Visual Monitoring of Points of Interest on a 2.5D Terrain using a UAV with Limited Field-of-View Constraint

Parikshit Maini, Pratap Tokekar and P.B. Sujit

Abstract

We study the problem of visually monitoring a set of points on a 2.5D terrain using an Unmanned Aerial Vehicle (UAV) with a downward-facing camera. The goal is to find a tour of minimum length for the UAV to visually inspect all points of interest. Varying terrain and limited field-of-view of the camera restrict the visibility of the UAV and can create obstacles in the flight path, making the problem challenging. The problem is NP-Hard and generalizes the Traveling Salesperson Problem (TSP). We present several algorithms to solve this problem. Our main theoretical contribution is a constant-factor approximation algorithm (assuming fixed parameters for the terrain). We also present a practical algorithm that uses the solution to a Generalized TSP sub-instance. We benchmark the GTSP based algorithm using a branch-and-cut Integer Linear Programming formulation and find that the proposed algorithm scales to much larger instances and is computationally fast. We also show proof-of-concept using field deployment of a UAV to visually monitor points of interest in the environment.

Index Terms

UAV; route planning; aerial monitoring; visibility-based planning; path planning;

I. Introduction

Visual surveillance and monitoring is an important application area for Unmanned Aerial Vehicles (UAVs). Crop management [1], area coverage [2], [3], terrain mapping [4], structural inspection [5], and disaster management [6], [7] 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 visual monitoring missions. In this paper, we address the visual monitoring problem on terrains using a UAV while accounting for camera FOV and terrain imposed visibility restrictions (Figure 1).

Specifically, the input to our 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 the UAV tour length is minimized. The energy and time consumed by the UAV to complete a path is usually proportional to the length of the path and hence is a useful metric to evaluate

Parikshit Maini is with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis MN 55455, USA, Email: (pmaini@umn.edu)

Pratap Tokekar is with the Department of Computer Science, University of Maryland, College Park MD 20742, USA, Email: (tokekar@umd.edu)

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

(a) (b)

Fig. 1: (a) A instance of the terrain modeled as 2.5D TIN (triangular irregular network) 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.

the quality of paths computed by a planner. The flight altitude maybe selected as a user input parameter based on the required image resolution and local regulations. It maybe constant or variable as in the case of constant resolution visual monitoring. The flight altitude for monitoring a point of interest needs to be higher than the altitude of the point of interest but need not be above the highest point in the terrain. As a result, parts of the terrain above the fixed altitude plane also act as obstacles on the flight path for the UAV, which further complicates the problem.

A naive strategy is to visit a point directly above each point of interest which is equivalent to solving the Traveling Salesperson Problem (TSP). The TSP problem is NP-Hard [8]. However, this strategy does not take into account the FOV of the camera. The UAV may be able to view a point of interest even when not directly above it and may also observe multiple points of interest from the same location. Thus the problem results in a combinatorial optimization problem that requires solving for the locations to monitor the points of interest and to find a minimum cost tour that visits them. We present several algorithms to address this problem.

Paper Organization: The rest of the paper is organized as follows. We present a literature survey and discuss the contributions and relevance of this work in Section II. We formalize the UAV routing problem on terrains and discuss preliminaries in Section III. In Section IV, we present the main theoretical results of the paper. In Section V, a practical GTSP-based algorithm is developed for route planning for the UAV. Sections VI and VII discuss evaluation results in simulation and field demonstration, respectively. We make concluding remarks and identify future directions in Section VIII.

II. Related Work

The visual monitoring problem broadly falls in the category of aerial coverage using UAVs. Applications include target monitoring [9], surveillance [10], [11], and persistent monitoring [12]. There has been relatively limited work on aerial coverage with UAVs that take into account constraints introduced by terrains. Terrain visibility, however, is a well-studied problem in computational geometry [13] and graphics [14] literature. In particular, terrain guarding [13] and watchtower problems [15] 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 terrain, turn out to be NP-Hard. The mobile version of the problem is known as the Watchman Routing Problem (WRP) [16]. In the typical formulation, the environment is a 2D polygon and the robot has omni-directional vision with no FOV constraints [17], [18]. In our previous work [19], [20], we developed algorithms for visibility-based coverage of piece-wise linear features within

a terrain. We model such features using a 1.5D representation and exploit the geometric structure to develop a discretization of the visibility constraints. In this paper, we address the visual monitoring problem on 2.5D terrains where no such geometric structure is available. Efrat et al. [21] studied the visual monitoring problem for a 2.5D terrain under some specific conditions; in particular, they assume that the robots move in straight lines and the camera's FOV spans a vertical plane that can sweep a terrain.

The most closely related work to ours, is that of Choi et al. [22] and Plonski et al. [23], [24]. Choi et al. [22] address the trajectory planning problem 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. [23], [24] account for the camera FOV and also give approximation guarantees. They address the problem of viewing a set of points using an aerial robot with a conical FOV but they do not consider any visibility restrictions due to the terrain. 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. We explicitly consider issues that arise due to obstructed visibility by the terrain and limited FOV of the camera. To the best of our knowledge, there does not exist any other work in the literature for route planning for a UAV for visual monitoring on a terrain while considering both camera field-of-view limitations and visibility restrictions due to terrain geometry.

Contributions: We formally introduce the visual monitoring problem on terrains for a UAV with limited FOV. Our main theoretical contribution is a constant-factor approximation algorithm for solving the planning problem (Section IV). The constant-factor depends on the terrain geometry related 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. We also present a practical algorithm that uses a Generalized Traveling Salesperson Problem (GTSP) solver as a subroutine (Section V). We empirically evaluate the performance of the algorithm using a new Integer Linear Programming (ILP) formulation implemented within a branch-and-cut framework as a baseline. We also carry out proof-of-concept field deployments using a UAV to monitor points of interest in a campus environment. Our methods are applicable to both constant and varying altitude UAV operations. For ease of explanation, we develop our methods and analysis considering constant altitude flight and present pointers for use in the case of variable altitude flight operations.

III. Problem Formulation and Preliminaries

In this section, we describe the notation used in the paper and formally define the problem addressed in this work. We also present background information to compute visibility regions for points on a terrain.

A. 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 1a. We model $\mathcal{E}$ as a 2.5D polyhedral terrain, represented as a Triangular Irregular Network (TIN) ([25], pg 352). Such a representation for the terrain ensures that there exists exactly one point on the surface of the terrain along the Z-axis for every point on the X-Y plane. The points of interest are located on the surface of $\mathcal{E}$. Without loss of generality, we assume that the lowest point on the terrain is at $z = 0$ and the highest point of interest is at $z = l$. Further, $\gamma > 0$, is the minimum angle made by any line joining two points on the terrain with respect to the Z axis. We assume that the UAV is equipped with a fixed downward-facing camera having a constant focal length and a finite field of view. While our solution methods do not impose a restriction on the shape of the FOV, for ease of exposition we consider a circular FOV. The FOV may then be represented as a fixed view angle, δ, in every direction. Thus, the camera casts a conical FOV on the terrain, as shown in Figure 1b. To aid in the development of our methods and analysis we consider

that the UAV operates at a fixed altitude $z = h$. We also assume that $h > l$, i.e., the altitude at which the UAV operates is above the highest point of interest to be monitored. This guarantees that any point of interest on the surface of the terrain can at least be observed from a location directly above it and hence there exists a feasible solution to the problem. We discuss the extension to variable altitude flight operations for each method in the corresponding sections.

We assume ideal operating conditions and ignore the effects due to wind and adverse weather. Further, the terrain is accurately modeled and the UAV has complete situation awareness of obstacles in the flight path. The UAV is assumed to have perfect localization and is equipped with a navigation controller that allows the UAV to travel between two waypoints. These are standard assumptions that allow modular computation of the UAV route. It may also be noted that the route planning problem is solved offline and relates to computation of a sequence of waypoints such that the objective of the problem (as given below) is met.

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 δ and 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 $\mathcal{P}$.

Our strategy for solving URPT works in two phases. In the first phase, we compute V_i, the 2D visibility region for each point of interest, p_i, in the fixed altitude flight plane. Visibility region of a terrain point is a closed connected space without holes in this plane. Due to occlusions with the terrain, these visibility regions may have complex geometric shapes (Figure 2b). If the robot visits any point within V_i, it can visually monitor p_i. In the second phase, we compute a tour for the aerial robot to visit at least one point in each visibility region. We present two strategies for solving this problem: (1) A polynomial-time[1] constant-factor approximation algorithm[2] that does not require any discretization of the visibility region (Section IV). (2) A practical algorithm that uses a solution to a GTSP sub-instance to solve URPT (Section V). We compare and benchmark the practical approach with a branch-and-cut ILP approach.

In the next subsection, we review a method that is employed to compute the visibility region for each point of interest.

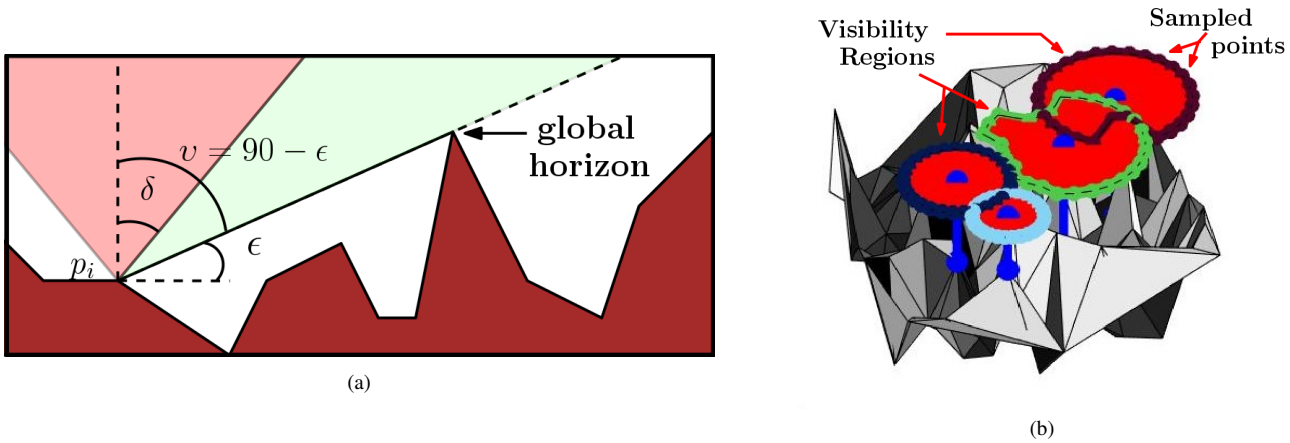
B. Visibility Computation

Consider a point, $p_i \in \mathcal{P}$, that needs to be monitored as shown in Figure 2a. To compute the visibility region for p_i on the terrain, we place a viewpoint at p_i and compute terrain visibility. The farthest visible point in a given radial direction, that obstructs all points beyond itself when viewed from a specific viewpoint is called the *global horizon point* [14] and is expressed in terms of the elevation angle ϵ. The locus of all global horizon points forms the global horizon. *Visibility angle* υ 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.

Horizon computation is a well-studied problem in the graphics literature ([14] and references within). We use the approximate horizon computation method developed by Stewart [27] to compute the horizon at each point of interest within the terrain (Figure 1a). However, our solution approach is independent of the horizon computation algorithm used and other methods in the literature may also be used. To compute

[1]An algorithm whose order of growth of its runtime is a polynomial function of the size of its input is referred to as a polynomial-time algorithm [26].

[2]An algorithm that returns near-optimal solutions is called an approximation algorithm [26]. Constant-factor refers to the approximation ratio of the approximation algorithm being independent of the size of the input.



(a) (b)

Fig. 2: (a) Global horizon point is the farthest visible point in a radial direction. ϵ is the elevation of the horizon point, υ is the visibility angle and δ is the camera view angle. The minimum of υ and δ 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.

the horizon, the radial space is sampled in a discrete number of values, d[3] and the minimum of υ and δ is computed in each direction. Linear interpolation of the extended projections in each direction on the flight plane is then used to compute the visibility region as shown in Figure 2b.

Stewart's method [27] computes an approximation of the global horizon. It divides the angular space into k sectors and computes a constant horizon value for each sector. To compute the elevation angle for a particular sector a coordinate transformation is performed to align the new axes with the a $(\pi/2)$ clockwise rotation of sector boundaries. It computes the maximum sectoral elevation as the maximum elevation angle amongst all points that lie in a sectoral space when observed from a viewpoint. This is repeated over all sectors to compute the maximum elevation (read global horizon). The elevation in turn is used to compute the visibility angle for each direction for all points of interest. The number of sectors, k, is an input parameter and affects the accuracy of the approximate horizon computation method. Narrow sectors (large k value) can miss terrain features and underestimate the elevation, while broad sectors (small value of k) could lead to unwanted over estimation. The interested reader may refer [27] for further details on the algorithm.

IV. Polynomial-Time Approximation Algorithm

In this section, we present a constant-factor polynomial-time approximation algorithm for solving URPT. The input to our algorithm is the set of visibility regions that are computed using the method described in Section III-B. 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 our 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 [28] and non-overlapping convex polygons [29].

The visibility regions in our case may not necessarily be polygonal (for example, they may contain circular arcs) or convex. In general, they can be overlapping. Further, the UAV flight plane may or may not be above the highest point on the terrain and parts of the terrain may act as obstacles along the flight path for the UAV. Nevertheless, we show how to approximate the visibility regions by possibly-overlapping disks of the same radius and consider the effect of obstacles on the path length in our analysis. We then show that this approximation still yields a tour whose length is bounded with respect to the optimal.

[3]Sampling resolution, d, is a user-input parameter discussed in Section V

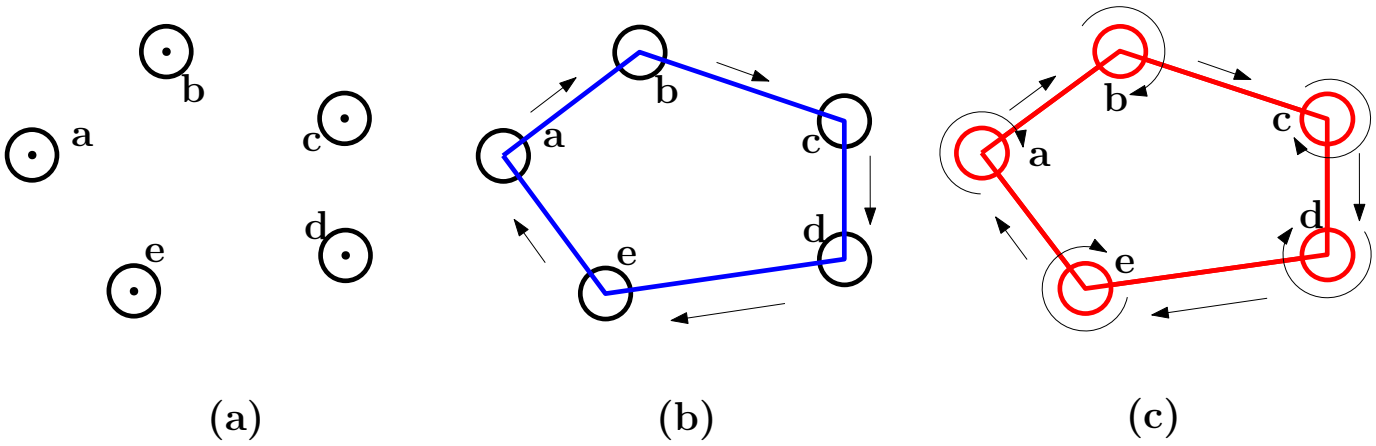


Fig. 3: To compute a tour that visits at least one point in each inner disk, the approximation algorithm works as follows. (a) It computes the maximum independent set of non-overlapping inner disks. (b) It then computes a $\frac{3}{2}$-approximation to the metric TSP tour that visits the center of the disks found in the first step. The tour is shown in blue. Remember that there may be obstacles in the environment that would be accounted for in the cost matrix. Hence, a metric TSP is computed. (c) Every time the tour enters a new disk, add a $2\pi\partial$ detour around the circumference of the disk and then move to the center. The tour with the detours added is shown in red.

A. Algorithm Description

Consider the set of visibility regions, $V = \{V_i : i \in [1, m]\}$, corresponding to the points of interest $p_i \in \mathcal{P}$ that the robot must monitor. We start by replacing each V_i by an inner disk, ∂_i, such that it is completely contained within V_i, i.e., $\partial_i \subseteq V_i$. The inner disk ∂_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 ∂ be the set of all inner disks. All inner disks have the same radius of $r_\partial \triangleq \min\{(h-l)\tan\gamma, (h-l)\tan\delta\}$.

We next find a tour that visits at least one point in each inner disk, ∂_i, as follows: (1) Find the maximum independent set of non-overlapping inner disks, say ∂^I. (2) Find a $3/2$–approximation to the optimal TSP tour that visits the center of all disks in ∂^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 you reach the same point again and then move towards the center. Note that this step adds a detour of length at most $2\pi r_\partial$ to the TSP tour.

In step (2), we find a TSP tour 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 [8]. 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, we can solve the problem 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 $3/2$–approximation algorithm using the Christofides heuristic [30]. In the analysis, we assume the worse $3/2$–approximation for step (2). The reason to add a detour around each disk on the tsp tour is to ensure, any overlapping disks that were not included in the set ∂^I are also visited by the UAV. We show that this algorithm returns a valid tour for URPT that is within a constant-factor of the optimal tour. Algorithm 1 shows the psuedocode for the approximation algorithm.

B. Theoretical Analysis

Let L_∂ be the length of the tour found by our algorithm. We first provide an upper bound to the length of this tour. Let L_V^* be the length of the optimal tour that visits at least one point in each visibility

Algorithm 1 Approximation Algorithm

Input: $V, \mathcal{P}$
Parameters: h, l, γ and δ
Output: a valid URPT tour for the UAV
1: Fit an inner disk, ∂_i of radius, $r_\partial \triangleq \min\{(h-l)\tan\gamma, (h-l)\tan\delta\}$, within each $V_i \in V$
2: Find the maximum independent set of non-overlapping disks
3: Compute a $\frac{3}{2}$-approximation to optimal metric TSP tour that visits the center of each disk
4: Follow the tour computed in step 3 and add a detour around each new disk that it enters

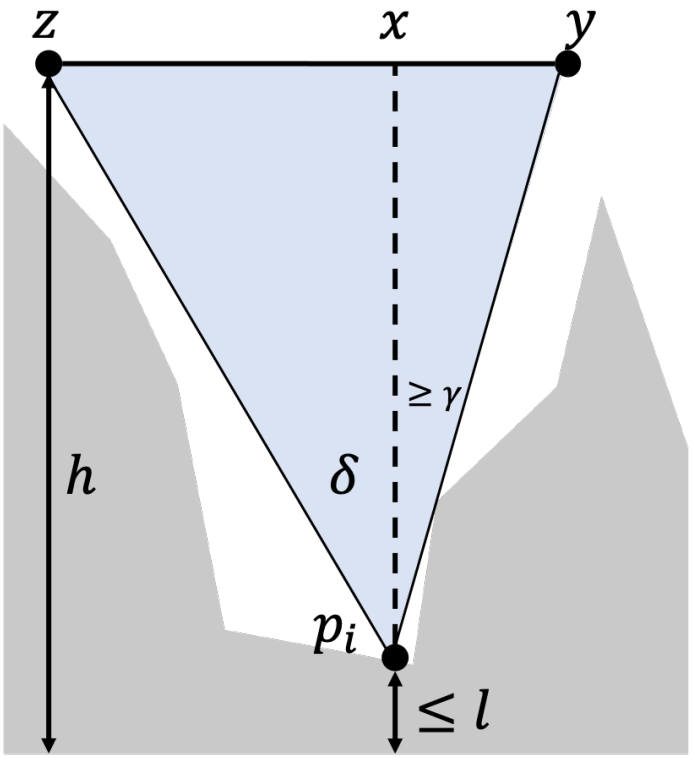


Fig. 4: A 1D slice of the terrain.

region, $V_i \in V$, i.e., the optimal solution to URPT. We start by showing that visiting all the inner disks is sufficient to visit all V_i, i.e., to monitor each point of interest.

Lemma 1: The inner disk ∂_i is completely contained in V_i.

Proof 1: Consider any point on the boundary of V_i. If we show that the distance between that point and the center of ∂_i, say x, is no less than r_∂, then we are done. We have 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 point z corresponds to the former and y corresponds to the latter. Therefore, we have $\angle z p_i x = \delta$ 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, $r_\partial = \min\{(h-l)\tan\gamma, (h-l)\tan\delta\}$, both y and z are no less than r_∂ away from the center of the disk. □

Next, we relate L_∂ and L_V^* as a function of the maximum number of non-overlapping inner disks.

Specifically, let $\partial^I \subseteq \partial$ be the largest set of inner disks, ∂_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 [28]. Let m_∂ be the number of disks in ∂^I.

Lemma 2: Let L_∂ be the length of the tour found using the proposed algorithm. Then we have:

$$L_\partial \leq \frac{3}{2}\left(L_V^* + 2m_\partial h \tan\delta\right) + 2m_\partial \pi r_\partial,$$

where m_∂ is the maximum number of non-overlapping inner disks, ∂^I.

Proof 2: The length of the tour, L_∂, is equal to the distance traveled to visit the centers of the disk in ∂^I (Step 2) and the detours added every time a new disk is visited (Step 3). Let L_{TSP}^* be the length of the optimal TSP tour that visits the center of the disks in ∂^I. Although finding L_{TSP}^* is NP-hard, there exists polynomial time approximation algorithms that find a tour whose length is at most $\frac{3}{2}L_{TSP}^*$ as described earlier. Therefore,

$$L_\partial \leq \frac{3}{2}L_{TSP}^* + 2m_\partial \pi r_\partial \tag{1}$$

$$\leq \frac{3}{2}\left(L_V^* + 2m_\partial r_\partial\right) + 2m_\partial \pi r_\partial. \tag{2}$$

The second inequality follows from the fact that we can always construct a tour that visits the center of the disks in ∂^I by first finding the optimal tour that visits at least one point in each disk in ∂^I (of length L_∂^*) and then adding a detour of at most $2r_\partial$ to visit the center. That is, $L_{TSP}^* \leq L_\partial^* + 2m_\partial r_\partial$ and $L_\partial^* \leq L_V^*$. □

We now show a lower bound on the length of the optimal tour. Recall that h is the height of the fixed-altitude flight plane and δ is the FOV angle. We construct a lower-bound approximation tour for the optimal one as follows. Replace each V_i by a 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 centered at the same x and y coordinates as that of p_i.

Lemma 3: The visibility region V_i is completely contained within the disk D_i.

Proof 3: 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 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 4: Let L_D^* be the length of the optimal tour that visits at least one point in each disk, $D_i \in D$. We have:

$$L_D^* \leq L_V^*.$$

Proof 4: From Lemma 3, we know 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, we lower bound the length of the optimal tour that visits at least one point in each V_i. We relate the lower bound to 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. Let m_D be the number of disks in D^I.

Lemma 5: Let L_V^* be the length of the optimal tour that visits at least one point in each V_i and $m_D \geq 3$.

We have:

$$L_V^* \geq \frac{m_D}{2}\alpha r_D$$

where $\alpha = 0.4786$.

Proof 5: From Theorem 1 in [31], we know that any tour of length L that visits at least one point in m_D disjoint disks of radius r_D satisfies: $L \geq \frac{m_D}{2}\alpha r_D$. Tekdas et al. in [31] also show that $\alpha = 0.4786$ when $m_D >= 3$. 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 4, we get the desired inequality.□

What remains to be shown is the relationship between m_D and m_∂. It is easy to see that $m_\partial \geq m_D$, i.e., the number of non-overlapping outer disks (in D) cannot be more than the number of non-overlapping inner disks (in ∂). We will show that m_∂ cannot be arbitrarily larger than m_D.

Lemma 6: Let m_D and m_∂ be the maximum number of non-overlapping outer disks, D_i, and inner disks, ∂_i, respectively. We have:

$$m_\partial \leq \left(\frac{2r_D}{r_\partial}\right)^2 m_D.$$

Proof 6: 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'. We now bound the maximum number of inner disks that can be packed within D_i' without any two overlapping. One inner disk has an area of πr_∂^2. Therefore, at most $\left(\frac{2r_D}{r_\partial}\right)^2$ non-overlapping inner disks are contained within D_i'. Since there are m_D non-overlapping outer disks, we get the desired result. □

We are now ready to state the main result of this section.

Theorem 1: Let L_∂ 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. We have:

$$L_\partial \leq \left(\frac{3}{2} + 155.17c\right) L_V^*, \tag{3}$$

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 altitude of the highest point of interest, δ 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 7: We know from Lemma 2,

$$\begin{aligned}
L_\partial &\leq \frac{3}{2}\left(L_V^* + 2m_\partial r_\partial\right) + 2m_\partial \pi r_\partial, \\
&\leq \frac{3}{2}\left(L_V^* + 2\left(\frac{2r_D}{r_\partial}\right)^2 m_D r_\partial\right) + 2\left(\frac{2r_D}{r_\partial}\right)^2 m_D \pi r_\partial, \\
&\leq \frac{3}{2}\left(L_V^* + 2\left(\frac{2r_D}{r_\partial}\right)^2 \frac{2}{\alpha r_D} L_V^* r_\partial)\right) + 2\left(\frac{2r_D}{r_\partial}\right)^2 \frac{2}{\alpha r_D} L_V^* \pi r_\partial, \\
&\leq \left(\frac{3}{2}\left(1 + 16\frac{h}{\alpha(h-l)}\frac{\tan\delta}{\min\{\tan\gamma, \tan\delta\}}\right) + \right. \\
&\quad \left. 16\pi\frac{h}{\alpha(h-l)}\frac{\tan\delta}{\min\{\tan\gamma, \tan\delta\}}\right) L_V^* \\
&\leq \left(\frac{3}{2} + 155.17c\right) L_V^* \\
&\leq O(1) L_V^*.
\end{aligned}$$

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 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.). □

Theorem 2: Let m be the number of points of interest (equal to number of visibility regions). Then the approximation algorithm in Section IV-A has a time complexity $\mathcal{O}(m^2 \log(m))$, which is polynomial in m i.e. the number of visibility regions (equal to number of points of interest).

Proof 8: The algorithm in Section IV-A comprises of three steps. The first step makes a pairwise check for overlap of inner disks corresponding to visibility regions. The number of pairs may be computed as a selection, C_2^m, and is equal to $\frac{m(m-1)}{2}$. The check for a single pair can be performed in constant time. Thus step one has a time complexity $\mathcal{O}(m^2)$. The second step invokes Christofides [30] $\frac{3}{2}$-approximate algorithm to compute a TSP tour to visit the centers of all non-overlapping disks. This step has time complexity $\mathcal{O}(m^2 \log(m))$ as shown in [30]. Step 3 adds a detour at each point on the TSP tour to follow the circumference of the disk. Computing the detour takes constant time for each point. Thus step 3 is linear in the number of points of interest ($\mathcal{O}(m)$). Hence the approximation algorithm in Section IV-A to compute a tour for the aerial robot to visit each visibility region has time complexity $\mathcal{O}(m^2 \log(m))$, which is polynomial in the number of points of interest. □

C. Extension to variable altitude flight operations

The same approximation algorithm may 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. The visibility cones in constant resolution flights would all be of the same size and the highest visibility region, V_i, would be at altitude $h + l$. In step (2) of the algorithm, the cost of paths between centers of inner disks within visibility regions must then account for a three dimensional traversal of the UAV. For doing the theoretical analysis, it is easy to see that the enclosing outer disks (D_i) and enclosed inner disks (∂_i), used to compute the lower and upper bounds on the UAV tour respectively, are still valid and a similar analysis can be used to find the constant approximation factor.

V. A Practical Algorithm for Route Planning

In the previous section, we presented a polynomial-time algorithm for solving URPT. Given fixed parameters for the terrain, it yields a constant-factor approximation which is appealing from a theoretical standpoint. In this section, we present a practical approach for finding the routes. Our approach is based on using the solution to a Generalized Traveling Salesman Problem (GTSP) sub-instance to solve the UAV route planning problem. GTSP can be thought of as the discrete version of TSPN, where each neighborhood is replaced by a cluster of points. The input to GTSP is a complete graph, as in TSP, and a cluster number for each vertex. The goal is to find the minimum-cost tour that visits at least one vertex in each cluster.

Consider a GTSP instance, where the clusters of vertices have a one to one mapping to the visibility regions, V, in the corresponding URPT instance. A GTSP tour that visits a vertex in each cluster can then be used to construct a tour for the UAV that visits each of the visibility regions. A naive approach to design such a GTSP instance is to discretize each visibility region, V_i. All sample points within V_i can then be clustered together in the input to GTSP. Visiting any one of these sample points on the tour will be sufficient to monitor p_i. While this is sufficient, it is not necessary to discretize all of V_i. Instead we show that it is sufficient to discretize only the boundaries of these 2D visibility regions. This leads to smaller sized instances and consequently faster computation times. Similar ideas have been used by Obermeyer et al. [4] for path planning for a non-holonomic robot through a set of polygonal spaces.

A. GTSP-Based Algorithm

We start by computing the visibility regions for all points of interest, as discussed in Section III-B. If a visibility region, say V_i, completely contains all other visibility regions, then V_i is redundant and can be ignored. This holds true since any tour that visits a point in V_i's interior also visits V_i. We can preprocess the visibility regions to find all non-redundant regions. For ease of exposition, in the following, we will assume that none of the m regions are redundant.

Next, we uniformly sample d points on the boundary of each visibility region. Let S_i be the set of sample points on the boundary of V_i. These sets of sample points then form the point clusters given as input to the GTSP sub-instance. In case of overlapping regions, some of these sample points may also lie in the interior (or on the boundary) of other visibility region(s). We duplicate such points and add a copy to the set of sample points corresponding to each of the overlapping regions (Figure 2b). A tour that visits a visibility region, V_i, must cross or touch the boundary of V_i as long as there is at least one other visibility region, say V_j, that does not overlap with V_i. If V_i overlaps with all other visibility regions in V, then such a tour must either cross the boundary of V_i or the boundary of an overlapping region in the interior of V_i, unless all visibility regions in V overlap at a single point— in which case the tour reduces to a single point. Hence, this construction of GTSP point clusters is sufficient to find a tour for the aerial robot that visits each visibility region.

The flow diagram shown in Figure 5 shows the steps to compute a GTSP sub-instance to solve URPT. Each point corresponds to a unique vertex in the input to the GTSP instance, $\mathcal{V} = \bigcup_{i=1}^{m} S_i$. We add an edge between every pair of vertices and define the cost function, $c_{ij} : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}^+$, as the length of path for the aerial robot to go from v_i to v_j, where $v_i, v_j \in \mathcal{V}$. In case of constant resolution monitoring (variable altitude flight), the cost function computation accounts for three dimensional cost of travel of the UAV. When v_i and v_j are copies of the same sample point, then the cost to travel between them is zero. 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. Although GTSP is NP-Hard, there are specialized solvers such

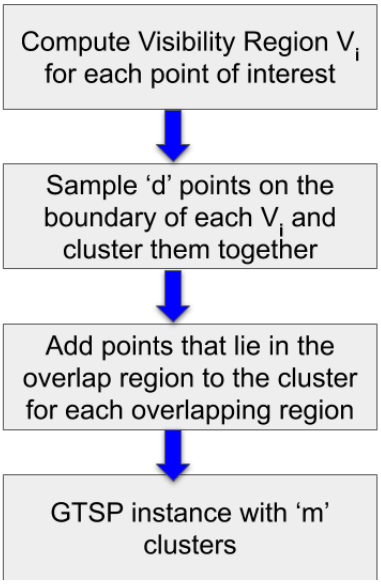


Fig. 5: Flowchart showing the steps to compute the GTSP sub-instance to find a solution to URPT.

as the one presented in [32], which can find efficiently the optimal solution for large instances. In the next section, we show empirical results in support of this. An alternative strategy is to solve the GTSP instance directly using an Integer Linear Programming (ILP) formulation with branch-and-cut. We give the ILP formulation in the supplementary document.

B. Extensions

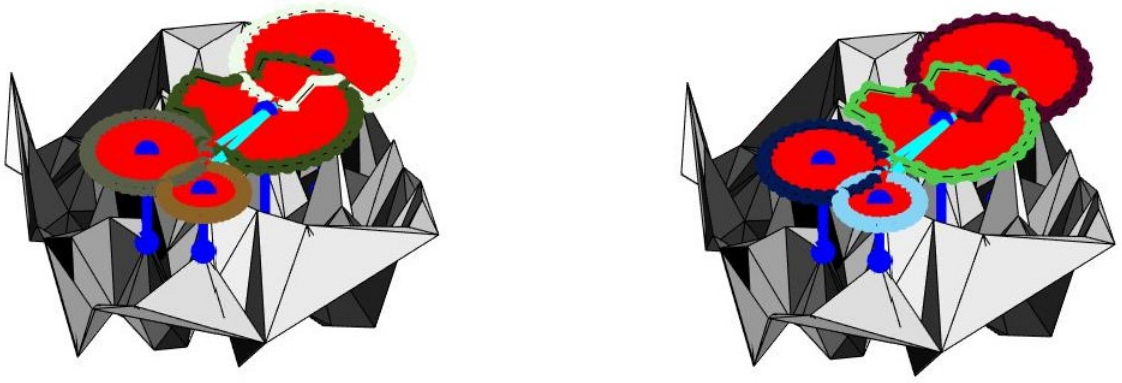
In some cases, it may be required for the robot to take images of the points of interest from a fixed height relative to the point of interest. In this case, the sampled points on the boundary of the visibility will be at different heights and the edges connecting them may require the robot to change altitudes. By virtue of considering the length of the tour within the objective function this only affects the way the edge costs are computed. Additionally, if the field of view of the camera is not a cone but instead is rectangular, it can easily be incorporated as long as the visibility regions can be computed.

VI. 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 [32] in Julia. For visualization, we use MATLAB R2017a with TIN based modeling of the terrain. The simulations were run on a laptop machine with an Intel i7 CPU with 2 cores with a RAM of 16 GB.

A. Instance Generation

To generate the simulation instances we use an environment of size 200×200 units. A 10×10 grid is placed on the environment and terrain altitude at each grid point is sampled randomly between 0 and 100 units. A TIN representation of the terrain was then generated by a piecewise triangular interpolation between neighboring grid points. The aerial robot flight altitude was fixed at 125 units (clear of all terrain features). Size of the set $\mathcal{P}$ of points to be monitored on the terrain was varied from $m = 4$ to 15. The camera FOV view angle δ was varied from 20° to 40° in steps of 10. The value of sampling resolution



(a) (b)

Fig. 6: 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	$\delta = 20^\circ$			$\delta = 30^\circ$			$\delta = 40^\circ$		
	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 I: 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.

parameter d, that determines the number of points sampled on the boundary of the terrain, was varied from 20 to 50 in steps of 10. For each combination of parameters we generated 20 different environments.

We use the GLNS solver 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.

B. Results

We discuss the 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 6. We report two sets of quantitative results. We first compare the performance of GLNS with respect to the ILP formulation. Then, we evaluate the scalability of using the GLNS solver to solve URPT.

The ILP solver (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. We use these lower bounds to benchmark our solutions and report the relative gap as the quality measure of the solution. Table I 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 an optimal solution. In table II, we show the relative gap for the ILP solutions. We observe that the ILP solver was not able to find any feasible solution for a sizable number of instances, especially as d and δ increase. While an increase in the time limit would result in more instances being solved, we expect the trend to be the same.

The GLNS method on the other hand, is able to find a feasible solution in all instances. In Table III, we show the relative gap of solutions computed by GLNS from ILP generated lower bounds. These results

Mean percentage relative gap of ILP generated solutions.

m	$\delta = 20^\circ$			$\delta = 30^\circ$			$\delta = 40^\circ$		
	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 II: 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	$\delta = 20^\circ$			$\delta = 30^\circ$			$\delta = 40^\circ$		
	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 III: Mean percentage relative gap of GLNS solutions w.r.t. ILP solver generated lower bounds. The numbers in bracket represent the standard deviation.

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 we show results for up to $m = 8$. The relative gap for the generated solutions rises quickly with increase in the value of m and δ (Tables II and III). 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. Tables IV and V show the average time taken by the two solvers to compute solutions to URPT. As the size of the instance grows the ILP witnesses an exponential increase in computation time. The computation time for GLNS based solver is less by up to two orders of magnitude for instances of this scale. The ILP based solver times out for all instances for m=8 and above.

We also show results for solving larger instances of URPT while using GLNS as a subroutine. Problem size for the GTSP sub-instance increases as the size of the URPT instance increases. We express the size of a problem instance in terms of the total number of vertices in the GTSP instance. Figure 7a 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 vertices for $m = 15$, as seen in Figure 7b. The corresponding computation time also increases but is still significantly faster than the ILP approach.

It is interesting to note from Figure 7c 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 (δ, d) to yield a small GTSP instance which can be solved quickly, without sacrificing on solution quality.

Time taken by GLNS solver for the optimization process in seconds.

m	$\delta = 20^\circ$			$\delta = 30^\circ$			$\delta = 40^\circ$		
	d = 20	d = 30	d = 40	d = 20	d = 30	d = 40	d = 20	d = 30	d = 40
4	0.49	0.55	0.53	0.53	0.52	0.59	0.53	0.57	0.64
6	0.57	0.56	0.71	0.58	0.7	0.81	0.62	0.75	1.00
8	0.59	0.79	0.94	0.71	1.02	1.6	0.95	2.1	4.8

TABLE IV: Table shows the average time taken (in seconds) by GLNS solver to compute the best solution.

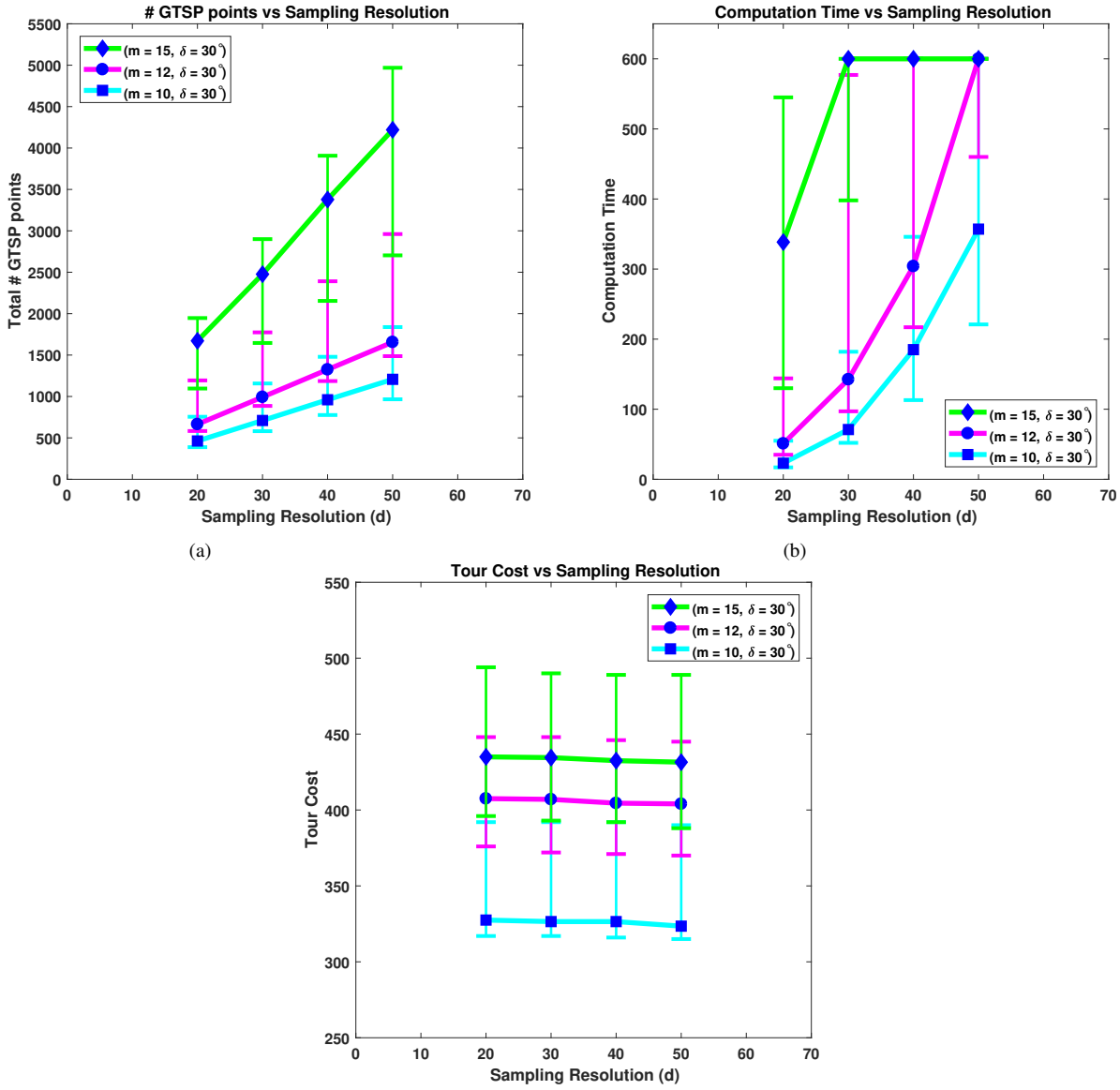


Fig. 7: Figure shows results for using GLNS to solve the GTSP subsinstance to find solutions to URPT. (a) Effect of sampling resolution on size of the GTSP instance. (b) Effect of sampling resolution on computation time. (c) Effect of sampling resolution on tour cost. The plots show the median values with whiskers marking the 1st and 3rd quartiles.

Mean time taken (in seconds) by the ILP solver for the optimization process.

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	13.3 (13.3)	166.4 (166.4)	342.8 (649.4)	22.9 (22.9)	195.3 (230.6)	249.4 (672.5)	31.4 (31.4)	113.3 (192.0)	257.8 (482.7)
6	575.2 (672.7)	- (900.3)	- (900.4)	420.9 (660.6)	- (900.4)	- (900.4)	447.20 (741.8)	- (900.3)	- (900.4)
8	- (900.4)	- (900.6)	- (900.5)	- (900.4)	- (900.4)	- (900.3)	- (900.6)	- (900.4)	- (900.32)

TABLE V: Table shows the average time taken by ILP solver to compute the optimal solution. The numbers in bracket represent the time taken by the solver to compute the best solution over all instances (includes instances for which the solver could not compute an optimal solution). All numbers are in seconds. A '-' signifies that the solver was not able to compute the optimal solution for any instance. The number in bracket in case of '-' , is always 900, signifying that the solver always timed out.

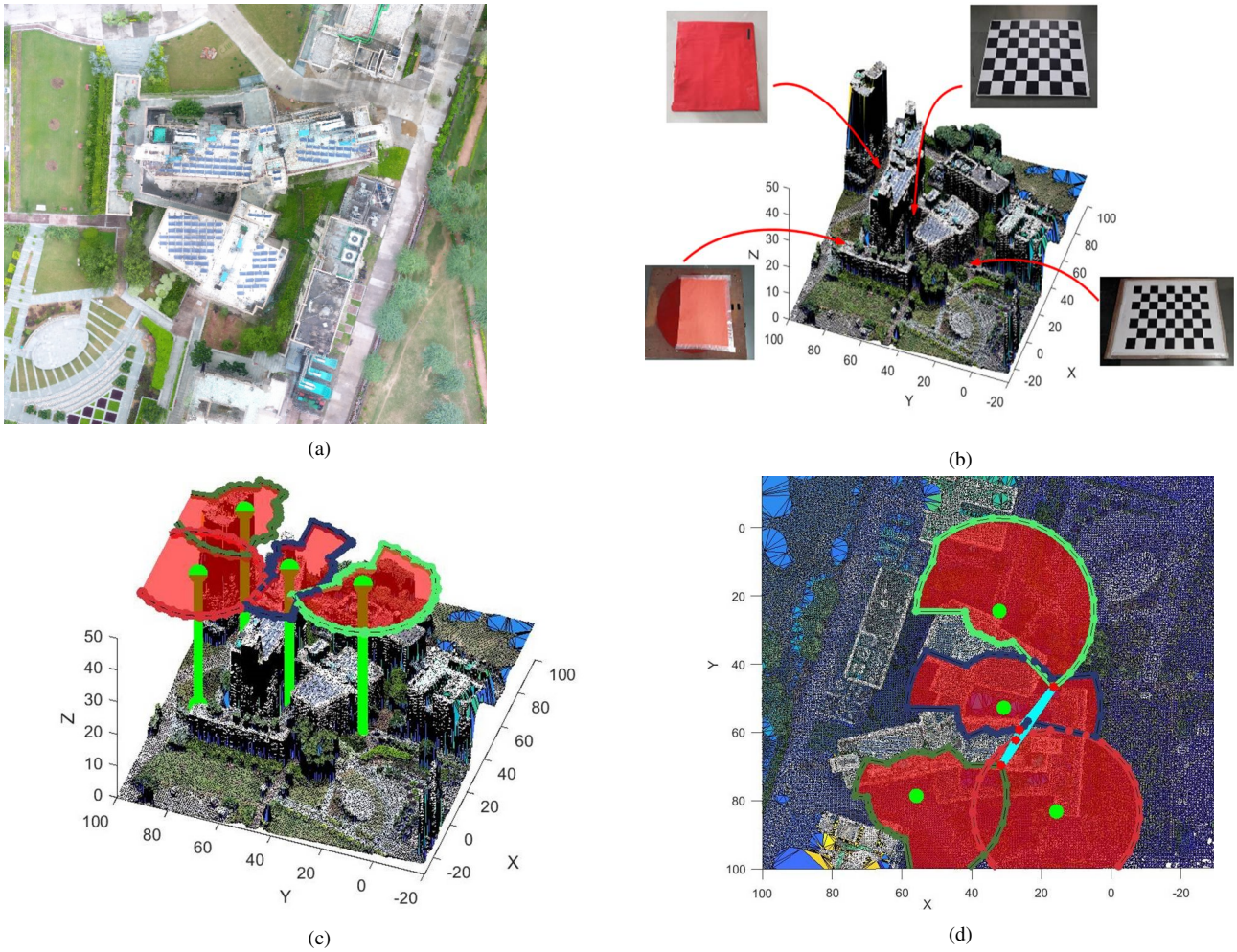
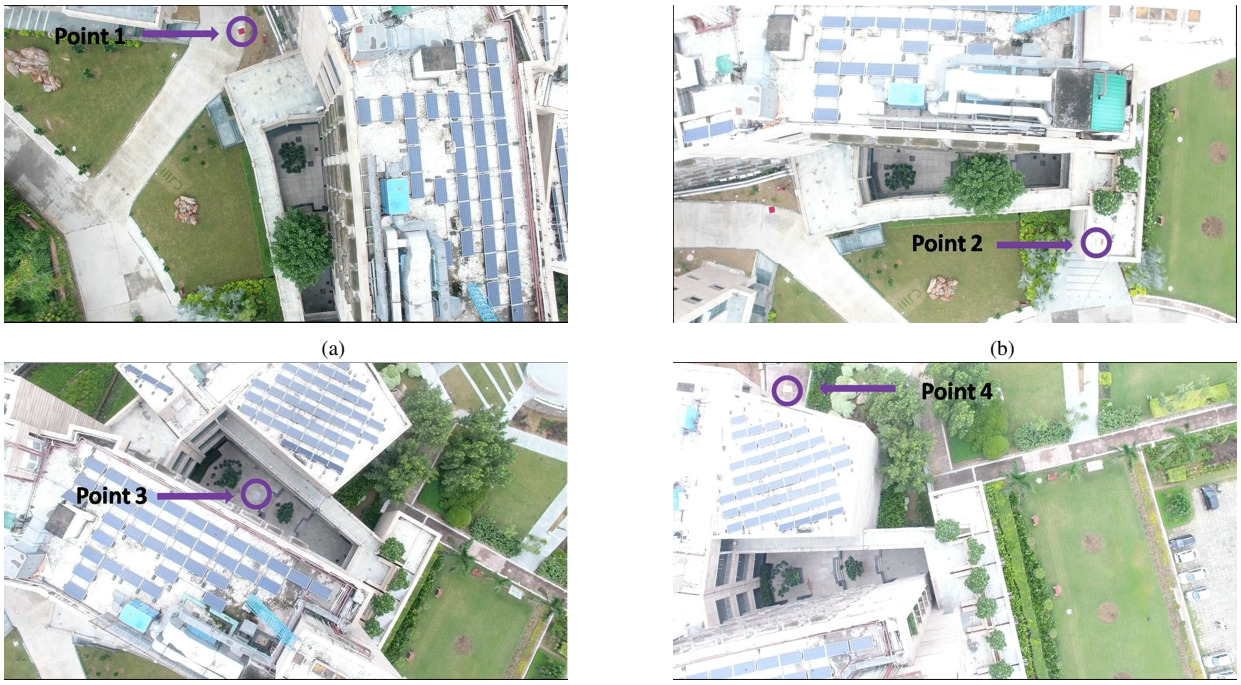


Fig. 8: (a) Operational area for the field trials. (b) DEM of the topography showing the placement of targets. (c) Visibility regions for the monitoring targets computed using the strategy discussed in Section III-B. (d) Top view of the operational area showing the UAV path in cyan color and visibility regions.

VII. Field Experiments

We demonstrate and validate the applicability of our solution method in field experiments conducted inside the campus of IIIT-Delhi. The operational area for the experiments is shown in Figure 8(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. We use the field of view as 30° in our 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 8(b). Visibility regions for each target were computed using the visibility computation strategy given in Section III-B (Figure 8(c)). Value of the sampling resolution parameter d was set to 30. Path for the quadrotor was computed as solution to the corresponding GTSP sub-instance solved using GLNS solver, as shown in Figure 8(d). The solver was run on a laptop machine with an Intel i7 CPU with 2 cores with a RAM of 16 GB. The aerial views of the four target points placed in the environment as observed by the quadrotor are shown in Figure 9. Experiment footage showing the UAV flight and camera imagery is available in the accompanying supplementary material. This serves to demonstrate that the presented method can be used in a practical setting with real-world operational conditions.



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

Fig. 9: Aerial views of points of interest from the quadrotor during experiments.

VIII. Conclusion and Future Work

In this paper, we study the problem of visually monitoring a set of points of interest on a terrain. Unlike previous visual monitoring work, our focus is explicitly on handling constraints that arise due to limited field-of-view of the robot's camera and obstruction to camera visibility as well as robot trajectory due to the terrain itself. We presented a polynomial time approximation algorithm with a constant approximation factor and a practical algorithm that solves a GTSP sub-instance to find solutions in reasonable amounts of time. Notably, the GTSP based solution outperforms a branch-and-cut ILP solver. We show proof-of-concept using field deployment of a UAV to visually monitor points of interest in a campus environment.

The work done in this paper can be extended in a number of ways. A straightforward extension is to plan with more degrees of freedom. For example, orientation of the camera may be controllable (e.g., with the aid of a gimbal). This can be incorporated by sampling the 3D pose at each point within the clusters, each of which becomes a separate vertex in the GTSP input graph. Kinematic constraints of the aerial robot as in the case of fixed-wing type UAVs may also be incorporated by modifying the edge cost computation to be the length of the shortest Dubins' path. Other important extensions include addressing the multiple UAV version and considering fuel limitations of the UAV. From a theoretical standpoint, the approximation ratio is a constant which is independent of the number of input points but does depend on terrain geometry. Removing this dependence and tightening the analysis is an immediate avenue of research.

Future directions include relaxing the assumption on accuracy of the terrain model and to account for observation/modeling errors within route planning. Route planning with online image processing and considering the time taken to process the images is another interesting direction of future study. Online solution methods that account for sensing, actuation and/or localization inaccuracies are also relevant problems to extend this work.

APPENDIX
BRANCH-AND-CUT ILP FORMULATION

To formulate the problem as an Integer Linear Program, we define binary decision variables, y_{ij}, for each pair of vertices v_i and v_j in the set $\mathcal{V}$. Recall that $\mathcal{P}$ is the set of points of interest that the UAV needs to monitor, V_i is the visibility region corresponding to p_i, $S_i \in S$ is the set (cluster) of sample points corresponding to V_i, $\mathcal{V} = \bigcup_{i=1}^{m} S_i$ is the set of vertices in the GTSP sub-instance and the function, $c : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}^+$, gives the cost for the UAV to travel between two vertices 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:

$$\min \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{V}} c_{ij} y_{ij} \tag{4}$$

Degree Constraints:

$$\sum_{v_j \in \mathcal{V} \setminus v_i} y_{ji} - \sum_{v_j \in \mathcal{V} \setminus v_i} y_{ij} = 0, \quad \forall v_i \in \mathcal{V} \tag{5}$$

$$\sum_{v_i \in S_k} \sum_{v_j \in \mathcal{V} \setminus S_k} y_{ji} = 1, \quad \forall k \in [1 \dots m] \tag{6}$$

Sub-tour Elimination Constraints:

$$\sum_{(i,j) \in \delta^+(s)} y_{ij} = 1, \quad \forall s \in \mathbb{P}(\mathcal{S}) \setminus \{\mathcal{S}, \phi\} \tag{7}$$

Variable Domain:

$$y_{ij} \in \{0, 1\} \quad \forall v_i, v_j \in \mathcal{V} \tag{8}$$

Equations (5) and (6) represent tour constraints and ensure each visibility region is visited. Equation (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, we employ 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 2) 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.

REFERENCES

[1] 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.

[2] J. F. Araújo, 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.

[3] 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.

[4] 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.

Algorithm 2 Separation Algorithm

Input: $\mathcal{P}, S, y$
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 to the formulation: $\sum_{(i,j)\in\delta^+(\kappa)} y_{ij} = 1$
    end for
end if

[5] 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.

[6] M. Erdelj, E. Natalizio, K. R. Chowdhury, and I. F. Akyildiz, "Help from the sky: Leveraging uavs for disaster management," *IEEE Pervasive Computing*, vol. 1, pp. 24–32, 2017.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] P. Maini, G. Gupta, P. Tokekar, and S. P. B., "Visibility-based monitoring of a path using a heterogeneous robot team," in *International Conference on Intelligent Robots and Systems*, 2018.

[20] P. Maini, K. Yu, S. P. B., 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*, 2018.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.