terrain, is varied from 20 to 50 in steps of 10. For each combination of parameters 20 different environments were generated.

The GLNS solver was used in *slow* mode setting and allowed a maximum time limit of 600 seconds to directly solve the GTSP instance. The baseline method, ILP formulation, was implemented in a branch-and-cut framework using the lazy callback functionality of IBM ILOG CPLEX library. The solver was allowed to run for a maximum time of 900 seconds for each instance.

### 4.5.2 Results

This section discusses simulation results for GTSP instances generated using visibility regions to compute tours for the aerial robot. Sample paths generated using the two solution methods are shown in Figure 4.4. Two sets

Mean percentage relative gap of ILP generated solutions.

| $m$ | $\delta = 20°$ | | | $\delta = 30°$ | | | $\delta = 40°$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | d = 20 | d = 30 | d = 40 | d = 20 | d = 30 | d = 40 | d = 20 | d = 30 | d = 40 |
| 4 | 0 (0) | 0 (0) | 5.3 (5.9) | 0 (0) | 1 (4.7) | 13.0 (16.9) | 0 (0) | 2.2 (7.7) | 14.2 (24.9) |
| 6 | 4.7 (9.8) | 25.9 (10.7) | 32 (11.4) | 18.2 (23.4) | 45.3 (16.8) | 51.3 (14.5) | 28.9 (28.1) | 58.5 (17.8) | 68.5 (18.7) |
| 8 | 33.7 (14.3) | 43.7 (17.6) | 45.3 (14.8) | 52.3 (18.2) | 65.1 (16.3) | 65.7 (13) | 74.99 (18.4) | 79.6 (14.3) | 87.5 (9.3) |

Table 4.2: Relative gap for best ILP solutions w.r.t. solver generated lower bound within a maximum time limit of 900 seconds. Numbers in bracket are standard deviation.

Mean percentage relative gap of GLNS solver solution w.r.t. ILP generated lower bounds.

| $m$ | $\delta = 20°$ | | | $\delta = 30°$ | | | $\delta = 40°$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | d = 20 | d = 30 | d = 40 | d = 20 | d = 30 | d = 40 | d = 20 | d = 30 | d = 40 |
| 4 | 0 (0) | 0 (0) | 4.8 (5.6) | 0 (0) | 1 (4.5) | 12.4 (16.7) | 0 (0) | 2.1 (7.4) | 14 (24.7) |
| 6 | 4.5 (9.5) | 24.7 (11.2) | 30.1 (12.2) | 17.8 (23.1) | 44.8 (16.8) | 52.3 (18.2) | 28.4 (27.9) | 57.3 (18) | 68 (19.8) |
| 8 | 31.6 (15.2) | 41.7 (17.8) | 48.2 (23.2) | 53.1 (20.8) | 68.2 (20) | 85.1 (20.2) | 76.2 (19.7) | 85 (16.1) | 94 (10.2) |

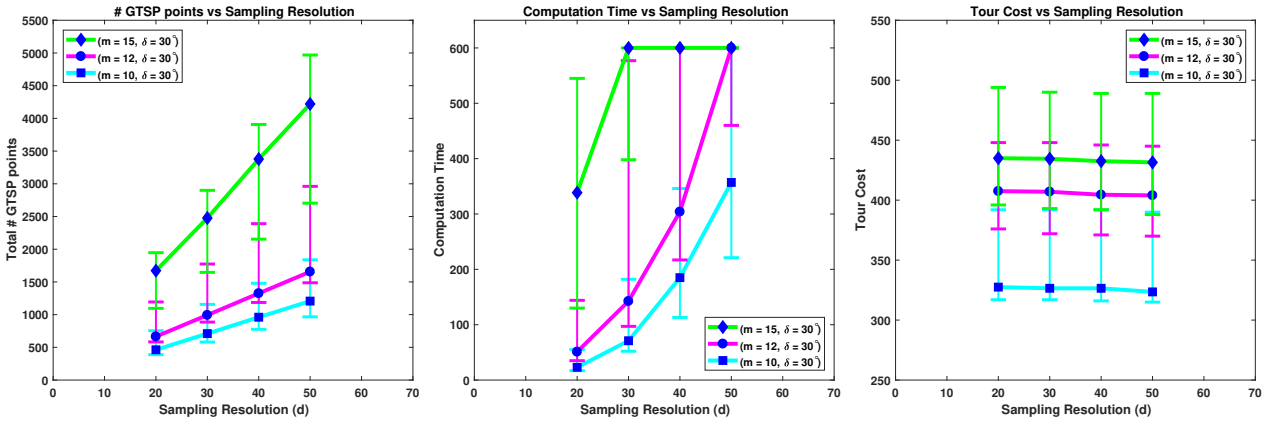
Table 4.3: Mean percentage relative gap of GLNS solutions w.r.t. ILP solver generated lower bounds. The numbers in bracket represent the standard deviation.

of quantitative results are reported. First, the performance of GLNS with respect to the ILP solver. Then, an evaluation of the scalability of the GLNS solver.

The ILP solver used (IBM ILOG CPLEX) returns a lower bound on the solution along with the solution found. When it finds the optimal solution, these two quantities are the same. These lower bounds are used to benchmark the solutions and the relative gap is reported as a quality measure of the solution. Table 4.1 gives the number of instances optimally solved by the ILP solver in 900 seconds. It also shows, in brackets, the number of instances for which the ILP solver was able to find a feasible, but not necessarily optimal solution. In table 4.2, the relative gap for the ILP solutions is shown. It is observed that the ILP solver was not able to find any feasible solution for a sizable number of instances, especially as $d$ and $\delta$ increase. While an increase in the time limit would result in more instances being solved, the trends expected to be the same.

The GLNS method on the other hand, is able to find a feasible solution in all instances. In table 4.3, the relative gap of solutions computed by GLNS from ILP generated lower bounds is presented. These results are only reported for instances for which the ILP solver could compute a lower bound. The ILP solver is not able to solve large instances ($m = 10$, $d = 50$) and hence results for up to $m = 8$ are shown. The relative gap for the ILP solver generated solutions rises quickly with increase in the value of $m$ and $\delta$ (Tables 4.3 and 4.2). Note that the relative gap is computed with respect to the *lower bound* and therefore zero relative gap is sufficient but not necessary condition to declare optimality.

The results for solving larger instances of URPT using GLNS are also shown. By modeling URPT as a GTSP instance, compute practical solutions for large instances can be computed quickly. The problem size for the GTSP instance increases as the size of the URPT instance increases. The size of a problem instance is




(a) (b) (c)

Figure 4.5: (a) Effect of sampling resolution on size of the GTSP instance. (b) Effect of sampling resolution on computation time using the GLNS solver. (c) Effect of sampling resolution on tour cost for tours computed by the GLNS solver. The plots show the median values with whiskers marking the 1st and 3rd quartiles.

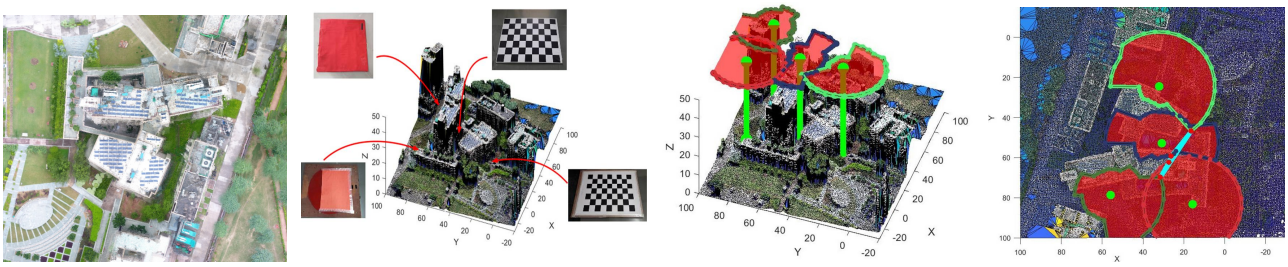
expressed in terms of the total number of vertices in the GTSP instance. Figure 4.5a shows the effect of increase in the value of the resolution parameter, $d$, and the number of points to be monitored in the environment. The size of the problem instances increases to more than 4000 points for $m = 15$, as seen in Figure 4.5b. The corresponding computation time also increases but is still significantly faster than the ILP approach.

It is interesting to note from Figure 4.5c that the quality of the tours, i.e., tour cost, does not improve proportionately with increase in the resolution. There is only a minimal effect of increasing the number of points on the boundary of a visibility region beyond 20. Thus, even for very large number of input points ($m$), one can use a coarser discretization ($\delta$, $d$) to yield a small GTSP instance which can be solved quickly, without sacrificing on solution quality.

## 4.6 Field Experiments

Field experiments were conducted inside the campus of IIIT-Delhi to demonstrate and validate the applicability of the solution methods. The operational area for the experiments is shown in Figure 4.6(a). A DJI Phantom 4 quadrotor was used to perform the experiments. The quadrotor was operated in the downward facing setting and operated at an altitude of 50 m. The DJI Phantom 4 has a field of view of $70^\circ \times 50^\circ$ in the downward facing mode. The field of view for the quadrotor was fixed at $30^\circ$ in these proof-of-concept experiment.

A DEM of the area was created using PIX4D and modeled using a TIN representation. Four target points were placed in the area as shown in Figure 4.6(b). Visibility regions for each target were computed using



(a) (b) (c) (d)

Figure 4.6: (a) Operational area for the field trials. (b) DEM of the topography showing the placement of targets. (c) Visibility regions for the monitoring targets computed using the strategy discussed in Section 4.2.2. (d) Top view of the operational area showing the UAV path in cyan color and visibility regions.


(a) (b) (c) (d)

Figure 4.7: Aerial views of points of interest from the quadrotor during experiments.

the visibility computation strategy given in Section 4.2.2 (Figure 4.6(c)). Value of the sampling resolution parameter $d$ was set to 30. Path for the quadrotor was computed as solution to the corresponding GTSP instance solved using GLNS solver, as shown in Figure 4.6(d). The aerial views of the four target points placed in the environment as observed by the quadrotor are shown in Figure 4.7. Experiment footage showing the UAV flight and camera imagery may be viewed in [94]. This serves to demonstrate that the presented method can be used in a practical setting with real-world operational conditions.

## 4.7 Conclusion

A two phase strategy to computer tours for a UAV to perform multi-point visual monitoring on a terrain is developed. The path planning problem is modeled as an instance of the TSPN problem and a constant-factor polynomial time approximation algorithm is designed for the class of TSPN instances. Further, GTSP based solution methods by discretizing the visiblity region boundaries were evaluated using two solution techniques – ILP formulation implemented in a branch-and-cut framework and GLNS (widely used GTSP solver). Field experiments were also conducted to verify the applicability and effectiveness of solution methods.

# Chapter 5

# Curvature Constrained Trajectory Planning for a Nonholonomic Mobile Robot through a sequence of points: A Perturbation - based Approach

Path Planning for curvature constrained (car-like) mobile robots is an important problem in many domains including environmental, industrial and defense applications. The motion of most wheeled ground mobile robots as also many Unmanned Aerial Vehicles (UAVs) can be effectively explained using a curvature constrained motion model. Path planning for such vehicles with a nonholonomic constraint has gained prominence over the past decade partly because of the rise in popularity of UAVs [95, 96, 97, 98, 99, 100, 101, 4]. A fixed-wing UAV operating in a constant altitude setting follows the curvature constrained motion model. Literature on path planning for UAVs in the presence of threat zones and obstacles, also makes use of the curvature constrained vehicle model [101, 102, 103]. The simple model for a car-like vehicle proposed by Dubins [1] is the most commonly used model. It provides a clean and efficient representation, independent of the dynamics of the vehicle, that allows development of path planning algorithms while considering only kinematic constraints. Bhatia et. al. [104], for instance, use Dubins' model for a UAV swarm rendezvous application. Sujit et. al. [95, 4] develop solutions for forest fire monitoring and terrain mapping applications using Dubins' model for UAVs.

This work addresses the path planning problem for a curvature constrained mobile robot through a given

sequence of $n$ points. For path planning in the euclidean space or for a robot that can turn in-place (such as quadrotors), a sequence immediately informs the path for the robot. However, in the case of nonholonomic motion planning, due to the large search space of orientation angles at each of the $n$ locations in the sequence, path planning is challenging and is known to be NP-hard. This work develops an algorithm, scalable to large sequences, that is able to compute approximate paths for a nonholonomic mobile robot in polynomial time.

## 5.1 Related Work

L.E. Dubins seminal work [1], is still one of the most conclusive results on path planning for curvature constrained vehicles in the literature. Dubins formalized the model for a forward moving car-like vehicle and developed a characterisation for the shortest path between any two states of the vehicle in an obstacle free environment. Dubins' results are discussed in detail in a later section. The logical extension of Dubins results to a sequence of 3 or more points, referred to as Generalized Dubins Path Problem (GDPP) and shown to be NP-hard by Manyam et. al. [105], is still an open problem in the general case. Lee et. al. [106] address GDPP and develop the only constant factor approximation algorithm that exists in the literature with a 5.03-approximation guarantee. In a recent work, Manyam et. al. [105] developed algorithms to compute tight lower bounds for GDPP. They relax the requirement for the departure angle at a location to be equal to the arrival angle $+\pi$. The difference in the two is instead upper bounded by a constant. Goaoc et. al. [107] proved that if the minimum separation between consecutive points is greater than $(4 \cdot R)$, where $R$ is the minimum turning radius of the vehicle, then the shortest path function through a sequence of points is convex and an optimal solution may be found using a gradient descent approach. They reduce the problem to a family of $n$-dimensional convex optimization subproblems, that may grow up to exponential size in the worst case. From a practitioner's perspective, their proofs are highly involved and the cost function (family of functions) is non trivial to implement.

The tour version of the problem for curvature constrained vehicles, referred to as Dubins Traveling Salesman Problem (DTSP), is a generalisation of TSP and is NP-hard. Rathinam et. al. [108] develop a constant factor approximation algorithm for the case when targets are atleast $(2 \cdot R)$ distance away from each other. Ny et. al. [109] design an approximation scheme, where the approximation guarantee is inversely proportional to the minimum distance between any two consecutive points in the sequence. Tang and Ozguner [110] develop gradient-based heuristics and Savla et. al. [111] develop heuristics based on the relaxed euclidean version of the problem. Heuristic algorithms [112, 113, 114] for DTSP have also been developed based on reductions to the one-in-a-set problem by discretizing the heading angle at each point. The solution requires large number of

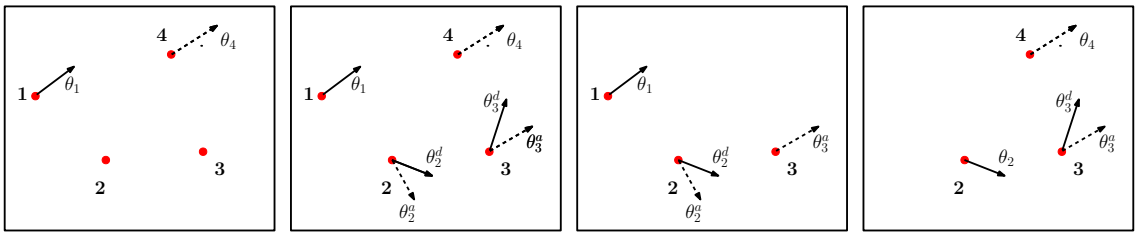



Figure 5.1: An instance of GDPP with 4 points is shown in (a). Points in the sequence are shown in small red circles, a solid arrow represents the departure angle and dashed arrow represents arrival angle at a point. (b) shows the arrival and departure angles computed by the lower bounding algorithm. An n-point sequence can be expressed as multiple overlapping 3-point sequence problems. The subsequences extracted from (b) are shown in (c) and (d). The instance in (c) needs to be solved first to find the heading angle at **2**, given as input to the instance in (d).

discretizations of the angular space and does not scale to large instances. Manyam et. al. [112] develop lower bounding algorithms for DTSP using a relaxation of the angular constraints.

This work develops an algorithm to compute approximate paths for a curvature constrained mobile robot through a given sequence of points. An approximate path implies that the location of the points on the path may be perturbed by a small amount and the path may not pass exactly through the given locations. The choice of exact coordinates of points to be visited by a robot may, in general, depend on application specific priors or considerations for the shortest path. From a practitioner's perspective, in applications like surveillance and mapping, small perturbations to the points to be visited, in favour of a shorter path and a low-complexity solution approach, is an acceptable trade-off in most applications. This thesis develops a practicable solution that is easy to implement and approximates paths developed by known lower bounds for GDPP [105]. The approach is computationally efficient and scalable to sequences of large number of points.

## 5.2 Application Scenario

GDPP extends the Dubins' path problem to a sequence of points. Dubins' algorithm allows the computation of shortest paths for a curvature constrained vehicle between two states in constant time. In the case of GDPP, an $n$ point sequence of more than two points and the orientation angles of the curvature constrained robot at the start and end locations, $\theta_1$ and $\theta_n$, are given as input (see Fig. 5.1a for an illustration). A solution would comprise of an $n$-tuple of $\theta_i$ (angle) values, $< \theta_1, \theta_2 \ldots \theta_n >$, where each $\theta_i$ for $i \in \{1, \ldots, n\}$ corresponds to the orientation of the robot at the $i^{th}$ point in the sequence. Given $n$ points and an $n$-tuple of the $\theta$ values, Dubins' algorithm may be used to compute the shortest path between consecutive points with linear time complexity. The challenge is to compute the optimal set of $\theta$ values such that the total length of the path is a minimum.

The solution approach is based on the recent work to compute lower bounds for GDPP by Manyam et. al. [105]. The term 'lower bounding algorithm' is henceforth used to refer to Manyam's solution, unless indicated otherwise. Their algorithm takes $\epsilon$ as a user input parameter and computes two $\theta$ values at every point in the sequence (except the first and last points) corresponding to arrival and departure angles of the robot (Fig. 5.1b). In a feasible solution to GDPP the departure angle must be exactly equal to arrival angle + $\pi$. The lower bounding algorithm permits the arrival and departure angles at any point in the sequence to be different but within the maximum angular difference, $\epsilon$ (Fig. 5.3). Any feasible solution to GDPP satisfies the relaxed constraint, thus making it a valid lower bound. In this work this maximum separation constraint between arrival and departure angles is exploited to compute approximate (feasible) paths using the lower bounding solutions. While feasibility is ensured by computing solution where the arrival angle is equal to the departure angle, the paths are allowed to deviate from the exact coordinate of the point in the sequence. The perturbation of locations in the interest of shorter path lengths leads to an efficient algorithm to solve GDPP.

## 5.3 Problem Formulation and Preliminaries

This work models the motion of a curvature constrained mobile robot using Dubins vehicle model [1] that describes a forward moving vehicle with a minimum turning radius, $R$, traveling at a constant speed. The state of a Dubins vehicle is defined using a triplet $< x, y, \theta >$, where $< x, y >$ represent the location in a 2D plane and $\theta$ gives the orientation of the vehicle. To solve the Generalized Dubins Path Problem, this work shows the computation for a 3-point sequence and extends the approach to an $n$-point problem.

### 5.3.1 Dubins Curves

Given initial and final vehicle states, $\mathbf{A} \equiv< \mathbf{x_1}, \mathbf{y_1}, \theta_\mathbf{1} >$ and $\mathbf{B} \equiv< \mathbf{x_2}, \mathbf{y_2}, \theta_\mathbf{2} >$, Dubins postulated that the shortest path from **A** to **B** in an obstacle free environment can be expressed as a 3-segment path of one of the following six configurations or their reduced forms: {RSR, LSL, RSL, LSR, RLR, LRL} (Fig. 5.2). R (Right) and L (Left) represent arcs on a circle of radius $R$ along clockwise and counter-clockwise direction, respectively and S represents a straight line segment. The terms path and curve interchangeably through the rest of the chapter. The six Dubins' curves can be represented using 3 generic curve types: $C^+SC^+$, $C^+SC^-$ and $C^+C^-C^+$, where $C^+$ and $C^-$ represent opposite directions of rotation along the circular arc. Curves belonging to the same generic type have similar properties and are symmetrical in nature. For example, in case of $C^+SC^- \equiv \{RSL, LSR\}$, all properties that hold true for RSL are also symmetrically true for LSR.

There exist combinations of $\mathbf{A}$ and $\mathbf{B}$, for which one or two segments of the shortest Dubins path may vanish. These are called reduced/degenerate curves/paths. The value of 3-tuple parameters corresponding to these curves are called degenerate points. These points hold special significance and characterize specific properties of the path length function. In the context of the solution method developed in this work, the $x$ and $y$ coordinates of both $\mathbf{A}$ and $\mathbf{B}$ and the orientation angle at $\mathbf{A}$ are fixed. This is discussed in more detail in Section 5.3.2. Hence, the only variable is the arrival angle or the orientation at $\mathbf{B}$. Hereafter, degenerate points are used to refer to the value of the arrival angle $(\theta_2)$ that corresponds to the occurrence of degenerate curves. The degenerate curves and corresponding degenerate points can be computed in constant time by exploiting the available geometric priors.

### 5.3.2 Problem Formulation

An $n$-point sequence can be expressed as a set of $(n-2)$ 3-point sequences, where consecutive sequences share two consecutive points (Fig 5.1). Each such subsequence forms an instance of the 3-point GDPP (Fig. 5.1c and 5.1d). Consider a 3-point sequence to be traversed by a Dubins vehicle. Let the three corresponding states be $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$, with tuples of the form $< x_1, y_1, \theta_1 >$, $< x_2, y_2, \theta_2 >$ and $< x_3, y_3, \theta_3 >$. $\theta_1$ and $\theta_3$ are departure and arrival angles at $\mathbf{A}$ and $\mathbf{C}$, respectively and are given as input. The location of all 3 points is also given. The only unknown is $\theta_2$. Given $\theta_2$, Dubins' algorithm may be used to compute the shortest path from $\mathbf{A}$ to $\mathbf{B}$ and $\mathbf{B}$ to $\mathbf{C}$, thus giving the shortest path through the sequence. In the rest of the chapter, unless otherwise mentioned the variation in path length function is always discussed as a function of $\theta_2$, considering the rest of the variables constant.

The output from the lower bounding algorithm (Fig 5.1b), generates for each 3-point subsequence, two $\theta_2$ values at $\mathbf{B}$, $\theta_2^a$ and $\theta_2^d$. $\theta_2^a$ is the arrival angle at $\mathbf{B}$ along the shortest Dubins' path from $\mathbf{A}$ and $\theta_2^d$ is the departure angle from $\mathbf{B}$ along the shortest Dubins path to $\mathbf{C}$. Also, $\|\theta_2^a$ - $\theta_2^b\| < \epsilon$. To compute a solution to the GDPP instance, from $\mathbf{A}$ to $\mathbf{C}$, find a value $\theta_2'$ ($\mathbf{B'}$ = $< x_2, y_2, \theta_2' >$), such that $L(\mathbf{A}, \mathbf{B'})$ + $L(\mathbf{B'}, \mathbf{C})$ is a minimum, where $L(\mathbf{A}, \mathbf{B})$ is the length of the shortest Dubins path from $\mathbf{A}$ to $\mathbf{B}$.

## 5.4 Strategy

In the case of a $(4 \cdot R)$ minimum separation assumption between consecutive points, the optimal orientation at B, $\theta_2$, may be computed as $\underset{\theta \in \{\theta_2^a, \theta_2^b\}}{\arg\min} (L(\mathbf{A}, \mathbf{B}(\theta))$ +$L(\mathbf{B}(\theta), \mathbf{C}))$, where $\mathbf{B}(\theta)$ signifies the vehicle state at $\mathbf{B}$ with orientation $\theta$. This is because the path length function is convex under this assumption and continuous in

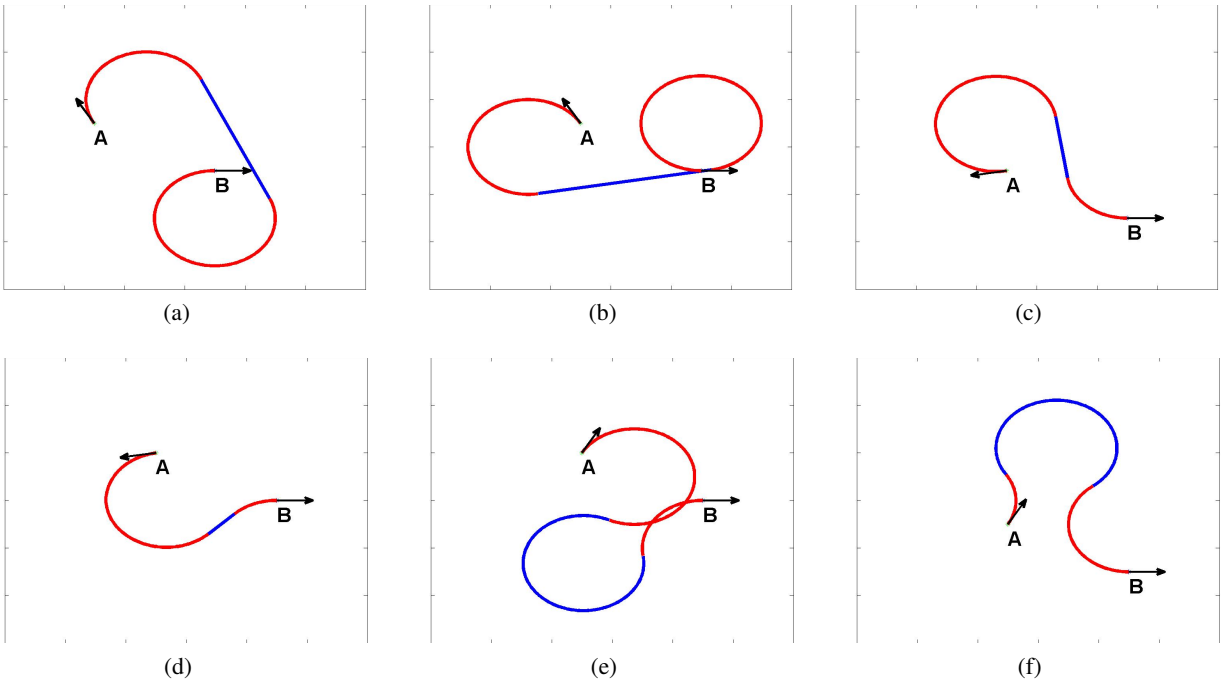



Figure 5.2: Dubins Curves: 3 segment paths of the form $C^+SC^+$ (a and b), $C^+SC^-$ (c and d) and $C^+C^-C^+$ (e and f). $C^+$ and $C^-$ are circular arcs in opposite directions of rotation and $S$ is a straight line segment.

the domain of $\theta_2$ [107].

When the assumption is relaxed to $(2 \cdot R)$, the optimal Dubins' path computation permits degenerate paths of the form $C^+C^-$, that do not exist for separations greater than $(4 \cdot R)$. This may be observed from the geometry of the curves. The generic curves of type $C^+SC^-$ do not exist between degenerate points corresponding to $C^+C^-$ curves [112]. In this case, path length as a function of $\theta_2$, other parameters remaining constant, is a *lower semi continuous function*. This implies that for small changes in the value of $\theta_2$, path length could increase abruptly. The abrupt behavior is observed only at degenerate points that correspond to $C^+C^-$ where both segments are of length less than $(\pi \cdot R)$. $C^+C^-$ curves with at least one segment greater than $(\pi \cdot R)$ in length occur as degenerate points in both $C^+C^-C^+$ and $C^+SC^-$ curves (Fig. 5.4) and thus act as transition points from one curve type to the other, thereby maintaining continuity in the path length function. While in the case when both segments of $C^+C^-$ type degenerate curve are less than $(\pi \cdot R)$ in length, the degeneracy occurs only in $C^+SC^-$ curves, thus making the optimal path length function discontinuous at precisely these points. The discontinuity can be observed directly from the geometry of the curves [112].

Consider a 3-point subsequence as extracted from the output of the lower bounding algorithm, with different arrival and departure angles, $\theta_2^a$ and $\theta_2^d$, at $\mathbf{B}$ (Fig. 5.3). Let $\mathbf{P_{AB}}$ and $\mathbf{P_{BC}}$ denote the path from $\mathbf{A}$ to $\mathbf{B}$ and $\mathbf{B}$ to $\mathbf{C}$, respectively.

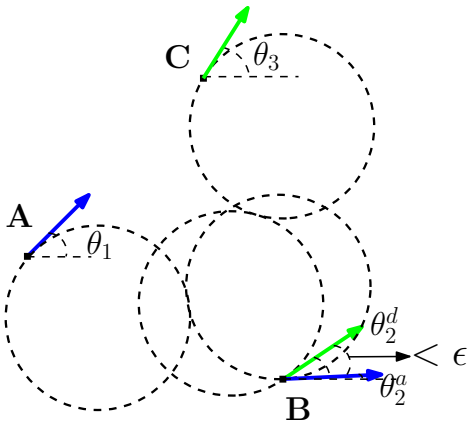



Figure 5.3: Problem Scenario for a 3-point subsequence extracted from the output of the lower bounding algorithm. To compute a feasible path from $\mathbf{A}$ to $\mathbf{C}$, the heading angle at $\mathbf{B}$ must be constant, i.e. arrival angle $(\theta_2^a)$ = departure angle $(\theta_2^d)$.

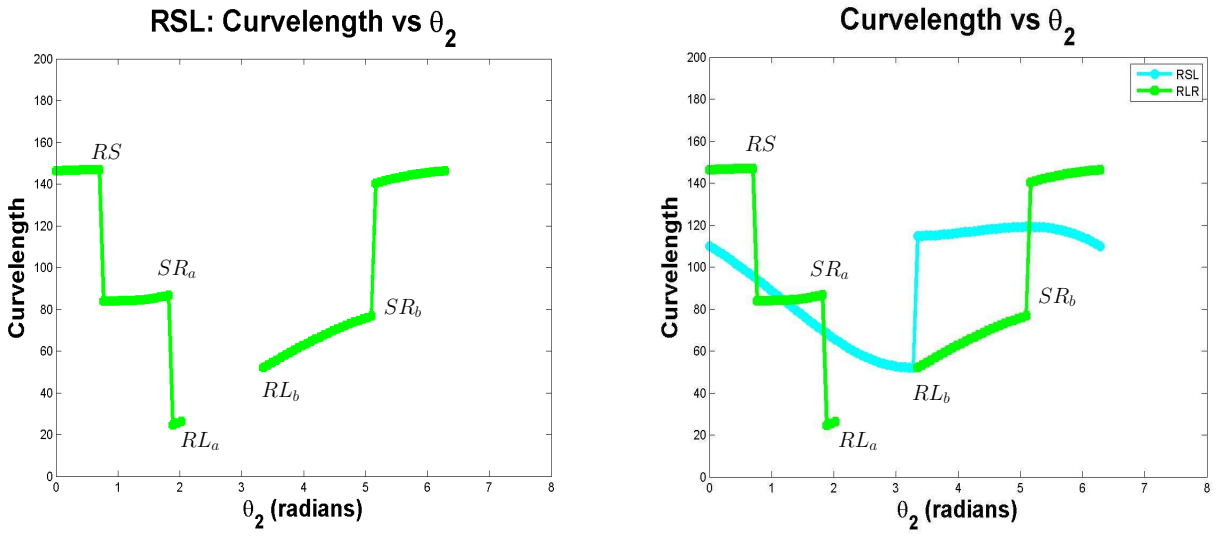
### 5.4.1 Preprocessing

The preprocessing step involves computing degenerate curves for the point sets, $< \mathbf{A}, \mathbf{B} >$ and $< \mathbf{B}, \mathbf{C} >$, and classifying them based on their characteristics.

- Compute all degenerate points on the path length function. Computation of degenerate curves is well studied in literature and any of the existing methods such as the one given in [105] or a sampling based method may be used. Degenerate points will be computed separately for $\mathbf{P_{AB}}$ and $\mathbf{P_{BC}}$. To compute Dubins curves or their degeneracies for $\mathbf{P_{BC}}$, direction of the path is considered in the reverse order, from $\mathbf{C}$ to $\mathbf{B}$. The departure angle from $\mathbf{C}$, for these computations, is considered as $(\theta_3 + \pi)$. Departure angle for the path from $\mathbf{B}$ to $\mathbf{C}$, would be derived similarly, as $\pi$ + arrival angle from $\mathbf{C}$ to $\mathbf{B}$.

- If both degenerate points ($\theta_2$ values) corresponding to $C^+C^-$ curves exist on the $C^+SC^-$ path length function, mark the interval between them for which the $C^+SC^-$ path does not exist. Let $I_{RL}^{AB}$ denote the interval for $RL$ degenerate curves for which $RSL$ path does not exist on $\mathbf{P_{AB}}$. Corresponding intervals on other $C^+SC^-$ curves are named similarly.

- Compute the shortest Dubins path at the points corresponding to $C^+C^-$ curves with both segments less than $(\pi \cdot R)$ in length. For each point, if the corresponding degenerate curve is of optimal length (i.e. minimum amongst all other Dubins curves), mark the point as a *point of discontinuity*. Let the set of such curves be denoted as $\Upsilon$.




(a) (b)

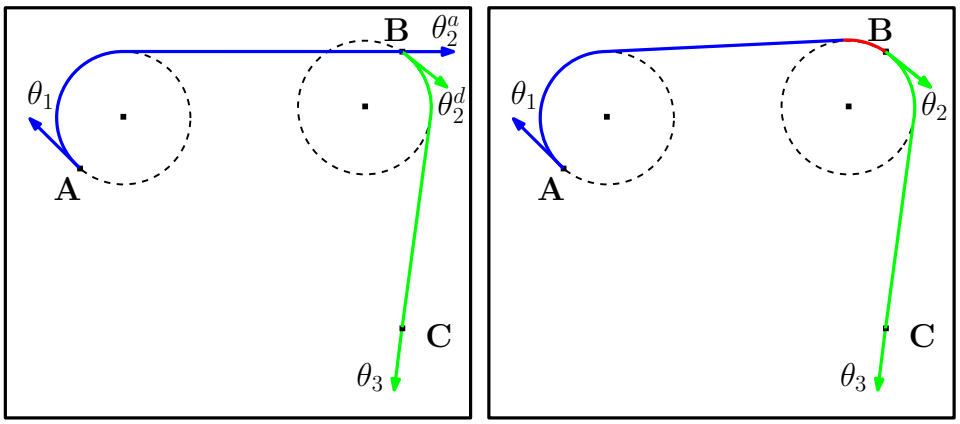
Figure 5.4: The Figure shows path length as a function of $\theta_2$. For RSL type path (a), this function is discontinuous between the two degenerate points corresponding to $RL$ curves. $RL$ curves occur as degeneracies in $RSL$, $RLR$ and $LRL$. $RLR$ and $LRL$ paths do not exist if the $L$ (respecively $R$) segment is less than $(pi \cdot R)$ in length [1]. $RL_a$ corresponds to the the case when both segments of the degenerate curve are less than $(pi \cdot R)$ in length. In case of $RL_b$, the $L$ segment is greater than $(pi \cdot R)$ in length and hence it also occurs as a degenerate curve in $RLR$ paths, as shown in (b). $RL_a$ occurs only in the length function for $RSL$ and hence leads to a discontinuity in the length function for the shortest Dubins path.

### 5.4.2 Solution Approach

The heading/orientation angle $\theta_2$ at $\mathbf{B}$ is computed differently for different combinations of curve types for $\mathbf{P_{AB}}$ and $\mathbf{P_{BC}}$. As noted earlier, the *points of discontinuity* on Dubins' path length function can occur only at RL or LR degenerate curves (when both segments are less than $(\pi \cdot R)$ length). Also, the interval $[\theta_2^a, \theta_2^d]$ may be clockwise or counter-clockwise and depends on independently computed optimal Dubins' curves for $\mathbf{P_{AB}}$ and $\mathbf{P_{BC}}$. Hence, on a given path even though both RL and LR degenerate points may occur within the given interval only one of them would incur a penalty in the path length function.

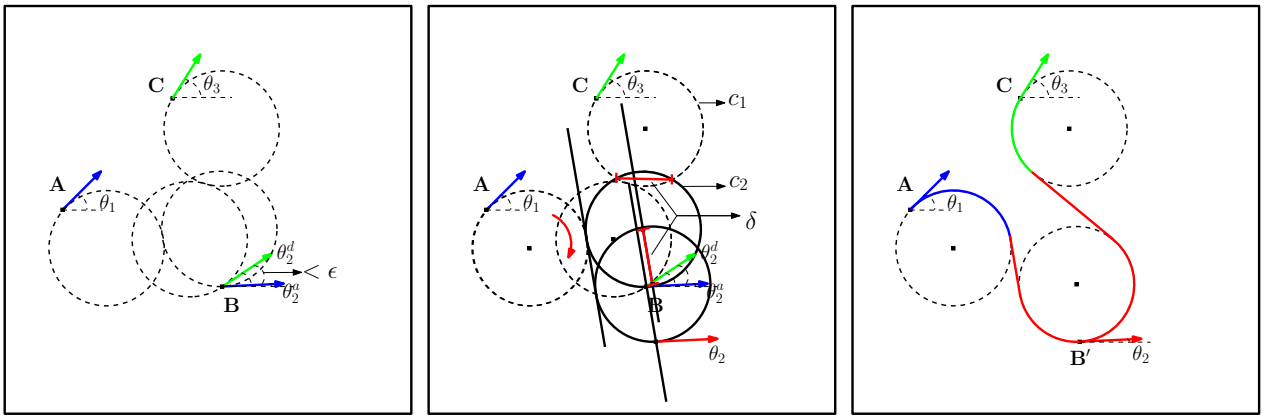
In the solution approach, it is determined if there exist any *points of discontinuity* within the interval $[\theta_2^a, \theta_2^d]$. Points on $\mathbf{P_{AB}}$ and $\mathbf{P_{BC}}$ are marked as $\theta_{RL}^a$ or $\theta_{LR}^a$ and $\theta_{RL}^d$ or $\theta_{LR}^d$, respectively, depending on the curve type. In the case conflicting points in $\Upsilon$ exist on $\mathbf{P_{AB}}$ and $\mathbf{P_{BC}}$ within the given interval, a perturbation mechanism is devised to dissolve the conflict. The location and orientation at $\mathbf{B}$ is computed to allow an approximation to the shortest path from $\mathbf{A}$ to $\mathbf{C}$ via $\mathbf{B}$ as enumerated in the steps below.

1. Check if there exists a *point of discontinuity* on $\mathbf{P_{AB}}$ or $\mathbf{P_{BC}}$ in $[\theta_2^a, \theta_2^d]$. If there are no such points




(a) (b)

Figure 5.5: Figure shows a case when there does not exist a *point of discontinuty* in the range $[\theta_2^a, \theta_2^d]$. The approximate path shown in 5.5b is computed using the approach outlined in Section 5.4.2.




(a) (b) (c)

Figure 5.6: Figure shows the application of the algorithm to a 3-point sequence, when both paths correspond to points of discontinuity in their respective length functions of the shortest Dubins path. Figure (a) shows the 3-point subsequence as extracted from the output of lower bounding algorithm, (b) shows the perturbation procedure and (c) shows the solution computed by the algorithm for a 3-point sequence.

within $[\theta_2^a, \theta_2^d]$, then the heading angle $\theta_2$ at $\mathbf{B}$ is computed as follows

$$\theta_2 = \underset{\theta \in \{\theta_2^a, \theta_2^d\}}{\arg\min} \left( L(\mathbf{A}, \mathbf{B}(\theta)) + L(\mathbf{B}(\theta), \mathbf{C}) \right)$$

2. If there exists a point in $\Upsilon$ on only one of $\mathbf{P_{AB}}$ and $\mathbf{P_{BC}}$, say $\theta_{RL}^a$ on $\mathbf{P_{AB}}$, check if the intersection condition, $I_{RL}^{AB} \cap (\theta_{RL}^a, \theta_2^d] \neq \phi$, is true. The intersection condition can be true for only one of $\theta_{RL}^a$ or $\theta_{LR}^a$ (similarly, $I_{RL}^{BC} \cap [\theta_2^a, \theta_{RL}^d) \neq \phi$ for $\theta_{RL}^d$ on $\mathbf{P_{BC}}$).

   If the intersection condition is not true, $\theta_2$ is computed as:

$$\theta_2 = \underset{\theta \in \{\theta_{RL}^a, \theta_2^d\}}{\arg\min} \left( L(\mathbf{A}, \mathbf{B}(\theta)) + L(\mathbf{B}(\theta), \mathbf{C}) \right)$$

   or as follows for $\theta_{RL}^d$ on $\mathbf{P_{BC}}$,

$$\theta_2 = \underset{\theta \in \{\theta_2^a, \theta_{RL}^d\}}{\arg\min} \left( L(\mathbf{A}, \mathbf{B}(\theta)) + L(\mathbf{B}(\theta), \mathbf{C}) \right)$$

   If intersection condition is true, $\theta_2$ is computed as:

$$\theta_2 = \theta_{RL}^a$$

   or as follows for $\theta_{RL}^d$ on $\mathbf{P_{BC}}$,

$$\theta_2 = \theta_{RL}^d$$

3. If a candidate pair of conflicting degnerate points exists, for example $\theta_{RL}^a$ on $\mathbf{P_{AB}}$ and $\theta_{LR}^d$ on $\mathbf{P_{BC}}$, check for the existence of a conflict.

   If $\theta_{RL}^a$ exists (similarly for $\theta_{LR}^a$) and $I_{RL}^{AB} \cap (\theta_{RL}^a, \theta_2^d] \neq \phi$, check if $\theta_{LR}^d$ (respectively $\theta_{RL}^d$) exists on $\mathbf{P_{BC}}$ in the range $(\theta_{RL}^a, \theta_2^d]$. If not, then

$$\theta_2 = \theta_{RL}^a$$

   If $\theta_{LR}^d$ exists on $\mathbf{P_{BC}}$ in the range $(\theta_{RL}^a, \theta_2^d]$ and $I_{LR}^{BC} \cap [\theta_{RL}^a, \theta_{LR}^d) \neq \phi$, there exists a conflict. In case of a conflict, none of $\theta_2^a$ or $\theta_2^d$ can be moved any further towards each other without incuring a cost penalty in the path length function.

In this scenario the location of $\mathbf{B}$ is perturbed in a systematic manner, as shown in Fig.5.6. The perturbation amount is equal to the length of the common chord between the circles $c_1$ and $c_2$ as shown in the figure. The direction of perturbation is along the tangent between the two segments of $\mathbf{P_{AB}}$ in the direction of rotation of its first segment. The perturbation direction is away from $\mathbf{C}$. The amount of perturbation, $\delta$, is computed as:

$$\delta = 2 \cdot R \cdot \sin(\cos^{-1}(\frac{D}{2 \cdot R}))$$

Let the new point be denoted as $\mathbf{B'} =< \mathbf{x'}, \mathbf{y'}, \theta_{\mathbf{2}}^{'} >$, where $\theta_2^{'} = \theta_2^a$.

## 5.5 Conclusion

In this work the n-point sequence problem for a curvature constrained mobile robot is addressed. Dubins car-like vehicle model [1] is used to model to represent the mobile robot. The resultant problem is known as Generalized Dubins Path Problem (GDPP). A practically useful algorithm is designed that generates approximate paths using known lower bounding solutions [105].

# Chapter 6

# Path Planning for a UAV with Kinematic Constraints in the Presence of Polygonal Obstacles

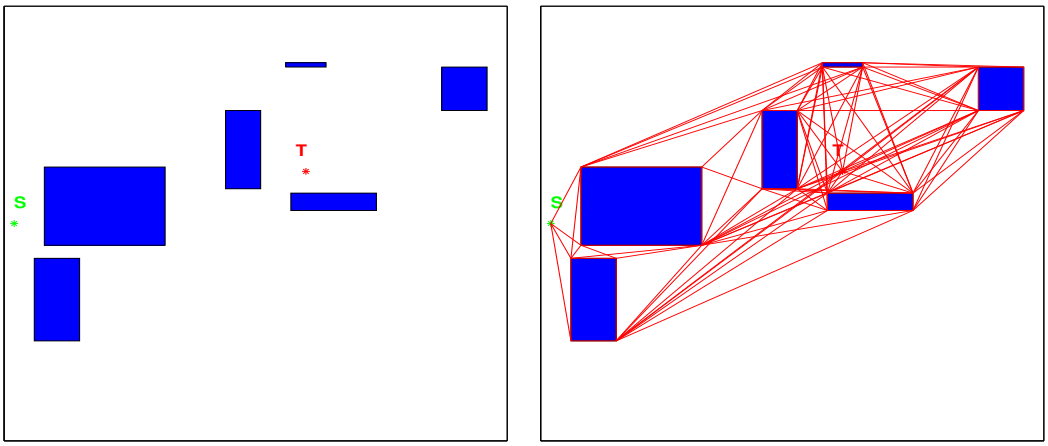
Path planning for unmanned aerial vehicles (UAVs) with obstacle avoidance is challenging attributed to the need for constant motion, unpredictable environmental factors like dynamically varying winds and added constraints on maneuvering. A popular approach to solve the path planning problem for UAVs used in the literature is to formulate it as an optimization problem [115]. However optimization is not straight forward since the path length needs to be optimized over free space while satisfying kinematic constraints. Some other techniques used in literature, include RRT [116], A* [117], Voronoi diagrams [118], visibility graphs [119], potential fields [120], road maps [117] and cell decomposition [121] based methods. This work aims to solve the path planning problem for a UAV with kinematic constraints while avoiding obstacles. A Dubins vehicle model applied to a point robot is used to approximate the kinematic properties of the UAV. The environment and obstacles are modeled using a visibility graph and a novel graph-theoretic interpretation of the steering angle constraint is devised that allows easy validation of the turning angle. A two step algorithm that is practically useful and easy to implement is designed to find feasible paths in the presence of polygonal obstacles. The first step comprises of a modified version of Dijkstra's shortest path algorithm for positive-weighted graphs. In the second step a reverse search is conducted on the visibility graph using results of the first step as priors, to find a feasible path for the UAV.

## 6.1 Related Work

L.E. Dubins' seminal work [1] on curvature constrained shortest paths in an obstacle free environment remains the most conclusive result on the path planning problem with constraints on steering angle. Dubins car-like vehicle model, as discussed in Chapter 5 describes a forward moving car with a constraint on the maximum curvature of the path. Dubins concluded that a smooth path of shortest length between any two states of the curvature constrained vehicle consists of at most *three* subpaths, each of which is either a straight line or a maximum curvature arc. Finding shortest paths for a Dubins car in an environment with arbitrary polygonal obstacles is known to be NP-hard [122]. Backer and Kirkpatrick [123] give a $(1+\epsilon)$-approximate algorithm with time complexity, polynomial in the number of obstacle vertices. The same authors [124] also give a polynomial time algorithm to find feasible unit-curvature paths in the presence of polygonal obstacles. They use feature points in the environment as anchors for turn segments and coupling between turns to devise a new normal form for the unit-curvature paths. Agarwal et. al. [125] give a characterization of the shortest curvature constrained paths in the presence of convex obstacles.

Even though a Dubins model can take discrete turns, path planning literature focuses on using binary values for the turning angle, *zero* or *maximum turning angle*. Eriksson-Bique et. al. [126] describe polygonal paths with discrete curvature and show that in the limiting case these reduce to Dubins paths. The paths are constrained by an upper bound on the maximum instantaneous change in the heading angle. The approach discussed in this thesis also uses the ability of a Dubins model to make discrete turns as long as the turn angle is less than the maximum steering angle of the robot. Chitsaz and LaValle [101] extend the Dubins car model to three dimensional space and give a characterization of shortest paths between two states of a Dubins Airplane. They classify the path planning problem using the goal altitude as, low, medium and high and give time-optimal characterizations of the shortest path. Choi [96] extends the Dubins car and Dubins airplane models to the case of path planning with unidirectional turning constraints. In a related line of work McGee and Hedrick [98][127] use non-linear optimization to do path planning for a kinematic airplane model in the presence of wind. Bortoff [118] gives a two-step path planner for UAVs in the presence of threat zones. The planner uses a Voronoi graph search to find an initial solution and improve on it using virtual force fields. Yang and Kapila [97] consider a class of 2-D optimal path-planning problems for UAVs with kinematic and tactical constraints. They use vector calculus to reduce the problem to parameter optimization and characterize a necessary condition for optimal path planning in the presence of tactical constraints. This work to plan an obstacle free path for a UAV with kinematic constraints in the presence of polygonal obstacles.

This work develops a two step planner to find an obstacle free path that satisfies kinematic constraints of




(a) (b)

Figure 6.1: (a). A sample environment showing polygonal obstacles (blue rectangles), start ($\mathbb{S}$) and terminal ($\mathbb{T}$) locations. The path planning problem needs to find a kinematics constrained path around the obstacles to start from $\mathbb{S}$ and reach $\mathbb{T}$. (b). Visibility graph as computed for the sample environment. The graph is augmented to include the start and terminal locations.

the UAV. The planner is practically useful and easy to implement. It uses a visibility graph representation of the environment and develops a modified version of Dijkstra's algorithm. A novel validation mechanism is built for the steering angle constraint. In the first step, a modification to Dijkstra's single source shortest path algorithm is designed to output paths suitable for a UAV. Time complexity of the algorithm is polynomial in the number of vertices in the graph. If a feasible path is not found in the first step, a heuristic search is conducted on the graph in the second step. Paths generated in the first step are used as priors to speed up the search. The search strategy however, is independent of step one and can be used as an individual algorithm in its own right. Further, the use of a graphical model of the environment and the validation mechanism for the steering angle constraint enables development of intelligent heuristics as extensions to this work. Simulation results that show paths computed by the planner in randomly generated environments are presented.

## 6.2 Problem Description and Preliminaries

Given an environment, $\mathbb{E}$ (see Figure 6.1a), plan a path, $\mathbb{P}$, for a fixed-wing UAV to travel from initial state, $\mathbb{S} = (x_0, y_0, \psi_0)$, to final state, $\mathbb{T} = (x_f, y_f, \psi_f)$, that avoids obstacles (polygonal) and respects kinematic constraints of the UAV. The UAV is assumed to travel at a constant altitude and modeled as a Dubins vehicle. The problem is reduced to path planning in two dimensions with a constraint on maximum curvature in the presence of arbitrary polygonal obstacles and is known to be NP-hard [122].

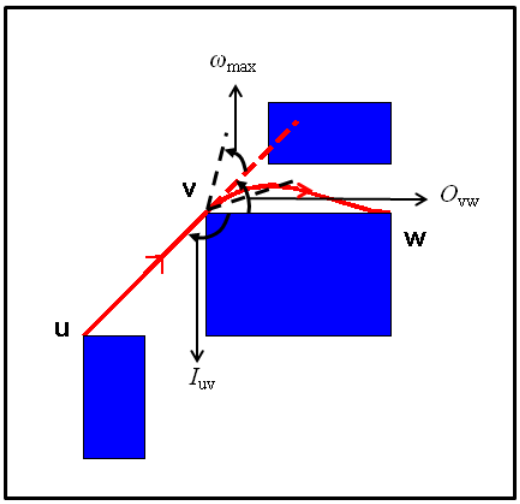



Figure 6.2: $I_{u,v}$ is the arrival angle, $O_{v,w}$ is departure angle and $\omega_{max}$ is the maximum steering angle. Transition from edge $(u, v)$ onto edge $(v, w)$ is not valid as it violates the constraint on steering angle. Transition, $\zeta = (u, v, w)$, where $\sigma_{vw}$ is the Dubins curve form $v$ to $w$ is valid, since $(|O_{vw} - (I_{uv} + \pi)| \leq \omega_{max})$ and the curve does not intersect any obstacle.

### 6.2.1 Visibility Graphs

Visibility graph [128] (Fig. 6.1b) is a tool that allows environments with polygonal obstacle to be viewed as graphs. The set of obstacle end-points forms the vertex set of the visibility graph. An edge exists between a pair of vertices in the graph if the corresponding obstacle end-points can be joined by a straight line that does not intersect any other obstacle. Edge weights are assigned as the Euclidean distance between the two end-points. Shortest path between a pair of vertices in the graph gives an obstacle-free path for a vehicle that follows a unicycle model [129]. However, due to constraints on the maximum steering angle, the shortest path so chosen may not be feasible for a Dubins curvature constrained vehicle. Specifically some of the transitions may be invalid because the turn angle is larger than the maximum instantaneous turn angle, $\omega_{max}$.

### 6.2.2 Dijkstra's Single Source Shortest Path Algorithm

Dijkstra's algorithm [130] solves the single source shortest path problem in a positive weighted graph. The problem is described as: consider an edge weighted graph, $G(V, E)$, with cost function $C : E \rightarrow \mathbb{R}^+$, such that $C(u, v) = \infty$, if there is no edge between $u$ and $v$. Let $s \in V$ be a vertex marked as source; find the shortest distance from $s$ to every vertex $t \in V \setminus \{s\}$. Dijkstra's algorithm computes a shortest path tree and returns the shortest distance from $s$ to each vertex. The algorithm has a time complexity of $O(|E| + |V| \log |V|)$.

Dijkstra's algorithm maintains two partitions, $S$ and $T$, of the graph. $S$ is initialized to $\{s\}$ and $T$ is

**Algorithm 9** UAVdijkstra$(G, s, \tau)$

$S$ = {s}
$T = V \backslash \{s\}$
dist($v$) = $\infty, \forall v \in T$
parent($v$) = $\phi, \forall v \in T$
dist($s$) = 0
parent($s$) = 0
**while** $T \neq \phi$ **do**
    d($v$) = $\min_{u \in S}$ {dist($u$) + $\tau_u(parent(u), v)$}, $\forall v \in T$
    pred($v$) = $\arg\min u \in S$ {dist($u$) + $\tau_u(parent(u), v)$}, $\forall v \in T$
    $minNode$ = $\arg\min v \in T$ d($v$)
    $T = T \backslash$ { $minNode$ }
    $S = S \bigcup$ { $minNode$ }
    dist($minNode$) = d($minNode$)
    parent($minNode$) = pred($minNode$)
**end while**

initialized to $V \setminus \{s\}$. At each iteration of the algorithm, a vertex is removed from $T$ and appended to $S$. It selects a vertex from $T$ that is connected to $s$, using the shortest path across the partition. Let this vertex be $v$. Since all edges are positive weighted, the last edge on the path from $s$ to $v$, $(u, v)$, goes across the partition i.e. $u \in S$. $v$ is removed from $T$ and added to $S$. Edge $(u, v)$ is also added to S. This produces a unique shortest path from $s$ to $v$ in S. The algorithm iterates until all vertices are added to $S$ or there is no edge that goes across the partition, in which case the graph is disconnected. Edges in $S$ form a shortest path tree and define a unique path from $s$ to each vertex.

## 6.3 Approach

### 6.3.1 Terminology

Information about the environment is available as a visibility graph, $G(V, E)$. $I_{uv}$ is the arrival angle at $v$ associated with the directed edge $(u, v)$ of the graph (see Fig. 6.2). In the search for an obstacle-avoiding path, the smallest independent unit of the path for a UAV is a $segment$. Segment, $\sigma^u_{vw}$, refers to the UAV's path from $v$ to $w$ when $u$ is the immediately preceding vertex of $v$ on the path. The arrival angle at $w$ along the segment $\sigma^u_{vw}$ $(\forall u \in V)$ is fixed to be $I_{vw}$. This prunes the search space by limiting the number of value for the arrival angle at a vertex. The departure angle from $v$ along the outgoing segment $\sigma^u_{vw}$ is denoted as $O^u_{vw}$. A segment, $\sigma^u_{vw}$, is deemed *feasible* if $|O^u_{vw} - (I_{uv} + \pi)| \leq \omega_{max}$. All angles are measured about the positive $x - axis$; a positive (negative) value denotes counter clockwise (clockwise) direction.

A segment, $\sigma_{vw}^{u}$, is restricted to either be along the edge $(v, w)$ of the graph or any of the six Dubins curve from $v$ to $w$. The heading angles at $v$ and $w$ used to compute the Dubins' curves are $I_{uv}$ and $I_{vw}$, respectively. Path along the directed edge of the graph is deemed feasible when the departure angle is within a maximum tolerance of the arrival angle at $v$. This admits the path along an arc of a circle with infinite turning radius. In the limiting case when $\omega_{max}$ tends to 0, the set of feasible segments reduce to one of the Dubins' curves. The curve shortest in length and avoiding all obstacles is selected as $\sigma_{vw}^{u}$. $\sigma_{vw}^{u} = \emptyset$, if $w$ is not reachable from $v$ by an obstacle avoiding path when $u$ is the last visited vertex.

**Definition 1.** *A* path *is a sequence of segments, $\mathbb{P}_{(s,t)} = \sigma_1, \sigma_2, \ldots, \sigma_m$, from an initial vertex, $s$, to a terminal vertex, $t$, such that consecutive segments share 2 vertices (in order).*

A path such that all it's constituent segments are *feasible*, is called a *feasible* path. By construction the path is always *obstacle free*. A *feasible* and *obstacle free* path is a *valid* path. The *shortest* path for a UAV is a *valid* path of shortest length between two vertices of the graph.

This leads to the definition of a *simple* path. In general, a *simple* path does not have a loop. A path that does not visit any vertex more than once, is a sufficient characterization of a *simple* path for a unicycle model. However, in case of a UAV, a *simple* path may visit a given vertex $v$ more than once, without resulting in a loop, as long as each visit is the result of a different segment.

**Definition 2.** *A* simple *path, for a UAV with kinematic constraints, is a path that does not traverse a segment, $\sigma_{vw}^{u}$, more than once.*

Let the number of vertices in the visibility graph be $n$, then a transition matrix, $\tau_v$, for a vertex $v \in V$ is a matrix of size $n \times n$ such that,

$$\tau_v(i,j) = \begin{cases} ||\sigma_{vj}^{i}|| & \text{if } \sigma_{vj}^{i} \neq \emptyset \\ \infty & \text{otherwise} \end{cases}$$

where, $||\sigma_{vw}^{u}||$ is the length of segment $\sigma_{vw}^{u}$. We compute the transition matrix for each vertex a priori.

**Algorithm 10** initializePathFinder($G, S, s, t, \tau, parent$)

\\ Output - $\mathbb{P}_{(s,t)}$: simple valid path from $s$ to $t$
1: $\mathbb{P}_{(s,t)}$ = NULL
2: pathLength = $\infty$
3: **for each** $v \in S$ **do**
4: **if** $(\tau_t(v,t) = 0)$ **then**
5: continue;
6: **end if**
7: $\mathbb{P'}_{(s,t)}$ = pathFinder($G, S, s, t, v, \tau, parent, \mathbb{P}_{(v,t)}$)
8: **if** (pathLength $\geq ||\mathbb{P'}_{(s,t)}||$) **then**
9: $\mathbb{P}_{(s,t)} = \mathbb{P'}_{(s,t)}$
10: pathLength = $||\mathbb{P}_{(s,t)}||$
11: **end if**
12: **end for**

### 6.3.2 Solution

A two-step algorithm is developed to find a *simple valid* path for a UAV in the presence of polygonal obstacles. As described in section 6.2.2, Dijkstra's algorithm maintains two partitions of the graph, $S$ and $T$. For each vertex in $S$, it stores the information of it's nearest neighbor in $T$. In every iteration, the neighbor of $S$ that is reachable by the shortest path from $s$ is removed from $T$ and added to $S$. The following modification is made to Dijkstra's algorithm to compute the path for the UAV. Let $v$ be a vertex in $S$ and $u$ be it's predecessor on the unique path from $s$ to $v$ in $S$. In case of $s$, the preceding vertex is considered to be $s$ itself. Then $v$'s nearest neighbor in $T$, $w$, is selected as the one reachable by the least cost feasible segment $\sigma^u_{vw}$. Thus, we select $w$ such that $\tau_v(u,w)$ is the minimum valued entry in the $u^{th}$ row of $\tau_v$. The neighbor reachable by the shortest path from $s$ is then selected as the vertex to be added to $S$. The modified version, called as *UAVdijkstra*, is described in pseudo-code format in Algorithm 9 and has a time complexity of $O(|V|^3)$. By construction it is clear, that the $s$-$t$ path returned by *UAVdijkstra* is *feasible* and *obstacle free* and hence, *valid*. However, it may so happen that *UAVdijkstra* does not find an $s-t$ path in the visibility graph even when one exists. This is because only the unique shortest paths already included in $S$ can be used to find valid paths to vertices in $T$. It is not hard to see, that a vertex not reachable from the paths in $S$ might still be reachable from $s$ using a transition not included in $S$. In the event, when $t$ is not included in $S$ by the end of *UAVdijkstra*, i.e. a *valid* $s-t$ path is not found, step two of the algorithm is executed.

The second step of the solution is developed as a recursive routine named *pathFinder* (Algorithm 11) and has a time complexity of $O(|V|^3)$. It performs a reverse search in the graph starting from $t$ for any *simple valid* $s-t$ path that exists in the visibility graph. *initializePathFinder* (Algorithm 10) is an initialization routine to *pathFinder*. It takes as input the output of step 1, namely the set $S$ and vector *parent* that comprises of the

**Algorithm 11** pathFinder($G, S, s, w, v, \tau, parent, \mathbb{P}_{(v,t)}$)

\\ Output - $\mathbb{P}_{(s,t)}$: simple valid path from $s$ to $t$
1: $\mathbb{P}_{(s,t)}$ = NULL
2: pathLength = $\infty$
3: **for all** $u \in S$ **do**
4: **if** $((\sigma^u_{vw} \in \mathbb{P}_{(v,t)})$ || $(\tau_v(u,w) = \infty))$ **then**
5: continue;

6: **else if** $(u == s)$ **then**
7: $\mathbb{P}'_{(s,t)} = \{\sigma^u_{vw}\} \cup \mathbb{P}_{(v,t)}$
8: **if** (pathLength $\geq ||\mathbb{P}'_{(s,t)}||$) **then**
9: $\mathbb{P}_{(s,t)} = \mathbb{P}'_{(s,t)}$
10: pathLength $= ||\mathbb{P}_{(s,t)}||$
11: **end if**

12: **else if** $(u == parent(v))$ **then**
13: $\mathbb{P}_{(u,t)} = \{\sigma^u_{vw}\} \cup \mathbb{P}_{(u,t)}$
14: $\mathbb{P}'_{(s,t)} = \mathbb{P}_{(s,u)} \cup \mathbb{P}_{(u,t)}$
15: **if** (pathLength $\geq ||\mathbb{P}'_{(s,t)}||$) **then**
16: $\mathbb{P}_{(s,t)} = \mathbb{P}'_{(s,t)}$
17: pathLength $= ||\mathbb{P}_{(s,t)}||$
18: **end if**

19: **else**
20: $\mathbb{P}_{(u,t)} = \{\sigma^u_{vw}\} \cup \mathbb{P}_{(u,t)}$
21: $\mathbb{P}'_{(s,t)}$ = pathFinder$(G, S, s, v, u, \tau, parent, \mathbb{P}_{(u,t)})$
22: **if** (pathLength $\geq ||\mathbb{P}'_{(s,t)}||$) **then**
23: $\mathbb{P}_{(s,t)} = \mathbb{P}'_{(s,t)}$
24: pathLength $= ||\mathbb{P}_{(s,t)}||$
25: **end if**
26: **end if**
27: **end for**

predecessors of vertices in $S$ on the unique shortest path from $s$, along with visibility graph of the environment and the vector of transition matrices. It makes a call to *pathFinder* for every $v \in S$ such that $\tau_t(v,t) \neq \infty$. The *pathFinder* routine takes the path, $\mathbb{P}_{(v,t)}$, constructed so far as one of the inputs. Each call to *pathFinder* adds a segment, $\sigma^u_{vw}$ to the path and makes a recursive call on the transition matrix of $u$ (lines 20-27). The recursion returns when any one of the termination conditions are met.

### Termination Conditions:

1. $\tau_v(s,w) \neq 0$

2. $\tau_v(parent(v),w) \neq 0$

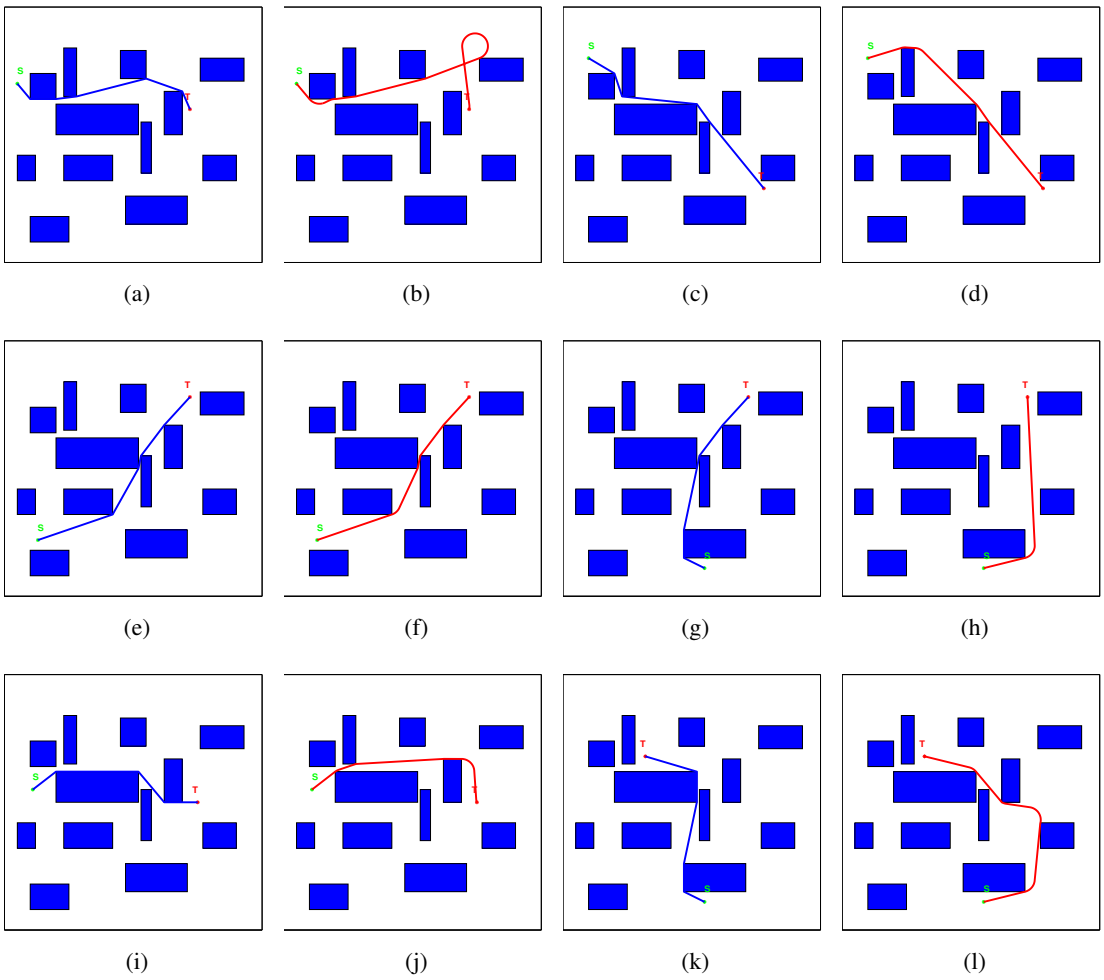



Figure 6.3: Sample simulation results conducted on a randomly generated environment for various initial and terminal configurations. (a), (c), (e), (g), (i) and (k) show the shortest path as returned by Dijkstra's algorithm. These paths violate the maximum turning angle constaint and are infeasible for a UAV. (b), (d), (f), (h), (j) and (l) show paths generated using our algorithm for the same configurations. The paths are successful in compromising obstacles and are traversable by a UAV.

3. $\forall u \in S, \tau_v(u, w) = \infty$

4. $\exists u \in S,$ s.t. $\tau_v(u, w) \neq \infty \cap \sigma_{vw}^u \in \mathbb{P}_{(v,t)}$

Conditions 1 (lines 7-12) and 2 (lines 13-19) above, correspond to the event when a *simple valid* $s - t$ path is found. Condition 1 implies, $\sigma_{vw}^s$ is a valid segment from $s$ onto $w$ through $v$. Then, $\mathbb{P}_{(s,t)} = \sigma_{vw}^s \cup \mathbb{P}_{(v,t)}$. Condition 2 means, $\sigma_{vw}^u$ is a valid segment such that $u$ = parent($v$). Thus, $u$ lies on the unique path, $\mathbb{P}_{(s,v)}$, from $s$ to $v$ as found in step one. Then, $\mathbb{P}_{(s,t)} = \mathbb{P}_{(s,v)} \cup \sigma_{vw}^u \cup \mathbb{P}_{(v,t)}$. Condition 3 (lines 5-6) occurs when there does not exist a $u \in S$ such that $\sigma_{vw}^u$ is a valid segment and condition 4 (lines 5-6) detects a loop. The algorithm always returns path if the instance is feasible and there exists a *simple valid* path from $s$ to $t$.

## 6.4 Results and Analysis

A two step solution to the problem of path planning for a UAV with kinematic constraints in the presence of arbitrary polygonal obstacles. Step one is a modified version of Dijkstra's shortest path algorithm that finds a feasible path for the UAV on visibility graph representation of the environment. The output of step one is a tree such that there is a unique path from $s$ to every vertex in $S$. By precomputing the transition matrix $\tau$ for each vertex, an on-line implementation of the algorithm in step one is achievable. Time complexity of augmenting the transition matrices in case of new obstacles is polynomial in the number of obstacle vertices in the environment. Step two makes a complete search for the existence of a *simple valid* $s-t$ path in the visibility graph of the environment and returns false if such a path cannot be found. The use of termination condition 2 for step two of the algorithm enables the use of paths from $s$ to $v$ generated in step one to compute a valid path from $s$ to $t$. This allows the solution to find a path from $s$ to $t$ by using information precomputed in step one. Output of the algorithm for a few instances in a simulated environment is shown in Fig. 6.3.

Simulations were conducted to verify the approach and observe properties of the generated paths. An environment of size 100 units x 100 units was populated with 12 randomly generated obstacles. Random initial and final states, $\mathbb{X}_0$ and $\mathbb{X}_f$, were generated for the vehicle. The minimum turning radius, that is a function of vehicle speed and model design, was fixed to be 6 units. Maximum instantaneous turn angle for the vehicle was assumed to be 30 $^\circ$ in the simulations. Fig. 6.3 shows some sample outputs generated by the algorithm. Paths returned by this planner are obstacle free and satisfy the kinematic constraints of a Dubins vehicle and are traversable by a UAV with kinematic constraints. In the limited simulations of the algorithm, we have observed step one to perform well and return paths in most cases. It may be observed that step two can be run independent of phase one of the algorithm, by replacing $S$ with $V$ and removing termination condition 2, in Algorithms 10 and 11. However during the simulations, step one was found to be a very strong prior and accelerated the search considerably. In most typical environments *UAVdijkstra* is able to find a *simple valid* path. Even when it fails to return a path, by virtue of computing a valid path to each vertex in $S$ it speeds up step two.

## 6.5 Conclusion

A two step algorithm to plan obstacle-free paths for a UAV with kinematic constraints in the presence of polygonal obstacles. It uses a graphical representation of the environment and proposes a novel interpretation that allows easy validation of the steering angle constraint. As step one of the algorithm, a modified version

of Dijkstra's algorithm is developed for Dubins' vehicles that runs in polynomial time and is suited for on-line implementation. An exhaustive reverse search is proposed as step two of the algorithm, in case step one does not find a path. The use of the results of step one as priors, sped the reverse search significantly. The solution performs well in practice and is able to find paths traversable by a UAV in randomly generated environments.

# Chapter 7

# Conclusion

Mobile robotic systems, single or multi-robot, can be used in a multitude of different tasks in varying operating conditions. This thesis addresses a few problems in this space and develops practical solutions. Important advancements are developed in solution methods for each of the problems addressed and new open threads are identifies that could be explored further. Fuel-constrained planning, for instance, is a very important problem and the solutions developed in this thesis are extensible to many scenarios and robotic hardware. Problems within the theme of visual coverage and monitoring using cooperative multi-robot systems are relevant to the community and demonstrate cooperation in differing capacities. Given the current thrust on intelligent field robotics, heterogeneous mobile robot systems are significant to operating successfully in unknown environments. In comparison to industrial settings, that offer a controlled setup, where the environment can be calibrated and the robot operations may be monotonic, robot operations in an outdoor natural setting can be very challenging. The exploitation of varying capabilities to complement limitations of different type of robots in a multi-robot system, to achieve a common goal is the way forward. For successful operation, this requires addressing the finer aspects of these cooperative missions. Aspects that relate to robot-based visual monitoring of marked geographical features and regions are addressed. Visibility constraints that arise as a result of terrain features and field of view limitation pose challenges to field operations. In this regard several problem are addressed including the coverage problem on a plane (Chapter 2), monitoring of linear features within a terrain (Chapter 3) and subsequently the multi-point monitoring problem on a terrain where the location of the points-of-interest is not restricted and can be random (Chapter 4). In the case of planar two dimensional environments, ground robots are utilized as refueling vehicles and a coordinated plan must be computed. For persistent monitoring of linear features on a terrain, both aerial and ground robots are utilized as robotic sensors that must cooperate to cover the areas of interest while optimizing individual robot costs. In either of the two

problems the path planning does not rely on any active communication between the robots. To make head start into the more difficult problem of monitoring on a terrain, some of the other aspects of the problem are simplified in Chapter 4. While mobile refueling vehicles are utilized in planar environments and stationary refuel depots are used in the case of persistent monitoring of piece-wise linear features on terrains, for monitoring points-of-interest that may be located anywhere within the terrain the use of a single aerial robot is explored. Algorithmic approaches that admit random terrains and compute paths for the aerial robot are developed. A constant-factor approximation algorithm and the construction of its proof is discussed. Practically useful and computationally efficient method based on reduction to an instance of GTSP is also discussed. Experiment and field deployments for each of these problems is discussed. These deployments are used to validate the developed algorithms and establish proof-of-concept in each case.

In the case of wheeled mobile robots and fixed-wing aerial robots, curvature constrained planar motion model provides a very clean interface to develop path planning algorithms. Within the second theme of problems addressed in this thesis, Dubins' model for a forward-moving car is used to develop path-planning approaches for robots with kinematic constraints. Computing smooth curvature constrained paths in practice is a challenge. It is shown in Chapter 5 that with small and precise perturbations to waypoint locations, a smooth path through a sequence of points may be computed in polynomial time. This result generalises the point-to-point path planning algorithm by Dubins to a sequence of points. Extension of Dubins' path planning results to an obstacle-laden environment where the obstacles are constrained to be polygonal is discussed in Chapter 6). A two-phase search algorithm is developed that encodes a graphical model and uses a pruning mechanism to improve search time for the shortest obstacle-free path within the graph. This algorithm is easy to implement and computes a valid path in polynomial time.

## 7.1 Future Directions

There are many directions, one could possibly extend each of these results. The fuel constrained UAV routing problem with mobile refueling station, discussed in Chapter 2 opens avenues for many interesting extensions. A very relevant extension is to consider persistent area coverage satisfying a given coverage rate for monitoring applications. The construction-heuristic based on a TSP tour of the targets and a repair algorithm, allows for the development of evolutionary techniques as extensions to this work that have been found to work reasonably well for many similar problems. A budgeted version of the problem that maximizes coverage with a cap on the number of refuels also makes an interesting variant of the problem. The literature on refueling problems also lacks any approximation schemes for the problem. Other interesting extensions to this problem, include

multiple cost functions (for practitioners), pareto-optimal solutions and multiple-UAV multiple-RV scenarios.

The persistent monitoring problem for piece-wise linear features on a terrain (Chapter 3) with stochastic fuel consumption is an open problem. It would involve the use of a probabilistic cost matrix and a multi-stage stochastic solution. Stochastic modeling captures many real-world constraints and would help compute solutions closer to field implementations. On the experimental validation front, a closed loop hardware implementation would be the way forward. It would be a challenging engineering problem and would also expose interesting research problems. Another set of problems is related to monitoring water bodies using aerial and surface vehicles which have similar properties in respect to visibility coverage but need more involved planning strategies. The multiple-point monitoring problem on terrains discussed in Chapter 4 opens a plethora of related problems. Computationally tractable solutions for complete coverage on terrains using a single robot is still an open problem and is extremely hard to solve. Extending these results to multi-robot systems that utilize both aerial and ground robots has not been explore before and forms for a challenging research problem given the computational hardness.

The trajectory planning problems for curvature constrained vehicles is a classic problem. Dubins' model is a simple and clean model that allows easy development of planning strategies. Design of navigation algorithms to exactly follow the Dubins' path for a mobile robot is difficult. Field implementations would add value to these results as well as open up new problems.

## 7.2 Concluding Remarks

In conclusion, research in using mobile robot systems for solving everyday problems holds promise. Robotic solutions for complex tasks in uncontrolled outdoor settings have evolved rather quickly over the last decade. There is interest and excitement towards the new set of solutions and I hope that we keep making progress. This thesis makes a small contribution towards a larger goal of intelligent and reliable use of mobile robots for performing tasks traditionally considered challenging. There are still many problems that would need to be addressed before much of these research evolve into products. These problems interest the community as a whole and would need concerted efforts from researchers, engineers and other experts from diverse backgrounds to solve them.

# Bibliography


[1] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, pp. 497–516, 1957.

[2] P. Tokekar, J. V. Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic uav and ugv system for precision agriculture," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1498–1511, Dec 2016.

[3] I. Colomina and P. Molina, "Unmanned aerial systems for photogrammetry and remote sensing: A review," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 92, pp. 79–97, 2014.

[4] P. B. Sujit, D. Kingston, and R. Beard, "Cooperative forest fire monitoring using multiple uavs," in *IEEE Conference on Decision and Control*, Dec 2007, pp. 4875–4880.

[5] A. Moitra, R. M. Mattheyses, V. A. DiDomizio, L. J. Hoebel, R. J. Szczerba, and B. Yamrom, "Multivehicle reconnaissance route and sensor planning," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 3, pp. 799–812, July 2003.

[6] H. Jachmann, "Evaluation of four survey methods for estimating elephant densities," *African J. of Ecology*, vol. 29, no. 3, pp. 188–195, 1991.

[7] J. Everaerts, "The use of unmanned aerial vehicles (uavs) for remote sensing and mapping," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 37, pp. 1187–1192, 2008.

[8] K. Sundar and S. Rathinam, "Algorithms for routing an unmanned aerial vehicle in the presence of refueling depots," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 1, pp. 287–294, 2014.

[9] T. E. Kannon, S. G. Nurre, B. J. Lunday, and R. R. Hill, "The aircraft routing problem with refueling," *Optimization Letters*, pp. 1–16, 2014.

[10] K. Sundar, S. Venkatachalam, and S. Rathinam, "Formulations and algorithms for the multiple depot, fuel-constrained, multiple vehicle routing problem," in *American Control Conference*, July 2016, pp. 6489–6494.

[11] S. Khuller, A. Malekian, and J. Mestre, "To fill or not to fill: The gas station problem," *ACM Transactions on Algorithms (TALG)*, vol. 7, no. 3, p. 36, 2011.

[12] G. Oh, Y. Kim, J. Ahn, and H.-L. Choi, "Task allocation of multiple uavs for cooperative parcel delivery," in *Advances in Aerospace Guidance, Navigation and Control*. Springer, 2018, pp. 443–454.

[13] M. Thammawichai, S. P. Baliyarasimhuni, E. C. Kerrigan, and J. B. Sousa, "Optimizing communication and computation for multi-uav information gathering applications," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, no. 2, pp. 601–615, April 2018.

[14] Y. Wang, T. Kirubarajan, R. Tharmarasa, R. Jassemi-Zargani, and N. Kashyap, "Multiperiod coverage path planning and scheduling for airborne surveillance," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, no. 5, pp. 2257–2273, Oct 2018.

[15] K. Sundar, S. Venkatachalam, and S. Rathinam, "Analysis of mixed-integer linear programming formulations for a fuel-constrained multiple vehicle routing problem," *Unmanned Systems*, 2017.

[16] S. G. Manyam, S. Rasmussen, D. W. Casbeer, K. Kalyanam, and S. Manickam, "Multi-uav routing for persistent intelligence surveillance reconnaissance missions," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2017, pp. 573–580.

[17] D. Levy, K. Sundar, and S. Rathinam, "Heuristics for routing heterogeneous unmanned vehicles with fuel constraints," *Mathematical Problems in Engineering*, vol. Article ID 131450, 2014.

[18] S. Erdoğan and E. Miller-Hooks, "A green vehicle routing problem," *Transportation Research Part E: Logistics and Transportation Review*, vol. 48, no. 1, pp. 100–114, 2012.

[19] S. Funke, A. Nusser, and S. Storandt, "Placement of loading stations for electric vehicles: No detours necessary!" *Journal of Artificial Intelligence Research*, vol. 53, pp. 633–658, 2015.

[20] J. Kim, B. D. Song, and J. R. Morrison, "On the scheduling of systems of uavs and fuel service stations for long-term mission fulfillment," *Journal of Intelligent & Robotic Systems*, vol. 70, no. 1-4, pp. 347–359, 2013.

[21] K. Kalyanam, M. Pachter, and D. Casbeer, "Average reward dynamic programming applied to a persistent visitation and data delivery problem," in *ASME 2017 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers, 2017, pp. V003T39A002–V003T39A002.

[22] N. Mathew, S. Smith, and S. Waslander, "Multirobot rendezvous planning for recharging in persistent tasks," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 128–142, Feb 2015.

[23] K. Yu, A. K. Budhiraja, S. Buebel, and P. Tokekar, "Algorithms and experiments on routing of unmanned aerial vehicles with mobile recharging stations," *Journal of Field Robotics*, vol. 36, no. 3, pp. 602–616, 2019.

[24] A. Gautam, P. Sujit, and S. Saripalli, "A survey of autonomous landing techniques for uavs," in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*. IEEE, 2014, pp. 1210–1218.

[25] ——, "Autonomous quadrotor landing using vision and pursuit guidance," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 10 501 – 10 506, 2017, 20th IFAC World Congress. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896317326204

[26] A. Rucco, P. Sujit, A. P. Aguiar, J. B. De Sousa, and F. L. Pereira, "Optimal rendezvous trajectory for unmanned aerial-ground vehicles," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, no. 2, pp. 834–847, 2018.

[27] K. A. Swieringa, C. B. Hanson, J. R. Richardson, J. D. White, Z. Hasan, E. Qian, and A. Girard, "Autonomous battery swapping system for small-scale helicopters," in *IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 3335–3340.

[28] K. A. Suzuki, P. Kemper Filho, and J. R. Morrison, "Automatic battery replacement system for uavs: Analysis and design," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1-4, pp. 563–586, 2012.

[29] P. Maini and P. Sujit, "On cooperation between a fuel constrained uav and a refueling ugv for large scale mapping applications," in *International Conference on Unmanned Aircraft Systems*, June 2015, pp. 1370–1377.

[30] M. Grötschel, M. W. Padberg *et al.*, "Polyhedral theory," *The Traveling Salesman Problem*, pp. 251–305, 1985.

[31] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations research*, vol. 21, no. 2, pp. 498–516, 1973.

[32] P. van Blyenburgh, "Unmanned aircraft systems: The global perspective," *UVS International, Paris, France*, 2007.

[33] "Experiment video," https://youtu.be/pilL0alZQ3I, 2018.

[34] S. G. Manyam, S. Rasmussen, D. W. Casbeer, K. Kalyanam, and S. Manickam, "Multi-uav routing for persistent intelligence surveillance & reconnaissance missions," in *International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2017, pp. 573–580.

[35] W. H. Chun and N. Papanikolopoulos, "Robot surveillance and security," in *Handbook of Robotics*. Springer, 2016, pp. 1605–1626.

[36] R. R. Murphy, *Disaster robotics*. MIT press, 2014.

[37] H. Ghazzai, A. Kadri, M. Ben Ghorbel, and H. Menouar, "Optimal sequential and parallel uav scheduling for multi-event applications," in *IEEE Vehicular Technology Conference*, June 2018, pp. 1–6.

[38] K. Máthé and L. Buşoniu, "Vision and control for uavs: A survey of general methods and of inexpensive platforms for infrastructure inspection," *Sensors*, vol. 15, no. 7, pp. 14 887–14 916, 2015.

[39] N. Lasla, H. Ghazzai, H. Menouar, and Y. Massoud, "Exploiting land transport to improve the uav's performances for longer mission coverage in smart cities," in *IEEE Vehicular Technology Conference*, April 2019, pp. 1–7.

[40] E. Resendiz, J. M. Hart, and N. Ahuja, "Automated visual inspection of railroad tracks," *Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 751–760, 2013.

[41] J. Katrasnik, F. Pernus, and B. Likar, "A survey of mobile robots for distribution power line inspection," *Transactions on Power Delivery*, vol. 25, no. 1, pp. 485–493, 2010.

[42] A. Ryan, X. Xiao, S. Rathinam, J. Tisdale, M. Zennaro, D. Caveney, R. Sengupta, and J. K. Hedrick, "A modular software infrastructure for distributed control of collaborating uavs," in *Guidance, Navigation, and Control Conference*. AIAA, 2006, p. 6455.

[43] S. Rathinam, P. Almeida, Z. Kim, S. Jackson, A. Tinka, W. Grossman, and R. Sengupta, "Autonomous searching and tracking of a river using an uav," in *American Control Conference (ACC)*. IEEE, 2007, pp. 359–364.

[44] S. Rathinam, Z. W. Kim, and R. Sengupta, "Vision-based monitoring of locally linear structures using an unmanned aerial vehicle," *Journal of Infrastructure Systems*, vol. 14, no. 1, pp. 52–63, 2008.

[45] R. W. Beard, T. W. McLain, D. B. Nelson, D. Kingston, and D. Johanson, "Decentralized cooperative aerial surveillance using fixed-wing miniature uavs," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1306–1324, 2006.

[46] P. Maini, K. Sundar, M. Singh, S. Rathinam, and P. Sujit, "Cooperative aerial–ground vehicle route planning with fuel constraints for coverage applications," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 6, pp. 3016–3028, 2019.

[47] S. G. Manyam, K. Sundar, and D. W. Casbeer, "Cooperative routing for an air-ground vehicle team–exact algorithm, transformation method, and heuristics," *IEEE Transactions on Automation Science and Engineering*, pp. 1–11, 2019.

[48] N. Mathew, S. Smith, and S. Waslander, "Multirobot rendezvous planning for recharging in persistent tasks," *Transactions on Robotics*, vol. 31, no. 1, pp. 128–142, Feb 2015.

[49] P. Tokekar and V. Kumar, "Visibility-based persistent monitoring with robot teams," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 3387–3394.

[50] S. Alamdari, E. Fata, and S. L. Smith, "Persistent monitoring in discrete environments: Minimizing the maximum weighted latency between observations," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 138–154, 2014.

[51] P. Maini and P. B. Sujit, "On cooperation between a fuel constrained uav and a refueling ugv for large scale mapping applications," in *International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, June 2015, pp. 1370–1377.

[52] F. Ropero, P. Muñoz, and M. D. R-Moreno, "Terra: A path planning algorithm for cooperative ugv–uav exploration," *Engineering Applications of Artificial Intelligence*, vol. 78, pp. 260–272, 2019.

[53] W. Chin and S. Ntafos, "Optimum watchman routes," in *Symposium on Computational Geometry*, ser. SCG '86. New York, NY, USA: ACM, 1986, pp. 24–33.

[54] X. Tan, "Fast computation of shortest watchman routes in simple polygons," *Information Processing Letters*, vol. 77, no. 1, pp. 27–33, 2001.

[55] J. S. Mitchell, "Approximating watchman routes," in *Symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2013, pp. 844–855.

[56] S. Carlsson, H. Jonsson, and B. J. Nilsson, "Finding the shortest watchman route in a simple polygon," *Discrete & Computational Geometry*, vol. 22, no. 3, pp. 377–402, Oct 1999.

[57] E. Packer, "Computing multiple watchman routes," *Experimental Algorithms*, pp. 114–128, 2008.

[58] J. Faigl, "Approximate solution of the multiple watchman routes problem with restricted visibility range," *Transactions on Neural Networks*, vol. 21, no. 10, pp. 1668–1679, Oct 2010.

[59] B. J. Nilsson and D. Wood, "Optimum watchmen routes in spiral polygons," in *Canadian Conference on Computational Geometry*, 1990, pp. 269–272.

[60] S. Carlsson, B. J. Nilsson, and S. Ntafos, "Optimum guard covers and m-watchmen routes for restricted polygons," in *Workshop on Algorithms and Data Structures*. Springer, 1991, pp. 367–378.

[61] K. Kalyanam, S. Manyam, A. Von Moll, D. Casbeer, and M. Pachter, "Scalable and exact milp methods for uav persistent visitation problem," in *IEEE Conference on Control Technology and Applications (CCTA)*, Aug 2018, pp. 337–342.

[62] S. K. K. Hari, S. Rathinam, S. Darbha, K. Kalyanam, S. G. Manyam, and D. Casbeer, "The generalized persistent monitoring problem," in *American Control Conference*, July 2019, pp. 2783–2788.

[63] N. Mathew, S. L. Smith, and S. L. Waslander, "Planning paths for package delivery in heterogeneous multirobot teams," *Transactions on Automation Science and Engineering*, vol. 12, no. 4, pp. 1298–1308, 2015.

[64] S. Carlsson and B. J. Nilsson, "Computing vision points in polygons," *Algorithmica*, vol. 24, pp. 50–75, 1999.

[65] Z. Drezner and H. W. Hamacher, *Facility location: applications and theory*. Springer Science & Business Media, 2001.

[66] "Experiment video," https://youtu.be/f0SCrqY9fbA, 2019.

[67] P. Tokekar, J. Vander Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic uav and ugv system for precision agriculture," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1498–1511, 2016.

[68] J. F. AraÃžjo, P. B. Sujit, and J. B. Sousa, "Multiple uav area decomposition and coverage," in *Symposium on Computational Intelligence for Security and Defense Applications*, 2013, pp. 30–37.

[69] P. Maini, K. Sundar, M. Singh, S. Rathinam, and P. B. Sujit, "Cooperative aerial-ground vehicle route planning with fuel constraints for coverage applications," *IEEE Transactions on Aerospace and Electronic Systems*, 2019.

[70] K. J. Obermeyer, P. Oberlin, and S. Darbha, "Sampling-based path planning for a visual reconnaissance unmanned air vehicle," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 2, pp. 619–631, 2012.

[71] G. Morgenthal and N. Hallermann, "Quality assessment of unmanned aerial vehicle (uav) based visual inspection of structures," *Advances in Structural Engineering*, vol. 17, no. 3, pp. 289–302, 2014.

[72] M. Erdelj, E. Natalizio, K. R. Chowdhury, and I. F. Akyildiz, "Help from the sky: Leveraging uavs for disaster management," *IEEE Pervasive Computing*, no. 1, pp. 24–32, 2017.

[73] P. Sujit, D. Kingston, and R. Beard, "Cooperative forest fire monitoring using multiple uavs," in *Conference on Decision and Control*. IEEE, 2007, pp. 4875–4880.

[74] J. S. Mitchell, "Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric tsp, k-mst, and related problems," *SIAM Journal on Computing*, vol. 28, no. 4, pp. 1298–1309, 1999.

[75] V. K. Shetty, M. Sudit, and R. Nagi, "Priority-based assignment and routing of a fleet of unmanned combat aerial vehicles," *Computers and Operations Research*, vol. 35, no. 6, pp. 1813 – 1828, 2008.

[76] M. Schwager, B. J. Julian, M. Angermann, and D. Rus, "Eyes in the sky: Decentralized control for the deployment of robotic camera networks," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1541–1561, 2011.

[77] A. Xu, C. Viriyasuthee, and I. Rekleitis, "Optimal complete terrain coverage using an unmanned aerial vehicle," in *International Conference on Robotics and Automation*, 2011, pp. 2513–2519.

[78] N. Mathew, S. L. Smith, and S. L. Waslander, "Multirobot rendezvous planning for recharging in persistent tasks," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 128–142, Feb 2015.

[79] E. Moet, M. Van Kreveld, and R. Van Oostrum, "Region intervisibility in terrains," *International Journal of Computational Geometry & Applications*, vol. 17, no. 04, pp. 331–347, 2007. [Online]. Available: https://doi.org/10.1142/S0218195907002367

[80] L. Floriani and P. Magillo, "Algorithms for visibility computation on terrains: A survey," *Environment and Planning B: Planning and Design*, vol. 30, no. 5, pp. 709–728, 2003.

[81] P. K. Agarwal, S. Bereg, O. Daescu, H. Kaplan, S. Ntafos, and B. Zhu, "Guarding a terrain by two watchtowers," in *Symposium on Computational Geometry*, 2005.

[82] A. Efrat, M. Nikkilä, and V. Polishchuk, "Sweeping a terrain by collaborative aerial vehicles," in *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, 2013, pp. 4–13.

[83] Y. Choi, Y. Choi, S. Briceno, and D. N. Mavris, "Three-dimensional uas trajectory optimization for remote sensing in an irregular terrain environment," in *International Conference on Unmanned Aircraft Systems*, June 2018, pp. 1101–1108.

[84] P. A. Plonski and V. Isler, "Approximation algorithms for tours of height-varying view cones," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 224–235, 2019.

[85] N. Stefas, P. A. Plonski, and V. Isler, "Approximation algorithms for tours of orientation-varying view cones," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–6.

[86] J.-R. Sack and J. Urrutia, *Handbook of computational geometry*. Elsevier, 1999.

[87] S. L. Smith and F. Imeson, "GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem," *Computers & Operations Research*, vol. 87, pp. 1–19, 2017.

[88] A. J. Stewart, "Fast horizon computation at all points of a terrain with visibility and shading applications," *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 1, pp. 82–93, Jan 1998.

[89] A. Dumitrescu and J. S. Mitchell, "Approximation algorithms for tsp with neighborhoods in the plane," *Journal of Algorithms*, vol. 48, no. 1, pp. 135–159, 2003.

[90] J. S. Mitchell, "A constant-factor approximation algorithm for tsp with pairwise-disjoint connected neighborhoods in the plane," in *Proceedings of the annual symposium on Computational geometry*. ACM, 2010, pp. 183–191.

[91] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group, Tech. Rep., 1976.

[92] O. Tekdas, D. Bhadauria, and V. Isler, "Efficient data collection from wireless nodes under the two-ring communication model," *International Journal of Robotics Research*, vol. 31, no. 6, pp. 774–784, 2012.

[93] P. Tokekar, A. K. Budhiraja, and V. Kumar, "Algorithms for visibility-based monitoring with robot teams," *arXiv preprint arXiv:1612.03246*, 2016.

[94] "Experiment video," https://youtu.be/84whBqdSCCs, 2020.

[95] P. B. Sujit, B. P. Hudzietz, and S. Saripalli, "Route planning for angle constrained terrain mapping using an unmanned aerial vehicle," *Journal of Intelligent & Robotic Systems*, vol. 69, no. 1, pp. 273–283, Jan 2013. [Online]. Available: https://doi.org/10.1007/s10846-012-9729-y

[96] H. Choi, "Time-optimal paths for a dubins car and dubins airplane with a unidirectional turning constraint," Ph.D. dissertation, The University of Michigan, 2014.

[97] G. Yang and V. Kapila, "Optimal path planning for unmanned air vehicles with kinematic and tactical constraints," in *Decision and Control (CDC), Conference on*, vol. 2. IEEE, 2002, pp. 1301–1306.

[98] T. G. McGee and J. K. Hedrick, "Optimal path planning with a kinematic airplane model," *Journal of guidance, control, and dynamics*, vol. 30, no. 2, pp. 629–633, 2007.

[99] M. Shanmugavel, A. Tsourdos, B. A. White, and R. Żbikowski, "Differential geometric path planning of multiple uavs," *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 5, pp. 620–632, 2007.

[100] T. G. McGee and J. K. Hedrick, "Path planning and control for multiple point surveillance by an unmanned aircraft in wind," in *American Control Conference (ACC)*. IEEE, 2006, pp. 4261–4266.

[101] H. Chitsaz and S. M. LaValle, "Time-optimal paths for a dubins airplane," in *Decision and Control (CDC), Conference on*. IEEE, 2007, pp. 2379–2384.

[102] P. Maini and P. B. Sujit, "Path planning for a uav with kinematic constraints in the presence of polygonal obstacles," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2016, pp. 62–67.

[103] Y. Lin and S. Saripalli, "Path planning using 3d dubins curve for unmanned aerial vehicles," in *Unmanned Aircraft Systems (ICUAS), International Conference on*. IEEE, 2014, pp. 296–304.

[104] A. Bhatia and E. Frazzoli, "Decentralized algorithm for minimum-time rendezvous of dubins vehicles," in *2008 American Control Conference*, June 2008, pp. 1343–1349.

[105] S. G. Manyam, S. Rathinam, D. Casbeer, and E. Garcia, "Tightly bounding the shortest dubins paths through a sequence of points," *Journal of Intelligent & Robotic Systems*, Jan 2017. [Online]. Available: https://doi.org/10.1007/s10846-016-0459-4

[106] J.-H. Lee, O. Cheong, W.-C. Kwon, S. Y. Shin, and K.-Y. Chwa, "Approximation of curvature-constrained shortest paths through a sequence of points," in *European Symposium on Algorithms*. Springer, 2000, pp. 314–325.

[107] X. Goaoc, H.-S. Kim, and S. Lazard, "Bounded-curvature shortest paths through a sequence of points using convex optimization," *SIAM Journal on Computing*, vol. 42, no. 2, pp. 662–684, 2013.

[108] S. Rathinam, R. Sengupta, and S. Darbha, "A resource allocation algorithm for multivehicle systems with nonholonomic constraints," *IEEE Transactions on Automation Science and Engineering*, vol. 4, no. 1, pp. 98–104, 2007.

[109] J. Le Ny, E. Feron, and E. Frazzoli, "On the dubins traveling salesman problem," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 265–270, 2012.

[110] Z. Tang and U. Ozguner, "Motion planning for multitarget surveillance with mobile sensor agents," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 898–908, 2005.

[111] K. Savla, E. Frazzoli, and F. Bullo, "Traveling salesperson problems for the dubins vehicle," *IEEE Transactions on Automatic Control*, vol. 53, no. 6, pp. 1378–1391, 2008.

[112] S. Manyam and S. Rathinam, "A tight lower bounding procedure for the dubins traveling salesman problem," in *International Symposium on Mathematical Programming.*, 2015.

[113] P. Oberlin, S. Rathinam, and S. Darbha, "Today's traveling salesman problem," *IEEE robotics & automation magazine*, vol. 17, no. 4, pp. 70–77, 2010.

[114] I. Cohen, C. Epstein, and T. Shima, "On the discretized dubins traveling salesman problem," *IISE Transactions*, vol. 49, no. 2, pp. 238–254, 2017.

[115] E. Forsmo, E. GrÃÿtli, T. Fossen, and T. Johansen, "Optimal search mission with unmanned aerial vehicles using mixed integer linear programming," in *Unmanned Aircraft Systems (ICUAS), International Conference on*. IEEE, 2013, pp. 253–259.

[116] E. Frazzoli, M. Dahleh, E. Feron *et al.*, "Real-time motion planning for agile autonomous vehicles," in *American Control Conference (ACC)*, vol. 1. IEEE, 2001, pp. 43–49.

[117] F. Yan, Y.-S. Liu, and J.-Z. Xiao, "Path planning in complex 3d environments using a probabilistic roadmap method," *International Journal of Automation and Computing*, vol. 10, no. 6, pp. 525–533, 2013.

[118] S. Bortoff, "Path planning for uavs," in *American Control Conference (ACC)*, vol. 1, no. 6. IEEE, 2000, pp. 364–368.

[119] F. Schøler, A. la Cour-Harbo, and M. Bisgaard, "Configuration space and visibility graph generation from geometric workspaces for uavs," in *AIAA Guidance, Navigation, and Control Conference*, 2011, p. 6416.

[120] S. Scherer, S. Singh, L. Chamberlain, and M. Elgersma, "Flying fast and low among obstacles: Methodology and experiments," *The International Journal of Robotics Research*, vol. 27, no. 5, pp. 549–574, 2008.

[121] D. Jung and P. Tsiotras, "Multiresolution on-line path planning for small unmanned aerial vehicles," in *American Control Conference (ACC)*. IEEE, 2008, pp. 2744–2749.

[122] J. Reif and H. Wang, "The complexity of the two dimensional curvature-constrained shortest-path problem," in *Robotics: The algorithmic Perspective: Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 1998, pp. 49–57.

[123] J. Backer and D. Kirkpatrick, "A complete approximation algorithm for shortest bounded-curvature paths," in *Algorithms and Computation*. Springer, 2008, pp. 628–643.

[124] ——, "Finding curvature-constrained paths that avoid polygonal obstacles," in *Symposium on Computational geometry (SCG)*. ACM, 2007, pp. 66–73.

[125] P. K. Agarwal, P. Raghavan, and H. Tamaki, "Motion planning for a steering-constrained robot through moderate obstacles," in *Symposium on Theory of Computing (STOC)*. ACM, 1995, pp. 343–352.

[126] S. Eriksson-Bique, D. Kirkpatrick, and V. Polishchuk, "On polygonal paths with bounded discrete-curvature: The inflection-free case," in *Discrete and Computational Geometry and Graphs*. Springer, 2014, pp. 44–64.

[127] T. G. McGee and J. K. Hedrick, "Path planning and control for multiple point surveillance by an unmanned aircraft in wind," in *American Control Conference (ACC)*. IEEE, 2006, pp. 4261–4266.

[128] M. De Berg, M. Van Kreveld, M. Overmars, and O. C. Schwarzkopf, *Computational Geometry*. Springer, 2000.

[129] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[130] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.