

Visibility-Based Persistent Monitoring Of Piece-wise Linear Features On A Terrain Using Multiple Aerial And Ground Robots

Parikshit Maini, Pratap Tokekare and P.B. Sujit

Abstract—Persistent monitoring on terrains using mobile robotic sensors requires coordinated planning. Terrain features add visibility obstacles and limited fuel capacity of aerial robots leads to range restrictions, that make the problem challenging. We address 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 previous works on visibility in specific classes of polygons and fuel-constrained routing, we develop a discrete representation of the problem that allows the design and application of discrete optimization techniques to find optimal solutions. We develop a Mixed Integer Linear Programming formulation and discuss a branch-and-cut implementation to compute exact solutions. We also develop a construction heuristic based on the idea of competitive construction of robot paths using a step-increment strategy. We report results from computational simulations and illustrate proof-of-concept using experiments on real robots.

Note to practitioners—This paper is motivated by the need to perform persistent monitoring in applications like border patrol, perimeter surveillance, etc. Unmanned aerial and ground robots can be used to perform these activities uninterrupted. However, aerial robots have limited fuel capacity and need periodic refueling. Hence, the number of refueling depots and their placement within the environment also affects the monitoring task. Also due to terrain variation, robots are subject to limited visibility. Therefore, we need to consider refueling constraints and terrain visibility aspects while planning optimal routes for the robots to perform visual monitoring. In this article, we present a general optimal routing formulation to compute exact solutions. We also present a fast heuristic for real time applications that produces feasible solutions. The algorithms are validated in simulations. We also show a proof-of-concept using experiments in limited outdoor settings.

I. INTRODUCTION

Visual monitoring using autonomous mobile robots has been transformative in applications like Intelligence, Surveillance and Reconnaissance operations (ISR) [1], [2], disaster management [3], [4] and structural monitoring [5], [6]. In this context, the utility of cooperative aerial and ground robot systems for monitoring tasks is an active area of research and has

Parikshit Maini is with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, USA. pmaini@umn.edu

P. Tokekare is with the Department of Computer Science at the University of Maryland, College Park, USA. ttokekare@umd.edu

P.B. Sujit is with the Department of Electrical Engineering and Computer Science at the Indian Institute of Science Education and Research Bhopal (IISERB), Bhopal – 462066, India. sujit@iiserb.ac.in

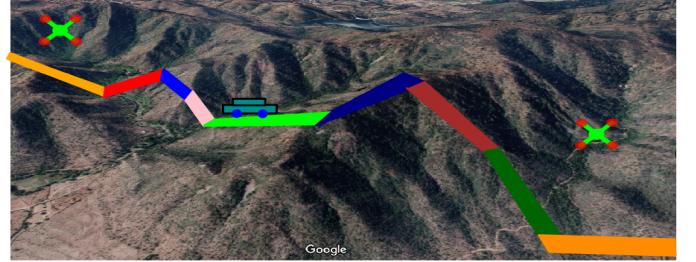


Fig. 1: An example scenario of a piece-wise linear feature (colored segments) on a terrain where a team of robots consisting of two aerial vehicles and one ground vehicle are monitoring.

garnered a lot of interest from the research community over the last decade. In this paper, we study the use of a cooperative heterogeneous robot system for persistent monitoring on a terrain. Applications like, rail-track monitoring [5], [7], power-line inspection [8], border & highway patrol [9] and river monitoring [10], 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. We model terrain features like roads, borders and pipelines as piece-wise linear ([10] and [11] also use similar models) and address the problem of persistent monitoring of such features using a cooperative system of aerial and ground robots (Figure 1). This class of problems are combinatorial in nature and are computationally intensive.

There exists literature that addresses cooperative routing for mobile robots [12]–[15], persistent monitoring [1], [13], [14], [16]–[20] 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. The problem includes negotiating visibility and fuel restrictions, joint 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. We model the mission cost 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. We use the terms tour, route and sequence interchangeably throughout the rest of the text.

The rest of the paper is organized as follows. Section II reviews the related work and contextualizes the technical contributions of this work. Section III discusses preliminaries on terrain model, robot models, persistent monitoring and the placement of refueling depots. Section IV describes the application scenario and establishes the problem statement. Section V develops the MILP and heuristic solution methods. Section VI describes the computational simulations including setup, instance generation, examples and results. Section VII describes the experiments used to illustrate the proof-of-concept. Section VIII concludes the paper and identifies future research directions.

II. RELATED WORK

Determining tours for a robot to fully observe a given space is popularly known as the watchman routing problem (WRP) [21], [22]. WRP is a well-studied problem. The simple version of WRP in a polygon without holes may be solved in polynomial time [22]; for polygon with holes, Mitchell [23] 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 [24], the goal is to compute tours for n robots (watchmen) to cover the environment. Carlsson et al. [24] 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 [25] and self-organizing map heuristics [26] have been developed. There have also been efforts to design algorithms for restricted polygon domains: spiral polygons [27], histograms [28], and street polygons [17], [24]. Exact [27], [28] and approximate [17] 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. In this work, we address a persistent version of n -WRP to monitor a piecewise linear feature on a terrain and extend the current state of the art by admitting the use of heterogeneous 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 [18], [29]–[31]. Alamdari et al. [18] 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. [1] address the homogeneous multi robot persistent monitoring problem in the context of a data delivery application with constraints on data freshness. Mathew et al. [32] solve a scheduling problem for mobile recharging stations to rendezvous with aerial robots for a persistent monitoring application. Ghazzai et al. [4] 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 [13], [14], [19], [20], [33]. Lasla et al. [6] 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. References [13], [14], [32], [33] 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. [34] addresses the problem of placement of recharging depots for electric automobiles ensuring they can travel long distances with minimal detours for recharging.

Contributions: In this work, we address 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:

- We address the persistent monitoring problem for piecewise linear features on altitude varying terrains using a heterogeneous set of ground and aerial mobile robotic sensors.
- Our 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.
- We develop a Mixed Integer Linear Programming (MILP) formulation for the persistent monitoring problem and discuss a branch-and-cut based implementation.
- We develop a construction heuristic called δ -search, based on competitive construction of robot paths using a step-increment strategy.
- We validate and compare our methods using large scale computational simulations.
- We also conduct outdoor experiments using multiple aerial and multiple ground robots to establish proof-of-concept.

A preliminary version of this paper was presented in [35] and [36]. This paper includes the framework from the preliminary version and generalizes it. In this paper, we extend the framework using a generalized MILP formulation for the monitoring problem, develop a construction heuristic (δ -search), and experimentally demonstrate using 2 UAVs and 2 ground rovers.

III. PRELIMINARIES

We discuss some preliminary ideas needed to develop our solution methods in this section. Specifically, we formalize the visibility-based model of the terrain and establish the definitions used in the rest of the paper. We also describe the chain visibility property and the robot model used.

A. Terrain Model

We use the model and terminology for terrain features as defined in [35]. For completeness, we give an overview here. The environmental feature to be monitored is modeled as a 1.5 dimensional structure. As shown in Figure 2, a piecewise linear feature within a terraneous environment may be straightened out in one dimension to build a 1.5D model.

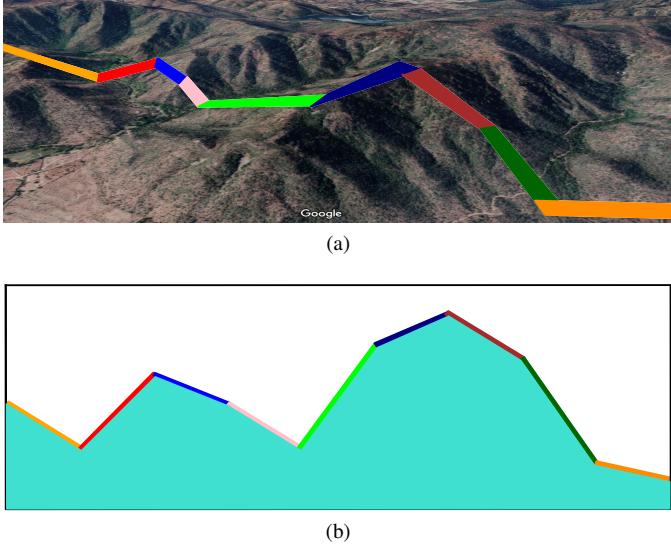


Fig. 2: A piece-wise linear feature within a terrain may be *straightened out* to build a 1.5D model. The individual edges of the feature are color coded in the two figures.

The 1.5D representation of a feature is *x*-monotone and is characterized by terrain points and reflex points (Figure 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 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 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 [17].*

Figure 3b shows the visibility polygon for a point x on the terrain. Examples of chain visible pairs includes street polygons and collapsed watchman routes [37], points on a 1.5D terrain and fixed altitude paths [17]. In our case, the fixed altitude path (line C in Figure 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 3c shows a visibility segment marked as a blue colored line on the terrain. When the context is clear from the text, we use ‘segment’ 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 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.

In the case of ground robots, we explicitly define the visibility region for a segment 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 3c mark the two viewpoints for the segment. Figure 3d shows the left and right view points for all visibility segments on the terrain.

B. Robot Model and System Assumptions

In this work, we are concerned with developing high level route planning algorithms and hence do not consider robot kinematics and dynamics. Further, the techniques developed here find application in offline planning within a static environment and do not consider communication between any of the robots during the monitoring task. We make the following system assumptions for ease of exposition:

- We assume the availability of a robust point-to-point navigation and obstacle avoidance system on each robot, including the availability of localization estimates.
- We assume ideal operating conditions and do not consider the effects of weather, wind or material properties of the terrain. 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

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

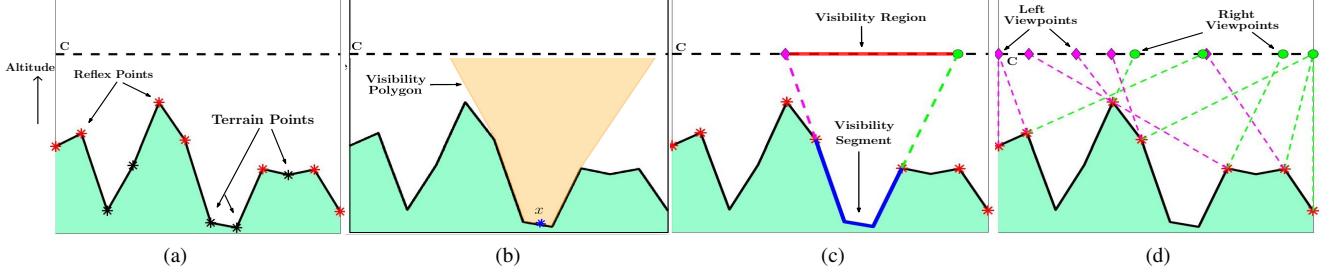


Fig. 3: (a) Terrain points, reflex points and a fixed altitude path, C, chain visible with the terrain. (b) Visibility polygon corresponding to a point x on the terrain. (c) A visibility segment and its corresponding visibility region. (d) Left and right viewpoints for all visibility segments.

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. We only consider fuel limitations 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.

C. 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, we restrict our search to repetitive sequences, wherein a certain finite sequence is repeated infinitely many times to build an infinite sequence. Alamdari et al. [18] 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 κ . In our 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 κ 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, we set $\kappa = 2$ in our 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 κ significantly increases the size of the search space (we discuss additional details on κ in Section V-A). To address this, we add a preprocessing step, 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.

D. 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 [30], decouple route planning and placement of refueling depots/sites [13], [19], [20], [32] or setup refueling depots/sites at a subset of points visited by the aerial robot [14].

This work significantly extends the state-of-the-art by jointly optimizing route planning for robots and the placement of refueling depots. We model the placement problem, on the lines of the classic facility location problem [38] where refueling depots must be opened at a subset of potential depot opening sites to ensure reachability of viewpoints. We develop a combined mathematical formulation to determine both robot paths and refueling depot placement while also accounting for associated costs in the optimization criteria.

IV. PROBLEM FORMULATION

Consider an environment \mathcal{E} as shown in Figure 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.

To this end, we design a cost function that 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. 1, minimizes the sum of maximum robot path cost and refueling depot opening costs .

$$\text{minimize} \left\{ \max_{i \in \mathcal{A}} \text{cost}(\Pi_i) + \sum_{d \in \mathbb{D}} \beta(w_d) \right\}, \quad (1)$$

where, $\text{cost}(\Pi_i)$ is the cost of the path Π_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, we do 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 our 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 1. We do not include this post-processing in this work.

V. SOLUTION APPROACH

We develop 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 δ -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. In this section, we first establish the notation used in the solution methods and then describe the two solution methods.

Let ν 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). Let $\mathcal{V}_c = \mathcal{V}_c^L \cup \mathcal{V}_c^R$, 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, we define 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 $V_c = \mathcal{V}_c \cup \mathbb{D}$.

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

A. MILP Formulation

We use the following sets of variables in the MILP formulation:

- $x_{ij}^{ckl_1 l_2}$, defined for each $c \in \mathcal{C}, k \in \mathcal{A}_c, i, j \in V_c$ and $l_1, l_2 \in \{1 \dots \kappa\}$. $x_{ij}^{ckl_1 l_2} = 1$, 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.
- $z_{ij}^{ckl_1 l_2}$, defined for each $c \in \mathcal{C}, k \in \mathcal{A}_c, i, j \in V_c$ and $l_1, l_2 \in \{1 \dots \kappa\}$. $z_{ij}^{ckl_1 l_2}$ 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} .

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 $V_c \setminus \mathcal{P}$. We now give an expanded form of the objective function described in Eq. 1 to be used in the MILP formulation.

$$\begin{aligned} \text{minimize} & \left\{ \left(\max_{c \in \mathcal{C}, k \in \mathcal{A}_c} \sum_{l_1 \in \{1 \dots \kappa\}} \sum_{l_2 \in \{1 \dots \kappa\}} \sum_{i \in V_c} \sum_{j \in V_c} \alpha_{ij}^c \cdot x_{ij}^{ckl_1 l_2} \right) \right. \\ & \left. + \sum_{d \in \mathbb{D}} \beta_d w_d \right\} \end{aligned} \quad (2)$$

The constraints used in the MILP are as follows:

Degree Constraints:

$$\sum_{l_2 \in \{1 \dots \kappa\}} \sum_{j \in V_c \setminus i} x_{ji}^{ckl_2 l_1} - \sum_{l_2 \in \{1 \dots \kappa\}} \sum_{j \in V_c \setminus i} x_{ij}^{ckl_2 l_2} = 0, \quad \forall c \in \mathcal{C}, k \in \mathcal{A}_c, i \in V_c, l_1 \in \{1 \dots \kappa\} \quad (3)$$

$$\sum_{l_2 \in \{1 \dots \kappa\}} \sum_{j \in V_c \setminus i} x_{ji}^{ckl_2 l_1} \leq 1, \quad \forall c \in \mathcal{C}, k \in \mathcal{A}_c, l_1 \in \{1 \dots \kappa\}, i \in V_c \quad (4)$$

$$\sum_{l_3 \in \{1 \dots \kappa\}} \sum_{j \in V_c \setminus i} x_{ji}^{ckl_3 l_1} - \sum_{l_3 \in \{1 \dots \kappa\}} \sum_{j \in V_c \setminus i} x_{ji}^{ckl_3 l_2} \geq 0, \quad \forall c \in \mathcal{C}, k \in \mathcal{A}_c, l_1, l_2 \in \{1 \dots \kappa\}, l_2 = l_1 + 1, i \in V_c \quad (5)$$

Visit Constraints:

$$y_i^{ck} - \sum_{l_2 \in \{1 \dots \kappa\}} \sum_{j \in V_c \setminus i} x_{ji}^{ckl_2 1} = 0, \quad \forall c \in \mathcal{C}, k \in \mathcal{A}_c, i \in V_c \quad (6)$$

$$y_d^{ck} - x_{jd}^{ckl_2 1} \geq 0, \quad \forall c \in \mathcal{C}, k \in \mathcal{A}_c, d \in \mathbb{D}, j \in V_c \setminus d, l_2 \in \{1 \dots \kappa\} \quad (7)$$

$$y_d^{ck} - \sum_{l_2 \in \{1 \dots \kappa\}} \sum_{j \in V_c \setminus d} x_{jd}^{ckl_2 1} \leq 0, \quad \forall c \in \mathcal{C}, k \in \mathcal{A}_c, d \in \mathbb{D} \quad (8)$$

$$w_d - y_d^{ck} \geq 0, \quad \forall c \in \mathcal{C}, k \in \mathcal{A}_c, d \in \mathbb{D} \quad (9)$$

$$w_d - \sum_{c \in \mathcal{C}} \sum_{k \in \mathcal{A}_c} y_d^{ck} \leq 0, \quad \forall d \in \mathbb{D} \quad (10)$$

Sub-tour Elimination Constraints:

$$\sum_{l_1 \in \{1 \dots \kappa\}} \sum_{l_2 \in \{1 \dots \kappa\}} \sum_{(i, j) \in \delta_c^+(\mathcal{P})} x_{ij}^{ckl_1 l_2} \geq 1 + \sum_{i \in \mathcal{P}} (y_i^{ck} - 1), \quad \forall c \in \mathcal{C}, k \in \mathcal{A}_c, \mathcal{P} \subseteq V_c \setminus s_c^k \quad (11)$$

Coverage Constraints:

$$\sum_{c \in \mathcal{C}} \sum_{k \in \mathcal{A}_c} \sum_{i \in \gamma_c(v)} y_i^{ck} \geq 1, \quad \forall v \in \nu \quad (12)$$

Refueling Constraints:

$$\begin{aligned} & \sum_{l_2 \in \{1 \dots \kappa\}} \sum_{j \in V_c} z_{ij}^{ckl_1 l_2} - \sum_{l_2 \in \{1 \dots \kappa\}} \sum_{j \in V_c} z_{ji}^{ckl_2 l_1} \\ &= \sum_{l_2 \in \{1 \dots \kappa\}} \sum_{j \in V_c} \alpha_{ij}^c \cdot x_{ij}^{ckl_1 l_2}, \\ & \forall c \in \mathcal{C} \setminus c_0, k \in \mathcal{A}_c, i \in \mathcal{V}_c, l_1 \in \{1 \dots \kappa\} \end{aligned} \quad (13)$$

$$z_{di}^{ckl_1 l_2} = \alpha_{di}^c \cdot x_{di}^{ckl_1 l_2}, \quad \forall c \in \mathcal{C} \setminus c_0, k \in \mathcal{A}_c, l_1, l_2 \in \{1 \dots \kappa\}, d \in \mathbb{D}, i \in V_c \quad (14)$$

$$z_{ij}^{ckl_1 l_2} \leq (U - \min_{d \in \mathbb{D}} \alpha_{jd}^c) \cdot x_{ij}^{ckl_1 l_2}, \quad \forall c \in \mathcal{C} \setminus c_0, k \in \mathcal{A}_c, l_1, l_2 \in \{1 \dots \kappa\}, i, j \in V_c \quad (15)$$

Variable Domain Constraints:

$$\begin{aligned} x_{ij}^{ckl_1 l_2} &\in \{0, 1\}, \\ &\forall c \in \mathcal{C}, k \in \mathcal{A}_c, l_1, l_2 \in \{1 \dots \kappa\}, i, j \in V_c \end{aligned} \quad (16)$$

$$y_i^{ck} \in \{0, 1\}, \quad \forall c \in \mathcal{C}, k \in \mathcal{A}_c, i \in V_c \quad (17)$$

$$w_d \in \{0, 1\}, \quad \forall d \in \mathbb{D} \quad (18)$$

$$\begin{aligned} z_{ij}^{ckl_1 l_2} &\in [0, U], \\ &\forall c \in \mathcal{C} \setminus c_0, k \in \mathcal{A}_c, l_1, l_2 \in \{1 \dots \kappa\}, i, j \in V_c \end{aligned} \quad (19)$$

The objective function (Eq 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 is a set of degree constraints that ensures that the result is a closed

Algorithm 1 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  $\exists_{l_1, l_2 \in \{1 \dots \kappa\}} x_{ij}^{clk_1 l_2} = 1$ 
    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. (11))
  
```

walk. Constraints 4 and 5 together ensure that no viewpoint is visited more than κ number of times. We note that, the number of variables and consequently the size of the search space increases significantly, with a small increase in the value of κ . Hence, it is to our advantage to restrict its value ($\kappa = 2$) as discussed in Section III-C. Constraints 6-8 populate visit variables for viewpoints and depots. Constraints 9 and 10 restrict the placement and number of refueling depots opened in the environment. Constraint 11 ensures that there does not exist a disconnected subtour unreachable from the starting point (s_c^k). Constraint 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 13 and 14 ensure fuel conservation at the viewpoints and when traveling out of a depot, respectively. Constraint 15 makes sure the aerial robot has enough fuel to reach the nearest depot. Constraints 16-19 specify the domain for decision variables used in the formulation.

Branch-and-Cut Algorithm

It may be observed that 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, we use a branch-and-cut framework that works as follows: we solve a relaxed version of the problem that does not include the sub-tour elimination constraints. When the solver obtains an integer feasible solution to this relaxed problem, we check if the feasible solution violates any of these constraints. If so, we add the violated constraints to the formulation and continue solving the problem. This process of adding constraints to the problem sequentially has been observed to be computationally efficient for many variants of TSP [39] and fuel constrained vehicle routing problems [13], [30].

The algorithm to identify violated constraints is called a separation algorithm. We present pseudocode for a separation algorithm in Algorithm 1. 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 $P \subseteq V_c \setminus \{s_c^k\}$, 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 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.

B. Construction Heuristic

Algorithm 2 δ -Search

- 1: initialize robot paths to starting locations
- 2: initialize segment coverage vector
- 3: **while** all segments are not covered **do**
- 4: **for all** robots **do**
- 5: compute- Δ -path
- 6: δ -increment = minimum cost Δ -path
- 7: update robot paths and coverage data
- 8: **return** robot paths

Owing to the computational hardness of HWPRPT, we cannot hope for an exact method like the MILP based branch-and-cut algorithm to scale well. In this light, we discuss the design of a computationally efficient heuristic that may be used to compute feasible solutions quickly. The construction heuristic, named as δ -search algorithm is based on competitive construction of robot paths using a step increment strategy. Algorithm 2 gives pseudo-code for δ -search algorithm. It has an iterative construction and incrementally builds robot paths until all segments are covered. The paths are competitively allowed a δ -increment in every iteration based on the value of a cost function. The cost function accounts for distance traveled (fuel consumed), new visibility segments covered, and refueling cost in the case of aerial robots.

Lemma 1. *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. \square

Lemma 2. *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. \square

The δ -search algorithm (Algorithm 2) 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 3-7) to incrementally build robot paths until all visibility segments are covered in combination by the robots. Within each iteration of the while loop, a Δ -path is computed for each robot using the algorithmic routine given in Algorithm 3. The Δ -path with the minimum cost is then marked as the δ -increment. Δ -paths are compared based on the following cost function:

$$\text{cost}(\Delta_i) = d_{\text{cost}}(i) + r_{\text{cost}}(i) + v_{rd}(i), \quad (20)$$

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

The δ -increment is then added to the corresponding robot's path. Segment coverage vector is updated to include new segments covered by the δ -increment. The while loop continues until all segments are covered. This ensures that δ -search algorithm terminates with a valid set of paths for each robot and ensures each visibility segment is covered.

We now describe, the Δ -path computation using the algorithmic routine given in Algorithm 3. It follows from the results in Lemmas 1 and 2, 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. We call these viewpoints as *end-viewpoints*. To compute the Δ -path, end-viewpoints on the robot's path, Π , are identified. Next, a neighbor (adjacent) viewpoint not currently on the robot's path, is computed for each end-viewpoint. A Δ -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 3 compute- Δ -path

```

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

Δ -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 4-12) in Algorithm 3 and works as follows. For ground robots, the Δ -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, we find the minimum cost traversal that includes a refueling visit to the nearest refueling depot. Any Δ -paths and the resultant robot paths, computed in this way satisfy fuel constraints. We restrict our search of Δ -paths 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 Δ -path is computed as the number of new visibility segments covered per unit length (line 13 in Algorithm 3). The path that maximizes the utility function is returned to the δ -search algorithm as the robot's Δ -path.

VI. COMPUTATIONAL SIMULATIONS

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

A. Simulation Setup

The MILP formulation requires significant computational resources. Therefore, the 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, whichever was reached first. Relative gap of a solution 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.

B. Instance Generation

The simulations were executed for a total of 2000 randomly generated instances. The values of various simulation parameter are summarized in Table I. 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 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 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

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 II

TABLE I: 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 II: Fuel capacities (U_k^c) for UAVs. UAV 1 and 2 were operated at altitude 1500 units. UAV 3 and 4 were operated at altitude 1800 units.

capacity (U_k^c) for the aerial robots were selected from five pre-generated set of values as given in Table II. 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 III-B. The ground robots incur an additional cost depending on the slope defined as uphill travel coefficient, $\bar{\gamma} = 0.35$, and a downhill travel coefficient, $\underline{\gamma} = -0.15$. Aerial robot cost computation uses the following parameters, $A_i, \forall i \in \{1 \dots (|\mathcal{C}| - 1)\}$ 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. The cost computation for ground robots and aerial robots is shown in Table III.

C. Illustrative Scenarios

Before considering the simulation results, we will show how the paths for the robotic sensors are generated by our planner to perform monitoring on a terrain. Figure 4 shows a few sample instances and the robot paths computed by the MILP solver. The robots available for monitoring are: 2 ground

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

TABLE III: Costs for different vehicles

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

TABLE IV: 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.

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, we add a small vertical offset 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 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 III. The route determined by the proposed MILP solution is shown in the Figure 4a. In this scenario, only UGV 1 and UAV 3 at altitude A2 are utilized. 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 2. In this instance, it includes the opening cost of two refueling depots (2000 units), and the maximum operating cost of a robot (9756 units), thus making a total of 11756 units.

Consider another scenario as shown in Figure 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 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 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 will also change.

D. Results

We will now analyze the effect of different fuel capacity and number of terrain points when using the MILP formulation and heuristics for the simulation instances described in Section V-A. Figure 5a shows the computation time taken by the solver as median values along with first and third quartiles. Figure 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

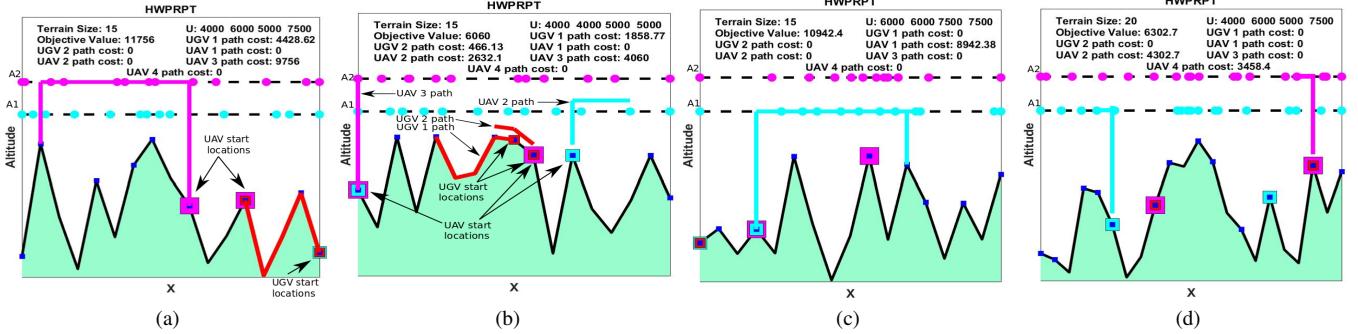


Fig. 4: Sample instances from the simulation set. The paths are color coded. Red color represents ground robots, cyan color represents robots flying at altitude 1 and magenta color represents aerial robots flying at altitude 2. Blue squares represent refueling depot opening sites. Starting location for 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.

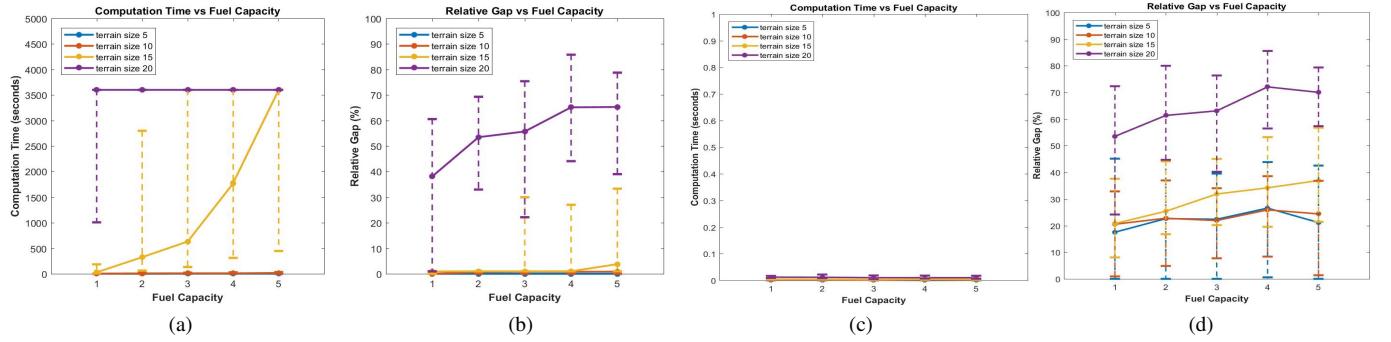


Fig. 5: Effect of change in fuel capacities (fuel capacity serial IDs corresponds to serial numbers in Table II) in terms of computation time and relative gap of the best solution computed by the solver within 3600 seconds for MILP approach in (a) and (b), and heuristics algorithm in (c) and (d).

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 5(c) and (d). Figure 5c shows the time taken to compute the solution against fuel capacity for the various instance sizes. We observe 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 5d, 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 IV. 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 5). This is attributed to the increase in search space with increasing values of fuel capacity.

VII. PROOF-OF-CONCEPT EXPERIMENTS

The solution methods proposed for coverage of a piecewise 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 6a was superimposed on the terrain shown in Figure 6b 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 ~ 188 meters. We computed paths for the robots using the MILP based solver (Figures 6d and 6c). The proof-of-concept demonstration 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

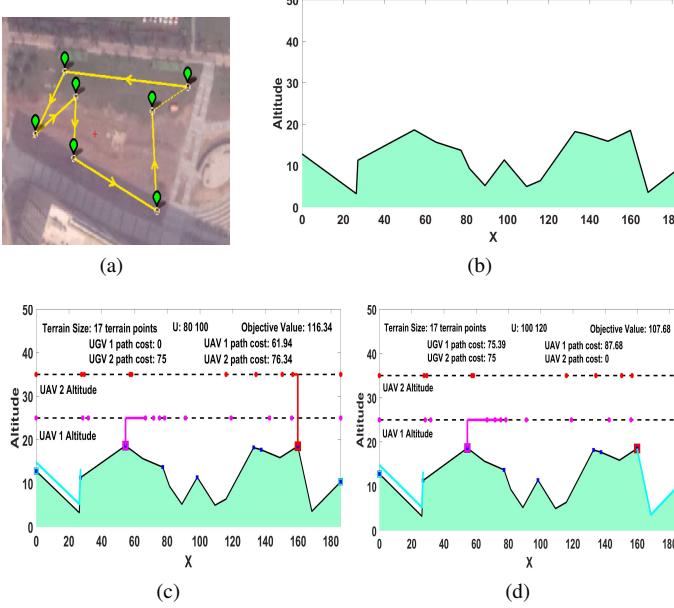


Fig. 6: (a) Simulated path-like feature used in the experiments. (b) 1.5 D terrain superimposed on the feature. (c) and (d) Robot paths generated by the MILP solver for experiment 1 and 2, respectively. Ground robot paths are shown in cyan color lines. Aerial robot #1 and #2 paths are shown in magenta 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.

geometry of the terrain and cost models for the robots, our methods can be used in real implementations.

Two 3DR Solo quad-copters were used as aerial robots in the experiments (Figure 7a). They were operated using the ArduCopter firmware and employed at 25 meters (id #1) and 35 meters (id #2) altitude respectively. 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 sets of experiments were conducted. The simulator visualization of robot paths generated for the experiment are shown in Figures 6c and 6d. 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 7b, 7c and 7d. 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 7e, 7f and 7g. The experiment video showing complete mission operation for both the experiments may be viewed in the multimedia attachment.

VIII. CONCLUSION

In this work, we address the persistent monitoring problem on piece-wise linear features on terrains. We have developed methods to 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, we design fast construction heuristic to compute feasible solutions. We evaluate and analyse our methods in simulation and also show experiment demonstrations, albeit using 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. Another interesting extension is to take uncertainty into account while planning the route for the vehicles. The uncertainty could be from the perceived information or vehicle failures or dynamically changing environment.

IX. ACKNOWLEDGMENTS

We thank G. Gupta and K. Yu for help with the preliminary work on this paper. This material is, in part, based upon work supported by the National Science Foundation under Grant No. 1566247. Parikshit Maini's research was partially supported by TCS PhD Fellowship and IIIT-Delhi. This work was partly supported by EPSRC GCRF Grant No. EP/P02839X/1.

REFERENCES

- [1] S. G. Manyam, S. Rasmussen, D. W. Casbeer, K. Kalyanam, and S. Manickam, "Multi-uav routing for persistent intelligence surveillance & reconnaissance missions," in *IEEE International Conference on Unmanned Aircraft Systems*, 2017, pp. 573–580.
- [2] W. H. Chun and N. Papanikopoulos, "Robot surveillance and security," in *Handbook of Robotics*. Springer, 2016, pp. 1605–1626.
- [3] R. R. Murphy, *Disaster robotics*. MIT press, 2014.
- [4] 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.
- [5] K. Máthé and L. Busoni, "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.
- [6] 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.
- [7] 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.
- [8] 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.

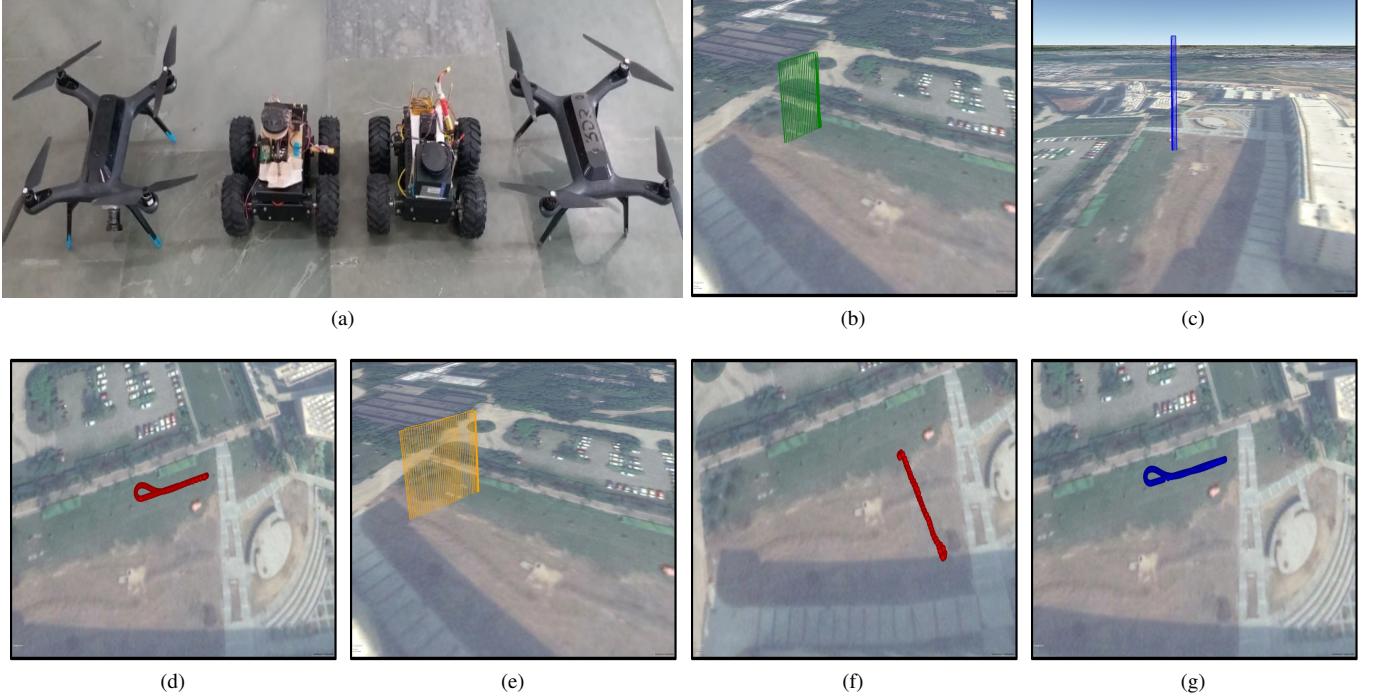


Fig. 7: Paths traversed by individual robots in the experiments. The paths were traced out using telemetry data collected during the robot mission. Visualizations were done in Google Earth Pro software. The paths also show the altitude profile for the aerial robots. Figures (a)-(c) show the paths traversed by UAV #1, UAV #2 and UGV #2 during experiment 1. Figures (d)-(f) show the paths traversed by UAV #1, UGV #1 and UGV #2 during experiment 2.

- [9] 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.
- [10] 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.
- [11] 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.
- [12] R. W. Beard, T. W. McLain, D. B. Nelson, D. Kingston, and D. Johanson, "Decentralized cooperative aerial surveillance using fixed-wing miniature uavs," *Proc. of the IEEE*, vol. 94, no. 7, pp. 1306–1324, 2006.
- [13] 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.
- [14] 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.
- [15] 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.
- [16] 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.
- [17] 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.
- [18] 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.
- [19] P. Maini and P. B. Sujit, "On cooperation between a fuel constrained uav and a refueling ugv for large scale mapping applications," in *IEEE International Conference on Unmanned Aircraft Systems*, June 2015, pp. 1370–1377.
- [20] 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.
- [21] 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.
- [22] X. Tan, "Fast computation of shortest watchman routes in simple polygons," *Information Processing Letters*, vol. 77, no. 1, pp. 27–33, 2001.
- [23] J. S. Mitchell, "Approximating watchman routes," in *Symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2013, pp. 844–855.
- [24] 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.
- [25] E. Packer, "Computing multiple watchman routes," *Experimental Algorithms*, pp. 114–128, 2008.
- [26] 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.
- [27] B. J. Nilsson and D. Wood, "Optimum watchmen routes in spiral polygons," in *Canadian Conference on Computational Geometry*, 1990, pp. 269–272.
- [28] 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.
- [29] 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.
- [30] K. Sundar and S. Rathinam, "Algorithms for routing an unmanned aerial vehicle in the presence of refueling depots," *Transactions on Automation Science and Engineering*, vol. 11, no. 1, pp. 287–294, 2014.
- [31] 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.

- [32] 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.
- [33] K. Yu, J. M. OKane, and P. Tokekar, "Coverage of an environment using energy-constrained unmanned aerial vehicles," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3259–3265.
- [34] 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.
- [35] P. Maini, G. Gupta, P. Tokekar, and P. B. Sujit, "Visibility-based monitoring of a path using a heterogeneous robot team," in *International Conference on Intelligent Robots and Systems, year=2018, pages=3765-3770, organization=IEEE*.
- [36] P. Maini, K. Yu, P. Sujit, and P. Tokekar, "Persistent monitoring with refueling on a terrain using a team of aerial and ground robots," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 8493–8498.
- [37] S. Carlsson and B. J. Nilsson, "Computing vision points in polygons," *Algorithmica*, vol. 24, pp. 50–75, 1999.
- [38] Z. Drezner and H. W. Hamacher, *Facility location: applications and theory*. Springer Science & Business Media, 2001.
- [39] M. Grötschel, M. W. Padberg *et al.*, "Polyhedral theory," *The Traveling Salesman Problem*, pp. 251–305, 1985.



Parikshit Maini is a post-doctoral researcher in the Department of Computer Science and Engineering at the University of Minnesota Twin-Cities, USA. He obtained his Ph.D. and M.Tech. degrees in computer science and engineering from the Indraprastha Institute of Information Technology Delhi, New Delhi, India, in 2020 and 2014, respectively. He received the B.Tech. degree in information technology from Guru Gobind Singh Indraprastha University, Delhi, India, in 2012. His research interests include motion planning, multi-robot cooperation and robot learning.



Pratap Tokekar is an Assistant Professor in the Department of Computer Science and UMIACS at the University of Maryland. Between 2015 and 2019, he was an Assistant Professor at the Department of Electrical and Computer Engineering at Virginia Tech. Previously, he was a Postdoctoral Researcher at the GRASP lab of University of Pennsylvania. He obtained his Ph.D. in Computer Science from the University of Minnesota in 2014 and Bachelor of Technology degree in Electronics and Telecommunication from College of Engineering Pune, India in 2008. He is a recipient of the NSF CAREER award (2020) and CISE Research Initiation Initiative award (2016). He serves as an Associate Editor for the IEEE Robotics & Automation Letters, IEEE Transactions of Automation Science & Engineering, and the ICRA and IROS Conference Editorial Board.



P.B. Sujit is an Associate Professor at the IISER Bhopal, Bhopal, India. He received the bachelors degree from Bangalore University, India, Masters degree from Visvesvaraya Technological University, India, and Ph.D. from the Indian Institute of Science, Bangalore, India. His research interests include unmanned aerial and underwater vehicles, multi-robot systems, and human-swarm interaction.