Visibility-Based Persistent Monitoring with Robot Teams

Pratap Tokekar and Vijay Kumar

Abstract— **We study the problem of planning paths for a team of robots motivated by coverage, persistent monitoring and surveillance applications. The input is a set of target points in a polygonal environment that must be monitored using robots with omni-directional cameras. The goal is to compute paths for all robots such that every target is visible from at least one path. The cost of a path is given by the weighted combination of the length of the path (travel time) and the number of viewpoints along the path (measurement time). The overall cost is given by the maximum cost over all robot paths and the objective is to minimize the maximum cost.**

In its general form, this problem is NP-hard. In this paper, we present an optimal algorithm and a constant factor approximation for two special versions of the problem. In both cases, the paths are restricted to lie on a pre-defined curve in the polygon. We show that if the curve satisfies a special property, termed *chain-visibility*, then there exists an optimal algorithm for monitoring a given set of target locations. Furthermore, if we restrict the input polygon to the class of street polygons, then we present a constant-factor approximation which is applicable even if the set of target locations is the entire polygon. In addition to theoretical proofs, we also present results from simulation studies.

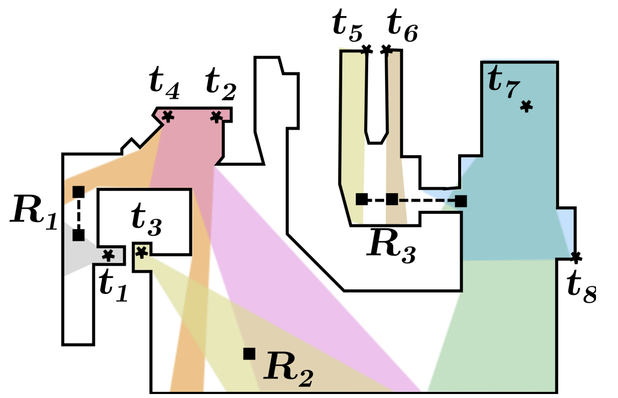


Fig. 1. Problem formulation. We are given a set of target points ($\star$) in a polygonal environment. Our goal is to find m paths ($- - -$) and discrete viewpoints on the paths ($\blacksquare$) such that each target is seen from at least one viewpoint. Here, R_1 sees t_1 and t_2, R_2 sees t_3 and t_4, and R_3 sees t_5–t_8. The cost of a path is a weighted combination of its length (travel time) and the number of viewpoints (measurement time). A path may consist of only one point. The objective is to minimize the maximum path cost.

I. Introduction

We study the problem of planning paths for teams of robots for visibility-based persistent monitoring in complex environments. Visibility-based monitoring problems commonly occur in many applications such as security and surveillance, infrastructure inspection [1], and environmental monitoring [2]. These problems have received significant interest recently [3]–[5], thanks in part, to the technological advances that have made it easy to rapidly deploy teams of robots capable of performing such tasks. For example, Michael et al. [6] demonstrated the feasibility of carrying out persistent monitoring tasks with a team of Micro Aerial Vehicles (MAVs) with onboard cameras.

Persistent monitoring problems are typically studied when the points of interest are given as input. The points may have associated weights representing their importance. A common objective is to find the order of visiting the points that minimizes the weighted latency. Alamdari et al. [7] showed that this problem is NP-hard and presented two $\log$ factor approximation algorithms. In many settings, the path to be followed by the robots is given as input as well and the speed of the robot must be optimized to minimize the maximum weighted latency. Cassandras et al. [8] presented an optimal control approach to determine the speed profiles for multiple robots when their motion is constrained to a given curve. Yu et al. [9] presented an optimal solution for computing speed profiles for a single robot moving along a closed curve to sense the maximum number of stochastically arriving events on a curve. Pasqualetti et al. [10] presented distributed control laws for coordination between multiple robots patrolling on a metric graph.

In this paper, we consider a richer version of the problem where the points to be visited by the robot are not given, and instead must be computed based on visibility-based sensing. We are given a set of target points in a polygonal environment. Each robot carries an omnidirectional camera and can see any target as long as the straight line joining them is not obstructed by the boundary of the polygon. Our goal is to compute paths for m robots, so as to ensure that each target is seen from at least one point on some path. Figure 1 shows an example scenario for $m = 3$.

Our problem is a generalization of the Art Gallery Problem (AGP) [11] and the Watchman Route Problem (WRP) [12]. The objective in AGP is to find the smallest set of "guard" locations, such that every point in an input polygon is seen from at least one guard. AGP is NP-hard for most types of input polygons [11], and very few approximation algorithms exist even for special cases. The objective in WRP is to find a tour of minimum length for a single robot (i.e., watchman) so as to see every point in an input polygon. There is an

P. Tokekar is with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, U.S.A. tokekar@vt.edu. The author was with the Department of Mechanical Engineer and Applied Mechanics, University of Pennsylvania when this work was performed.

V. Kumar is with the Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, PA, U.S.A. kumar@seas.upenn.edu.

This material is based upon work supported in part by ONR grant N00014-09-1-1051.

optimal algorithm for solving WRP in polygons without any holes [13] and a $\mathcal{O}(\log^2 n)$ approximation algorithm for n-sided polygons with holes [14]. Carlsson et al [13] introduced m–WRP where the goal is to find m tours such that each point in the environment is seen from at least one tour. The objective is to minimize the total length of m tours. They showed that the problem is NP-hard (in fact, no approximation guarantee is possible).

Using the length of a tour as the cost is reasonable when a robot is capable of obtaining images as it is moving. However, in practice, obtaining high-resolution images while moving may lead to motion blur or cause artifacts to appear due to rolling shutter cameras. This is especially the case when MAVs are to be used. It would be desirable for the robot to stop to obtain a measurement. Instead of finding a continuous path, we would like to find a set of discrete viewpoints on m paths. The cost of a path can be modeled as the weighted sum of the length of the path (travel time) and the number of measurements along the path (measurement time). Wang et al. [15] first introduced this objective function for WRP for the case of a single robot and termed it the Generalized WRP (GWRP). They showed that GWRP is NP-hard and presented a $\mathcal{O}(\text{polylog}\, n)$ approximation for the restricted case when each viewpoint is required to see a complete polygon edge. Fekete et al. [16] presented an approximation algorithm for the special case of rectilinear grid polygons and unit sensing range for a single robot.

In this paper, we introduce the m robot version of GWRP. This problem in general is NP-hard since it generalizes the NP-hard problems of GWRP and m-WRP. Hence, we consider special instances of the problem and present a number of positive results in these directions. In particular, we characterize the conditions under which the problem has an optimal algorithm (Section III) and a constant-factor approximation algorithm (Section IV). In addition to theoretical analysis, we perform simulations to study the effect of the number of robots and targets on the optimal cost (Section V). We formally state the problems considered in this paper and our contributions in the following section.

II. Problem Formulation

The environment P is an n-sided 2D polygon without holes (in one of the problems we consider a 1.5D terrain environment). We are given a set of target points X within P. We have m robots each carrying an omnidirectional camera. Let each robot travel with unit speed and let the time to obtain an image be t_m. Let Π_i denote both the i^{th} path and V_i denote the discrete set of viewpoints along this path. Let Π denote the collection of m paths. Let $\mathbf{VP}(p)$ denote the visibility polygon of a point p in P and $\mathbf{VP}(V_i)$ denote the union of visibility polygons of all viewpoints on Π_i.

In this paper, we study two problems for visibility-based persistent monitoring. In the first problem, we are given a curve in the polygon along which we must determine the set of viewpoints. The curve can be, for example, the boundary of the environment for border patrolling, or a safe navigation path within the environment. The goal to find m paths along this curve and corresponding viewpoints to see every point in X. We show that if the set X and the curve satisfy a property, termed *chain-visibility*, then this problem can be solved optimally.

Definition 1 *Let X be a set of points and C be some curve in a 2D environment. The pair (X, C) is said to be chain visible if the intersection of the visibility polygon of any point $x \in X$ with C, i.e., $\mathbf{VP}(x) \cap X$ is either empty or a connected chain.*

Although restrictive, the chain-visibility property is satisfied by various curves. Figure 2 shows some examples.

Chain-visibility was used by Carlsson and Nilsson [17] to show that there always exists a *collapsed watchman path* satisfying chain-visibility in street polygons. A street polygon is a polygon without holes with the property that its boundary can be partitioned into two chains, U and D, and any point in U is visible from some point in D and vice versa. Figure 2(b) gives an example. We give more examples of chain-visibility in the following proposition.

Proposition 1 *The following pairs of target points of interest X and curves C all satisfy the chain-visibility property:*

- *X is any set of points in a street polygon and C is a collapsed watchman route.*
- *X is any set of points in a polygon without holes and C is the shortest path between any pair of points s and t in the polygon. In particular, C can be a straight line within the polygon.*
- *X is any set of points on a 1.5D terrain and C is a fixed altitude path.*

Naturally, if the intersection of the visibility region of a point with C is empty, we will not be able to compute a viewpoint on C to see the point. Hence, in the rest of the paper, we consider only those situations where each point in X is visible from some point in C while satisfying the chain-visibility property. Formally, the first problem we consider is the following:

Problem 1
Input: *set of points of interest X and a curve C such that (X, C) are chain-visible*
Output: *m paths $\Pi = \{\Pi_i | \Pi_i \subseteq C\}$ each with viewpoints V_i where for all $x \in X$ we have $x \in \cup \mathbf{VP}(V_i)$.*
Objective: $\underset{\Pi}{\text{minimize}} \max_{\Pi_i \in \Pi} (l(\Pi_i) + |V_i| t_m)$.
Contribution: *Optimal algorithm (Section III).*

The optimal algorithm for this problem is given in Section III.

In the second problem, X is simply the set of all points in the polygon. Our goal is to find m paths, restricted to a chain-visible curve, so as to monitor every point in the environment. Here, we focus on the case of street polygons where we know there always exists a curve, namely the collapsed watchman route, that is chain-visible for any subset of points in the polygon [17]. Street polygons have previously been studied in the context of robot navigation in

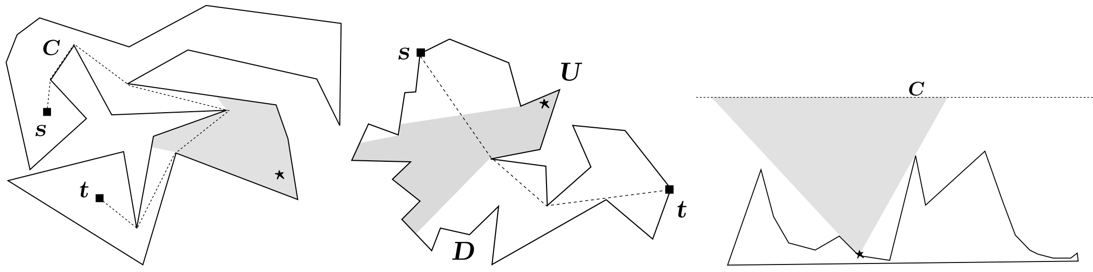


Fig. 2. Examples of curves and environments satisfying the chain visibility property (Definition 1). (a) Shortest path between any two points in a polygon without holes. (b) A collapsed watchman route in a street polygon. (c) 1.5D terrain with a fixed altitude path.

unknown environments [18], [19].

Problem 2
Input: *street polygon P and a chain-visible curve C*
Output: *m paths $\Pi = \{\Pi_i | \Pi_i \subseteq C\}$ each with viewpoints V_i where for all $x \in X$ we have $x \in \cup \mathbf{VP}(V_i)$.*
Objective: $\underset{\Pi}{\text{minimize}} \max_{\Pi_i \in \Pi} (l(\Pi_i) + |V_i| t_m)$.
Contribution: *4–approximation algorithm (Section IV).*

Carlsson and Nilsson [17] presented an optimal algorithm to find the fewest number of viewpoints along C to see every point in P. One way to compute paths would be to first find this smallest set of viewpoints and then distribute them into m paths. Unfortunately, this approach can lead to paths that are arbitrarily longer and consequently arbitrarily worse than optimal paths (Figure 3). Nevertheless, we present a 4–approximation algorithm for Problem 2.

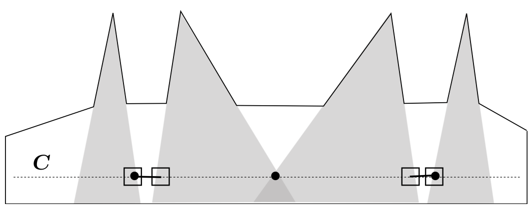


Fig. 3. Problem 2. The optimal solution with $m = 2$ paths consists of two short paths with two viewpoints (☐) each. However, first finding the minimum number of discrete viewpoints (●) on the input curve C and then finding the $m = 2$ paths to visit them may give arbitrarily long solutions (at least one path will have to visit two ●).

III. Optimal Algorithm when Given A Set of Targets and Chain-Visible Curve

In this section, we present an optimal algorithm for solving Problem 1. Here, we are given as input a set of points X in an n-sided 2D polygon that must be visually monitored. The goal is to find a set of viewpoints V along with m watchmen paths that visit V. The watchmen paths are restricted to an input curve C that along with X satisfies the chain-visibility property (Definition 1). In this section we show how to compute V and find the m watchmen paths optimally.

In general, the viewpoints V can be anywhere along C, i.e., no finite candidate set for V is given. We will first establish that there always exists an optimal solution in which V is restricted to either endpoint of the intersection of $\mathbf{VP}(x)$ with C, where x is some point in X. Let C_x denote the segment $\mathbf{VP}(x) \cap C$ for $x \in X$. For ease of notation, we will assign an ordering of points on the curve C. This allows us to define the left and right endpoints for C_x (equivalently, first and last points of C_x along C). We have,

Lemma 1 *There exists an optimal solution for Problem 1 with viewpoints V^* such that for any $v \in V^*$, if v is the left (respectively, right) endpoint of some path Π_i^*, then v must be the right (respectively, left) endpoint of some C_x.*

The proof is given in the appendix.

This lemma allows us to restrict our attention only to the set of finite (at most $2|X|$) points on C. Furthermore, we need to consider only the right endpoints of all C_x for starting a path, and only the left endpoints for ending a path. We will use dynamic programming to find the optimal starting and ending points of m paths. Before we describe the dynamic programming solution, we present a subroutine that is useful in computing the cost of a path when the first and last viewpoint on the path is given.

Algorithm 1: OptimalSinglePath

Input: i, j: first and last viewpoints for a path Π_i on C
Input: X': set of target points such that $\forall x \in X', \mathbf{VP}(x) \cap \Pi_i \neq \emptyset$
Output: V_i: optimal set of viewpoints on Π_i to cover X' (including i and j)
Output: J_i: optimal cost for Π_i to cover X'

```
Mark all points in X' as uncovered
Mark all points in X' visible from either i or j as covered
V_i ← {i, j}
p ← i
while ∃ an uncovered point in X' do
    q ← first point to the right of p such that q is the right endpoint of C_x for some uncovered x ∈ X'
    V_i ← V_i ∪ {q}
    Mark all x ∈ X' visible from q as covered
    p ← q
end
J_i = l(Π_i) + |V_i| t_m
return V_i and J_i
```

The subroutine given in Algorithm 1 takes as input a path Π_i defined by its first and last viewpoint on C. It also takes as input a set of target points X' that are visible from at least one point along Π_i. The output of the subroutine is the optimal set of viewpoints V_i (subject to the condition that first and last point of Π_i are included) and the optimal cost J_i of this path. The following lemma proves the correctness of this algorithm.

Lemma 2 *Let X' be a set of target points and C be a chain-visible curve. Let Π_i be some path along C and $X' \subseteq X$ be target points visible from Π_i. If the first and last viewpoints of Π_i are i and j respectively, then Algorithm 1 computes the optimal set of viewpoints and the optimal cost for Π_i.*

The proof of correctness in given in the appendix.

From Lemma 1 we know that all paths in an optimal solution start and end at the right and left endpoints of C_x. Denote the set of all right and left endpoints by R and L respectively. We build a table of size $|L| \times |R| \times m$. The entry $T(i, j, k)$ gives the maximum cost of the first k paths, with the k^{th} path starting at some $i \in R$ and ending at some $j \in L$, and all $k' < k$ paths ending before i. To correctly fill in the entry $T(i, j, k)$, we must ensure that there does not exists any C_x that starts after the $(k-1)^{th}$ path and ends before k^{th} path (Figure 4). Let $I(j', i)$ be a binary indicator that is 1 if there exists a C_x that is strictly contained between points j' and i, $i > j'$ (but does not contain i and j'). Let `Inf` be a very large number. We compute $T(i, j, k)$ as follows.

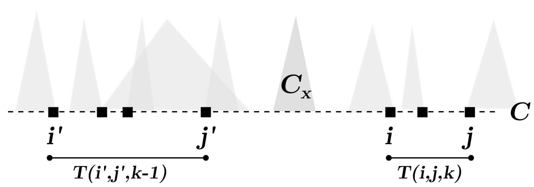


Fig. 4. Dynamic Programming. We search for the start and end points for m paths. Table entry $T(i, j, k)$ gives the cost of the k^{th} part starting at i and ending in j. We must make sure that there is no point x such that the C_x lies completely between the $(k-1)^{th}$ and k^{th} paths.

Let s be the first point on C. In initializing the entry $T(i, j, k = 1)$ we must ensure that there is no target such that C_x ends before i. Thus, for all $i = 1$ to $|R|$ and $j = 1$ to $|L|$ we initialize,

$$T(i, j, 1) = \begin{cases} \text{OPTIMALSINGLEPATH}(i, j) & I(s, i) = 0 \\ \texttt{Inf} & I(s, i) > 0 \end{cases}$$

To fill the rest of the entries, we first find points j' and i' given i, j, k,

$$[j', i'] = \underset{j' < i, i' \leq j'}{\arg\min} \{T(i', j', k-1) + \texttt{Inf} \cdot I(j', i)\}. \quad (1)$$

The term $I(j', i)$ ensures that there is no C_x that starts after j' but ends before i. Furthermore, since the $(k-1)^{th}$ path ends at $j' < i$, we know all C_x that start before j' will be covered. This only leaves two types of points to consider: (i) C_x starts after j' but does not end before i, and (ii) C_x starts after i. While filing in $T(i, j, k)$ we first compute j', i' according to Equation 1. Then, we verify if there exists any point x belonging to either of the two types listed above. If not, then all points in X have already been covered by the first $k-1$ paths. Hence, we set $T(i, j, k) = T(i', j', k-1)$. If there is exists a point belonging either of the two types listed, then

$$T(i, j, k) = \max\{\text{OPTIMALSINGLEPATH}(i, j), \\ T(i', j', k-1)\}.$$

Additionally, if $k = m$ we must check if there is any point that has not been covered. Let t be the rightmost point of the curve C. If $I(j, t) = 1$, we set $T(i, j, m)$ to $+\texttt{Inf}$.

To recover the final solution, we have to find the entry $T(i, j, m)$ with the least cost. Using additional book-keeping pointers, we can recover the optimal solution by standard dynamic programming backtracking. The following theorem summarizes our main result for this section.

Theorem 1 *There exists a polynomial time algorithm that finds the optimal solution for Problem 1.*

The property in Lemma 1 allowed us search over a finite set of points for computing the endpoints of the paths. This comes from the finiteness of the set X. Now, consider the case when *all* points in a polygon are to be monitored by the robots. We may have possibly infinite candidate endpoints for the m optimal paths along C. Nevertheless, in the next section we will show how to compute an approximation for the optimal paths in finite time.

IV. 4–APPROXIMATION FOR STREET POLYGONS

In this section, we present a 4–approximation algorithm for Problem 2. The input to the problem is a street polygon P (see Figure 2(b) for an example) and a curve C that satisfies the chain-visibility property for all points in P. Carlsson and Nilsson [17] showed that there always exists such a curve for street polygons, known as the collapsed watchman route. They also presented an algorithm to compute the smallest set of discrete viewpoints along such a curve to see every point in P. As shown in Figure 3 constructing paths directly from the optimal set of discrete viewpoints can lead to arbitrarily bad solutions for Problem 2. Nevertheless, in this section, we will present an algorithm that yields a 4–approximation starting with the smallest set of discrete viewpoints.

Let g_s and g_t be the first and last points of the input curve C. Let p and q by any points on C (p to the left of q). Let $C[p, q]$ denote the set of all points on C between p and q. We use the following definition of a *limit point* of a point p adapted from [17].

Definition 2 *The limit point of a point p on C, denoted by $lp(p)$, is defined as the first point on C to the right of p such that $lp(p)$ is the right endpoint of C_x for any $x \in closure(P \setminus \mathbf{VP}(C[g_s, p]))$.*[1]

[1]The closure of a set of points is the union of the set of points with its boundary.

In other words, $lp(p)$ is the right endpoint of a C_x closest to p and to its right, such that x is not visible from any point to the left of p, including p. Figure 5 shows an example.

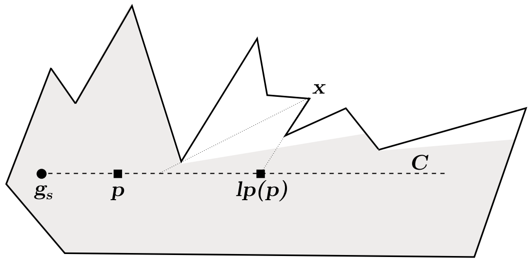


Fig. 5. Limit point $lp(p)$ is the leftmost point to the right of p that is also the right endpoint of C_x for some x not in $\mathbf{VP}(C[g_s, p])$.

In the previous section, we implicitly used the concept of a limit point in Line 6 of Algorithm 1. The limit point of any point p given a curve C can be computed efficiently in polynomial time [17], which we will use in our algorithm. For the first limit point, we have the following result.

Lemma 3 ([17]) *If g_1 is the first point on C to the right g_s such that g_1 is the right endpoint of C_x for any $x \in P$, then $\mathbf{VP}(C[g_s, g_1]) = \mathbf{VP}(g_1)$.*

Carlsson and Nilsson [17] also presented an algorithm to compute the first viewpoint g_1 satisfying the above condition.

It is easy to see that any solution where the first path starts before g_1 can be converted to another valid solution of equal or less cost where the first path starts at g_1. The subroutine given in Algorithm 2 starts with g_1 to compute m paths. Let T^* be the cost of the optimal solution for some instance of Problem 2. The subroutine takes as input a guess for T^*, say $\hat{T}$. We first compute the smallest set of viewpoints G^* sufficient to see every point in the environment using [17]. The rest of the algorithm constructs m paths such that the cost of each path is at most $4\hat{T}$.

Lemma 6 shows if the set of paths computed by the algorithm does not see every point in P, then our guess for T^* is too small. Thus, we can start with a small initial guess for T^*, say $\hat{T} = t_m$, and use binary search to determine the optimal value T^*. Since all tours computed cost less than $4\hat{T}$, we obtain a 4–approximation when our guess $\hat{T} = T^*$.

We first prove that the discrete set of viewpoints computed by Algorithm 2 are correct.

Lemma 4 *Let r be the last point on the last path and $V = \cup V_i$ be the set of all viewpoints over all paths given by Algorithm 2. If a point $x \in P$ is visible from $C[g_s, r]$, then x is also visible from the discrete set of viewpoints V.*

Proof: Suppose there is a point x visible from $C[g_s, r]$ which is not visible from any point in V. Let x_l and x_r be the left and right endpoints of C_x. x_l must lie to the left of r since x is visible from $C[g_s, r]$. Similarly, x_r must lie to the left of r otherwise since $r \in V$, x will be covered by r. From Lemma 3 and the fact that $g_1 \in V$, x_l must lie to the right of g_1. Thus we have $g_1 < x_l \leq x_r < r$. We omit the rest of the proof since it is similar to that of Lemma 2.

Algorithm 2: StreetSubroutine

Input: P: a street polygon
Input: C: collapsed watchman route from g_s to g_t
Input: t_m: measurement cost per viewpoint
Input: $\hat{T}$: guess for the cost of optimal solution

$G^* \leftarrow \{g_i\}$: smallest set of viewpoints on C ([17])
$l \leftarrow g_1$
for $i = 1$ *to* m **do**
  $r \leftarrow$ point along C length $\hat{T}$ away from l (set to g_t if no such point exists)
  $V_i' \leftarrow \{g_i | g_i \in G^*, l < g_i < r\}$
  if $|V_i'| > \dfrac{\hat{T}}{t_m}$ **then**
    $V_i' \leftarrow$ first $\lceil \hat{T}/t_m \rceil$ viewpoints in V_i'
    $r \leftarrow$ last point in V_i'
  end
  $\Pi_i \leftarrow$ path starting at l and ending at r
  $V_i = \{l\} \cup V_i' \cup \{r\}$ `// viewpoints on` Π_i
  $l \leftarrow lp(r)$
end
return Success if $\cup \mathbf{VP}(V_i) = P$, Failure otherwise.

■

Lemma 5 *Let Π be the set of paths computed using Algorithm 2 and r be the last endpoint of the rightmost path. Let Π' be any set of m paths with r' the last endpoint of the rightmost path, such that all points visible from $C[g_s, r']$ are covered by Π'. If the cost of Π' is at most $\hat{T}$, then r' cannot be to the right of r.*

Proof: From Lemma 3 and the definition of limit point, we can say that the first path in Π' starts from g_1. We will prove the lemma by induction on the index of the path. Specifically, we will show that the right endpoint of the i^{th} path in Π, say Π_i, cannot be to the left of the right endpoint of the i^{th} path in Π', say Π_i'. For ease of notation, we will refer to the left endpoint of the i^{th} path by l_i and correspondingly r_i.

Base case. We have two possibilities: (i) r_1 is $\hat{T}$ away from g_1, (ii) Π_1 contains at least $\lceil \hat{T}/t_m \rceil$ viewpoints from G^*. For (i), since the cost of Π_1' is at most $\hat{T}$, its length cannot be greater than $\hat{T}$. Hence, r_1' cannot be to the right of r_1. For (ii), suppose r_1' is to the right of r_1. Then Π_i' must contain at least $\lceil \hat{T}/t_m \rceil$ viewpoints from the optimality of G^*. Thus, the cost of Π_1' is greater than $\hat{T}$ which is a contradiction. The base of the induction holds.

Inductive step. Suppose that r_{i-1}' is to the left or coincident with r_{i-1}. We claim that l_{i-1}' must be to the left or coincident with l_i. Suppose not. By construction, $l_i = lp(r_{i-1})$. Let x be a point such that l_i is the right endpoint of C_x and x is not visible from $C[g_s, r_{i-1}]$. Such an x always exists according to the definition of a limit point. Hence, C_x lies completely between r_{i-1}' and l_i' implying x is not covered by Π', which is a contradiction. Hence, l_i' is to the left or coincident with l_i. Now, using an argument similar to the base case, we can show that r'_i cannot be to the right of r_i proving the induction.

Hence, for all i, r'_i cannot be to the right of r_i proving the lemma. ■

Lemma 6 *If Algorithm 2 returns* FAILURE *then* $\hat{T} < T^*$.

Proof: Suppose $T^* \leq \hat{T}$. Let Π^* be the optimal set of m paths. Let r and r^* be the rightmost points on Π and Π^* respectively. From Lemma 5, we know r^* cannot be to the right of r.

Algorithm 2 returns FAILURE when there exists some point, say $x \in P$, not covered by Π. From Lemma 4, x must not be visible from $C[g_s, r]$. Thus, x cannot be visible from $C[g_s, r^*]$. Hence, the optimal set of paths do not see every point in P which is a contradiction. ■

We now bound the cost of the solution produced by our subroutine.

Lemma 7 *The maximum cost of any path produced by Algorithm 2 is at most* $4\hat{T}$.

Proof: By construction, the length of any path is at most $\hat{T}$. The number of measurements are at most $\lceil \hat{T}/t_m \rceil + 2 \leq 3\hat{T}/t_m$. Hence, the cost of any path is at most $4\hat{T}$. ■

Combining all the above lemmas, we can state our main result for this section.

Theorem 2 *There exists a* 4*–approximation for Problem 2.*

The minimum number of discrete viewpoints $|G^*|$ can be significantly larger than the number of vertices of the polygon. G^* can be computed in time polynomial in the input size (i.e., the number of sides in the input polygon) and the output size (i.e., $|G^*|$). Consequently, the running time for each invocation of the subroutine in Algorithm 2 is also polynomial in the input and output size. The optimal value T^* can be computed in $\mathcal{O}(\log T^*)$ invocations of the subroutine via binary search. We can reduce the overall running time by terminating the binary search early, at the expense of the approximation factor.

V. SIMULATIONS

In this section, we report results from simulations using the optimal algorithm presented in Section III. The algorithm was implemented in MATLAB using the VisiLibity [20] library for floating-point visibility computations. The polygonal environment shown in Figure 6(a) was used to generate all the simulation instances.

The simulation results are presented in Figure 6. The plots show mean and standard deviation of the costs for 50 trials. The positions of the targets are randomly generated for each trial. All target locations are generated such that they satisfy the chain-visibility property with respect to the the dashed path shown in Figure 6(a). The measurement time t_m was set to $1\,\mathrm{s}$ for all trials.

Figure 6(b) shows the effect of varying the number of robots on the optimal cost (makespan). 15 target locations are randomly generated for each trial. Figure 6(c) shows the effect of varying the number of targets. The number of robots were fixed to 3. Figure 7 shows the computation time required for running the dynamic programming, as a function of the number of robots and the targets.

VI. DISCUSSION

Our approach in this paper was to formulate a geometric version of the persistent monitoring problem, abstracting away some of the more practical concerns. Even with this formulation, the problem turns out to be challenging. Geometric abstractions allow us to focus on the inherent challenges of the problem without the additional practical complexities. In the rest of this section, we first discuss how our work can be extended to handle the practical complexities. We then highlight some related open problems.

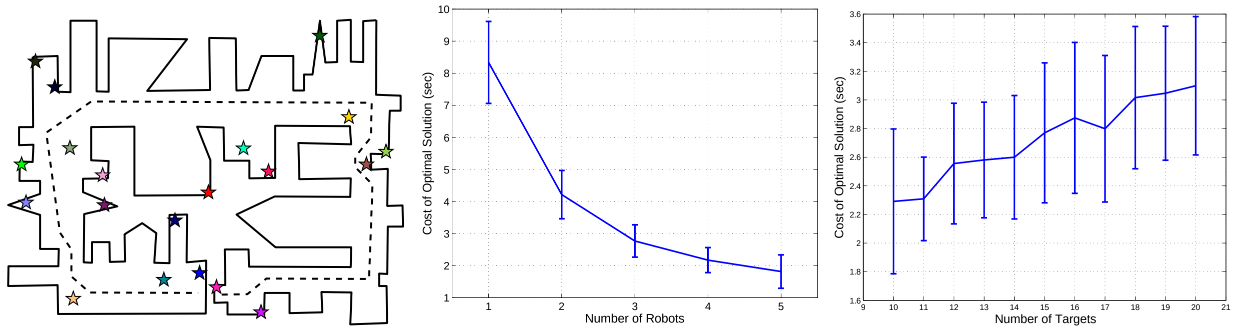
A. Practical Extensions

The analysis presented in this paper is based on the key property of chain-visibility that is satisfied by limited classes of environments. In particular, for Problem 2 we require the environment to be a street polygon. One approach of extending our algorithm for general environments would be to first decompose the environment into street polygons, and then apply our algorithm separately in each component. While algorithms for decomposition into street polygons are not known, there is an optimal algorithm for decomposing a polygon without holes into the fewest number of monotone subpolygons [21]. The class of monotone polygons are included in the class of street polygons, and thus can be used as valid inputs to our algorithm. However, the approximation guarantee will in general not hold.

We can address practical constraints, e.g., robot dynamics, finite extents, etc., by starting with the geometric solutions given in this paper and refining them to incorporate the constraints. Such a two-level refinement approach was presented by Turpin et al. [22] for the related problem of assigning goal positions and trajectories to teams of robots with complex dynamics and strict collision avoidance constraints. We will adapt these refinements as part of our future work on experimental evaluation on a system of MAVs. Our ongoing efforts on implementation in simulated environments can be seen in the accompanying video.

B. Open Problems

The main open problem from a theoretical standpoint is whether there are polygons for which we can compute optimal solutions or constant-factor approximations for watchmen routes without requiring a candidate input curve. One possible approach would be to determine settings in which we can first compute a discrete set of viewpoints and then find m paths to visit them. There are existing algorithms for finding m paths of minimum maximum length to visit a set of points (see e.g., [23]). The key property would be to show that a tour restricted to the discrete set of viewpoints thus computed is at most a constant factor away from an optimal tour. The approximation algorithm presented in Section IV follows this principle. Investigating similar results for richer environments is part of our ongoing work.



(a) (b) (c)

Fig. 6. (a) Representative simulation instance. The target points ($\star$) are generated randomly for each trial within the polygonal environment. (b) Varying the number of robots on the optimal solution cost (makespan). The number of targets is fixed to 15 for all trials. (c) Varying the number of targets on the optimal solution cost (makespan). The number of robots is fixed to 3 for all trials.

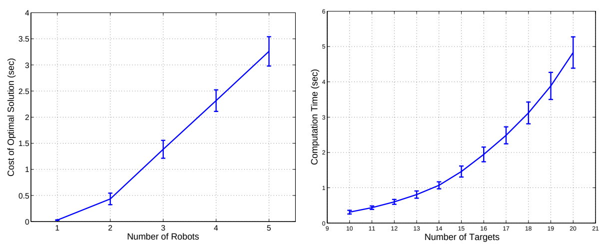


Fig. 7. Computation time as a function of the number of robots with 15 targets (left), and as a function of the number of targets with 3 robots (right).

VII. Conclusion

In this paper, we introduced a new formulation for persistent monitoring problems. The typical objective in persistent monitoring is to find robot tours that visit a set of target points in an environment. In this paper, we relax this constraint and seek to find robot tours that only need to visit locations that can see the targets. The objective is to find m robot paths such that every target is seen by at least one robot and the maximum path cost is minimized. This formulation generalizes two well-known NP-hard problems, namely the Art Gallery Problem and the Generalized Watchman Route Problem. Nevertheless, in this paper we characterize the class of environments for which we can obtain strong performance guarantees. Specifically, we show that if the paths are restricted to a curve satisfying a special property, termed chain-visibility, then we obtain algorithms with suboptimality bounds.

References

[1] T. Ozaslan, S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, “Inspection of penstocks and featureless tunnel-like environments using micro uavs,” in *International Conference on Field and Service Robotics*, 2013.

[2] R. N. Smith, M. Schwager, S. L. Smith, B. H. Jones, D. Rus, and G. S. Sukhatme, “Persistent ocean monitoring with underwater gliders: Adapting sampling resolution,” *Journal of Field Robotics*, vol. 28, no. 5, pp. 714–741, 2011.

[3] S. L. Smith, M. Schwager, and D. Rus, “Persistent robotic tasks: Monitoring and sweeping in changing environments,” *Robotics, IEEE Transactions on*, vol. 28, no. 2, pp. 410–426, 2012.

[4] J. Yu, M. Schwager, and D. Rus, “Correlated orienteering problem and its application to informative path planning for persistent monitoring tasks,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[5] X. Lan and M. Schwager, “Planning periodic persistent monitoring trajectories for sensing robots in gaussian random fields,” in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2415–2420.

[6] N. Michael, E. Stump, and K. Mohta, “Persistent surveillance with a team of mavs,” in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 2708–2714.

[7] 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.

[8] C. G. Cassandras, X. Lin, and X. Ding, “An optimal control approach to the multi-agent persistent monitoring problem,” *Automatic Control, IEEE Transactions on*, vol. 58, no. 4, pp. 947–961, 2013.

[9] J. Yu, S. Karaman, and D. Rus, “Persistent monitoring of events with stochastic arrivals at multiple stations,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[10] F. Pasqualetti, J. W. Durham, and F. Bullo, “Cooperative patrolling via weighted tours: Performance analysis and distributed algorithms,” *Robotics, IEEE Transactions on*, vol. 28, no. 5, pp. 1181–1188, 2012.

[11] J. O’rourke, *Art gallery theorems and algorithms*. Oxford University Press Oxford, 1987.

[12] W.-p. Chin and S. Ntafos, “Optimum watchman routes,” *Information Processing Letters*, vol. 28, no. 1, pp. 39–44, 1988.

[13] 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, 1999.

[14] J. S. Mitchell, “Approximating watchman routes,” in *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2013, pp. 844–855.

[15] P. Wang, R. Krishnamurti, and K. Gupta, “Generalized watchman route problem with discrete view cost,” *International Journal of Computational Geometry & Applications*, vol. 20, no. 02, pp. 119–146, 2010.

[16] S. P. Fekete, J. S. Mitchell, and C. Schmidt, “Minimum covering with travel cost,” *Journal of combinatorial optimization*, vol. 24, no. 1, pp. 32–51, 2012.

[17] S. Carlsson and B. J. Nilsson, “Computing vision points in polygons,” *Algorithmica*, vol. 24, no. 1, pp. 50–75, 1999.

[18] C. Icking, R. Klein, E. Langetepe, S. Schuierer, and I. Semrau, “An optimal competitive strategy for walking in streets,” *SIAM Journal on Computing*, vol. 33, no. 2, pp. 462–486, 2004.

[19] C. Icking, T. Kamphans, R. Klein, and E. Langetepe, “On the competitive complexity of navigation tasks,” in *Sensor Based Intelligent Robots*. Springer, 2002, pp. 245–258.

[20] K. J. Obermeyer and Contributors, “The VisiLibity library,” `http://www.VisiLibity.org`, 2008, r-1.

[21] J. M. Keil, “Decomposing a polygon into simpler components,” *SIAM Journal on Computing*, vol. 14, no. 4, pp. 799–817, 1985.

[22] M. Turpin, K. Mohta, N. Michael, and V. Kumar, "Goal assignment and trajectory planning for large teams of interchangeable robots," *Autonomous Robots*, pp. 1–15, 2014.

[23] E. M. Arkin, R. Hassin, and A. Levin, "Approximations for minimum and min-max vehicle routing problems," *Journal of Algorithms*, vol. 59, no. 1, pp. 1–18, 2006.

APPENDIX

PROOF OF LEMMA 1

Proof: Consider the case when v is the left endpoint of a path Π_i in an optimal solution. We will prove by contradiction. If v is not also the right endpoint of some C_x, then find the first right endpoint, say v', of some C_x that is to the right of v along C. All points in X visible from v are also visible from v'. Thus, we can let v' be the new left endpoint of Π_i^* to give a valid solution of lesser length, i.e. lesser cost, which is a contradiction. The case for the right endpoint is symmetrical.

■

PROOF OF LEMMA 2

Proof: We first verify that all targets in X' will be covered by the algorithm (and thus the algorithm terminates). Suppose not. Let x be a target that is not covered. By definition of X', C_x intersects with Π_i. Let x_l and x_r be the left and right endpoints of C_x. If x_l is to the left of i, then x is visible from i and will be marked covered. If x_r is to the right of j, then x is visible from j and will be marked covered. Hence, x_l and x_r lie between i and j.

Consider the closest viewpoint in V_i lying to the left of x_l, say v (we know at least one such viewpoint exists, namely i). Let v' be the first viewpoint in V to the right of v (we know at least one such viewpoint exists, namely j). Now v' cannot be to the left of x_l, else v is not the closest viewpoint to the left of x_l. Similarly, v' cannot be to the right of x_r since v' will not satisfy the condition in Line 6 in Algorithm 1. This leaves the case where v' is between x_l and x_r, in which case x is visible from a viewpoint in V_i.

Next, we verify that the optimal set of viewpoints and cost is correctly computed. The length of Π_i is fixed since i and j are given as input. Let X'' be the subset of X' such that any $x \in X''$ is not visible from either i or j. It remains to show that $|V_i \setminus \{i, j\}|$ is the least number of measurements required to cover X''. Denote the viewpoints in $V_i \setminus \{i, j\}$ by $v_1, \dots, v_n$. Along with i and j, this defines $n+1$ partitions: $[v_0 := i, v_1], [v_1, v_2], \dots, [v_n, v_{n+1} := j]$.

For contradiction, suppose there is a $V' = \{v'_i\}$ with $n-1$ viewpoints that cover X''. Then, there must exist at least one $[v_i, v_{i+1}]$ partition that does not contain any v'_j. From Line 6, this implies that there is some point x whose interval lies completely between two consecutive viewpoints in V'. Thus, V' does not cover all elements in X which is a contradiction.

■