On Realistic Target Coverage by Autonomous Drones

AHMED SAEED, Georgia Institute of Technology
AHMED ABDELKADER, University of Maryland, College Park
MOUHYEMEN KHAN, Georgia Institute of Technology
AZIN NEISHABOORI and KHALED A. HARRAS, Carnegie Mellon University
AMR MOHAMED, Qatar University

Low-cost mini-drones with advanced sensing and maneuverability enable a new class of intelligent sensing systems. To achieve the full potential of such drones, it is necessary to develop new enhanced formulations of both common and emerging sensing scenarios. Namely, several fundamental challenges in visual sensing are yet to be solved including (1) fitting sizable targets in camera frames; (2) positioning cameras at effective viewpoints matching target poses; and (3) accounting for occlusion by elements in the environment, including other targets. In this article, we introduce Argus, an autonomous system that utilizes drones to collect target information incrementally through a two-tier architecture. To tackle the stated challenges, Argus employs a novel geometric model that captures both target shapes and coverage constraints. Recognizing drones as the scarcest resource, Argus aims to minimize the number of drones required to cover a set of targets. We prove this problem is NP-hard, and even hard to approximate, before deriving a best-possible approximation algorithm along with a competitive sampling heuristic which runs up to 100× faster according to large-scale simulations. To test Argus in action, we demonstrate and analyze its performance on a prototype implementation. Finally, we present a number of extensions to accommodate more application requirements and highlight some open problems.

CCS Concepts: • **Theory of computation → Computational geometry**; • **Computer systems organization → Sensor networks**;

Additional Key Words and Phrases: 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. 2019. On Realistic Target Coverage by Autonomous Drones. *ACM Trans. Sen. Netw.* 15, 3, Article 32 (June 2019), 33 pages.
https://doi.org/10.1145/3325512

A preliminary version of this work was published in the following conference paper:
Ahmed Saeed, Ahmed Abdelkader, Mouhyemen Khan, Azin Neishaboori, Khaled A. Harras, and Amr Mohamed. 2017. Argus: Realistic target coverage by drones. In *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'17)*, 12 pages.

Authors' addresses: A. Saeed and M. Khan, School of Computer Science, Institute of Technology, Atlanta, USA; emails: amsmti3@gatech.edu, mouhyemen.khan@gmail.com; A. Abdelkader, Department of Computer Science, University of Maryland, College Park, MD, USA; email: akader@cs.umd.edu; A. Neishaboori and K. A. Harras, School of Computer Science, Carnegie Mellon University, Pittsburgh, USA; emails: kharras@cs.cmu.edu, azin.neishaboori@gmail.com; A. Mohamed, Department of Computer Science and Engineering, Qatar University, Doha, Qatar; email: amrm@ieee.org.

1550-4859/2019/06-ART32 $15.00
https://doi.org/10.1145/3325512

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 prevention (La Vigne et al. 2011). Moreover, visual sensor systems enable the automation of complex tasks like crowd counting, event detection, object tracking, target identification, and activity recognition (Denman et al. 2015). 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).

With the increasing availability of low-cost mini-drones, visual sensors are currently finding applications beyond surveillance in disaster response (Erdelj et al. 2017), structural inspection (Bircher et al. 2017), sport streaming (Wang et al. 2017), and cinematography (Joubert et al. 2016). We begin by examining the application needs where such compact mobile sensors can be exploited by more sophisticated sensor systems. Before proceeding to describe the system, we design to fill this gap; we review recent progress in mini-drone technologies, which is the driving force behind our work, and highlight the anticipated developments which will enable more advanced systems like the one we propose. Finally, we summarize our contributions and provide an overview of the structure of this article.

1.1 Practical Motivation

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 (Blanz et al. 2005) or within a certain viewing angle (Bay et al. 2006).
- The relevant features of targets should be fully captured, preferably by a single camera to avoid the challenging task of stitching images from multiple viewpoints (Lin et al. 2015).
- As a prerequisite, occlusions and blind spots should be avoided by positioning cameras accordingly (Weinland et al. 2010).

An extreme approach to some of these challenges is to increase the density of deployed cameras such that any target, within the area of interest, is covered from all angles (Wang and Cao 2011a, 2011b). This approach requires a large number of cameras, incurring a rather high cost (Yu et al. 2015). 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 that targets will be facing the camera (Tezcan and Wang 2008). It is clear that modeling targets by more than mere "blips" can improve the quality of the collected footage and therefore enable more effective visual sensing systems.

To the best of our knowledge, no earlier work in smart surveillance tackled these challenges simultaneously. Very recently, Nägeli et al. (2017) and Galvane et al. (2017) presented novel drone systems for automated cinematography, taking into account framing objectives, occlusions, and collision-avoidance. However, they focus on scenarios involving a known number of actors where a user specifies the desired location of each actor in the camera frame. More crucially, their experiments utilize a motion capture system where actors wear helmets equipped with tracking chips. In contrast, our system leverages weaker setups to estimate target parameters and deliver close-up views of the targets of interest without any user intervention.

1.2 Next Generation Sensors

The recent years have witnessed rapid developments in mini-drone technologies. Amateurs and professionals alike now have access to a wide range of drone sizes and capabilities for fun and profit. A number of proposed drone classifications, along with multiple examples, can be found in Hassanalian and Abdelkefi (2017) and Shakeri et al. (2018). We are particularly interested in drones at the lower end of the spectrum exemplified by robotic flying insects (Wood 2008), which are now available as open-hardware (Vanhoutte et al. 2017); see Helbling and Wood (2017) for a recent review. It is remarkable that such small platforms can be equipped with high end sensors, which enables a multitude of previously unforeseen applications. Specifically, there has been equally remarkable progress in camera sensor technologies with unprecedented feature sizes (Koppal 2016). Despite the stringent constraints of weight and energy consumption on top of the complexity of the required tasks, there has been steady progress in the capabilities of these micro-drones (Tijmons et al. 2017).

We anticipate the utilization of these platforms for futuristic applications through a combination of advanced system architectures and novel user-friendly deployments. As seen in the parallel developments of the *Internet of Things (IoT)* (Aazam et al. 2018a, 2018b; Perera et al. 2014; Xu et al. 2014; Zanella et al. 2014), these sensing platforms have a great potential in virtually every application domain. With the introduction of commercial intelligent personal assistants and home automation technologies like Alexa (Amazon.com 2014) and Google Home (Google 2016), users are becoming more familiar with such ambient devices. We envision that such systems will soon be endowed with mobile agents that live symbiotically with users both indoors and outdoors (Essameldin and Harras 2017; Gedawy et al. 2018; Saeed et al. 2015).

1.3 Overview

In this article, we introduce Argus, an autonomous system that tackles the challenges identified in Section 1.1 by exploiting the rapid advancements in mini-drone technologies and their anticipated applications in surveillance, crowd monitoring (Finn and Wright 2012), infrastructure inspection (Bircher et al. 2017), and cinematography (Joubert et al. 2016). To go beyond traditional coverage, Argus accumulates target information by dynamically controlling the available sensors to estimate target parameters before assigning mobile drones to eliminate blind spots and capture frontal views of the subjects of interest. In particular, the proposed two-tier architecture leverages recent progress in persistent coverage (Schwager et al. 2011). Argus employs a top tier subsystem to detect targets within the monitored area and maintain estimates of their states. Given this information, the lower tier subsystem is responsible for controlling the available drones to obtain high-quality views of the targets of interest. A crucial aspect of the lower tier is the modeling of targets and the positioning of drones to capture target footage. Argus utilizes the *Oriented Line Segment Target Model* (*OLS*), a new geometric model we develop to incorporate target pose, size, and potential occluders. *With drones being the most valuable resource, 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 target and fits a line segment and orientation to estimate its size and pose. While still being simple, the new model is more complex than plain points and requires a more advanced system to estimate its parameters and new algorithms to utilize it. We show that minimizing the number of drones under *OLS* is NP-hard and even hard to approximate. Then, we proceed to 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 100× faster in our simulations with large numbers of targets and various target and camera parameters.

We implement a fully autonomous prototype of Argus with two AR.Drone 2.0 quadcopters fitted with camera sensors and a fixed PTZ-camera. 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. Thus, higher quality coverage can be achieved through a powerful lightweight target model suitable for real-time applications.

1.4 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 particularly relevant applications that have recently received a growing interest.

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 (Bircher et al. 2017). In such scenarios, the number of targets can be very large. We analyze the scalability of Argus in Section 8.2.

Cinematography, both in reality or in virtual worlds, frequently considers the planning of camera trajectories to obtain the desired shots in a given scene (Joubert et al. 2015; Lino et al. 2011). 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 9.

Argus can readily be used in several other applications, e.g., defense and public safety operations (Fraga-Lamas et al. 2016) as well as disaster response (Erdelj et al. 2017). The proposed algorithms can help plan the deployment of autonomous search and rescue robots and troops to scan and secure an area while minimizing the resources allocated to each task. Since Argus takes into account target size, pose, and relative location with respect to the various obstacle in a given area (e.g., buildings, terrain, and foliage), besides possibly more application-specific constraints, it is able to suggest the best locations to engage the targets of interest. The efficiency of the proposed algorithms make them well-suited to the dynamic nature of such scenarios where the deployment configuration may need to be updated frequently.

1.5 Summary of Contributions and Article Organization

The contributions of this article are fourfold:

- We propose and develop 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 prove that computing the minimum number of drones to cover a set of *OLS* targets is NP-hard and even hard to approximate.

— 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 Argus, a complete prototype of the system, to demonstrate the superior quality of coverage the system can offer, and gauge the overhead of the proposed algorithms in action.

The rest of the article is organized as follows. In Section 2, we describe the system architecture with an emphasis on the top tier. Then, we present the new target model and formulate the coverage problem under this model, which is the focus of the lower tier, in Section 3. We establish the hardness of covering *OLS* targets in Section 4. We proceed to analyze the coverage constraints in Section 5 before developing the coverage algorithms in Section 6. In Section 7, we report on the implementation of a full prototype of Argus, which we use together with simulations to evaluate the proposed algorithms in Section 8. Recognizing scenarios where the proposed system may not be adequate, we present a number of possible extensions and other open problems in Section 9. Finally, we present an extensive survey of related work in Section 10 and conclude the article in Section 11.

2 SYSTEM ARCHITECTURE

Argus is a fully autonomous system that aims to capture high-quality video footage of identified targets of interest subject to coverage constraints. Argus employs a two-tier architecture. The top tier, used for coarse grain coverage, provides the location, width, and pose of targets and obstacles. The lower tier uses the information provided by the top tier to provide fine grain coverage using mobile drones; a setup we believe will become more convenient as drones get smaller, e.g., Mulgaonkar and Kumar (2014). Having a hierarchy of surveillance systems allows each tier to be responsible for different tasks (Kulkarni et al. 2005); see the survey in Natarajan et al. (2015).

For the top tier, Argus leverages recent work on persistent coverage (Palacios-Gasós et al. 2017; Schwager et al. 2011, 2017) to monitor an area of interest and estimate basic target information; see the surveys in Nigam (2014) and Khan et al. (2018). Traditionally, static PTZ cameras suffice for this function, especially in indoor environments. For instance, PTZ cameras were used in Chang et al. (2013) to identify the type of bags carried by subjects based on the locations determined by static cameras. Alternatively, the location and pose information of targets can be provided by non-visual means like radar and device-free localization systems (e.g., Adib et al. (2014), which can detect both the location and pose of human subjects using Wi-Fi signals). A more general system was described in Schwager et al. (2011), which accommodates heterogeneous sensors of varying degrees of freedom to achieve the required coverage. More recent proposals enable continuous adaptive coverage of changing environments (Schwager et al. 2017) possibly containing obstacles (Palacios-Gasós et al. 2017). Furthermore, these systems can be extended to take into account both energy (Derenick et al. 2011) and communication constraints (Orfanus et al. 2016; Wang et al. 2016). Using the information collected by the top tier sensors, the lower tier positions mobile drones to eliminate blind spots and capture high-quality views of the targets of interest.

Figure 1 depicts the operational flow of the Argus system. The system consists of three main components: (1) *Coarse Grain Context Detector*, (2) *OLSC Solver*, and (3) *Fine Grain Context Detector*. The *Coarse Grain Context Detector* is responsible for obtaining basic target information, using the information provided by the top tier, which the *OLSC Solver* uses to determine a strategy for positioning the drones in the lower tier. The *OLSC Solver* requires as input the location, width, and pose of each target. The output of the *OLSC Solver* is used to position the lower tier drones to capture high-quality unobstructed footage of whole targets. These images can then be further

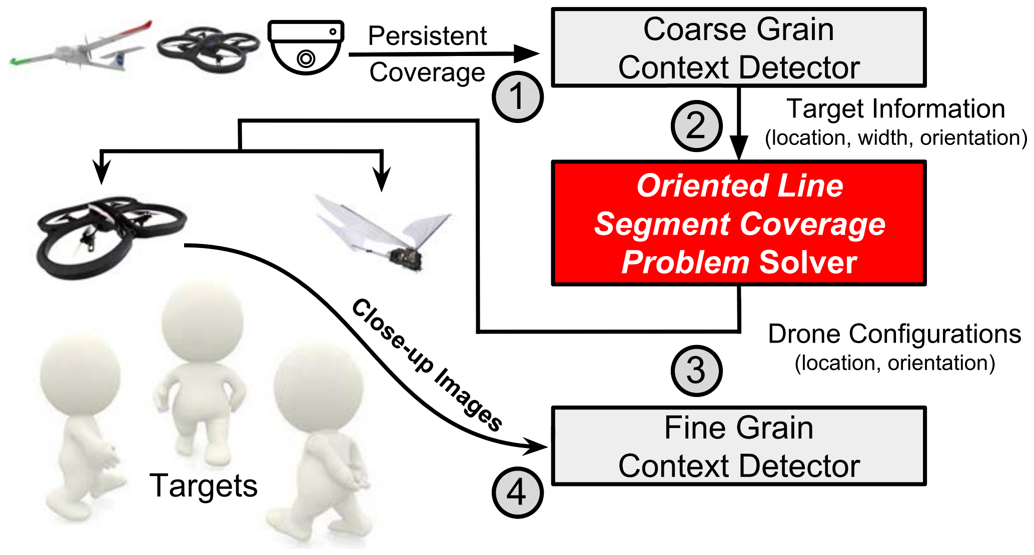

Persistent Coverage
1
Coarse Grain Context Detector
2
Target Information
(location, width, orientation)
Oriented Line Segment Coverage Problem Solver
Drone Configurations
(location, orientation)
3
Close-up Images
Fine Grain Context Detector
4
Targets

Fig. 1. Operational flow of Argus.

processed, through the *Fine Grain Context Detector*, by different context extraction algorithms (Hu et al. 2004).

We realize that implementing each component is challenging in its own right with many open research problems. In particular, we propose a generic design which can be adapted to the specific application and context in question. Figure 1 hints at the different options for implementing each layer using the appropriate combination of sensors that can range from high altitude surveillance drones to flapping wing micro-drones like the one shown in the figure from Sahai et al. (2013). Hence, we focus on the *OLSC Solver* and present vanilla approaches to the other components of Argus using typical hardware that can be found at most research labs.

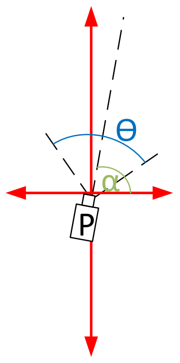
3 THE COVERAGE PROBLEM

In this section, we formulate the problem of covering a set of targets under the new *OLS* model as required by the lower tier of the Argus system. We start by defining a convenient abstract model for both sensors and targets along with the required conditions for coverage. Then, we formally define the coverage problem that aims to minimize the number of drones needed for coverage. Finally, we highlight and justify the assumptions we make on the top tier in order to provide the information needed to provide the input to the coverage problem.

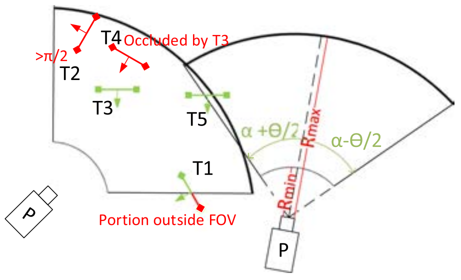
3.1 Definitions

We define the geometric models we use to capture the coverage problem at hand; see Figure 2.

3.1.1 Sensor Model. We think of sensors as autonomous mini-drones, 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 2(a)), θ is the **Angle of View (AOV)**, and R_{min} and R_{max} are the minimum and maximum allowable distances between the camera and any target for acceptable viewing quality. Similar models have been used for anisotropic or directional sensors (e.g., Amac Guvensan and Gokhan Yavuz (2011)).



(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.

Fig. 2. Camera parameters.

Definition 1. Field of View (FOV) (Figure 2(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.

3.1.2 Oriented Line Segment Target Model (OLS). We model targets in 2D as line segments whose lengths are the width of the targets, and pose 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 poses. 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 pose vector. Furthermore, we define M_j as the midpoint of the target and let W_j denote its width. We assume $W_j \ll R_{max}\ \forall j$.

3.1.3 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 $\{\langle P_1^s, P_1^e \rangle, \langle P_2^s, P_2^e \rangle, \dots\}$, which block visibility but, unlike targets, have no pose.

3.1.4 Coverage Model. A sensor S_i is said to fully cover a target T_j if the following conditions apply: (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 $\overrightarrow{M_j P_i}$ is $\leq \pi/2$, meaning that S_i can capture frontal views of T_j. See Figure 2(b) for a summary.

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^*}.

3.2 Minimizing the Number of Drones

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 $\theta < 360°$ (Section 4).

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 $\mathcal{T}$ (Chvatal 1979), which, by our lower-bound results, is the best-possible for *OLSC*.

3.3 Modeling Assumptions

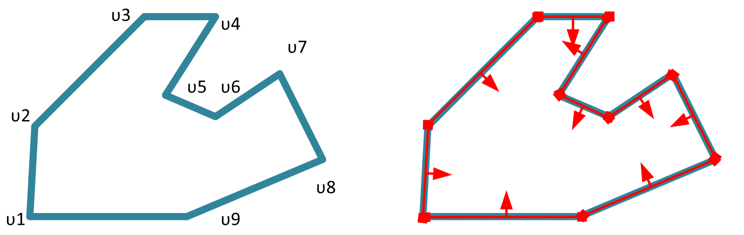
The main assumption we make in this work is that target locations and poses can be estimated by a coarse grain coverage system. As described in Section 2, we rely on a suitable coarse grain context detector to fulfill this requirement. The *Coarse Grain Context Detector* leverages established results in target detection and tracking using fixed cameras, e.g., visual sensors for pedestrian tracking (Chen and Odobez 2012), or other contextual sensors, e.g., device-free RF-based techniques (Abdelnasser et al. 2015a, 2015b; Adib et al. 2014). In addition, recent advances in persistent coverage (Palacios-Gasós et al. 2017; Schwager et al. 2017) greatly extend the range of application contexts where such information can be robustly collected within complex dynamic environments. The *OLS* model is essentially proposed to obtain close-up views of relevant targets using mobile cameras and provide fine grain surveillance as needed. This approach is a natural next step in multi-tier camera sensor networks where coarse grain information may be acquired via higher tier cameras providing low granularity coverage sufficient for detection and localization, but insufficient for identification, recognition, or activity monitoring (Kulkarni et al. 2005). We point out that for a variety of scenarios, and in particular for monitoring public spaces, there are numerous contextual hints that can be exploited to simplify the information that has to be estimated. For example, for human subjects, a median value of target width suffices for the operation of the system, without having to estimate more accurate values per target.

4 ON THE HARDNESS OF OLSC

As discussed in the previous section, the OLSC Solver is a key focus of this article. We analyze the hardness of OLSC by relating it to polygon illumination problems. We relate the two problems by looking at the generic Omni-OLSC. Then, we leverage our earlier results on the inapproximability of polygon illumination (Abdelkader et al. 2015) and a reduction by Eidenbenz et al. (Eidenbenz et al. 2001) to prove the inapproximability of OLSC.

4.1 Omni-OLSC is Hard

We consider a special case with camera parameters fixed to $R_{min} = 0$, $R_{max} = \infty$, and AOV $= 360°$. We call this problem Omni-OLSC as all cameras are now omnidirectional and their viewing



(a) Point Guard AGP instance $\Upsilon = \{v_1, v_2, \ldots, v_9\}$. (b) The corresponding OLSC instance $\mathcal{T} = \{T_1, T_2, \ldots, T_9\}$.

Fig. 3. Reducing the Point Guard Art Gallery Problem (PGAGP) to the Oriented Line Segment Coverage Problem (OLSC).

direction (VD) is no longer relevant. Omni-OLSC allows for a direct reduction from the point guard art gallery problem as it is solely defined by the set of targets $\mathcal{T}$. This enables us to develop the intuition behind the proof of hardness for the general OLSC.

We define a polygon Υ as an ordered sequence of n vertices $v_1, v_2, \ldots, v_n$ where $n \geq 3$, as shown in Figure 3(a). Υ forms a closed planar region bounded by the edges $\overline{v_1v_2}, \overline{v_2v_3}, \ldots, \overline{v_{n-1}v_n}, \overline{v_nv_1}$. A simple polygon Υ, without holes, divides the plane into three faces relative to Υ: interior (Υ_i), exterior (Υ_e), and boundary (Υ_b). A point is said to lie in Υ iff it belongs to $\Upsilon_i \cup \Upsilon_b$. Two points x and y in Υ are said to be mutually visible if the line segment $\overline{xy}$ lies completely in Υ. Finally, we use $cover(x)$ to denote the subset of points in Υ_b visible from a point $x \in \Upsilon$.

Our proof makes use of the *Point Guard Art Gallery Problem (PGAGP)*. A set of points $G \subset \Upsilon$, referred to as a guard set, is said to cover the boundary of Υ iff $\Upsilon_b \subset \cup_{x \in G} cover(x)$. PGAGP is to find a guard set G^* to cover the boundary of given polygon Υ, such that $|G^*|$ is minimum. PGAGP for boundary coverage was shown to be NP-hard (Eidenbenz et al. 2001) even when limited to convex (Culberson and Reckhow 1988) or star-shaped guard views (Lee and Lin 1986).

Theorem 4.1. *Omni-OLSC is NP-Hard.*

Proof. We encode a given PGAGP instance Υ as an Omni-*OLSC* instance $\mathcal{T}$, e.g., as shown in Figure 3. We map each edge $\overline{v_{i-1}v_i}$ to a target T_i whose P_i^s is v_{i-1}, P_i^e is v_i, and $\overrightarrow{D_i}$ points to the interior of Υ. By Definition 2, the placement of cameras is limited to the interior of Υ. Hence, the minimum number of omnidirectional sensors required to cover all targets is exactly the minimum number of point guards required to cover the polygon. It follows that Omni-*OLSC* is at least as hard as PGAGP. □

The same reduction presented above can also be used to reduce PGAGP for polygons with holes to Omni-OLSC. The edges bounding each polygonal hole are mapped to OLS targets facing the interior of the polygon.

4.2 Polygon Illumination and OLSC

Our study of polygon illumination is motivated by the natural reduction to our problem as described in the previous subsection. Two additional restrictions to the art gallery problem are needed to capture the OLS model. First, since we use cameras having a limited FOV, we turn our attention to the Point α-Floodlight Illumination Problem for art galleries with Holes (PFIPH), where the art gallery is to be covered with α-floodlights, i.e., floodlights of angle α. Second, we further require that each edge is *fully covered* by at least one α-floodlight to get a restricted variant that we call F-PFIPH. Illumination of polygons without holes by α-floodlights where floodlights can only be placed at vertices of the polygon is NP-hard (Bagga et al. 1996). However, the hardness of PFIPH

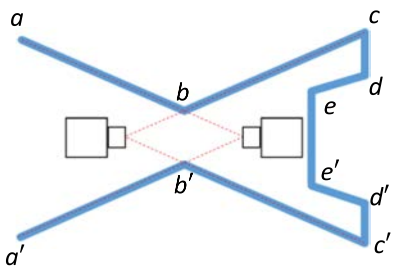


Fig. 4. Point Floodlight Gadget.

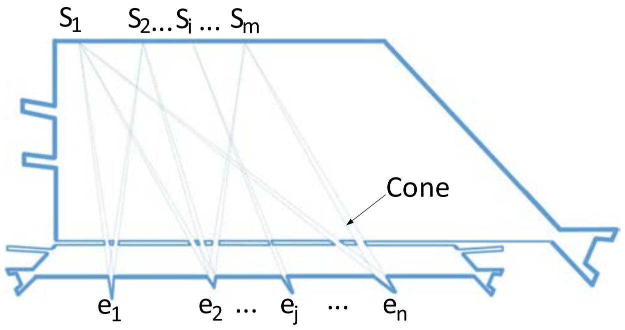


Fig. 5. Reducing SET-COVER to Point α-Floodlight Illumination for polygons with Holes (PFIPH).

is an open problem (Urrutia et al. 2000). In summary, we will be dealing with the two problems below.

Definition 4. Point α-Floodlight Illumination Problem for polygons with Holes (PFIPH): Given a polygon P with holes and angle α, find the minimum number of α-floodlights to illuminate the whole polygon such that α-floodlights can be placed anywhere inside the polygon.

Definition 5. Full-coverage Point α-Floodlight Illumination Problem for polygons with Holes (F-PFIPH): Given a polygon P with holes and angle α, find the minimum number of α-floodlights to illuminate the whole polygon such that α-floodlights can be placed anywhere inside the polygon and each edge is fully illuminated by at least one α-floodlight.

It is clear that *PFIPH* and *F-PFIPH* are relevant to many surveillance problems, especially with the increasing interest in coverage with directional sensors (Amac Guvensan and Gokhan Yavuz 2011). Hence, it is necessary to establish lower-bounds on the efficiency of algorithms for such problems to enable more principled approaches for tackling them in practice.

4.3 OLSC is Hard, Even to Approximate

We develop the Point Floodlight Gadget (PFG) that we use in combination with the gadgets of Eidenbenz et al. (2001) and Abdelkader et al. (2015) for our reduction. We prove the hardness and inapproximability of *PFIPH* using a reduction that immediately yields the same results for both *F-PFIPH* and OLSC. With that, we settle the open problem regarding the hardness of PFIPH (Urrutia et al. 2000) and establish the hardness of OLSC.

The ***Point Floodlight Gadget*** **(PFG)** (Figure 4) is used to force the placement of floodlights inside the polygon, rather than on its edges or vertices. This simple gadget facilitates the encoding of PFIPH constraints which leverages earlier hardness proofs for the art gallery problem (e.g., Bagga et al. (1996) and Culberson and Reckhow (1988)) by forcing floodlights to be placed inside the polygon while allowing some control on where floodlights are pointed.

As shown in Figure 4, a PFG requires exactly two point floodlights to be fully covered. The intuition behind this design is to have a shape that can only be covered by a single camera from a unique configuration. The PFG is designed by intersecting two isosceles triangles with apex angle $\alpha_{PFG} = \min(\alpha, 60^\circ)$, an arbitrary upper bound chosen for convenience. The right triangle has two dents to force the placement of the left camera, while the left triangle acts as an interface to allow plugging the PFG into larger constructs. Such constructs also need to have dents that force the placement of the right camera. Each of the two triangles forming a PFG can be covered by a single α-floodlight (i.e., a floodlight with angle $\alpha \geq \alpha_{PFG}$) placed at its apex.

We use the PFG to adapt the construction in Eidenbenz et al. (2001) in order to obtain a reduction from `SET-COVER` to *PFIPH*. Given a set system (E, S), the `SET-COVER` problem asks for the minimum number of subsets from S to cover all elements in E, where $S \subseteq 2^E$. As shown in Figure 5, the reduction naturally maps the set system to a polygon with holes such that each element in E is represented by a dent on the bottom side and each subset in S is a potential floodlight configuration at the top edge. A horizontal barrier with tapered corridors is added to enforce this correspondence by limiting which dents are visible from each location.

We define a *cone* as the maximal convex area inside the polygon that includes one dent and one corridor. A floodlight placed above the barrier may illuminate a dent only if it belongs to one of the cones containing it. The polygon is designed with the barrier at a certain height such that at most two cones from different dents intersect above the barrier (Eidenbenz et al. 2001).

We extend the construction in Eidenbenz et al. (2001) as follows: (1) Three PFGs are added to illuminate everything other than the dents. This incurs four more *guards* than the design in Eidenbenz et al. (2001). (2) We set the height of the top side of the polygon to ensure that none of the guards requires an angle of view greater than α to cover the subset of dents assigned to it. This height, y_0, is an arbitrary constant independent of the width of the polygon (Eidenbenz et al. 2001). It is clear the required changes do not violate any of the properties of the original construction and the results carry through.

Theorem 4.2. *PFIPH is NP-hard.*

For hardness of approximation, we reduce from a restricted variant of `SET-COVER` using the same procedure outlined above. `RESTRICTED-SET-COVER` simply restricts `SET-COVER` by requiring $|S| \leq |E|$ and is easily shown at least as hard to approximate as `DOMINATING-SET` (Lemma 9, (Eidenbenz et al. 2001)). Since our version of the reduction only changes the cost of a solution by a constant additive factor of 4, we also get an equivalent of the promise problem in (Lemma 10, Eidenbenz et al. (2001)) with slightly different constants, i.e., $c + 6$ instead of $c + 2$. This effectively yields an approximation preserving reduction from `RESTRICTED-SET-COVER` and we get a corresponding result.

Theorem 4.3. *PFIPH cannot be approximated by a polynomial time algorithm with an approximation ratio of $\frac{1-\epsilon}{12} \ln n$ for any $\epsilon > 0$, where n is the number of vertices of the input polygon, unless $NP \subseteq TIME(n^{O(\log \log n)})$.*

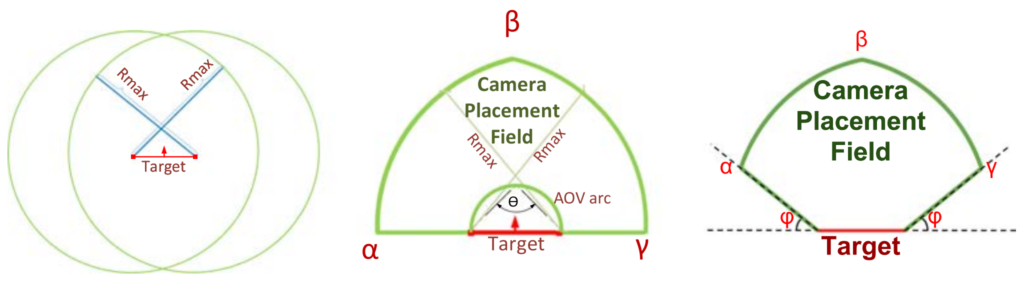
We observe that all edges in our reductions are fully covered by at least one guard. This means that the above results extend to *F-PFIPH* as well. Finally, using the simple approximation-preserving reduction outlined in Figure 3, the same applies to OLSC. Hence, the following is a corollary of Theorems 4.1, 4.2, and 4.3.

Corollary 4.4. *OLSC is NP-hard.*

Corollary 4.5. *OLSC cannot be approximated by a polynomial time algorithm with an approximation ratio of $\frac{1-\epsilon}{12} \ln n$ for any $\epsilon > 0$, where n is the number of vertices of the input polygon, unless $NP \subseteq TIME(n^{O(\log \log n)})$.*

5 COVERING OLS TARGETS

Our drone placement algorithms rely on a decomposition of space by the various coverage constraints per target. Recalling the definitions of Section 3, 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 pose (i.e., capturing the target from its significant perspective), and occlusion avoidance (i.e., having no target or obstacle occluding the target of



(a) Boundary of points no farther than R_{max} from any point on the target. (b) BCPF for R_{max}, AOV and target pose constraints. (c) BCPF parametrized by viewing angle ϕ for quality control.

Fig. 6. Basic Camera Placement Field (BCPF).

interest). First, we focus on satisfying all these constraints for a single target, which allows us to develop the essential tools needed to compute drone placements. Then, we show the extension to a pair of targets using a convenient approach to covering multiple targets simultaneously. Arbitrary subsets of targets can then be covered by satisfying their coverage constraints in a pairwise fashion.

5.1 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 6(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 pose.

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 6(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 (Chen et al. 2008a), the accuracy of object matching and point matching between two images drop significantly for rotations larger than 45° (Bay et al. 2006). 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 6(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 7. 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.

5.2 Covering a Pair of Targets by a Single Camera

For a single camera to cover two targets T_a and T_b, it must fall in the CPF of each, meaning that camera placement is limited to the Intersection Area (IA) of their CPFs. This guarantees a

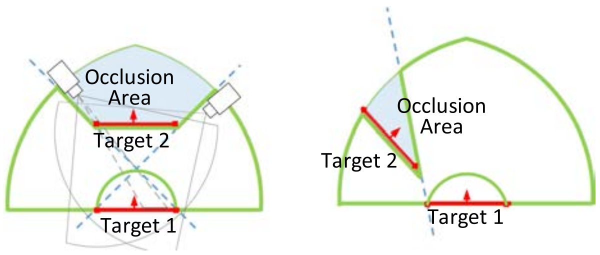
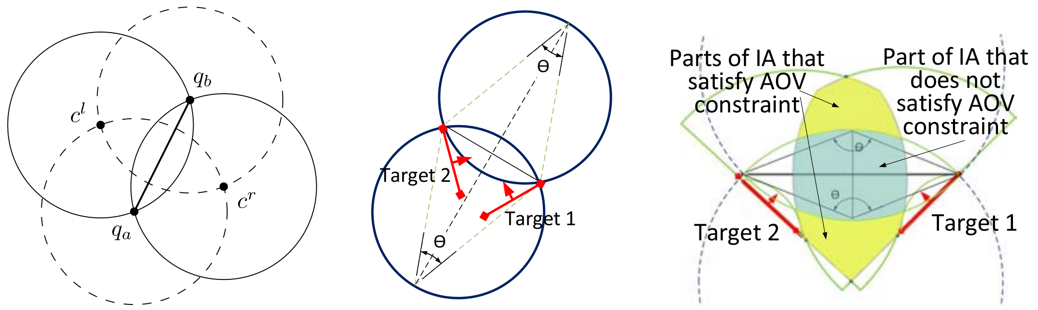


Fig. 7. Camera Placement Field (CPF) of Target 1 after excluding areas blocked by some occluder (labeled as Target 2).


q_b
c^l
c^r
q_a
θ
Target 2
Target 1
θ
Parts of IA that satisfy AOV constraint
Part of IA that does not satisfy AOV constraint
Target 2
Target 1

(a) Construction of AOV circle pairs (solid) using helper circles (dashed).

(b) AOV circles ($\theta < \frac{\pi}{2}$).

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

Fig. 8. AOV circle pairs and intersection area (IA).

placement that satisfies range, pose, and occlusion constraints for both targets as shown in Figure 8(c). The AOV constraint, on the other hand, requires for a candidate camera location x and any choice of points $q_a \in T_a$ and $q_b \in T_b$ that $\angle q_a x q_b \leq \theta$. This constraint on camera locations can be conveniently encoded by a pair of *AOV circles* that we define next.

Definition 5.1 (AOV circle pair). For any pair of points (q_a, q_b), the *AOV circle pair* is the two congruent circles sharing $\overline{q_a q_b}$ as a chord at an inscribed angle equal to the AOV θ.

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. It is easy to verify that the circles having $\overline{q_a q_b}$ as a chord at an inscribed angle θ have radius $r_{q_a,q_b}(\theta) = \frac{|q_a q_b|}{2 \sin \theta}$. One way to construct the centers of these circles is to compute the two points of intersection for the two helper circles with radius $r_{q_a,q_b}(\theta)$ centered at q_a and q_b as shown in Figure 8(a).

For each pair of targets (T_a, T_b) we need a set of *AOV circles* to exclude all locations that cannot fully cover both targets simultaneously. We use the four diagonals connecting one end point from each target to generate four AOV circle pairs which can be shown to contain all AOV circles for all pairs of points (q_a, q_b) on the two targets. We defer the formal proof to Appendix A.

Intuitively, consider any pair of points (q_a, q_b) on the two targets and a camera location x where a camera is to be placed to cover both q_a and q_b simultaneously. Fixing x, observe that each of q_a and q_b can be moved to one of the end points on their respective target segments to make $\angle q_a x q_b$ larger. It follows that if a camera at x can cover all diagonals, then it can also cover any such pair of points and consequently the whole two targets. Hence, it suffices to exclude all camera locations that cannot cover any diagonal. Figure 8 shows two examples of AOV circle pairs. In concurrent work, an alternative approach to covering pairs of targets was developed in Lino and Christie (2015); using Euler angles to parameterize the set of camera placements yielding a specific on-screen composition of the given pair of targets, the best viewpoint is found by an interval-based search in the parametric space defined by the Euler angles. In contrast, our approach works directly in the Cartesian space of camera placements yielding a more intuitive spatial decomposition which is easier to incorporate with the other constraints we consider for the purposes of coverage as required in surveillance and similar applications.

6 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*.

6.1 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.

6.1.1 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.

Per Section 5, 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.

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 6.1. *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 poses, 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 $O(n^2N^2)$ (see Figure 7). 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 $|\mathcal{P}| = O(N^4)$.

6.1.2 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 6(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.

6.2 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 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 be 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 $sweep(x) = \{\hat{\alpha}_1, \hat{\alpha}_2, \ldots\}$ 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 $O(N^2)$ (Vegter 1990). 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 toward 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 $f_\infty(x, \hat{\alpha}, \alpha) = \max_{T_j \in cov(x, \hat{\alpha})} d(x, \alpha, T_j)$.

Remark: By design, the *Angular Discretizer* is restricted to the candidate locations returned by the *Spatial Discretizer*. However, recall that such candidate locations are merely suggested as witnesses that certain subsets of targets can be covered by a single camera. It is possible to choose better locations to cover a given subset of targets than the representative location provided by the *Spatial Discretizer*. This is further discussed in Section 9.4.

6.3 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 $\frac{|\mathcal{R}_{greedy}|}{|\mathcal{R}_{opt}|} = O(\log |\mathcal{T}|)$ (Chvatal 1979). 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.

7 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 (Khan et al. 2016; Saeed et al. 2014). Figure 9 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 $30Hz$. 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 (Krajník et al. 2011) 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.

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 500MHz and 1GB 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

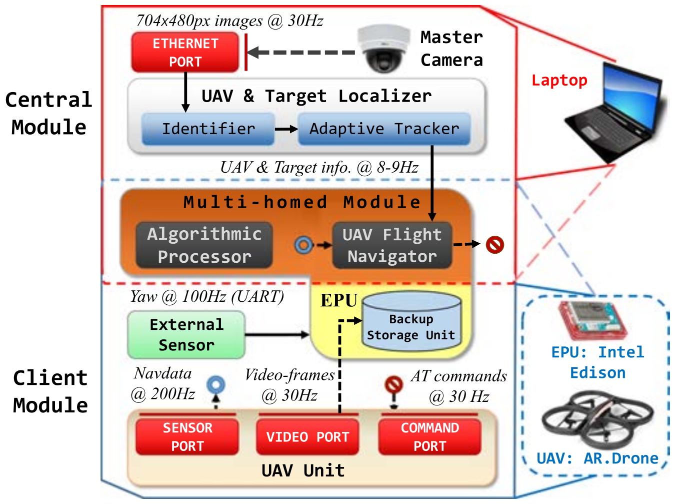


Fig. 9. Architecture of the Argus prototype.

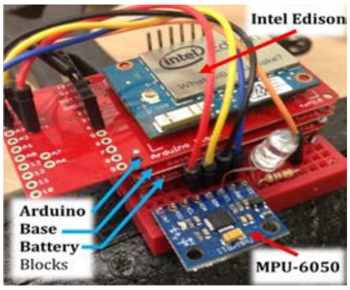


Fig. 10. EPU setup.

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 10.

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 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 $30m^2$ where we place one to five synthetic targets positioned in configurations that require a maximum of two drones (Figure 11); 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 pose estimation.

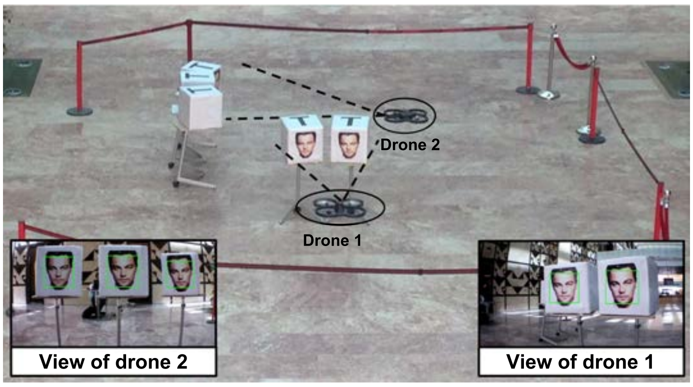


Fig. 11. Experimental setup: target layout, drone configurations, and captured images.

Table 1. Energy Consumption of Parrot AR.Drone 2.0 Maneuvers Measured Over Time (i.e., Power) and Distance Traveled

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

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 8, 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 (Kim and Shim 2013). The *Central Module* uses the EPU to evaluate local strategies to minimize the cost of the update, based on the energy footprint of each maneuver. Table 1 summarizes the energy consumption of Parrot AR.Drone 2.0 maneuvers. To avoid large rotations or displacements, drones cooperate to keep targets in view (Hausman et al. 2015). In our implementation, we follow a simple greedy heuristic for saving energy, based on the energy profile of the Parrot AR.Drone 2.0. The drone used in our experiments uses more energy for rotation compared to horizontal motion. Hence, the *Central Module* prefers coverage configurations that can be reached using minimal rotation. This autonomous behavior also serves as a fallback strategy if the communication link between the *Central Module* and the drone is broken.

Note that for other drones, the energy footprint of different maneuvers will be different, and the *Central Module* will thus adjust its policy accordingly. This topic has been the subject of earlier work, which can be used for more careful energy conservation schemes for target coverage (Zorbas et al. 2013) and autonomous reconfiguration (Bregu et al. 2016).

8 EVALUATION

We aim to assess the performance of a working instance of Argus in real-time and verify the efficiency of the proposed algorithms. To this end, we demonstrate the advantages of the OLS model, compared to the traditional model of targets as mere points, through the prototype we implement per Section 7. In particular, we analyze the overhead of the *OLSC Solver* within the system and establish the feasibility of adopting this enhanced model in a real surveillance system. The prototype we employ for this evaluation leverages typical hardware that can be found at most research labs and is comparable in scale to the experiments reported on closely related systems (Galvane et al. 2017; Nägeli et al. 2017). In addition, we present a set of large-scale simulations that compare the performance of the proposed algorithms against a baseline and establish the sampling heuristic as the method of choice, which we employ on the prototype.

8.1 Argus Evaluation

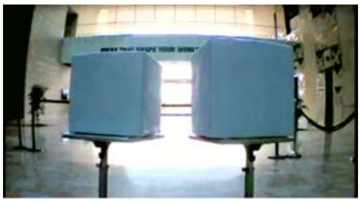
We demonstrate the pitfalls of traditional target coverage algorithms, where target size and pose are not taken into account (Neishaboori et al. 2014a), by comparing them to Argus in a realistic setting. Then, we break down the delays in the presented system and compare against the delay introduced by 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 12, 13, and 14. 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 12(a) shows two targets covered from the opposite direction of their significant perspective because typical coverage algorithms do not take target pose into account. Moreover, typical target coverage algorithms do not take target size and potential occlusions between targets into account, which is demonstrated in Figure 13(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 12(b) and 13(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 15 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., $120ms$ per frame for human body pose estimation (Flohr et al. 2015)).

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 16. 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 16. In our experiments, the *UAV Localizer* has to be invoked at a minimum frequency of $8Hz$ for smooth control of the drone.

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 1). Figure 17 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


(a) Targets may be covered from behind.

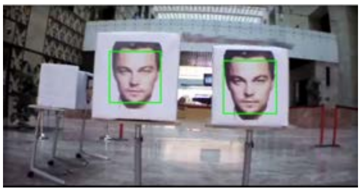
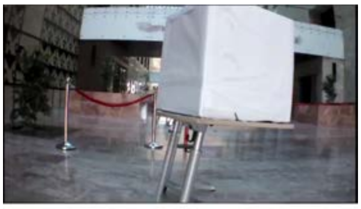

(b) Target pose is taken into account.

Fig. 12. Comparing the view from Drone 1 under a typical target coverage algorithm (top) and OLSC (bottom).


(a) Targets may occlude one another.

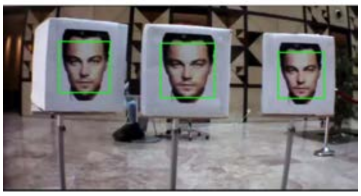
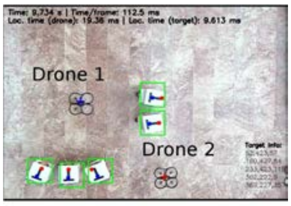
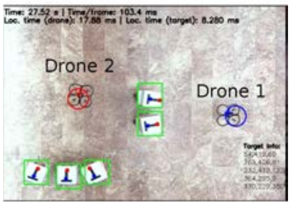

(b) Potential occlusions are taken into account.

Fig. 13. 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): 19.38 ms | Loc. time (target): 9.613 ms

(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

(b) Drones properly cover all targets.

Fig. 14. Top views from the master camera showing drone configurations corresponding to Figures 12 and 13.

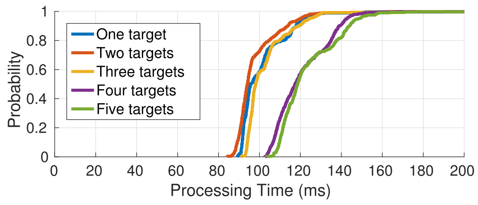


Fig. 15. CDF of processing time per frame including image fetching, target and drone localization, and drone communication.

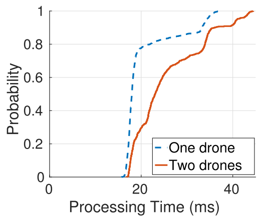


Fig. 16. CDF of processing time of the *UAV Localizer*.

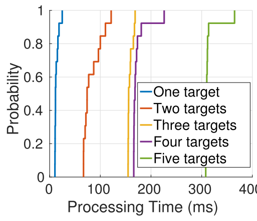


Fig. 17. CDF of processing time of the *OLSC Solver*.

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).

8.2 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 $\epsilon \rightarrow 0$, 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

Table 2. Simulation Parameters

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

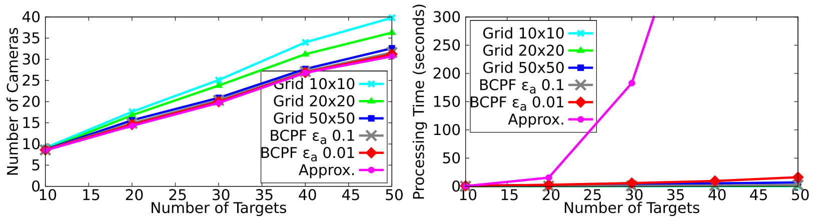


Fig. 18. Comparing the performance of all algorithms for increasing numbers of targets.

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.

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 poses 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 10×10 (sparse), Grid 20×20 (medium), and Grid 50×50 (dense) for ϵ set to $10m$, $5m$ and $2m$, respectively.

Figure 18 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 $\epsilon_a = 0.1rad$.

Figure 19 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 $0.1rad$, BCPF sampling outperforms grid sampling requiring 12% less cameras and running up to 2× faster on 140 targets.

Figure 20 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

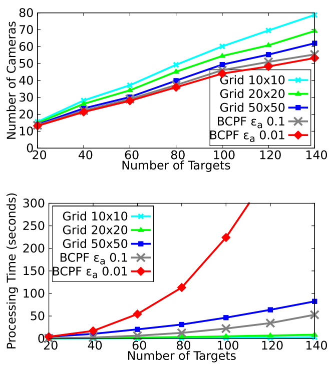


Fig. 19. Comparing all heuristics for increasing numbers of targets.

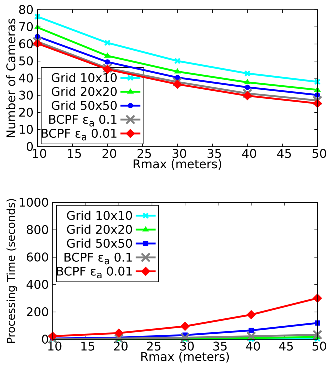


Fig. 20. Effect of varying R_{max} for a fixed number of targets.

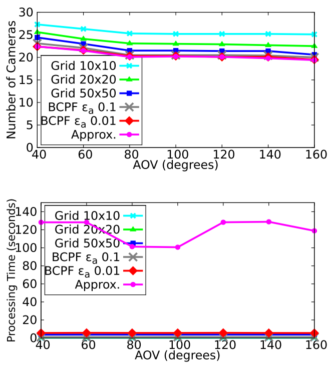


Fig. 21. Effect of varying angle of view (AOV) for a fixed number of targets.

the other hand, changing the value of the AOV, shown in Figure 21, 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.

9 EXTENSIONS AND OPEN PROBLEMS

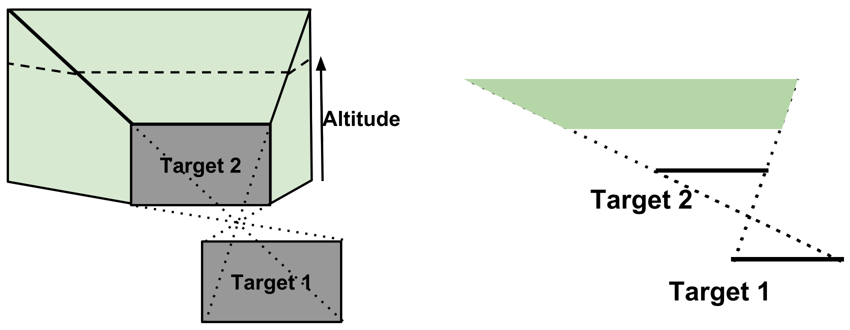
OLSC presents a new powerful model that can be harnessed to capture many scenarios with little to no modifications. In this section, we discuss a few of those scenarios with two goals in mind: (1) facilitate the porting of this model to be used in other domains, and (2) illustrate future research directions where this model can be extended or applied to improve target models in smart surveillance and visual sensing systems.

9.1 Coverage in 3D

The coverage model adopted in this article, per Section 3.1, is aimed to enhance traditional coverage models used primarily in the surveillance literature. We developed the OLS model as a convenient alternative to modeling targets by mere points, which can be incorporated with little overhead. On the other hand, the OLS model does not fully capture the 3D nature of targets and their coverage constraints. We propose a simple adaptation that accounts for the changes in visibility and allows the placement of drones at different altitudes.

In scenarios where cameras are mounted on flying robots, there are many more configurations available for covering any given set of targets. In particular, for ground targets it is possible to mitigate occlusion effects by flying at a higher altitude. To make our OLS model even more realistic, information about the 3D shape of the target should be taken into account to calculate the best camera locations. The main parameter we consider here is target height. Although we could extend our coverage constraints to 3D and attempt a similar approach to what we have done in 2D, we propose a sampling strategy that builds on the heuristics we developed and tested in 2D.

We propose a heuristic solution to the problem in this new setting, by defining a new 2D problem at each discretization of flying altitudes h. The 2D problems we define are almost identical to the situation we had before, except in the following: (1) we get occlusion constraints by intersecting the 3D shadow prism with the horizontal plane at height h as shown in Figure 22; (2) we adjust



(a) Occlusion volume for a 3D model (green). (b) Occlusion area in 2D at altitude h (green).

Fig. 22. 3D shadow prism and occlusion region at a given height mapped to a 2D setting.

R^h_{max} and R^h_{min} at each height; the new ranges are calculated as a function of h such that the calculated ranges do not violate the original range restrictions; (3) angular sweeping will not be performed in 2D as mentioned in Section 6.2, but rather in 3D to cover both possible camera pans as well as camera tilts.

The candidates sampled at each height h are all added to the set of candidates and we map to SET-COVER as before. For every given grouping of targets, we may get redundant candidates at different heights that all cover the same group. Before solving the SET-COVER instance, we may filter these redundant candidates by only keeping ones with preferred altitudes or any other criteria. After we select the set of candidates to use for coverage, we may follow with a pan-tilt optimization to get the best quality of coverage. A 3D version of this algorithm will require target heights, in addition to target locations, poses, and widths.

For applications dealing with relatively large objects (e.g., buildings and statues), this may require breaking big targets into multiple smaller targets. Several endeavors in surveillance and computer visions are directed toward extracting 3D information from 2D pictures or videos (Aubry et al. 2014; Ramakrishna et al. 2012) or depth sensors (Henry et al. 2012) to capture the relevant features of objects. Such algorithms and sensors can provide a 3D version of *OLSC* with the necessary input for its operation.

9.2 OLSC Using a Fixed Number of Cameras

Availability of a limited number of mobile cameras is to be expected in most scenarios. We propose a two-phase algorithm that first solves OLSC and suggests a minimal number of camera configurations. The second phase is to solve an instance of the Multiple Traveling Salesmen Problem (mTSP) where mobile cameras try to visit all the identified locations and take the required shots through the most efficient (e.g., shortest) trajectories (Bektas 2006). While this does not guarantee optimal coverage, it provides a nice extension for the proposed model to handle a common scenario. Solving the actual problem requires an extended formulation that captures both SET-COVER and TSP, which we consider as a future research direction in improving the current model. A closely related problem was studied in Wang et al. (2007).

9.3 Target Quality of View Requirements

Several metrics of quality of coverage can be considered when evaluating a coverage algorithm. On top of that, different quality constraints might be required for each type of targets. Quality requirements include angle of view, target-camera distance, tracking performance, and accounting for device constraints (i.e., pan, tilt, and zoom limitations) (Krahnstoever et al. 2008). Those requirements vary from one application to the other, and even from one target to the other.

OLSC captures most of those quality requirements by having a flexible representation of the camera placement field, which encodes range, viewing-angle, and occlusion constraints. In addition, the target can basically have two effective widths: a total width used to evaluate occlusion constraints for other targets and a *feature width* used to ensure the target itself is adequately covered. Tracking performance, however, remains a challenge, which we further discuss in the next section.

9.4 Open Problems

In this section, we discuss some open problems that are not treated in our study of Argus.

Continuously Updating Camera Configurations. Recall that Argus relies on the top tier to perform target tracking as part of its persistent coverage functionality; this is the responsibility of the *Target Localizer* module. For drone navigation, we also assume that the top tier provides suitable positioning information to help plan efficient and safe paths to position each drone at its assigned location; which is implemented in the *UAV Localizer* module. In addition, to cope with positioning errors, each drone can factor in the readings from its on-board sensors to better position itself with respect to the targets it is assigned to cover; this is handled by the *Adaptive Tracker* module.

When it comes to dynamically updating camera configurations, target handover and mobility management are essential issues (Foresti et al. 2005). Argus relies on mobile cameras, which means that as targets move, cameras need to hand over targets among themselves and possibly move to maintain all targets in view, or the target configuration will necessitate adding new cameras due to new occlusion or pose conditions. Beyond updating camera locations and assignments, the system is also required to optimize the energy used to move the cameras and the quality of coverage of targets during handover.

While handling such scenarios can simply be achieved by running the algorithm every δt seconds, and updating the locations of cameras to adopt the new solution, this would result in considerable overhead and delays as no relationship is assumed between the camera configuration U_t and the configuration U_{t+1}, for a time instant t. Several algorithms were studied to address such scenarios. We highlight some of those approaches and leave it to future work to select the approach most suitable for Argus and the specific application at hand.

One approach relies on computer vision algorithms to detect targets as they move between camera views without requiring cameras to move (Chen et al. 2008b). Cameras can then move slightly in order to better cover the new targets that entered its field of view to satisfy OLSC constraints. Another approach relies on cameras bidding on which of them will be required to cover a target that just moved, based on a coverage utility function (Esterle et al. 2014). An OLSC implementation of this approach can introduce as a utility the amount of energy consumed by the camera-mounted robot to move in order to better cover the target (He et al. 2009; Yu and Lee 2014).

Positioning Tolerance and Camera Calibration. As the provable approximation algorithm used candidate locations defined by the intersection points of coverage constraints, such locations may not be the best to use in practice. In particular, we have shown that such locations will lie on the boundary of a region where a single camera may cover the same set of targets. Attempting to place cameras at the boundary means that any inaccuracy in target localization and camera control or the expected movement of targets can all lead to missing some of the targets. Hence, the solution returned by OLSC only suggests a feasible partitioning of targets, where each partition can be covered by a single camera. This suggests that once OLSC optimization identifies a set of locations for camera placement, each camera should follow by optimizing its own configuration for the purpose of covering the set of targets it has been assigned. This becomes more relevant when planning the trajectory each camera should execute as it transitions from one configuration to another, possibly in response to target mobility.

10 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, or to guarantee quality communication over a wireless network among clients in that area. Several approaches to area coverage have been studied including static randomly deployed sensors (Carmi et al. 2007) and strategically placed mobile sensors (Dhillon and Chakrabarty 2003) using either isotropic (Hexsel et al. 2011) or anisotropic sensors (Yildiz et al. 2014). The related problem of barrier coverage was studied in Kumar et al. (2005), 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 (Abdelkader et al. 2012; Ai and Abouzeid 2006; Johnson and Bar-Noy 2011), or strategically placed directional sensors or antennas (Berman et al. 2007; Neishaboori et al. 2014a, 2014b). 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 (Johnson and Bar-Noy 2011). For antenna placement to serve a set of static targets with bounds on the bandwidth demand per antenna, a 3-approximation algorithm was presented in Berman et al. (2007). In order to satisfy connectivity requirements between antennas, Han et al. (2008) gave a 9-approximation algorithm. *We propose a more realistic model for target coverage by visual sensors that greatly enhances the point model typically used by earlier algorithms (Amac Guvensan and Gokhan Yavuz 2011). 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 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 (Wang and Cao 2011b). Wu and Wang (2012) studies the necessary conditions for full-view coverage in static camera deployments and Hu et al. (2014) studies full-view coverage using heterogeneous mobile cameras. Full-view barrier coverage was then introduced (Wang and Cao 2011a) and further extended to accommodate stochastic deployments in Yu et al. (2015). Taking self-occlusions into account, ensuring all sides of a convex target are always visible was studied in Tokekar and Isler (2014). *Our proposed approach is different in two aspects: (1) It overcomes occlusion scenarios and takes target size into account in addition to target pose. (2) It is concerned with target coverage rather than area coverage which is the main concern of full-view coverage.*

Persistent Coverage. In a seminal paper (Schwager et al. 2011) a decentralized control strategy was introduced for the deployment of heterogeneous cameras for coverage tasks, which significantly improved upon earlier methods (Cortes et al. 2004). In follow-up works, the control strategy was further enhanced to robustly learn and adapt to changes in the environment (Palacios-Gasós et al. 2016; Schwager et al. 2017), while emphasizing decentralized control over a communication network. More recently, the control strategy was endowed with optimal path planning while avoiding obstacles in the environment (Best et al. 2017; Palacios-Gasós et al. 2017; Tokekar et al. 2014) and target unpredictability (Hönig and Ayanian 2016). Several challenging aspects of persistent coverage have also been studied: energy-awareness (Derenick et al. 2011), connectivity (Orfanus et al. 2016), adaptive streaming (Wang et al. 2016), and dynamic priorities (da Silva et al. 2017). For a more thorough survey, we refer the reader to Nigam (2014) and Khan et al. (2018). *The proposed* *system leverages established results in persistent coverage through a two tier architecture. We make critical use of the information provided by the top tier providing persistent coverage over the environment to plan the placement of mobile cameras in the lower tier to obtain high -quality views of the targets of interest.*

Video Capture Using Drones. There has been a growing interest in using drones and drone swarms for surveillance and video capture (Bürkle 2009), e.g., for sport streaming (Wang et al. 2017). In such applications, several challenges including target mobility and low-quality footage (e.g., due to distance) were studied in Hsu and Chen (2015). For mobile target tracking, using either a single drone (Naseer et al. 2013) or multiple drones (Mueller et al. 2016) 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 (Joubert et al. 2016). Building upon earlier work in virtual cinematography (Lino et al. 2011), this exciting line of work has recently been witnessing very interesting developments (Galvane et al. 2017; Joubert et al. 2015; Nägeli et al. 2017). In earlier work, we developed several target coverage algorithms for targets represented as points and deployed them on our testbed (Khan et al. 2016; Neishaboori et al. 2014a, 2014b; Saeed et al. 2014). *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 (Urrutia et al. 2000). Several variants were introduced constraining guard placement (e.g., point, vertex, or edge) and coverage (i.e., convex, star shaped, or spiral shaped) (Culberson and Reckhow 1988; Lee and Lin 1986). In particular, the art gallery illumination problem considered guards having a limited angle of view (Bagga et al. 1996; Bose et al. 1997). Visibility algorithms have found many applications in wireless communications, sensor networks, surveillance, and robotics. However, several variants were shown to be NP-hard (O'Rourke and Supowit 1983), and more recently even $\exists\mathbb{R}$-complete (Abrahamsen et al. 2017). In addition, inapproximability results for art gallery coverage with and without holes were shown in Eidenbenz et al. (2001) and also for the illumination of art galleries without holes (Abdelkader et al. 2015). On the approximation side, the works in Deshpande et al. (2007) and González-Baños (2001) presented algorithms for the coverage of art galleries with and without holes, respectively. *We settle the hardness and approximability of art gallery illumination for polygons with holes and use this to prove the hardness of OLSC. 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 (Deshpande et al. 2007) or sampling (González-Baños 2001) while assuming omnidirectional cameras.

11 CONCLUSION

We presented Argus, an autonomous system that utilizes drones to provide better coverage of targets taking into account their size, pose, 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 100× 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.

APPENDIX

A A TECHNICAL LEMMA FOR HANDLING AOV CONSTRAINTS

Recall that for any pair of points (q_a, q_b), an *AOV circle pair* was defined in Section 6.1.1 as the two congruent circles sharing $\overline{q_a q_b}$ as a chord at an inscribed angle equal to the AOV θ. Now, for two targets T_a and T_b represented by the line segments (P_a^s, P_a^e) and (P_b^s, P_b^e), respectively, we define the set of diagonals as

$$T_a \otimes T_b = \{(P_a^s, P_b^s), (P_a^s, P_b^e), (P_a^e, P_b^s), (P_a^e, P_b^e)\}.$$

The next lemma shows that the four AOV circle pairs corresponding to the diagonals $T_a \otimes T_b$ suffice to capture the AOV constraints for a pair of targets T_a and T_b, as needed for the comprehensive discretization of the search space for camera placement per Section 6.1.1.

LEMMA A.1. *For a fixed AOV angle θ and any pair of points $(q_a, q_b) \in T_a \times T_b$, the corresponding pair of AOV circles is completely contained in the union of AOV circles corresponding to the diagonals $T_a \otimes T_b$.*

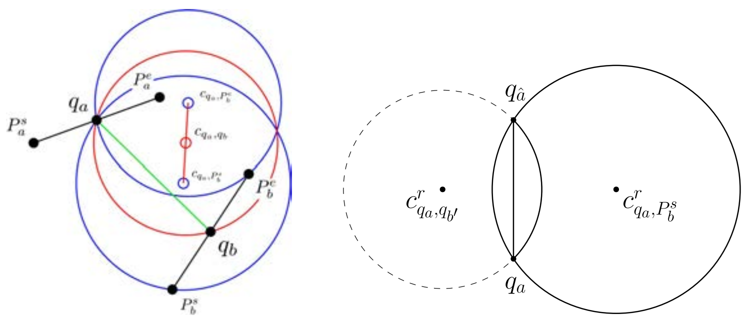
PROOF. For any pair of points (q_1, q_2) let $\{C^r_{q_1,q_2}, C^l_{q_1,q_2}\}$ be the corresponding pair of AOV circles for an AOV angle θ and let $\{c^r_{q_1,q_2}, c^l_{q_1,q_2}\}$ be their centers with $c^r_{q_1,q_2}$ to the right of $\overrightarrow{q_1 q_2}$ and $c^l_{q_1,q_2}$ to its left. Without loss of generality, we will show that

$$C^r_{q_a,q_b} \subseteq C^r_{q_a,P_b^s} \cup C^r_{q_a,P_b^e} \subseteq \left(C^r_{P_a^s,P_b^s} \cup C^r_{P_a^e,P_b^s}\right) \cup \left(C^r_{P_a^s,P_b^e} \cup C^r_{P_a^e,P_b^e}\right).$$

The key claim is that for any pair of points (q_a, q_b), the circle $C^r_{q_a,q_b}$ is contained in the union of the two circles obtained by replacing one of the points with the two end points on its segment, i.e., $C^r_{q_a,P_b^s} \cup C^r_{q_a,P_b^e}$. Observe that the point we do not replace, i.e., q_a, will be shared by all three circles.

Recall that the circle $C^r_{q_a,q_b}$ has radius $\frac{|q_a q_b|}{2\sin\theta}$, which is linear in $|q_a q_b|$. It follows that fixing q_a and moving q_b to any point $q_{b'}$ on the closed line segment $\overline{P_b^s P_b^e}$, the radius of the intermediate circle $C^r_{q_a,q_{b'}}$ varies linearly as $\frac{|q_a q_{b'}|}{2\sin\theta}$. Consequently, the centers of all intermediate circles $c^r_{q_a,q_{b'}}$ lie on the line segment between $c^r_{q_a,P_b^s}$ and $c^r_{q_a,P_b^e}$; denote this line segment by $l_{s,e}$. It follows that all intermediate circles pass through not only q_a, but both points of intersection between $C^r_{q_a,P_b^s}$ and $C^r_{q_a,P_b^e}$; we denote the other point by $q_{\hat{a}}$ as it is the mirror image of q_a about $l_{s,e}$. See Figure 23(a) for an example.

We argue that $C^r_{q_a,q_{b'}} \subseteq C^r_{q_a,P_b^s} \cup C^r_{q_a,P_b^e}$ for all points $q_{b'} \in \overline{P_b^s P_b^e}$, such as q_b. Using the common chord $\overline{q_a q_{\hat{a}}}$, we cut the circle $C^r_{q_a,q_{b'}}$ into two arcs $\widehat{q_a q_{\hat{a}}}^s$ and $\widehat{q_a q_{\hat{a}}}^e$ where the first intersects $l_{s,e}$ closer to $c^r_{q_a,P_b^s}$ and the latter intersects $l_{s,e}$ closer to $c^r_{q_a,P_b^e}$. Since any two circles intersect in at most two points and $\{q_a, q_{\hat{a}}\}$ are the two points of intersection between $C^r_{q_a,q_{b'}}$ and $C^r_{q_a,P_b^s}$, which may also coincide if $q_{b'} = P_b^s$, it follows that $\widehat{q_a q_{\hat{a}}}^s$ cannot exit $C^r_{q_a,P_b^s}$. Hence, $\widehat{q_a q_{\hat{a}}}^s \subseteq C^r_{q_a,P_b^s}$ and similarly $\widehat{q_a q_{\hat{a}}}^e \subseteq C^r_{q_a,P_b^e}$, as shown in Figure 23(b).



(a) Illustrating the two properties of AOV circles that we use in the proof.

(b) Arcs of a common chord are fully contained in the other circle.

Fig. 23. The key elements in the proof of Lemma A.1.

This shows that indeed $C^r_{q_a,q_{b'}} \subseteq C^r_{q_a,P_b^s} \cup C^r_{q_a,P_b^e}$, as required. By symmetry, we also obtain $C^r_{q_a,q_{b'}} \subseteq C^r_{P_a^s,q_{b'}} \cup C^r_{P_a^e,q_{b'}}$. Now, setting $q_{b'} = P_b^s$ yields $C^r_{q_a,P_b^s} \subseteq C^r_{P_a^s,P_b^s} \cup C^r_{P_a^e,P_b^s}$ and $q_{b'} = P_b^e$ yields $C^r_{q_a,P_b^e} \subseteq C^r_{P_a^s,P_b^e} \cup C^r_{P_a^e,P_b^e}$, which completes the proof. □

REFERENCES

Mohammad Aazam, Sherali Zeadally, and Khaled A. Harras. 2018b. Fog computing architecture, evaluation, and future research directions. *IEEE Communications Magazine* 56, 5 (2018), 46–52.

Mohammad Aazam, Sherali Zeadally, and Khaled A. Harras. 2018a. Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities. *Future Generation Computer Systems* 87 (2018), 278–289.

Ahmed Abdelkader, Moamen Mokhtar, and Hazem El-Alfy. 2012. Angular heuristics for coverage maximization in multi-camera surveillance. In *AVSS'12*.

Ahmed Abdelkader, Ahmed Saeed, Khaled Harras, and Amr Mohamed. 2015. The inapproximability of illuminating polygons by α-floodlights. In *CCCG'15*.

Heba Abdelnasser, Khaled A. Harras, and Moustafa Youssef. 2015a. UbiBreathe: A ubiquitous non-invasive WiFi-based breathing estimator. In *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. ACM, 277–286.

Heba Abdelnasser, Moustafa Youssef, and Khaled A. Harras. 2015b. Wigest: A ubiquitous wifi-based gesture recognition system. In *IEEE Conference on Computer Communications (INFOCOM'15)*. IEEE, 1472–1480.

Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. 2017. The art gallery problem is $\exists\mathbb{R}$-complete. *CoRR* abs/1704.06969 (2017). arxiv:1704.06969 http://arxiv.org/abs/1704.06969

Fadel Adib, Zach Kabelac, Dina Katabi, and Robert C. Miller. 2014. 3D tracking via body radio reflections. In *NSDI'14*.

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.

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.

Inc. Amazon.com. 2014. Amazon Alexa. http://alexa.amazon.com/.

Mathieu Aubry, Daniel Maturana, Alexei A. Efros, Bryan C. Russell, and Josef Sivic. 2014. Seeing 3D chairs: Exemplar part-based 2D-3D alignment using a large dataset of CAD models. In *CVPR'14*.

Jay Bagga, Laxmi Gewali, and David Glasser. 1996. The complexity of illuminating polygons by alpha-flood-lights. In *CCCG'96*.

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

Tolga Bektas. 2006. The multiple traveling salesman problem: An overview of formulations and solution procedures. *Omega* (2006).

Piotr Berman, Jieun Jeong, Shiva Prasad Kasiviswanathan, and Bhuvan Urgaonkar. 2007. Packing to angles and sectors. In *SPAA'07*.

Graeme Best, Jan Faigl, and Robert Fitch. 2017. Online planning for multi-robot active perception with self-organising maps. *Autonomous Robots* 42 (Dec. 2017), 715–738. DOI: https://doi.org/10.1007/s10514-017-9691-4

Andreas Bircher, Kostas Alexis, Ulrich Schwesinger, Sammy Omari, Michael Burri, and Roland Siegwart. 2017. An incremental sampling-based approach to inspection planning: The rapidly exploring random tree of trees. *Robotica* 35, 6 (2017), 1327–1340. DOI: https://doi.org/10.1017/S0263574716000084

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*.

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), 153–163.

Endri Bregu, Nicola Casamassima, Daniel Cantoni, Luca Mottola, and Kamin Whitehouse. 2016. Reactive control of autonomous drones. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 207–219.

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

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

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*.

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*.

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

Kuan-Wen Chen, Chih-Chuan Lai, Yi-Ping Hung, and Chu-Song Chen. 2008b. An adaptive learning method for target tracking across multiple cameras. In *CVPR'08*.

Vasek Chvatal. 1979. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research* 4, 3 (1979), 233–235.

J. Cortes, S. Martinez, T. Karatas, and F. Bullo. 2004. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation* 20, 2 (April 2004), 243–255. DOI: https://doi.org/10.1109/TRA.2004.824698

Joseph C. Culberson and Robert A. Reckhow. 1988. Covering polygons is hard. In *FOCS'88*.

L. C. Batista da Silva, R. M. Bernardo, H. A. de Oliveira, and P. F. F. Rosa. 2017. Unmanned aircraft system coordination for persistent surveillance with different priorities. In *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*. 1153–1158. DOI: https://doi.org/10.1109/ISIE.2017.8001408

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), 9449–9467.

J. Derenick, N. Michael, and V. Kumar. 2011. Energy-aware coverage control with docking for robot teams. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 3667–3672. DOI: https://doi.org/10.1109/IROS.2011.6094977

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.

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

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

M. Erdelj, E. Natalizio, K. R. Chowdhury, and I. F. Akyildiz. 2017. Help from the sky: Leveraging UAVs for disaster management. *IEEE Pervasive Computing* 16, 1 (Jan. 2017), 24–32.

Aliaa Essameldin and Khaled A. Harras. 2017. The hive: An edge-based middleware solution for resource sharing in the internet of things. In *Proceedings of the 3rd Workshop on Experiences with the Design and Implementation of Smart Objects*. ACM, 13–18.

Lukas Esterle, Peter R. Lewis, Xin Yao, and Bernhard Rinner. 2014. Socio-economic vision graph generation and handover in distributed smart camera networks. *ACM Transactions on Sensor Networks (TOSN)* 10, 2 (2014), 20:1–20:24.

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.

Fabian Flohr, Madalin Dumitru-Guzu, Julian F. P. 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.

Gian Luca Foresti, Christian Micheloni, Lauro Snidaro, Paolo Remagnino, and Tim Ellis. 2005. Active video-based surveillance system: The low-level image and video processing techniques needed for implementation. *IEEE Signal Processing Magazine* 22, 2 (2005), 25–37.

Paula Fraga-Lamas, Tiago M. Fernández-Caramés, Manuel Suárez-Albela, Luis Castedo, and Miguel González-López. 2016. A review on internet of things for defense and public safety. *Sensors* 16, 10, Article 1644 (2016).

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

Q. Galvane, C. Lino, M. Christie, J. Fleureau, F. Servant, F.-L. Tariolle, and P. Guillotel. 2017. Directing cinematographic drones. *ArXiv e-prints* (Dec. 2017). arxiv:1712.04216

Hend Gedawy, Karim Habak, Khaled A. Harras, and Mounir Hamdi. 2018. Awakening the cloud within: Energy-aware task scheduling on edge IoT devices. In *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 191–196.

Héctor González-Baños. 2001. A randomized art-gallery algorithm for sensor placement. In *SoCG'01*.

LLC Google. 2016. Google Home. http://home.google.com/.

Xiaofeng Han, Xiang Cao, Errol L. Lloyd, and Chien-Chung Shen. 2008. Deploying directional sensor networks with guaranteed connectivity and coverage. In *SECON'08*. IEEE, 153–160.

M. Hassanalian and A. Abdelkefi. 2017. Classifications, applications, and design challenges of drones: A review. *Progress in Aerospace Sciences* 91, Supplement C (2017), 99–131. DOI: https://doi.org/10.1016/j.paerosci.2017.04.003

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.

Y. He, I. Lee, and L. Guan. 2009. Distributed algorithms for network lifetime maximization in wireless visual sensor networks. *IEEE Transactions on Circuits and Systems for Video Technology* 19, 5 (May 2009), 704–718. DOI: https://doi.org/10.1109/TCSVT.2009.2017411

Elizabeth Farrell Helbling and Robert Wood. 2017. A review of propulsion, power, and control architectures for insect-scale flapping-wing vehicles. *Applied Mechanics Reviews* 70 (Dec. 2017), 010801–010801-9. DOI: https://doi.org/10.1115/1.4038795

Peter Henry et al. 2012. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *The International Journal of Robotics Research* 31, 5 (2012), 647–663.

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

Hwai-Jung Hsu and Kuan-Ta Chen. 2015. Face recognition on drones: Issues and limitations. In *DroNet'15*.

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.

Yitao Hu, Xinbing Wang, and Xiaoying Gan. 2014. Critical sensing range for mobile heterogeneous camera sensor networks. In *INFOCOM'14*.

W. Hönig and N. Ayanian. 2016. Dynamic multi-target coverage with robotic cameras. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 16)*. 1871–1878. DOI: https://doi.org/10.1109/IROS.2016.7759297

Niels Joubert, Dan B. Goldman, Floraine Berthouzoz, Mike Roberts, James A. Landay, Pat Hanrahan, et al. 2016. Towards a drone cinematographer: Guiding quadrotor cameras using visual composition principles. *arXiv preprint arXiv:1610.01691* (2016).

Niels Joubert, Mike Roberts, Anh Truong, Floraine Berthouzoz, and Pat Hanrahan. 2015. An interactive tool for designing quadrotor camera shots. *ACM Transactions on Graphics.* 34, 6 (Oct. 2015), 238:1–238:11. DOI: https://doi.org/10.1145/2816795.2818106

A. Khan, B. Rinner, and A. Cavallaro. 2018. Cooperative robots to observe moving targets: Review. *IEEE Transactions on Cybernetics* 48, 1 (Jan. 2018), 187–198. DOI: https://doi.org/10.1109/TCYB.2016.2628161

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*.

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.

S. J. Koppal. 2016. A survey of computational photography in the small: Creating intelligent cameras for the next wave of miniature devices. *IEEE Signal Processing Magazine* 33, 5 (Sept. 2016), 16–22. DOI: https://doi.org/10.1109/MSP.2016.2581418

Nils Krahnstoever, Ting Yu, Ser-Nam Lim, Kedar Patwardhan, and Peter Tu. 2008. Collaborative real-time control of active cameras in large scale surveillance systems. In *Workshop on Multi-Camera and Multi-Modal Sensor Fusion Algorithms and Applications-M2SFA2 2008*.

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

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

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

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

Chung-Ching Lin, Sharathchandra U. Pankanti, Karthikeyan Natesan Ramamurthy, and Aleksandr Y. Aravkin. 2015. Adaptive as-natural-as-possible image stitching. In *CVPR'15*.

Christophe Lino and Marc Christie. 2015. Intuitive and efficient camera control with the toric space. *ACM Transactions on Graphics* 34, 4 (July 2015), 82:1–82:12. DOI: https://doi.org/10.1145/2766965

Christophe Lino, Marc Christie, Roberto Ranon, and William Bares. 2011. The director's lens: An intelligent assistant for virtual cinematography. In *Proceedings of the 19th ACM International Conference on Multimedia (MM'11)*. ACM, New York, NY, 323–332. DOI: https://doi.org/10.1145/2072298.2072341

Matthias Mueller, Gopal Sharma, Neil Smith, and Bernard Ghanem. 2016. Persistent aerial tracking system for UAVs. In *IROS'16*.

Yash Mulgaonkar and Vijay Kumar. 2014. Towards open-source, printable pico-quadrotors. In *Proc.eedings of the Robot Makers Workshop, Robotics: Science Systems Conference*.

Tobias Nägeli, Lukas Meier, Alexander Domahidi, Javier Alonso-Mora, and Otmar Hilliges. 2017. Real-time planning for automated multi-view drone cinematography. *ACM Transactions on Graphics* 36, 4 (July 2017), 132:1–132:10. DOI: https://doi.org/10.1145/3072959.3073712

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

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.

Azin Neishaboori, Ahmed Saeed, Khaled A. Harras, and Amar Mohamed. 2014b. Low complexity target coverage heuristics using mobile cameras. In *MASS'14*.

Azin Neishaboori, Ahmed Saeed, Khaled A. Harras, and Amr Mohamed. 2014a. On target coverage in mobile visual sensor networks. In *MobiWac'14*.

Nikhil Nigam. 2014. The multiple unmanned air vehicle persistent surveillance problem: A review. *Machines* 2, 1 (2014), 13–72. DOI: https://doi.org/10.3390/machines2010013

D. Orfanus, E. P. de Freitas, and F. Eliassen. 2016. Self-organization as a supporting paradigm for military UAV relay networks. *IEEE Communications Letters* 20, 4 (April 2016), 804–807. DOI: https://doi.org/10.1109/LCOMM.2016.2524405

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

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

J. M. Palacios-Gasós, E. Montijano, C. Sagüés, and S. Llorente. 2016. Distributed coverage estimation and control for multirobot persistent tasks. *IEEE Transactions on Robotics* 32, 6 (Dec. 2016), 1444–1460. DOI: https://doi.org/10.1109/TRO.2016.2602383

J. M. Palacios-Gasós, Z. Talebpour, E. Montijano, C. Sagúés, and A. Martinoli. 2017. Optimal path planning and coverage control for multi-robot persistent coverage in environments with obstacles. In *2017 IEEE International Conference on Robotics and Automation (ICRA'17)*. 1321–1327. DOI: https://doi.org/10.1109/ICRA.2017.7989156

C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. 2014. Context aware computing for the internet of things: A survey. *IEEE Communications Surveys Tutorials* 16, 1 (First 2014), 414–454. DOI: https://doi.org/10.1109/SURV.2013.042313.00197

Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. 2012. Reconstructing 3D human pose from 2D image landmarks. In *Computer Vision–ECCV 2012*. Springer, 573–586.

Ahmed Saeed, Mostafa Ammar, Khaled A. Harras, and Ellen Zegura. 2015. Vision: The case for symbiosis in the internet of things. In *Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services (MCS'15)*. ACM, New York, NY, 23–27. DOI: https://doi.org/10.1145/2802130.2802133

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*.

R. Sahai, K. C. Galloway, and R. J. Wood. 2013. Elastic element integration for improved flapping-wing micro air vehicle performance. *IEEE Transactions on Robotics* 29, 1 (Feb. 2013), 32–41. DOI: https://doi.org/10.1109/TRO.2012.2218936

M. Schwager, B. J. Julian, M. Angermann, and D. Rus. 2011. Eyes in the sky: Decentralized control for the deployment of robotic camera networks. *Proceedings of IEEE* 99, 9 (Sept. 2011), 1541–1561. DOI: https://doi.org/10.1109/JPROC.2011.2158377

M. Schwager, M. P. Vitus, S. Powers, D. Rus, and C. J. Tomlin. 2017. Robust adaptive coverage control for robotic sensor networks. *IEEE Transactions on Control of Network Systems* 4, 3 (Sept. 2017), 462–476. DOI: https://doi.org/10.1109/TCNS.2015.2512326

Reza Shakeri, Mohammed Ali Al-Garadi, Ahmed Badawy, Amr Mohamed, Tamer Khattab, Abdulla Al-Ali, Khaled A. Harras, and Mohsen Guizani. 2018. Design challenges of multi-UAV systems in cyber-physical applications: A comprehensive survey, and future directions. *arXiv preprint arXiv:1810.09729* (2018).

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

S. Tijmons, G. C. H. E. de Croon, B. D. W. Remes, C. De Wagter, and M. Mulder. 2017. Obstacle avoidance strategy using onboard stereo vision on a flapping wing MAV. *IEEE Transactions on Robotics* 33, 4 (Aug. 2017), 858–874. DOI: https://doi.org/10.1109/TRO.2017.2683530

P. Tokekar and V. Isler. 2014. Polygon guarding with orientation. In *2014 IEEE International Conference on Robotics and Automation (ICRA'14)*. 1014–1019.

P. Tokekar, V. Isler, and A. Franchi. 2014. Multi-target visual tracking with aerial robots. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 3067–3072. DOI: https://doi.org/10.1109/IROS.2014.6942986

Jorge Urrutia et al. 2000. Art gallery and illumination problems. *Handbook of Computational Geometry* 1, 1 (2000), 973–1027.

E. Vanhoutte, F. Ruffier, and J. Serres. 2017. A quasi-panoramic bio-inspired eye for flying parallel to walls. In *2017 IEEE SENSORS*. 1–3. DOI: https://doi.org/10.1109/ICSENS.2017.8234110

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

Pengpeng Wang, Ramesh Krishnamurti, and Kamal Gupta. 2007. Metric view planning problem with traveling cost and visibility range. In *2007 IEEE International Conference on Robotics and Automation*. IEEE, 1292–1297.

Xiaoli Wang, Aakanksha Chowdhery, and Mung Chiang. 2016. SkyEyes: Adaptive video streaming from UAVs. In *Proceedings of the 3rd Workshop on Hot Topics in Wireless (HotWireless'16)*. ACM, New York, NY, 2–6. DOI: https://doi.org/10.1145/2980115.2980119

X. Wang, A. Chowdhery, and M. Chiang. 2017. Networked drone cameras for sports streaming. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS'17)*. 308–318. DOI: https://doi.org/10.1109/ICDCS.2017.200

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

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

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.

R. J. Wood. 2008. The first takeoff of a biologically inspired at-scale robotic insect. *IEEE Transactions on Robotics* 24, 2 (April 2008), 341–347. DOI: https://doi.org/10.1109/TRO.2008.916997

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

L. D. Xu, W. He, and S. Li. 2014. Internet of things in industries: A survey. *IEEE Transactions on Industrial Informatics* 10, 4 (Nov. 2014), 2233–2243. DOI: https://doi.org/10.1109/TII.2014.2300753

Enes Yildiz, Kemal Akkaya, Esra Sisikoglu, and Mustafa Y. Sir. 2014. Optimal camera placement for providing angular coverage in wireless video sensor networks. *IEEE Transactions on Computers* 63, 7 (2014), 1812–1825.

S. Yu and C. S. G. Lee. 2014. Network lifetime maximization in mobile visual sensor networks. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 3800–3805.

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*.

A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. 2014. Internet of things for smart cities. *IEEE Internet of Things Journal* 1, 1 (Feb. 2014), 22–32. DOI: https://doi.org/10.1109/JIOT.2014.2306328

Dimitrios Zorbas, Tahiry Razafindralambo, Francesca Guerriero, et al. 2013. Energy efficient mobile target tracking using flying drones. *Procedia Computer Science* 19 (2013), 80–87.

Received May 2017; revised January 2019; accepted March 2019