Argus: Realistic Target Coverage by Drones

Ahmed Saeed
Georgia Institute of Technology
ahmed.saeed@gatech.edu

Ahmed Abdelkader
University of Maryland, College Park
akader@cs.umd.edu

Mouhyemen Khan
Georgia Institute of Technology
mouhyemen.khan@gatech.edu

Azin Neishaboori
Carnegie Mellon University
azin.neishaboori@gmail.com

Khaled A. Harras
Carnegie Mellon University
kharras@cs.cmu.edu

Amr Mohamed
Qatar University
amrm@qu.edu.qa

ABSTRACT

Low-cost mini-drones with advanced sensing and maneuverability enable a new class of intelligent visual sensing systems. This potential motivated several research efforts to employ drones as standalone surveillance systems or to assist legacy deployments. However, several fundamental challenges remain unsolved including: 1) Adequate coverage of sizable targets; 2) Target orientation that render coverage effective only from certain directions; 3) Occlusion by elements in the environment, including other targets.

In this paper, we present *Argus*, a system that provides visual coverage of wide and oriented targets, using camera-mounted drones, taking into account the challenges stated above. Argus relies on a geometric model that captures both target shapes and coverage constraints. With drones being the scarcest resource in Argus, we study the problem of minimizing the number of drones required to cover a set of such targets and derive a best-possible approximation algorithm. Building upon that, we present a sampling heuristic performs favorably yet is up to 100x faster compared to the approximation algorithm. We implement a complete prototype of Argus to demonstrate and evaluate the proposed coverage algorithms within a fully autonomous surveillance system. Finally, we evaluate the proposed algorithms via simulations to compare their performance at scale under various conditions.

CCS CONCEPTS

•Theory of computation →Computational geometry; •Computer systems organization →Sensor networks;

KEYWORDS

target coverage; full-view coverage; drone-based surveillance; art gallery problems; visibility; approximation algorithm

ACM Reference format:
Ahmed Saeed, Ahmed Abdelkader, Mouhyemen Khan, Azin Neishaboori, Khaled A. Harras, and Amr Mohamed. 2016. Argus: Realistic Target Coverage by Drones. In *Proceedings of The 16th ACM/IEEE International Conference on Information Processing in Sensor Networks, Pittsburgh, PA USA, April 2017 (IPSN 2017),* 12 pages.
DOI: http://dx.doi.org/10.1145/3055031.3055078

1 INTRODUCTION

Public spaces such as airports, train stations, shopping malls and schools, are usually monitored with the aid of security cameras mounted at key locations. Such cameras greatly help overview the area of interest and guide first responders in the event of an emergency, which can have a significant impact on crime [36]. Moreover, visual sensor systems enable the automation of complex tasks like crowd counting, event detection, object tracking, target identification, and activity recognition [17]. The automation of these tasks has the potential of providing better solutions to several operational and security issues in public spaces (e.g., queue length estimation and perimeter protection).

There are several theoretical and practical challenges associated with the design of effective and efficient visual sensor systems as exemplified by recent work on surveillance. Such intelligent systems with advanced features like automatic identification and recognition impose a set of requirements on video footage:

- Subjects should be facing the camera [9] or within a certain viewing angle [7].
- Relevant portions of subjects should be fully captured, preferably by a single camera to avoid stitching images from multiple viewpoints, which can be challenging [38].
- As a prerequisite, occlusions and blind spots should be avoided by positioning cameras properly [54].

To the best of our knowledge, no earlier work in smart surveillance tackled these challenges simultaneously.

An extreme approach to some of these challenges is to increase the density of deployed cameras such that any object, within the area of interest, is covered from all angles [52, 53]. This requires a large number of cameras incurring a rather high cost [57]. Furthermore, targets are typically modeled as mere points which results in two issues. First, mutual occlusion between targets and occlusion by obstacles in the area are not accounted for, which can create blind spots. Second, assuming sizable targets can be represented by multiple points, there is no guarantee that the target will be fully captured in the frame of at least one camera if each point is treated separately and may be covered by a different camera. Another approach is to optimize the orientations of cameras in a static deployment to minimize occlusions, however, this does not ensure the target will be facing the camera [48]. It is clear that modeling targets by more than mere "blips" can improve the quality of the collected data which enables more realistic sensing and more effective systems.

In this paper, we introduce *Argus*, a system that tackles all identified challenges by exploiting the rapid advancements in mini-drone technologies and their anticipated applications in surveillance, crowd monitoring [21], infrastructure inspection [8] and cinematography [30]. In particular, camera-mounted drones are dispatched and dynamically positioned to eliminate blind spots and capture frontal views of the subjects of interest. Argus uses the *Oriented Line Segment Target Model* (*OLS*), a new geometric model we develop to capture target orientation, size, and potential occluders. Such information can be estimated with high accuracy from visual sensors [14] or other contextual sensors (e.g., device-free RF-based techniques [3]). Argus relies on static cameras which capture coarse grain footage providing the information needed to estimate *OLS* target traits. *With drones being the most valuable resource in Argus, we focus on the problem of drone placement to cover targets under the OLS model while minimizing the number of drones needed.*

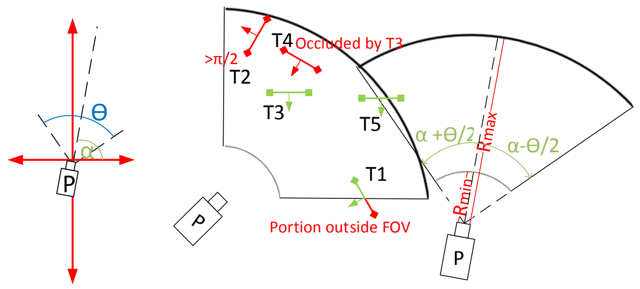
Intuitively, *OLS* looks at a cross-section through the object and fits a line segment and orientation to estimate its size and facing direction. While still being simple, the new model is more complex than plain points and requires a more advanced system to estimate it and new algorithms to utilize it. We show that minimizing the number of drones under *OLS* is NP-hard and even hard to approximate [46]. In this paper, we develop a best-possible $O(\log n)$-approximation algorithm, where n is the number of targets. The algorithm is based on a novel spatial subdivision of the search space for camera placement by the various coverage constraints, which elucidates the treatment of the new *OLS* model for computation. We leverage these insights to develop a more efficient coverage heuristic that almost matches the performance of the approximation algorithm while running up to 100x faster in our simulations with large numbers of targets and various target and camera parameters.

We prototype a fully autonomous Argus with two AR.Drone 2.0 quadcopters fitted with camera sensors and a fixed PTZ-camera as the source of target information. We use the prototype to demonstrate the drastic difference in coverage quality enabled by *OLS* compared to the traditional model of targets as mere blips on the radar. Our experiments with synthetic targets show that adopting the enhanced *OLS* model does not introduce significant overheads with respect to the navigation and control algorithms already running in the system.

The contributions of this paper are three fold:

- We present Argus, a fully autonomous system that controls drones to provide high quality unobstructed coverage of targets from appropriate viewpoints based on a novel *Oriented Line Segment Target Model (OLS)*.
- We design a best-possible $O(\log n)$-approximation algorithm and an efficient heuristic for coverage. We compare the proposed algorithms through extensive simulations.
- We implement and analyze a fully autonomous prototype of Argus to demonstrate the superior quality of coverage it can offer, and gauge the overhead of the proposed algorithms within a realistic system.

The rest of the paper is organized as follows. In Section 2, we describe Argus and the novel *OLS* model along with related applications that can benefit from Argus. We proceed to study *OLS* and develop the coverage algorithms in Section 3. Section 4 discusses the implementation of the Argus prototype, which we use together with simulations to evaluate Argus in Section 5. Finally, we highlight related work in Section 6 and conclude the paper in Section 7.



(a) AOV (θ) and VD (α).

(b) Parameters controlling the shape and size of a camera's FOV (right). Violations of coverage constraints for targets T1, T2, T4 (left). Target T5 is covered but not fully covered.

Figure 1: Camera parameters.

2 ARGUS OVERVIEW

Argus is a fully autonomous system that aims to capture high quality video footage of identified targets of interest subject to coverage constraints. In this section, we introduce a new target model that captures both width and orientation, and formally define the coverage problem for this new model, taking occlusions into account. Then, we introduce the system components and properly define the scope of this study. Finally, we present trending applications we envision Argus can execute or help improve significantly.

2.1 Coverage Model and Problem Formulation

Sensor Model: We think of sensors as autonomous quadcopters, equipped with cameras. The configuration of a sensor is a tuple $S_i = \langle P_i, \alpha_i, \theta, R_{min}, R_{max} \rangle$, where: P_i is the location of the sensor in 2D and α_i is the **Viewing Direction (VD)** measured counter-clockwise from the positive x-axis (Figure 1(a)), θ is the **Angle of View (AOV)**, R_{min} and R_{max} are the minimum and maximum allowable distances between the camera and any target for acceptable viewing quality. A similar model has been presented for several anisotropic sensors [5].

Definition 1. Field of View (FOV) (Figure 1(b)): *The unoccluded area that can be viewed by a sensor with an acceptable quality. Formally, it is the spherical frustum having the camera at P as its apex with an axis at angle α, an opening angle of θ, and limited by R_{min} and R_{max} with any occlusions subtracted.*

Oriented Line Segment Target Model (OLS): We model targets in 2D as line segments whose lengths are the width of the targets, and orientation is a vector perpendicular to the line segment. Larger targets can be represented by one or more line segments representing their different aspects and their corresponding orientations. Formally, the configuration of a target is the tuple $T_j = \langle P_j^s, P_j^e, \overrightarrow{D_j} \rangle$, where P_j^s and P_j^e are the start and end points of the line segment and $\overrightarrow{D_j}$ is the orientation vector. Furthermore, we define M_j as the midpoint of the target and let W_j denote its width. We assume $W_j \ll R_{max}$ $\forall\, j$.

Obstacle Model: We reuse the line segment primitive to represent obstacles by the segments along their boundaries. Obstacle O_k is a chain of segments $\{\langle P_1^s, P_1^e\rangle, \langle P_2^s, P_2^e\rangle, \ldots\}$, which block visibility but, unlike targets, have no orientation.

Coverage Model: A sensor S_i is said to fully cover a target T_j if the following conditions apply: (Figure 1(b)): 1) T_j falls in the FOV of S_i which means that it is neither too far nor too close and that a line segment from S_i to any point on T_j does not intersect any other target or obstacle. 2) The angle between $\overrightarrow{D_j}$ and $\overrightarrow{M_j S_i}$ is $\leq \pi/2$, meaning that S_i can capture frontal views of T_j.

Definition 2. Full Coverage: *A target T_j is fully covered if $\overline{P_j^s P_j^e}$ is fully contained in the FOV of some camera S_{i^*}, with T_j facing S_{i^*}.*

Modeling Assumptions: The main assumption we make in this work is that target locations and orientations can be estimated by a coarse grain surveillance system. This assumption leverages recent advances in target detection and tracking using fixed cameras (e.g., [14] for pedestrians). *OLS* is essentially proposed to obtain close-up views of targets using mobile cameras and provide fine grain surveillance as needed. This approach is supported by work on multi-tier camera sensor networks where coarse grain knowledge may be acquired via higher tier cameras providing low granularity coverage sufficient for detection and localization, but insufficient for identification, recognition, or activity monitoring [34].

The Coverage Problem: We formally define the coverage problem for *OLS* targets and briefly discuss its hardness and the approaches we take to compute a solution.

Definition 3. *Oriented Line Segment Coverage Problem (OLSC): Let $\mathcal{T}$ be a set of n oriented line segments, that may only intersect at their end points, and O be a set of u obstacles. Find the minimum number of mobile directional visual sensors, with uniform $\langle \theta, R_{min}, R_{max}\rangle$, required to fully cover all segments in $\mathcal{T}$.*

It is necessary to establish lower-bounds on the efficiency of algorithms for such problems to better understand how to tackle them in practice. To this end, we show that *OLSC* is NP-Hard and even hard to approximate by studying a variant of the Art Gallery Problem with an AOV $\theta < 360°$ [46].

Solving *OLSC* requires the generation of a set of candidate camera placement configurations (i.e., location and orientation pairs) and selecting a set of configurations that cover all targets while minimizing the number of cameras. This approach relies on subdividing the search space (i.e., the plane) by the various coverage requirements of the targets in $\mathcal{T}$. These subdivisions produce a finite set of potential camera location and orientation pairs ($\mathcal{R}$) which is convenient for computation. We consider $\mathcal{R}$ to be *comprehensive* if it contains at least one representative for each region of space where cameras could be placed to cover any given subset of targets. With that, *OLSC* is reduced to picking a subset of $\mathcal{R}$ to cover all targets in $\mathcal{T}$, which is equivalent to solving the SET-COVER problem over $(\mathcal{T}, \mathcal{R})$. Hence, applying the well-known greedy selection scheme guarantees an $O(\log n)$-approximation of the minimum number of cameras needed to cover $\mathcal{T}$ [16], which, by our lower-bound results [46], is the best-possible for *OLSC*.

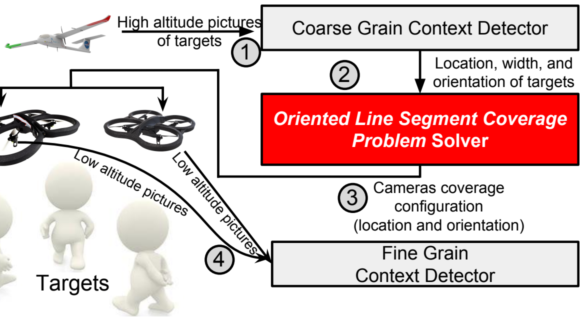

High altitude pictures of targets
1
Coarse Grain Context Detector
2
Location, width, and orientation of targets
Oriented Line Segment Coverage Problem Solver
Low altitude pictures
Low altitude pictures
3
Cameras coverage configuration (location and orientation)
4
Fine Grain Context Detector
Targets

Figure 2: Operational flow of Argus.

2.2 The Argus System

Argus relies on two tiers of surveillance systems. The top tier, used for coarse grain surveillance, provides the location, width, and orientation of targets and obstacles. The lower tier uses the output of the top tier to provide fine grain coverage using mobile drones; a setup we believe will become more feasible as drones get smaller (e.g., [40]). Having a hierarchy of surveillance systems allows each tier to be responsible for different tasks [34]; see the survey in [42]. This concept was also used in [13] where PTZ cameras are used to identify the type of bags carried by subjects based on the locations determined by static cameras. Alternatively, the location and orientation information of targets can be provided by non-visual means like radar and device-free localization systems (e.g., [3] which can detect both the orientation and location of humans using Wi-Fi signals). Camera-mounted drones in the lower tier can use the locations estimated by such RF-based systems to capture video footage of the detected targets.

Figure 2 depicts the operational flow of the Argus system. The system consists of four components: 1) *Coarse Grain Context Detector*, 2) *OLSC Solver*, 3) *Drone Controller*, and 4) *Fine Grain Context Detector*. The *Coarse Grain Context Detector* is responsible for obtaining basic target information, from high altitude cameras or coarse grain RF sensors, that the *OLSC Solver* uses to determine the positioning strategy of mobile cameras. The inputs that the *OLSC Solver* requires are the location, width, and orientation of each target. The output of the *OLSC Solver* is used by the *Drone Controller* to move the low altitude cameras to capture high quality unobstructed images of whole targets. These images can then be further processed, through the *Fine Grain Context Detector*, by different context extraction algorithms [27]. *We realize that implementing each of these components is challenging in its own right with many open research problems. Hence, we focus on the OLSC Solver and present vanilla approaches to the other components of Argus.*

2.3 Further Applications for Argus

Although surveillance is the natural use case for Argus, the same workflow can immediately be used to plan a deployment of static cameras to cover a set of static targets (e.g., artifacts in a museum). To further demonstrate the utility of the proposed system, we discuss two particularly relevant applications that have received a growing interest recently.

In structural inspection, Argus is able to represent the components to be inspected as wide objects that can occlude one another as well as provide a limited number of viewpoints that need to be visited [8]. In such scenarios, the number of targets can be very large. We analyze the scalability of Argus in Section 5.2.

Cinematography, both in reality or in virtual worlds, frequently considers the planning of camera trajectories to obtain the desired shots in a given scene [30]. Argus can generate candidate viewpoints given a description of anticipated target locations, which may even be planned by a director in a cinematographic context. Argus also accommodates additional requirements on camera placements to help optimize the shots as we discuss in Section 3.2.

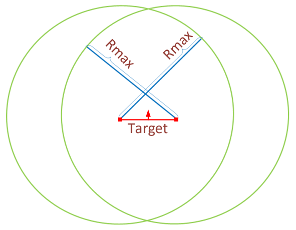
3 DRONE PLACEMENT: THE OLSC SOLVER

Deploying drones requires configuring each with a location to move to and a direction to point its camera sensor. There are infinitely many possible configurations spanning every location where a drone can be positioned and every direction it can be covering. In order to get a handle on the problem of drone placement, the key step is to reduce the space of configurations into a small finite subset. The goal of the *OLSC Solver* is to compute a set of configurations that covers a given set of *OLS* targets using the minimum number of drones. The *OLSC Solver* can be broken down into three modules: 1) A *Spatial Discretizer* responsible for finding a small subset of points to work with, 2) An *Angular Discretizer* that determines the relevant directions to consider at each of the points selected by the *Spatial Discretizer*, and 3) A *Configuration Selector* to pick a subset of the configurations generated by the *Angular Discretizer*.

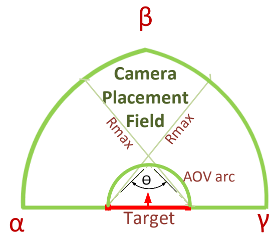
3.1 Spatial Discretizer

The goal of the *Spatial Discretizer* is to generate the candidate locations for camera placement. Each candidate location can be used to view a subset of targets under the coverage model. A key characteristic of the *Spatial Discretizer* is the nature of the set of candidate locations it generates. We define two types of candidate sets: 1) comprehensive and 2) heuristic, denoted by $\mathcal{P}$ and $\hat{\mathcal{P}}$, respectively. ***Comprehensive representation of the search space means that the set of candidate locations is guaranteed to include all optimal configurations, up to an equivalence. Two configurations are equivalent with respect to a subset of targets if both configurations can cover these targets under the same constraints.*** Heuristic sets are not guaranteed to be comprehensive but are an effective alternative which is also practical as they include fewer locations allowing faster computation of drone configurations at the expense of a potential increase in the number of drones.

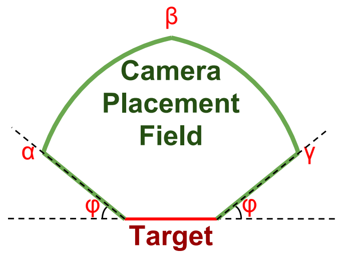
Formally, given a comprehensive set of candidate locations $\mathcal{P}$ we are able to obtain an $O(\log n)$-approximation algorithm. However, generating the $O(N^4)$ candidates required for a comprehensive set can be an overkill and incurs much higher overhead. This, in turn, slows down both the *Angular Discretizer* and *Configuration Selector* as they would have to go through too many candidates. To remedy this, we develop a heuristic spatial discretizer that generates $O(N)$ candidates $\hat{\mathcal{P}}$, enabling the *OLSC Solver* to handle larger numbers of targets.



(a) Boundary of points no farther than R_{max} from any point on the target.



(b) BCPF for R_{max}, AOV and target orientation constraints.



(c) BCPF parametrized by viewing angle ϕ for quality control.

Figure 3: Basic Camera Placement Field (BCPF).

3.1.1 Comprehensive Spatial Discretization. Our objective is to identify candidate locations that comprehensively represent the search space through spatial subdivisions based on target, obstacle, and camera constraints. We have four constraints for a camera to cover a target: range (i.e., being within R_{min} and R_{max} from the target), angle of view (i.e., being within the camera's FOV of width θ), target orientation (i.e., capturing the target from its significant perspective) and occlusion avoidance (i.e., having no target or obstacle occluding the target of interest). First, we focus on satisfying all these constraints for a single target, which allows us to develop the essential tools needed to compute camera placements. Then, we show the extension to a pair of targets using a convenient approach to covering multiple target simultaneously. Finally, we generalize to arbitrary subsets of targets by satisfying their coverage constraints in a pairwise fashion.

Covering a single target by a single camera: We aim to determine the region around a target where a camera can be placed and oriented to fully cover this target. We call this region the *Camera Placement Field (CPF)*. It is more convenient to define the CPF by introducing one constraint at a time.

Starting with range constraints, Figure 3(a) shows how the space around a target is restricted by R_{max} to the intersection of two circles each centered at one end point of the target segment, since target width is $\ll R_{max}$ ($R_{min} = 0$ was used to simplify the figure). Next, for the AOV constraint, we exclude locations too close to the target such that the angle required for full coverage would be larger than θ. The area to exclude is bounded by an *AOV arc* with the target segment as a chord at an inscribed angle of θ. Then, we exclude everything behind the target to account for target orientation.

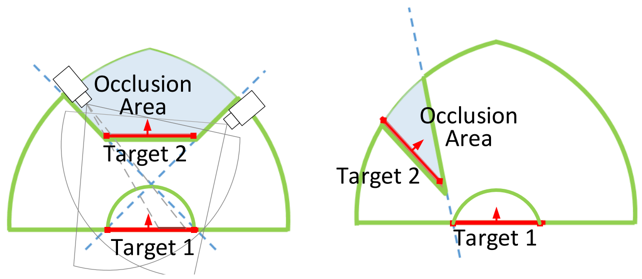


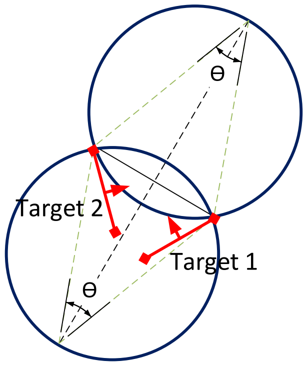
Figure 4: Camera Placement Field (CPF) of Target 1 (in thick line) after applying occlusion constraints by some occluder (marked here as Target 2).

Applying the first three constraints only results in an area we refer to as the *Basic Camera Placement Field (BCPF)*. The BCPF is bounded by three arcs and two line segments as illustrated in Figure 3(b), which assumes that a camera can cover targets at 90° rotations. While some tasks like face detection can still yield high accuracy at 90° rotations [15], the accuracy of object matching and point matching between two images drop significantly for rotations larger than 45° [7]. To incorporate notions of quality in the coverage model, the BCPF can be restricted to only include locations within a certain rotation with respect to the target. This is achieved by a controllable parameter ϕ that constrains the range of acceptable rotations as illustrated in Figure 3(c).

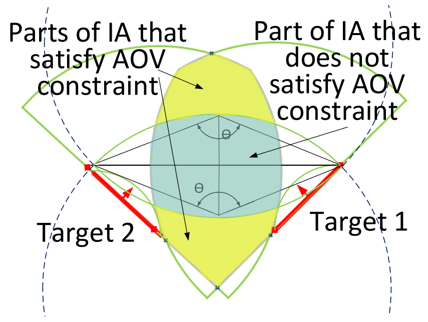
Applying the last constraint, if other targets or obstacles intersect the BCPF of the target at hand, it is necessary to exclude the *occlusion area* of all points within the BCPF where any camera cannot provide full coverage of this target. This is obtained by the lines connecting opposite ends of the occluding segment and the target segment as illustrated in Figure 4. The vertices along the boundary of the CPF will be referred to as the *critical points* of the CPF as they play a crucial role in our algorithms.

Covering a pair of targets by a single camera: For a single camera to cover two targets, it must fall in the CPF of each, meaning that camera placement is limited to the Intersection Area (IA) of their CPFs as shown in Figure 5(a). This guarantees a placement that satisfies range, orientation, and occlusion constraints for both targets. As for the AOV constraint, we need a set of *AOV circles* to exclude all locations that cannot fully cover both targets simultaneously. Instead of using targets themselves as chords to define the circles as in the case of a single target, we use the four line segments connecting their end points. For each such line segment, the area to exclude is determined by the two circles having that segment as a chord with an inscribed angle of θ; we call these the *AOV circle pair*. For $\theta \leq \frac{\pi}{2}$, we exclude the union of the AOV circle pair while for $\theta > \frac{\pi}{2}$ we exclude their intersection. Note that the camera never lies inside both AOV circles as the intersection is always excluded. Hence, we can use individual AOV circles to enforce one constraint at a time. Using all four line segments results in 4 AOV circle pairs which can be shown to contain all relevant AOV circles for these two targets. Figure 5(b) shows two examples of AOV circle pairs.

Covering a set of targets using a single camera: For a single camera to cover three or more targets, the camera must fall in the IA of all of their CPFs and outside some of their AOV circles. It is clear that any computation on the power set of $\mathcal{T}$, examining all subsets to generate all possible IAs, would take an exponential number of steps. We avoid this paradigm of enumerating IAs explicitly, and only compute discrete representatives for them.



(a) AOV circle pair ($\theta < \frac{\pi}{2}$).


Parts of IA that satisfy AOV constraint
Part of IA that does not satisfy AOV constraint
θ
θ
Target 2
Target 1

(b) Intersection Area (yellow) and its restriction by only one of the 4 AOV circle pairs for two targets ($\theta > \frac{\pi}{2}$).

Figure 5: AOV circle pairs and intersection area (IA).

Comprehensive representation of the search space: The representatives we compute are the intersection points of the geometric coverage constraints. Note that the vertices along the boundary of any potential region for camera placement to cover a given subset of targets are either critical points of a CPF, intersection points between CPFs, or intersection points between CPFs and AOV circles; we use $\mathcal{P}$ to denote the set of all such vertices. We prove that $\mathcal{P}$ is a comprehensive representation.

THEOREM 3.1. *Given an* OLSC *instance* $\langle \mathcal{T}, \theta, R_{min}, R_{max} \rangle$, *the set* $\mathcal{P}$ *contains at least one representative for each feasible coverage configuration for all subsets of* $\mathcal{T}$.

PROOF. Let $S \subseteq \mathcal{T}$ be a subset of k targets that can be covered simultaneously by a single camera c placed at point x. The case where $k = 1$ is trivial, since any critical point on the CPF of a single target can be used as a representative for covering that target. Since $\mathcal{P}$ contains all critical points of all CPFs, we are done. For $k \geq 2$, let A_k be the region around x to which c can be moved and rotated accordingly while still being able to cover all k targets in S. Clearly, A_k must lie in the intersection of CPFs of all targets in S. Otherwise, by the definition of a CPF, at least one of the orientation, range (R_{max} and R_{min}) or occlusion constraints would be violated for at least one target in S, a contradiction. Moreover, A_k must lie outside at least one of the AOV circles generated by all pairs of targets in S. Otherwise, by the definition of an AOV circle pair, c would not be able to simultaneously cover at least two of the targets in S by an AOV θ, again a contradiction. We may therefore think of A_k as a region enclosed in a set of CPFs with some parts taken out by a set of AOV circles. This implies that A_k is bounded by at least one CPF and possibly some AOV circles. This allows us to describe A_k by the curves outlining its boundary and their intersection points. Regarding A_k as the equivalence class of points where a camera can be placed to cover S, any of these intersection points can serve as a representative. As there is at least one CPF boundary for A_k, these intersection points must contain either an intersection point of two CPFs or an intersection point of a CPF and an AOV circle. By construction, $\mathcal{P}$ contains all such intersection points. □

We consider the complexity of generating $\mathcal{P}$. Letting $N = n + u$, each CPF can be represented by up to $O(N)$ pieces as all other n targets and u obstacles can split the BCPF into several parts. Thus, the operation of intersecting two CPFs is $O(N^2)$ and performing this operation pairwise for all targets is $O(n^2N^2)$. The operation of intersecting a CPF with an AOV circle is $O(N)$, and is repeated $O(n^2)$ times for all AOV circles resulting in an $O(n^2N)$ operation per target. Hence, repeating this operation $O(n)$ times takes $O(n^3N)$. This amounts to a total of $O(n^2(n^2 + nu + u^2))$. We relax this expression to $O(N^4)$ and loosely bound $|\mathcal{P}| = O(N^4)$.

3.1.2 Heuristic Spatial Discretization. The $O(N^4)$ candidates generated by the approximation algorithm are too demanding for real-time applications. On top of that, we can still produce good solutions using far fewer candidates at the cost of missing tightly packed configurations corresponding to small intersection areas. The reason is that each additional target further restricts the region of space where cameras can be placed to cover the set of targets simultaneously. In practice, such configurations are neither robust to errors in target localization and drone navigation nor stable enough to capture the anticipated views before targets move apart. This motivates a more efficient and robust approach to the generation of candidates. We propose the *Basic Camera Placement Field (BCPF) sampling*.

An intuitive approach to yield $O(n)$ candidate locations is to sample a constant number of points per target taking occlusion into account. However, an easy first order relaxation is that any camera placement covering a given target must fall in its BCPF of that target (Figure 3(b)). The advantage of using the BCPF instead of the actual CPF, is that BCPF can be computed in $O(1)$ per target compared to $O(N)$ for the CPF. Once the BCPF is known, uniformly sampling its interior should capture most of the useful configurations. Note that the intersection of multiple BCPFs gets sampled proportionally which favorably reduces the probability of missing good candidate points. However, as suggested by our simulations with uniformly random target where adversarial arrangements are unlikely, it suffices to sample points along the boundary of the BCPF. Letting ρ be the sum of the central angles of the two BCPF arcs and the apex angle of the triangle in between, we can fix suitable BCPF sampling steps ϵ_a and ϵ_r for the angular and radial axes, respectively. With that, we generate $O(\frac{\rho \cdot R_{max}}{\epsilon_a \cdot \epsilon_r} \cdot n)$ candidate locations that we call $\hat{\mathcal{P}}$. Our experiments show the promise of this almost agnostic approach to candidate generation as it is able to match the quality of the approximation algorithm while being much faster.

3.2 Angular Discretizer

Once a camera is placed at a given location x from either $\mathcal{P}$ or $\hat{\mathcal{P}}$, we need to determine the relevant viewing directions (VDs) to consider. We achieve this in two stages: First, we perform an angular sweep to identify one representative VD for each subset of targets that can covered simultaneously from the location in question. Then, we optimize representative VDs for better footage quality.

Angular sweep: This step identifies a set of representative VDs $sweep(x) = \{\hat{\alpha}_1, \hat{\alpha}_2, \dots\}$ for each maximal subset of targets that can be covered simultaneously by a camera placed at x. Each such maximal subset can be covered by a range of viewing directions $[\alpha_i^l, \alpha_i^h]$. The application may specify a criteria for selecting the best direction from this range. As a default setting, we use $\hat{\alpha}_i = (\alpha_i^l + \alpha_i^h)/2$. Let $cov(x, \alpha)$ denote the maximal subset of targets covered by a camera at x when its VD is set to α. Observe that if we perform a radial sort around x of the end points of all target segments visible from x, no two targets overlap. Given the radial sort of all end points, we can easily determine which targets are visible by discarding segments interrupted by a closer point and enumerate *sweep* in $O(N)$. The radial sort can easily be found in $O(N \log N)$. Alternatively, a *visibility diagram* for the set of segments can be constructed in $O(N^2)$ [51]. Using the diagram, *sweep* queries take $O(N)$.

Viewing direction optimization: Ideally, surveillance footage should provide clear frontal views by an assignment of cameras to targets with each camera-target pair nearly facing one another. This easily breaks down when the camera's viewing direction is not directly towards the target. Given a candidate location for camera placement x, each maximal subset of targets $cov(x, \hat{\alpha}_i)$ may be covered by any viewing direction $\alpha \in [\alpha_i^l, \alpha_i^h]$. Within this range, one extreme might favor certain targets placing them right at the center of the FOV, while other targets barely fit at the side. Depending on the spread of these targets and the direction each of them is facing, a camera positioned at x can adjust its VD to obtain the best views possible. A natural objective is to minimize the *deviation*, defined as the angle between the camera's VD and the line-of-sight from x to the target's midpoint. Let $d(x, \alpha, T_j)$ denote the deviation for target T_j when viewed by a camera at x with VD α. With that, we seek to minimize the total deviation over all targets $f_1(x, \hat{\alpha}, \alpha) = \sum_{T_j \in cov(x,\hat{\alpha})} d(x, \alpha, T_j)$. The optimal VD α^* can then be chosen as $\arg\min_{\alpha \in [\alpha^l, \alpha^h]} f_1(x, \hat{\alpha}, \alpha)$. Alternatively, we may choose to minimize the worst deviation for any one target $f_\infty(x, \hat{\alpha}, \alpha) = \max_{T_j \in cov(x,\hat{\alpha})} d(x, \alpha, T_j)$.

3.3 Configuration Selector

With the output of the *Angular Discretizer* as the set of configurations $\mathcal{R} = \{cov(x, \hat{\alpha}) \mid x \in \mathcal{P}, \hat{\alpha} \in sweep(x)\}$, our goal is to find a *minimum set cover* which is a subset $\mathcal{R}_{opt} \subseteq \mathcal{R}$ whose union is $\mathcal{T}$ with $|\mathcal{R}_{opt}|$ minimized. Using the standard greedy approximation scheme, we compute a cover $\mathcal{R}_{greedy}$ with a guaranteed bound

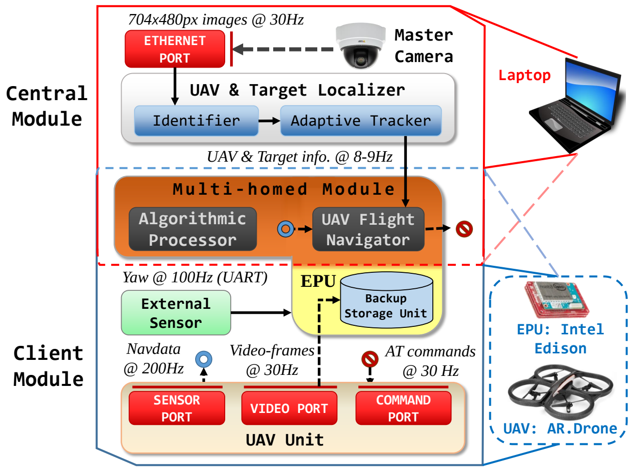


Figure 6: Architecture of the Argus prototype.

$\frac{|\mathcal{R}_{greedy}|}{|\mathcal{R}_{opt}|} = O(\log |\mathcal{T}|)$ [16]. In each round, the algorithm greedily picks the set that covers the largest number of uncovered targets, updates the sets and repeats until all targets are covered. Using the notion of *deviation* we used for optimizing the viewing direction per candidate location, we can also rank different candidate locations according to the quality of coverage they can offer. At iteration i, among all candidates $\{(x, \hat{\alpha})\}$ that can cover the maximum number of targets, we favor the one achieving the minimum $f_1(x, \hat{\alpha}, \alpha^*)$. The greedy algorithm will then return a coverage scheme providing better views while still approximating the minimum number of cameras needed.

To obtain an $O(\log n)$-approximation, the comprehensive set of candidates $\mathcal{P}$ is used. As sweeping over $\mathcal{P}$ to generate $\mathcal{R}$ takes $O(N^5)$ steps, we loosely bound the time complexity of the proposed approximation algorithm by $O(nN^5)$. Similarly, defining a set of configurations $\hat{\mathcal{R}}$ using $\hat{\mathcal{P}}$ from the heuristic spatial discretizer results in an $O(\frac{\rho \cdot R_{max}}{\epsilon_a \cdot \epsilon_r} \cdot n^2 N)$ algorithm.

4 IMPLEMENTING ARGUS

Our goal is to develop a fully autonomous instance of Argus to measure the overhead of the *OLSC Solver* under realistic conditions. We build upon our earlier work on developing an autonomous testbed for multi-drone experiments [31, 47]. Figure 6 depicts the architecture of the Argus prototype that we fully implement as three modules: *Central*, *Client*, and *Multi-homed*.

The Central Module is responsible for localizing quadcopters and targets in 2D and running the *OLSC Solver*. The *OLSC Solver* runs only the *BCPF Sampling* algorithm as it is more efficient while being competitive to the approximation algorithm. In our setup, the *Central Module* is run on a Lenovo ThinkPad Y50. The *Central Module* uses a master camera to obtain the input for its *UAV and Target Localizer* component. We use an Axis 213 PTZ network camera located directly above the testbed area. The master camera is connected to the *Central Module* through an Ethernet cable and provides images at a frequency of 30 *Hz*. The *UAV and Target Localizer* filters the noise and locates all quadcopters and targets in the image. Each image is then passed to the *Adaptive Tracker* which makes use of the last known location of each drone or target to localize it in the scene. This approach reduces the processing time of the localization step by performing local searches in the image for drones and targets.

A Client Module is the mobile camera component of the system. We choose a quadcopter platform for its low-cost, small size, and maneuverability even in small spaces. In particular, we use the Parrot AR.Drone 2.0 [33] which is equipped with an ARM processor running an embedded Linux 2.6.32 BusyBox. The Parrot AR. Drone 2.0 is also equipped with two cameras: a front 720p camera with a 93° lens and a vertical QVGA camera with a 64° lens. We mainly use the front camera in our experiments. We allow the client to add as many sensors as needed which can help obtain more surveillance information (e.g., depth sensors) or better navigate the drone (e.g., accelerometers). To this end, we use an External Processing Unit (EPU) which collects recorded video from the camera and sensory readings from the external sensors. Communication between the drone and the EPU is performed over Wi-Fi.

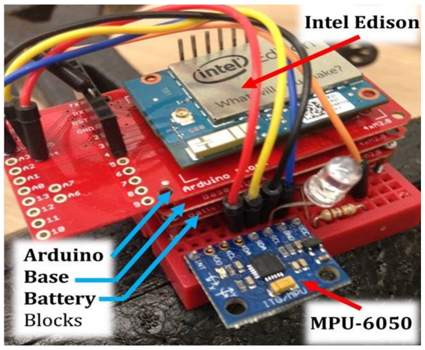


Figure 7: EPU setup.

For the EPU, we use Intel Edison which is an ultra-small computing device powered by an Atom system-on-chip dual-core CPU at 500 *MHz* and 1 *GB* RAM. Intel Edison has integrated Wi-Fi, Bluetooth, 50 multiplexed GPIO interfaces, and runs Yocto Linux. The EPU is powered by a Battery Block. Additional sensors are hardwired into the EPU using an Arduino block. We use an Inertial Measurement Unit (IMU) as the external sensor in this setup. The IMU improves the autonomous navigation of drones by providing finer grain yaw angles to help with drone orientation. Another benefit of mounting EPUs on quadcopters is the extra processing power and added flexibility they offer. We can install our own drivers, operating systems, integrated sensors, and overcome the typical closed-nature restriction of off-the-shelf quadcopters. We attach the EPU on top and close to the center of gravity of the drone to avoid disturbing the balance and stability of the vehicle. The EPU setup is shown in Figure 7.

The Multi-Homed Module is a special set of sub-modules that can belong to either the *Central* or *Client* module. The flexibility of housing its sub-modules allows easy migration between a centralized and a distributed platform. We use two such sub-modules: *UAV Flight Navigator* and *Algorithmic Processor*. The *UAV Flight Navigator* receives a set of parameters from the *UAV Localizer* (i.e., 2D

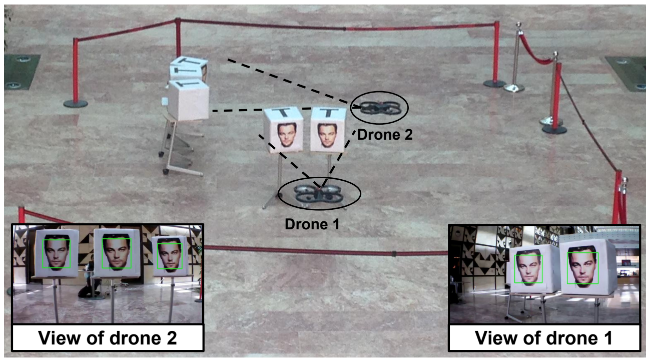


Figure 8: Experimental setup.

	Horizontal Motion	Rotation	Hovering
Power (Watt)	65.625	68.750	61.125
Energy per meter (Joule/m)	65.630	68.750	N/A

Table 1: Energy consumption of typical drone maneuvers measured over time (i.e., power) and distance traveled.

coordinates) and the *IMU* (i.e., yaw angles) and controls the drone through its navigation parameters to properly fly to the desired coordinates. The *Algorithmic Processor* handles any sensory information processing (i.e., *Fine Grain Context Detector* functionalities).

Experimental setup: The testbed covers an area of 30 m^2 where we place one to five synthetic targets positioned in configurations that require a maximum of two drones (Figure 8); for scenarios with one to three targets, we only need one drone for coverage, and for scenarios with four or five targets, we need two drones. Our target apparatus is a white box mounted on top of a podium with a printed face attached to one of its vertical sides to represent the significant perspective. A letter "T" on the top side of the box helps simplify location and orientation estimation.

Noting that drone control and sensory information analysis are processing-intensive operations, we aim to achieve real-time processing with minimal latency. To this end, we handle the autonomous control of drones on the *Central Module* and distribute the processing of video feeds from each drone under the *Client Module* running on the drone's EPU to detect faces on the sides of targets. We set $R_{max} = 2\ m$ and $\theta = 75°$, which is slightly smaller than the camera's actual AOV, to avoid cases where covered targets barely fit in the captured frame.

Real-time adaptation to target mobility: Argus needs to repeatedly invoke the *OLSC Solver* to respond to updates in the locations of either targets or obstacles. As shown in Section 5, the algorithm can take up to a few seconds based on the number of targets. Until a drone is assigned a new configuration, decisions have to be made locally by each drone to respond to target mobility in real-time. Argus allows drones to hover in place or move horizontally for short distances to maintain target coverage using standard tracking algorithms [32]. Local decisions, based on the energy footprint of each maneuver, are computed on the EPU to minimize the cost of the proposed strategy. Table 1 summarizes the power consumption of typical drone maneuvers. To avoid large rotations or displacements, drones cooperate to keep targets in view [24]. This autonomous behavior also serves as a fallback strategy if the communication link between the *Central Module* and the drone is broken.

5 EVALUATION

We start our evaluation by examining the behavior of Argus on the testbed. We compare the new *OLS* model to the traditional model of targets as mere points and analyze the overhead of the *OLSC Solver* within the system. Then, we evaluate Argus at scale via simulations and compare the proposed algorithms for the *OLSC* against a baseline.

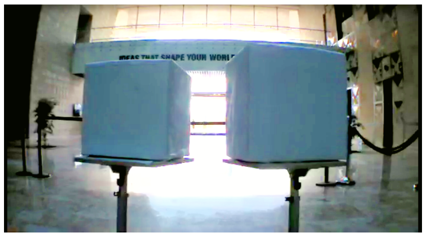
5.1 Argus Evaluation

We demonstrate the pitfalls of traditional target coverage algorithms, where target size and orientation are not taken into account [44], by comparing them to Argus in a realistic setting. Then, we break down the delays in the presented system and compare against the delay introduced by the *OLSC Solver*.

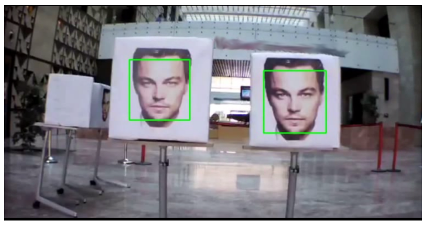
OLS vs. blips on the radar: To demonstrate the advantages of the proposed model, we take for example the surveillance footage in Figures 9, 10, and 11. Recall that these images are captured by the master camera and the front cameras on each drone. We choose this particular target configuration to put the quality of coverage of a typical target coverage algorithm in contrast with *OLS*. Figure 9(a) shows two targets covered from the opposite direction of their significant perspective because typical coverage algorithms do not take target orientation into account. Moreover, typical target coverage algorithms do not take target size and potential occlusions between targets into account, which is demonstrated in Figure 10(a) where one target occludes two other targets. When *OLS* is employed, these issues are resolved and cameras are positioned to properly cover the targets as shown in Figures 9(b) and 10(b). Note that the generated configurations are based on target width and camera constraints (e.g., R_{max} of 2 m) which represents a constraint on the quality of images used for face detection.

Implementation delay breakdown: Figure 12 shows the CDF of the processing time per frame, which captures the overall processing performed by the *Central Module* apart from the *OLSC Solver*. This processing includes image fetching, decoding and preprocessing, target localization, drone localization and communication. Our target apparatus can be detected efficiently within a few milliseconds. This reduces the processing time per frame as complex targets would take longer to detect (e.g., 120 ms per frame for human body orientation estimation [22]).

The difference in processing time per frame for one to three targets and four and five targets is dominated by the overhead of handling the extra drone. This added overhead can be seen in the CDF of localization time in Figure 13. Recall that when the drone makes large displacements, locality over consecutive frames is lost. This occasionally forces the algorithm to search the entire frame, resulting in the skewed shape of the CDF observed in Figure 13. In our experiments, the *UAV Localizer* has to be invoked at a minimum frequency of 8 Hz for smooth control of the drone.

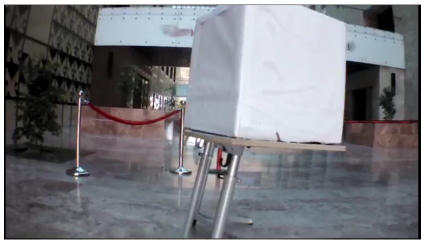


(a) Targets may be covered from behind.

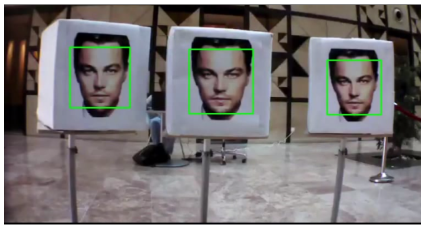


(b) Target orientation is taken into account.

Figure 9: Comparing the view from Drone 1 under a typical target coverage algorithm (top) and OLSC (bottom).

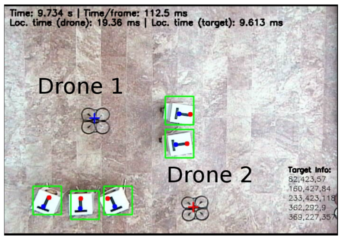


(a) Targets may occlude one another.

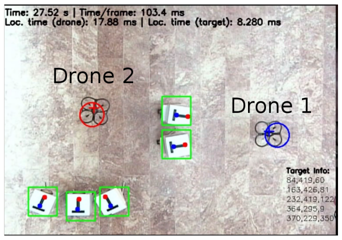


(b) Potential occlusions are taken into account.

Figure 10: Comparing the view from Drone 2 under a typical target coverage algorithm (top) and OLSC (bottom).


Time: 9.734 s | Time/frame: 112.5 ms
Loc. time (drone): 19.36 ms | Loc. time (target): 9.613 ms
Drone 1
Drone 2
Target info:

(a) Drones cover targets from wrong angles.


Time: 27.52 s | Time/frame: 103.4 ms
Loc. time (drone): 17.88 ms | Loc. time (target): 8.280 ms
Drone 2
Drone 1
Target info:

(b) Drones properly cover all targets.

Figure 11: Top views from the master camera showing drone configurations corresponding to Figures 9 and 10.

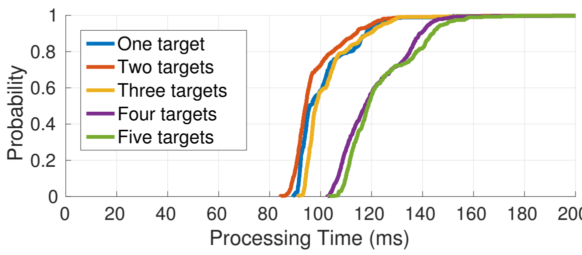


Figure 12: CDF of processing time per frame including image fetching, target and drone localization, and drone communication.

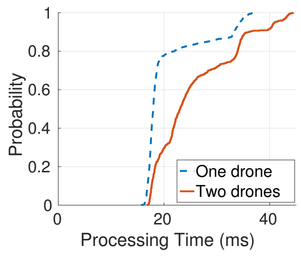


Figure 13: CDF of processing time of the *UAV Localizer*.

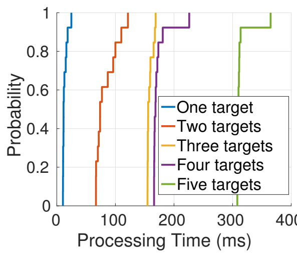


Figure 14: CDF of processing time of the *OLSC Solver*.

OLSC as a component of a surveillance system: We compare the processing time per frame, which corresponds to the overhead of the *Coarse Grain Context Detector*, to the overhead of the *OLSC Solver* (Figure 2). Figure 14 shows the CDF of the processing time of the *OLSC Solver* for the number of targets in our tests. The solver is implemented in MATLAB and we expect it can be significantly optimized. Still, with five targets, the solver can be invoked once for every three processed frames. As mentioned in the previous section, several techniques can be exploited to maintain target coverage while the solver is running. This task is made easier by the ability to invoke the solver at a relatively high frequency (i.e., one third the frequency of updates in the input parameters).

5.2 Argus at Scale

We evaluate, through MATLAB simulations, the performance of the proposed coverage algorithms under large scale conditions that we cannot test on the prototype. We compare the performance of the approximation algorithm to the BCPF sampling heuristic with two levels of granularity for angular sampling using an ϵ_a of 0.01 and 0.1 rad and an ϵ_r of R_{max}.

As a baseline for comparison, we present a **grid sampling heuristic**. We use a simple discretization of the search space: a uniform grid of $\epsilon \times \epsilon$ cells. As $\epsilon \longrightarrow 0$, grid points would hit all possible intersection areas of target CPFs. If $w \times h$ are the dimensions of the bounding box of $\mathcal{T}$, the number of grid points will be $O(\frac{w \cdot h}{\epsilon^2})$, but is otherwise independent of $|\mathcal{T}|$. Treating these points as candidate locations, we generate representative coverage configurations at each point by an angular sweep before running the greedy selection scheme, which amounts to a runtime of $O(\frac{w \cdot h}{\epsilon^2} \cdot nN)$. We use this naive approach to verify the effectiveness of our proposed method in finding appropriate candidate points to minimize the number of cameras needed. To do so, we use relatively small instances of *OLSC* such that ϵ need not be too small and the runtime and memory requirements of the grid heuristic are feasible.

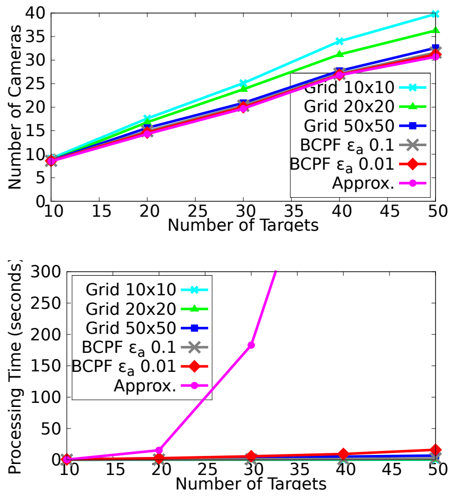


Figure 15: Comparing all algorithms for increasing numbers of targets.

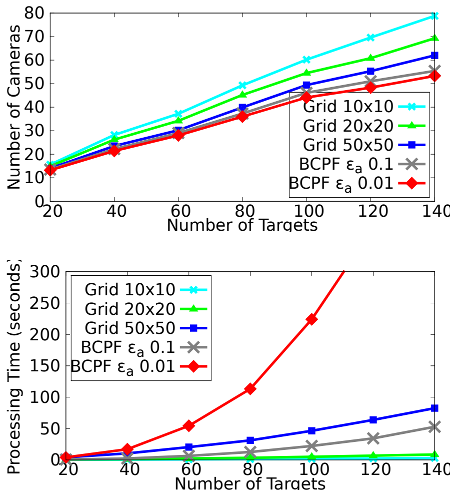


Figure 16: Comparing all heuristics for increasing numbers of targets.

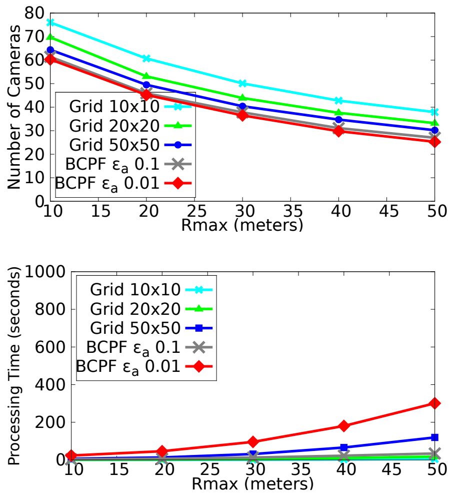


Figure 17: Effect of varying R_{max} for a fixed number of targets.

Parameter	Range	Nominal value
Dimensions	$100\ m \times 100\ m$	$100\ m \times 100\ m$
Target Width	$1\ m$	$1\ m$
AOV	$40° - 140°$	$100°$
Target count	10 - 140	30 (small), 80 (large)
$R_{\max}$	$10\ m$ - $50\ m$	$20\ m$ (small), $30\ m$ (large)

Table 2: Simulation parameters.

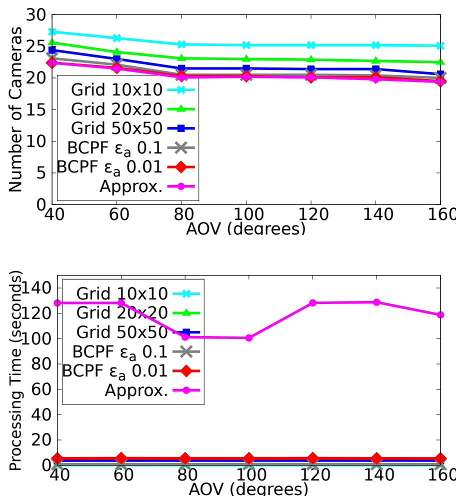


Figure 18: Effect of varying angle of view (AOV) for a fixed number of targets.

We evaluate the algorithms in the extreme case where all present objects are targets (i.e., no obstacles). Since both targets and obstacles act as occluders while only targets need to be covered, this setup requires maximal computations for the chosen number of objects. The goal of this evaluation is to show the effect of changing the number of targets, range, and AOV on the number of drones and processing time required to perform the coverage task under the proposed model. Targets are placed at random locations with random orientations over the area of interest such that they do not overlap. The default values of simulation parameters are shown in Table 2 for both small and large scenarios. We use small scenarios to evaluate the approximation algorithm, which suffices to show the advantages of sampling, and use larger scenarios to compare the different heuristics. We use three resolutions of grid sampling: Grid 10x10 (sparse), Grid 20x20 (medium) and Grid 50x50 (dense) for ϵ set to 10 m, 5 m and 2 m, respectively.

Figure 15 shows the effect of increasing the number of targets. The approximation algorithm produces the best performance in terms of the number of cameras required while taking orders of magnitude more time than sampling approaches due to its higher complexity. The approximation algorithm exceeds a minute per calculation for less than 25 targets while BCPF sampling computes a coverage for 140 targets in around a minute with $\epsilon_a = 0.1\ rad$.

Figure 16 contrasts the performance of sampling approaches in large scale scenarios. Grid sampling provides a comparable number of cameras for small numbers of targets where it is unlikely to have compact configurations of CPF intersections that grid sampling might miss. However, it is clear that BCPF sampling is superior in terms of the number of cameras. Moreover, for a moderate ϵ_a of 0.1 rad, BCPF sampling outperforms grid sampling requiring 12% less cameras and running up to 2x faster on 140 targets.

Figure 17 shows the effect of increasing R_{max} which increases the area covered by each CPF. Having larger CPFs increases the number of regions to be considered in the approximation algorithm and the number of CPFs that a sample point can belong to in the sampling approaches. On the other hand, changing the value of the AOV, shown in Figure 18, does not impact the processing time by much as it does not increase the area of the CPF considerably. However, it increases the number of targets included at each step of sweeping which reduces the number of cameras needed for coverage.

Based on our simulation results, BCPF sampling is the method of choice for a wide range of scenarios as it combines time and resource efficiency especially for large numbers of targets.

6 RELATED WORK

Area and target coverage: The goal of area coverage algorithms is to detect any activity of interest within a certain area in a sensor network deployment. Several approaches to area coverage have been studied including static randomly deployed sensors [12] and strategically placed mobile sensors [19] using either isotropic [25] or anisotropic sensors [56]. The related problem of barrier coverage was studied in [35], where the objective is to detect any targets crossing the barrier into an area of interest. To cover a set of targets within an area, target coverage algorithms were studied in randomly deployed sensors [1, 4, 29], or strategically placed directional sensors [43, 44]. Target coverage using static randomly deployed Pan-Tilt-Zoom (PTZ) cameras, that possibly zoom in to obtain better views, was shown to be NP-hard and a 2-approximation algorithm was presented [29]. *We propose a more realistic model for target coverage by visual sensors that greatly enhances the point model typically used by earlier algorithms [5]. Our approach requires fewer sensors compared to area coverage techniques as it only attempts to cover the present targets rather than the whole area of interest. We establish lower bounds on minimizing the number of sensors required by the new model [46] and develop a matching approximation algorithm in Section 3.*

Full-view coverage: Full-view coverage is a variant of area coverage with the extra objective of ensuring that any target is covered from all angles [53]. [55] studies the necessary conditions for full-view coverage in static camera deployments and [28] studies full-view coverage using heterogeneous mobile cameras. Full view barrier coverage was then introduced [52] and further extended to accommodate stochastic deployments in [57]. Taking self-occlusions into account, ensuring all sides of a convex target are always visible was studied in [49]. *Our proposed approach is different in two aspects: 1) It overcomes occlusion scenarios and takes target size into account in addition to target orientation. 2) It is concerned with target coverage rather than area coverage which is the main concern of full-view coverage.*

Video capture using drones: There has been a growing interest in using drones and drone swarms for surveillance and video capture [11]. In such applications, several challenges including target mobility and low quality footage (e.g., due to distance) were studied in [26]. For mobile target tracking, using either a single drone [41] or multiple drones [39] can be used for persistent tracking. Such applications focus on target coverage without restricting the angles from which targets are viewed. Autonomous cinematography is another application for drones, beyond coverage and tracking, the aesthetic quality plays a key role in viewpoint planning [30]. In earlier work, we developed several target coverage algorithms for targets represented as points and deployed them on our testbed [31, 43, 44, 47]. *In this paper, our work leverages recent advances in drone technologies to develop an autonomous system that utilizes our enhanced target model and demonstrate the feasibility of running the proposed coverage algorithms on a real system. We envision extensions of the proposed model to accommodate specific aesthetic or gesture capture requirements to allow more control over the quality of coverage as required for persistent tracking and cinematography.*

Art Gallery Problem: A classical problem in discrete and computational geometry asks for the minimum number of guards required to see every point in an art gallery represented by a polygon with or without holes [50]. Several variants were introduced constraining guard placement (e.g. point, vertex, or edge) and coverage (e.g., star shaped) [37]. In particular, the art gallery illumination problem considered guards having a limited angle of view [6, 10]. Visibility algorithms have found many applications in wireless communications, sensor networks, surveillance, and robotics. However, several variants were shown to be NP-hard [45]. In addition, inapproximability results for art gallery coverage with and without holes were shown in [20] and also for the illumination of art galleries without holes [2]. On the approximation side, the works in [18, 23] presented algorithms for the coverage of art galleries with and without holes, respectively. *We settle the hardness and inapproximability of art gallery illumination for polygons with holes and use this to prove the hardness of OLSC [46]. We also present a best-possible approximation algorithm for OLSC based on a spatial subdivision derived from the coverage constraints. The novelty of our algorithm lies in the incorporation of a limited angle of view camera model with our newly proposed target model.* Earlier approximation algorithms relied on triangulations [18] or sampling [23] while assuming omnidirectional cameras.

7 CONCLUSION

We presented Argus, a system that uses drones to provide better coverage of targets taking into account their size, orientation, and potential occlusions. We started by introducing *OLS*, a novel geometric model that captures wide oriented targets and the conditions necessary for their coverage. Then, we formulated the *Oriented Line Segment Coverage Problem (OLSC)* that aims at minimizing the number of cameras required to cover a set of targets represented by this new model. We devised a best-possible $O(\log n)$-approximation algorithm and a sampling heuristic that runs up to 100x faster while performing favorably compared to the provably-bounded approximation algorithm. Finally, we developed a fully autonomous prototype that uses quadcopters to monitor synthetic targets in order to measure the overhead of the proposed algorithms in realistic scenarios and show the improved quality of coverage provided by the new model.

8 ACKNOWLEDGMENTS

This work was made possible by the NPRP grant #4-463-2-172 from the Qatar National Research Fund (a member of Qatar Foundation). The authors would like to thank the IPSN Shepherd, Tam Vu, and the anonymous reviewers for providing excellent feedback.

REFERENCES

[1] Ahmed Abdelkader, Moamen Mokhtar, and Hazem El-Alfy. 2012. Angular Heuristics for Coverage Maximization in Multi-camera Surveillance. In *AVSS'12*.

[2] Ahmed Abdelkader, Ahmed Saeed, Khaled Harras, and Amr Mohamed. 2015. The Inapproximability of Illuminating Polygons by α-Floodlights. In *CCCG'15*.

[3] Fadel Adib, Zach Kabelac, Dina Katabi, and Robert C. Miller. 2014. 3D Tracking via Body Radio Reflections. In *NSDI'14*.

[4] Jing Ai and Alhussein A Abouzeid. 2006. Coverage by directional sensors in randomly deployed wireless sensor networks. *Journal of Combinatorial Optimization* 11, 1 (2006), 21–41.

[5] M Amac Guvensan and A Gokhan Yavuz. 2011. On coverage issues in directional sensor networks: A survey. *Ad Hoc Networks* 9, 7 (2011), 1238–1255.

[6] Jay Bagga, Laxmi Gewali, and David Glasser. 1996. The Complexity of Illuminating Polygons by alpha-flood-lights. In *CCCG'96*.

[7] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. 2006. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*.

[8] Andreas Bircher, Kostas Alexis, Michael Burri, Philipp Oettershagen, Sammy Omari, Thomas Mantel, and Roland Siegwart. 2015. Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics. In *ICRA'15*.

[9] Volker Blanz, Patrick Grother, P Jonathon Phillips, and Thomas Vetter. 2005. Face recognition based on frontal views generated from non-frontal images. In *CVPR'05*.

[10] Prosenjit Bose, Leonidas Guibas, Anna Lubiw, Mark Overmars, Diane Souvaine, and Jorge Urrutia. 1997. The floodlight problem. *International Journal of Computational Geometry & Applications* 7, 01n02 (1997).

[11] Axel Bürkle. 2009. Collaborating miniature drones for surveillance and reconnaissance. In *Proc. of SPIE Unmanned/Unattended Sensors and Sensor Networks VI*, Vol. 7480.

[12] Paz Carmi, Matthew J Katz, and Nissan Lev-Tov. 2007. Covering points by unit disks of fixed location. In *Algorithms and Computation*. 644–655.

[13] Ronald Chang, Teck Wee Chua, Karianto Leman, Hee Lin Wang, and Jie Zhang. 2013. Automatic cooperative camera system for real-time bag detection in visual surveillance. In *ICDSC'13*.

[14] Cheng Chen and Jean-Marc Odobez. 2012. We are not contortionists: coupled adaptive learning for head and body orientation estimation in surveillance video. In *CVPR'12*.

[15] Hsiuao-Ying Chen, Chung-Lin Huang, and Chih-Ming Fu. 2008. Hybrid-boost learning for multi-pose face detection and facial expression recognition. *Pattern Recognition* 41, 3 (2008), 1173–1185.

[16] Vasek Chvatal. 1979. A greedy heuristic for the set-covering problem. *Mathematics of operations research* 4, 3 (1979), 233–235.

[17] Simon Denman, Tristan Kleinschmidt, David Ryan, Paul Barnes, Sridha Sridharan, and Clinton Fookes. 2015. Automatic surveillance in transportation hubs: No longer just about catching the bad guy. *Expert Systems with Applications* 42, 24 (2015).

[18] Ajay Deshpande, Taejung Kim, Erik Demaine, and Sanjay Sarma. 2007. A Pseudopolynomial Time $O(\log n)$-Approximation Algorithm for Art Gallery Problems. *Algorithms and Data Structures* (2007), 163–174.

[19] Santpal Singh Dhillon and Krishnendu Chakrabarty. 2003. Sensor placement for effective coverage and surveillance in distributed sensor networks. In *WCNC'03*.

[20] Stephan Eidenbenz, Christoph Stamm, and Peter Widmayer. 2001. Inapproximability results for guarding polygons and terrains. *Algorithmica* 31, 1 (2001), 79–113.

[21] Rachel L Finn and David Wright. 2012. Unmanned aircraft systems: Surveillance, ethics and privacy in civil applications. *Computer Law & Security Review* 28, 2 (2012), 184–194.

[22] Fabian Flohr, Madalin Dumitru-Guzu, Julian FP Kooij, and Dariu M Gavrila. 2015. A probabilistic framework for joint pedestrian head and body orientation estimation. *IEEE Transactions on Intelligent Transportation Systems* 16, 4 (2015), 1872–1882.

[23] Héctor González-Baños and Jean-Claude Latombe. 2001. A randomized art-gallery algorithm for sensor placement. In *SoCG'01*.

[24] Karol Hausman, Jörg Müller, Abishek Hariharan, Nora Ayanian, and Gaurav S Sukhatme. 2015. Cooperative multi-robot control for target tracking with onboard sensing. *The International Journal of Robotics Research* 34, 13 (2015), 1660–1677.

[25] Bruno Hexsel, Nilanjan Chakraborty, and K Sycara. 2011. Coverage control for mobile anisotropic sensor networks. In *ICRA'11*.

[26] Hwai-Jung Hsu and Kuan-Ta Chen. 2015. Face Recognition on Drones: Issues and Limitations. In *DroNet '15*.

[27] Weiming Hu, Tieniu Tan, Liang Wang, and Steve Maybank. 2004. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 34, 3 (2004), 334–352.

[28] Yitao Hu, Xinbing Wang, and Xiaoying Gan. 2014. Critical Sensing Range for Mobile Heterogeneous Camera Sensor Networks. In *INFOCOM'14*.

[29] Matthew P Johnson and Amotz Bar-Noy. 2011. Pan and scan: Configuring cameras for coverage. In *INFOCOM'11*.

[30] Niels Joubert, Dan B Goldman, Floraine Berthouzoz, Mike Roberts, James A Landay, Pat Hanrahan, and others. 2016. Towards a Drone Cinematographer: Guiding Quadrotor Cameras using Visual Composition Principles. *arXiv preprint arXiv:1610.01691* (2016).

[31] Mouhyemen Khan, Sidra Alam, Amr Mohamed, and Khaled A Harras. 2016. Simulating Drone-be-Gone: Agile Low-Cost Cyber-Physical UAV Testbed (Demonstration). In *AAMAS'16*.

[32] Jeong Woon Kim and David Hyunchul Shim. 2013. A vision-based target tracking control system of a quadrotor by using a tablet computer. In *International Conference on Unmanned Aircraft Systems (ICUAS'13)*. 1165–1172.

[33] Tomáš Krajník, Vojtěch Vonásek, Daniel Fišer, and Jan Faigl. 2011. AR-drone as a platform for robotic research and education. In *International ConferenceResearch and Education in Robotics*.

[34] Purushottam Kulkarni, Deepak Ganesan, Prashant Shenoy, and Qifeng Lu. 2005. SensEye: a multi-tier camera sensor network. In *MM'05*.

[35] Santosh Kumar, Ten H Lai, and Anish Arora. 2005. Barrier coverage with wireless sensors. In *MobiCom'05*.

[36] Nancy G La Vigne, Samantha S Lowry, Joshua A Markman, and Allison M Dwyer. 2011. *Evaluating the Use of Public Surveillance Cameras for Crime Control and Prevention.* Urban Institute, Justice Policy Center.

[37] Der-Tsai Lee and Arthurk Lin. 1986. Computational complexity of art gallery problems. *IEEE Transactions on Information Theory* 32, 2 (1986), 276–282.

[38] Chung-Ching Lin, Sharathchandra U. Pankanti, Karthikeyan Natesan Ramamurthy, and Aleksandr Y. Aravkin. 2015. Adaptive As-Natural-As-Possible Image Stitching. In *CVPR'15*.

[39] Matthias Mueller, Gopal Sharma, Neil Smith, and Bernard Ghanem. 2016. Persistent Aerial Tracking system for UAVs. In *IROS'16*.

[40] Yash Mulgaonkar and Vijay Kumar. 2014. Towards Open-Source, Printable Pico-Quadrotors. In *Proc. Robot Makers Workshop, Robotics: Science Systems Conf.*

[41] Tayyab Naseer, Jürgen Sturm, and Daniel Cremers. 2013. Followme: Person following and gesture recognition with a quadrocopter. In *IROS'13*.

[42] Prabhu Natarajan, Pradeep K. Atrey, and Mohan Kankanhalli. 2015. Multi-Camera Coordination and Control in Surveillance Systems: A Survey. *ACM Transactions on Multimedia Computing, Communications, and Applications* 11, 4 (2015), 57:1–57:30.

[43] Azin Neishaboori, Ahmed Saeed, Khaled A. Harras, and Amar Mohamed. 2014. Low Complexity Target Coverage Heuristics Using Mobile Cameras. In *MASS'14*.

[44] Azin Neishaboori, Ahmed Saeed, Khaled A. Harras, and Amr Mohamed. 2014. On Target Coverage in Mobile Visual Sensor Networks. In *MobiWac'14*.

[45] Joseph O'Rourke and Kenneth Supowit. 1983. Some NP-hard polygon decomposition problems. *Transactions on Information Theory* 29, 2 (1983), 181–190.

[46] Ahmed Saeed, Ahmed Abdelkader, Mouhyemen Khan, Azin Neishaboori, Khaled A Harras, and Amr Mohamed. 2017. On Realistic Target Coverage by Autonomous Drones. *arXiv preprint arXiv:1702.03456* (2017).

[47] Ahmed Saeed, Azin Neishaboori, Amr Mohamed, and Khaled A. Harras. 2014. Up and away: A visually-controlled easy-to-deploy wireless UAV Cyber-Physical testbed. In *WiMob'14*.

[48] Nurcan Tezcan and Wenye Wang. 2008. Self-orienting wireless multimedia sensor networks for occlusion-free viewpoints. *Computer networks* 52, 13 (2008), 2558–2567.

[49] Pratap Tokekar and Volkan Isler. 2014. Polygon guarding with orientation. In *ICRA'14*.

[50] Jorge Urrutia and others. 2000. Art gallery and illumination problems. *Handbook of computational geometry* 1, 1 (2000), 973–1027.

[51] Gert Vegter. 1990. The visibility diagram: A data structure for visibility problems and motion planning. In *2nd Scandinavian Workshop on Algorithm Theory*.

[52] Yi Wang and Guohong Cao. 2011. Barrier coverage in camera sensor networks. In *MobiHoc '11*.

[53] Yi Wang and Guohong Cao. 2011. On full-view coverage in camera sensor networks. In *INFOCOM'11*.

[54] Daniel Weinland, Mustafa Özuysal, and Pascal Fua. 2010. Making action recognition robust to occlusions and viewpoint changes. In *European Conference on Computer Vision*. 635–648.

[55] Yibo Wu and Xinbing Wang. 2012. Achieving full view coverage with randomly-deployed heterogeneous camera sensors. In *ICDCS'12*.

[56] Enes Yildiz, Kemal Akkaya, Esra Sisikoglu, and Mustafa Y Sir. 2014. Optimal Camera Placement for Providing Angular Coverage in Wireless Video Sensor Networks. *IEEE Trans. Comput.* 63, 7 (2014), 1812–1825.

[57] Zuoming Yu, Fan Yang, Jin Teng, A.C. Champion, and Dong Xuan. 2015. Local face-view barrier coverage in camera sensor networks. In *INFOCOM'15*.