Mobile Target Coverage and Tracking on Drone-Be-Gone UAV Cyber-Physical Testbed

Mouhyemen Khan, Karel Heurtefeux, Amr Mohamed, *Senior Member, IEEE*, Khaled A. Harras, *Senior Member, IEEE*, and Mohammad Mehedi Hassan, *Member, IEEE*

***Abstract*—Mobile wireless sensor networks have been extensively deployed for enhancing environmental monitoring and surveillance. The availability of low-cost mobile robots equipped with a variety of sensors makes them promising in target coverage tasks. They are particularly suitable where quick, inexpensive, or nonlasting visual sensing solutions are required. In this paper, we consider the problem of low complexity target tracking to cover and follow moving targets using flying robots. We tackle this problem by clustering targets while estimating the camera location and orientation for each cluster separately through a cover-set coverage method. We also leverage partial knowledge of target mobility to enhance the efficiency of our proposed algorithms. Three computationally efficient approaches are developed: *predictive fuzzy*, *predictive incremental fuzzy*, and *local incremental fuzzy*. The objective is to find a compromise among coverage efficiency, traveled distance, number of drones required, and complexity. The targets move according to one of the following three possible mobility patterns: random waypoint, Manhattan grid, and reference point group mobility patterns. The feasibility of our algorithms and their performance are also tested on a real-world indoor testbed called *drone-be-gone*, using Parrot AR.Drone quadcopters. The deployment confirms the results obtained with simulations and highlights the suitability of the proposed solutions for real-time applications.**

***Index Terms*—Algorithm design and analysis, autonomous agents, clustering algorithms, cyber-physical systems, unmanned vehicles.**

I. Introduction

LOW-COST mobile robots equipped with a variety of sensors have become popular in quick, inexpensive, or nonlasting visual sensing scenarios such as disaster recovery, geographical surveys, traffic monitoring, etc. The units deployed for coverage tasks are either static or dynamic. Dynamic units often take the form of unmanned ground or aerial vehicles (UGVs and UAVs). The small size and maneuverability of UAVs make them particularly suitable to access what would otherwise be inaccessible regions. Such agility deems UAVs attractive even in many indoor cyber-physical and smart IoT applications such as health and smart home monitoring, in addition to diverse outdoor applications such as cinematography, and home and public security [1]–[3]. However, due to the necessity of effective and agile sensing coverage, efficient algorithms need to be developed, and testbeds for both indoor and outdoor contexts are highly desirable.

In our context, we are motivated by the need for computationally efficient algorithms for the autonomous control of mobile visual sensors, which are limited in their energy and computational capabilities. We have considered how to find the minimum number of cameras to cover a high ratio of a set of targets as a clustering problem [4]. Targets in proximity to each other are clustered, and the cameras' location and orientation are then estimated for each cluster. However, the problem of optimal camera location to maximize coverage has been shown to be NP-complete in many variations for both area and target coverage in both isotropic [5] and anisotropic sensors [6]. Therefore, it has been simplified in many forms in the field of robotics and sensor networks [6]–[8]. Despite all these efforts, finding a computationally efficient algorithm with an acceptable number of cameras for an arbitrary number of targets, while considering target mobility, has remained a challenge.

In this paper, we consider moving targets, while leveraging partial knowledge of their mobility patterns, and propose three algorithms based on a fuzzy *C* clustering and cover-set (FCCC) algorithm [4]. The three algorithms are *predictive fuzzy algorithm* (PFA), *predictive incremental fuzzy algorithm* (PIFA), and *local incremental fuzzy algorithm* (LIFA). Each algorithm is based on the fuzzy C-means (FCM) algorithm for clustering targets and our own cover-set coverage method (CSCM) for determining a camera's pose (location and orientation) to cover each cluster. These algorithms aim to enhance target coverage and reduce the traveled distance of the mobile cameras. Such an aim is determined in order to facilitate the implementation of the proposed algorithms in real-time coverage and tracking scenarios of mobile targets while avoiding long flying distances and time-consuming UAV navigation that may lead to missing instant target locations.

We evaluate our algorithms via extensive simulation with a wide range of metrics and parameters that reflect different applications. We use the following four metrics to assess the performance of our algorithms and their capabilities in real-time: coverage ratio, number of mobile cameras, traveled distance, and time complexity. We also utilize three representatives of different mobility patterns for our targets, namely, *random waypoint* (RWP), *Manhattan grid* (MG), and the *reference point*

Manuscript received January 21, 2017; revised May 25, 2017 and September 5, 2017; accepted November 16, 2017. Date of publication December 14, 2017; date of current version November 22, 2018. This work was supported by King Saud University and Qatar University under GCC Co-Funding Program Grant GCC-2017-009. The work of K. A. Harras was supported by Qatar Foundation Seed Funding to Carnegie Mellon University. *(Corresponding author: Amr Mohamed.)*

M. Khan, K. Heurtefeux, and A. Mohamed are with the Department of Computer Science and Engineering, Qatar University, Doha, Qatar (e-mail: mouhyemen.khan@gmail.com; heurtefeux@gmail.com; amrm@qu.edu.qa).

K. A. Harras is with the Computer Science Department, Carnegie Mellon University, Doha, Qatar (e-mail: kharras@cs.cmu.edu).

M. M. Hassan is with the College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia (e-mail: mmhassan@ksu.edu.sa).

Digital Object Identifier 10.1109/JSYST.2017.2777866

1937-9234 © 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

group (RPG) mobility patterns. This variety allows us to investigate the behavior of our algorithms in a wider context of situations and applications. We show that the three proposed algorithms exhibit complementary behaviors, which deem them applicable in a variety of applications. On one hand, PFA and PIFA use mobility prediction of targets; hence, they are generally more efficient in coverage and traveled distance but have higher time complexity and number of mobile cameras. On the other hand, LIFA is lightweight and requires lesser cameras but exhibits a lower coverage ratio.

Finally, to obtain better insights into the performance of our algorithms in real-world situations, we present a case study of the proposed algorithms on our *drone-be-gone (DbeG)* testbed. DbeG is an agile, inexpensive, and general-purpose cyber-physical system (CPS) testbed that we developed using off-the-shelf UAVs with centralized or distributed control and/or processing. *DbeG* comes with features that include vision-based two-dimensional (2-D) localization, autonomous navigation for multiple UAVs, a simulation environment, and an external processing unit (EPU) mounted on the UAV. Overall, the drones, running our algorithms, were capable of reacting in real-time to the moving targets, and performance results were consistent with those obtained in our simulations.

The remainder of this paper is organized as follows. Section II explores the related work on target coverage/tracking and testbed implementations. In Section III, we formally state the assumptions and present our system model. We describe our algorithms in Section IV, present mobility patterns in Section V, and evaluate the performance of the proposed algorithms in Section VI. We describe the *DbeG's* testbed architecture in Section VII, and present our case study that integrates our proposed solutions with DbeG in Section VIII. Finally, we conclude and present our future work in Section IX.

II. Related Work

Full-view coverage: It was introduced as a variant of area coverage with the added goal that a target is covered from all angles [9]. Using an algorithm based on potential field [10], the goal is for one UAV to cover maximum space in minimum time without revisiting regions. A related problem called barrier coverage has the goal to detect any target crossing a barrier within area of interest [11]. *Our work differs from the above-presented studies in its objectives. Our goal is to design low-complexity and efficient coverage algorithms for low-cost UAV(s) while considering moving targets.*

Camera coverage: It generally falls into the following two categories: covering an area and covering a set of targets. The first category is akin to the old art gallery problem [12], wherein the goal is to place the minimum number of guards that can observe the whole gallery together. However, replacing guards (isotropic sensors) with cameras (anisotropic sensors) makes the problem more complex due to the limited range and angle of view (AoV) of cameras. Even though space and camera orientations are discretized, the problem remains unscalable [13]–[15]. As for the second category, maximizing target coverage has been proven to be NP-complete for most of its variant formulations [6], [16]. As a result, many studies convert these problems to more tractable forms by simplifying assumptions and providing heuristics. The most common coverage condition is that a target is visible with an acceptable quality from at least one camera [17]–[21]. However, in some applications, a target needs to be covered from all angles, i.e., full angular coverage [8]. In this paper, we assume that targets are covered if they are seen by at least one camera from any angle. *We differ from the aforementioned since we aim to maximize coverage over a given set of mobile targets, rather than a given area or target distribution function.*

Optimal camera location/orientation: Cheong *et al.* [17]–[21] propose solutions for this problem using simplified assumptions. In [17], fixed camera location and unknown discrete camera pans are assumed. Camera locations are determined for directional targets as opposed to simplifying targets as points or blips in the system [22]. In [19], randomly scattered set of directional sensors are assumed from which a set is selected one at a time. A similar problem is studied in [18] where an "active" set of camera sensors and their orientations are selected from a larger pool of already placed ones. In [20], fixed camera locations with varying pan and tilt are assumed to keep track of mobile targets using neighbor cooperation. In [21], rotating directional sensors are used to optimize different coverage objectives. Since the optimal solution to these problems is also NP-complete, heuristics are suggested. In [4], camera location/orientation to cover mobile targets activities is addressed with the objective of maximizing average coverage quality during the course of target movement in approximated linear paths. *We differ from all of the aforementioned, except [4], in assuming that camera locations are neither fixed nor chosen from a set of random selected points. In [4], the specific objective of camera coverage is considered only for targets moving in linear paths.*

Search and tracking (SAT): Coordinated control techniques are used to autonomously search and track multiple targets using Bayesian filtering [23]. UAVs can switch their operational mode from search to track. The proposed algorithm was applied to a marine search and rescue scenario. A distributed sensor network along with a team of UAVs using distributed mission area probability maps to search and track mobile ground targets [24]. A vision-based persistent search and track technique using multiple UAVs is addressed. The main contribution is target detection and task assignment methods [25]. Keeping track of all discovered targets while searching for new targets by controlling the direction of the vision sensor and UAV's motion is addressed [26]. Only one UAV is used in this work. A team of fixed wing UAVs perform searching and tracking tasks over urban environment [27]. However, the extracted road map is assumed known to all UAVs. *Our goal differs from the aforementioned in its objectives. We are not addressing SAT techniques or algorithms. We focus on the tracking segment alone using low-complexity coverage algorithms while assuming that the location of targets is known. Additionally, our work includes different mobility patterns and hence the trajectory of UAVs is not known beforehand.*

Testbed implementations: Expensive motion capture systems are typically utilized for indoor testbeds [28]–[31]. Attempts are made to develop low-cost testbeds using off-the-shelf AR.Drones as UAVs. Researchers are motivated to integrate research and education using AR.Drones and RoboDuino [32]. Up&Away testbed uses AR.Drone and is an initial attempt to build a low-cost general-purpose CPS testbed [33]. However, these testbeds are either custom-built for specific applications, perform simulated or coarse-grained control of UAVs, or lack distributed control and/or processing. Efforts are invested to add EPUs on the AR.Drone, which enables distributed control and/or processing. Localization based on a GPS-barometer hybrid and drone-control estimation algorithm is implemented

on EPU Beaglebone to control AR.Drone [34]. University of Tubingen mounted three EPUs on AR.Drone, namely, ATMEL ATMega, Gumstix Overo Fireboard, and Hardkernel Odroid-U2, and demonstrated figure flying and on-board computer vision [35]. *We set ourselves apart by developing an agile, low-cost, general-purpose indoor testbed with the ability to switch between centralized and distributed control and/or processing seamlessly.*

III. System Model for Target Coverage

In this section, we first discuss the FCM algorithm for clustering fixed targets based on their locations. We then summarize the model for designing the coverage algorithms based on the cover-set problem.

A. Fuzzy C Clustering and Cover-Set Algorithm

Let $\mathcal{T}_t$ be a set of N fixed targets $T_i, \quad i \in \{1 \dots N\}$ at time t. We then define the cover-set, and the coverage problem as follows.

Definition 1: cover-set A cluster of targets $T_i \in \mathcal{T}, \quad i \in \{1 \dots n\}$ form a cover-set with respect to a certain camera specifications, i.e., given range R_{max} and AOV, if it is feasible to cover all of them by one such camera.

Definition 2: camera coverage Given a set of targets $\mathcal{T}_t$ in a 2-D plane and using homogeneous horizontal cameras with a given maximum AOV and maximum coverage range R_{max}, find the minimum number of clusters such that each cluster is a cover-set, the best camera position P_k and orientation for each cover-set such that all targets are visible by at least one camera.

Based on the definitions mentioned above, the C clustering and cover-set (CCC) algorithm searches for the minimum number of clusters such that each cluster is a cover-set and all targets of all cover-sets are visible by at least one camera. The FCCC seeks an overall target coverage ratio TCR by allowing targets to belong to multiple clusters with different probabilities while trying to recluster the uncovered targets to achieve the overall TCR.

To divide targets into clusters using arbitrary number of cluster C, the FCM algorithm [36] is used. The FCM algorithm partitions a set of targets x_i into a set of C fuzzy clusters. The main objective of FCM is to minimize the following objective function:

$$\text{FCM}(\mathcal{T}_t) : \min_{C} \sum_{c=1}^{C} \sum_{i=1}^{N} u_{i,c}^{m} \|x_i - v_c\|. \tag{1}$$

N and C represent the number of targets and the number of clusters, respectively. x_i is the location vector of target T_i, m is the fuzziness index with $m \in [1, \infty]$, v_c represents the kth center-of-cluster, $u_{i,c}$ represents the membership of the ith target location to the cth cluster, and $||x_i - v_c||$ is the Euclidean distance between the ith target location and the cth cluster center. Hence, the overall complexity of FCM is $O(CN)$.

Once the clusters are formed, the camera location and orientation are found according to the coverage method for one cover-set. More details on how the CSCM is derived can be found in [4]. The overall complexity of the CSCM is $O(N2\pi/\Delta\phi)$. The overall complexity of the FKCC algorithm can then be estimated as $O(MN2\pi/\Delta\phi + MCN)$, where M is the complexity of the search scheme for the best number of clusters, which is a function of the maximum number of cameras C_m, e.g., for binary search $M = \log(C_m)$. Although the complexity of FCCC is linear in N and C, which may deem the algorithm feasible by default for real-time target tracking, it may not be feasible in many applications to run the algorithm to find the new locations and orientations of cameras at every time step, because of the following reasons.

1) From one time step to another, we have to select which camera to move from current position i to new position j, which may require complex path planning techniques [37].
2) Frequent navigation of cameras mounted on UAVs could be a time-consuming process and may lead to increasing the number of uncovered targets while the UAV is navigating across locations.
3) Also, frequent mobility of UAVs increases the flying distances and the mechanical energy consumption, and hence minimizes the flying time.

Therefore, for real-time target tracking, it is highly desirable to address the tradeoff between running the FCCC algorithm at every time step to guarantee the coverage of moving targets at all times while reducing the flying distances and time-consuming navigation of UAVs to avoid missing the instant locations of targets.

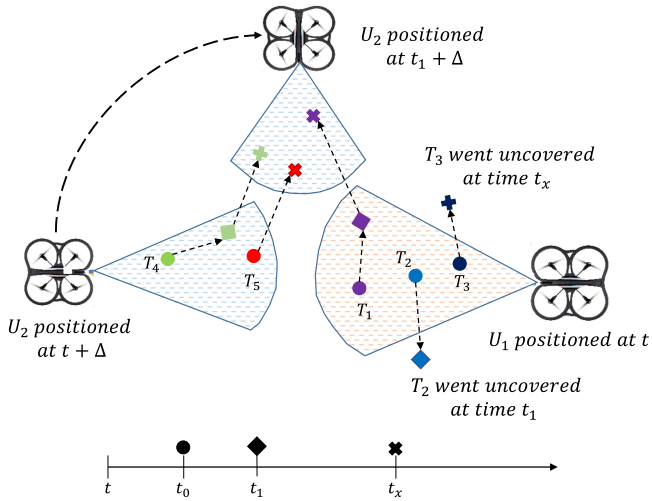

Fig. 1. System model: UAVs are abbreviated as U_x, targets as T_x, and time as t_x. A generic model is shown where a number of UAVs cover n targets using the coverage algorithm. The position of the targets evolve with time based on our mobility patterns. Uncovered targets are addressed based on a prediction model.

The system model is illustrated in Fig. 1. UAVs are abbreviated as U_x, targets as T_x, and time as t_x. The number of UAVs and their positions are determined based on our coverage algorithms, as discussed in Section IV. Target locations evolve with time based on one of our mobility patterns, as discussed in Section V. As seen from the figure, targets T_2 and T_3 go uncovered. Since targets T_1, T_4, T_5 are being covered by UAV U_3, UAVs U_1 and U_3 need to update their positions to cover targets T_2 and T_3.

IV. Mobile Target Coverage and Tracking

In this section, we present our proposed algorithms for mobile target coverage and tracking, and discuss their major characteristics, assuming they have partial or full knowledge about target locations during the whole coverage time T_{max}.

Algorithm 1: Predictive Fuzzy Algorithm.

1 $t \leftarrow 1$;
2 **while** $t \leq t_{max}$ **do**
3 Determine the set $\mathcal{T}_{t \rightarrow t+\Delta t}$, the location of the mobile targets for the next Δt timesteps ;
4 $c \leftarrow c_0$;
5 **while** $CR_{t \rightarrow t+\Delta t} < TCR$ **do**
6 $\mathcal{C}_{t \rightarrow t+\Delta t} = FCM(\mathcal{T}_{t \rightarrow t+\Delta t}, c)$;
7 $Cam_i = CSCM(\mathcal{C}_{t \rightarrow t+\Delta t})$;
8 Compute $CR_{t \rightarrow t+\Delta t}$, the coverage ratio obtained with Cam on $\mathcal{T}_{t \rightarrow t+\Delta t}$;
9 $c \leftarrow f(c)$;
10 **end**
11 $t \leftarrow t + \Delta t$;
12 **end**

A. Predictive Fuzzy Algorithm

For low target mobility, the predictive fuzzy algorithm (PFA) is based on the FCM algorithm defined above, which clusters real targets at time t, in addition to predicted targets at times $t + i \quad \forall i \in \{1 \dots \Delta t\}$. PFA predicts the movement for targets and includes the predicted new target locations along with current real targets. The entire set of current real targets and predicted target locations are then divided into cluster set $\mathcal{C}_{t+\Delta t}$. The algorithm's pseudo-code is depicted in Algorithm 1. In line 4, PFA starts from an initial number of clusters c_0 and performs a binary search for the best number of cameras Cam_i required to achieve the target coverage ratio TCR. In line 6, for each trial, the algorithm uses fuzzy clustering FCM to cluster the target set $\mathcal{T}_{t \rightarrow t+\Delta t}$. In line 7, it then calculates the cameras' locations and orientations through the angular search algorithm CSCM [4]. After Δt given time steps, the FCM algorithm is run again and new clusters along with their corresponding camera locations and orientations are determined.

The normalized complexity over time for PFA is $O(f(\hat{N})/\Delta t)$, where $\hat{N}$ is the total number of current and future target locations at time steps $t \rightarrow t + \Delta t$, and $f(\hat{N}) = M\hat{N}2\pi/\Delta\phi + MC\hat{N}$, as explained in Section III-A above. The pseudo-code of PFA is given in Algorithm 1.

B. Predictive Incremental Fuzzy Algorithm

For high target mobility, $\hat{N} \gg N$, which deems PFA inefficient even if Δt is large since the percentage of uncovered targets will be proportional to Δt. Therefore, we propose the predictive incremental fuzzy algorithm (PIFA), which is also based on the FCCC algorithm and uses predictions like PFA.

With FCM and CSCM as its basis, and akin to PFA, the prediction model is also used in PIFA. However, unlike PFA, PIFA introduces predicted target locations incrementally. The current location of targets at t is fed to FCM and CSCM to determine camera poses. The coverage ratio is computed. Then, predicted locations of t_1 are included with current location of targets to redetermine the coverage ratio for t. If the ratio is higher, predicted locations of t_2 are included to the original locations and the ratio is computed once again. This iteration breaks only when a lesser ratio is obtained or a time threshold is met. After Δt time steps, the FCM algorithm is run again and new clusters, cameras' locations, and orientations are determined.

Algorithm 2: Predictive Incremental Fuzzy Algorithm.

1 $t \leftarrow 1$;
2 **while** $t \leq t_{max}$ **do**
3 Determine $\mathcal{T}_t, \mathcal{T}_{t+1}, ..., \mathcal{T}_{t+\Delta t}$, the sets of the targets for the next Δt timesteps ;
4 $c \leftarrow c_0$;
5 **while** $CR_t < TCR$ **do**
6 $\mathcal{C}_t = FCM(\mathcal{T}_t, c)$;
7 $Cam_i = CSCM(\mathcal{C}_t)$;
8 Compute CR_t, the coverage ratio obtained with Cam on $\mathcal{T}_t$;
9 $c \leftarrow f(c)$;
10 **end**
11 $j \leftarrow 1$;
12 **while** $j \leq \Delta t$ **do**
13 Add $\mathcal{T}_{t+j}$ to the closest cluster ;
14 $Cam'_i = CSCM(\mathcal{C}_t)$;
15 Compute CR_{t+j}, the coverage ratio obtained with Cam' on $\mathcal{T}_t$;
16 $j \leftarrow j + 1$;
17 **end**
18 $t \leftarrow t + \Delta t$;
19 **end**

For PIFA, the FCM is run every Δt with a normalized time complexity of $O(MCN/\Delta t)$, while the cover-set is run each time step $i \quad \forall i \in 1 \dots \Delta t$, which in total is $O(N2\pi/\Delta\phi)$. Therefore, the normalized time complexity over time for PIFA is $O(f(N)/\Delta t + N2\pi/\Delta\phi)$, and the pseudo-code is given in Algorithm 2.

C. Local Incremental Fuzzy Algorithm

Unlike PFA and PIFA, the local incremental fuzzy algorithm (LIFA) is not based on prediction. Instead, the algorithm works as follows. Target locations at t are clustered using FCM and then provided as input to the CSCM to determine camera pose(s). The coverage ratio is computed. At t_1, FCM is not run, thereby keeping the same clusters. However, the CSCM is run for determining the camera pose(s) at every time step. This will handle target tracking. After Δt given time steps, FCM is run once again to determine new set of cluster(s). Trying to keep the targets associated with the same cluster, only the new camera's location and orientation are computed at each time step to minimize the uncovered targets. Therefore, LIFA is expected to provide low complexity solution for application scenarios where targets tend to move in well-behaved clusters, e.g., group mobility pattern.

Similar to PIFA, the normalized time complexity of LIFA is $O(f(N)/t_{\text{max}} + N2\pi/\Delta\phi)$, and the pseudo-code is given in Algorithm 3.

V. Mobility Patterns

In this section, we present and discuss three different mobility patterns, which are used to model a target's or set of targets' motion. The mobility model represents various characteristics such as location, velocity, and motion of a target over time.

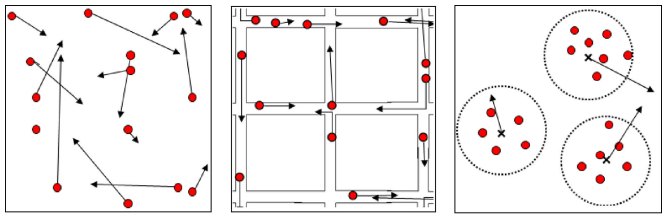

Fig. 2. Pictorial representation of (a) *Random waypoint*, (b) *Manhattan grid*, and (c) *reference point group* mobility patterns. A red disc corresponds to a target with the vector pointing to the direction it is headed.

Algorithm 3: Local Incremental Fuzzy Algorithm.

$t \leftarrow 1$;
$c \leftarrow c_0$;
while $CR_t < TCR$ **do**
  $\mathcal{C}_t = FCM(\mathcal{T}_t, c)$;
  $Cam_i = CSCM(\mathcal{C}_t)$;
  compute CR_t, coverage ratio obtained for the targets at time t ;
  $c \leftarrow f(c)$;
end
while $t \leq t_{max}$ **do**
  $Cam_i = CSCM(\mathcal{C}_t)$;
  $t \leftarrow t + 1$;
end

Characterizing this model helps to investigate an algorithm's potential in handling target behaviors in different settings or contexts. For example, target behavior in an urban setting on the streets or roads is different from within a shopping mall or stadium. Moreover, apart from accounting for various contexts, the pros and cons of an algorithm can also be noticed through different mobility patterns.

A. Random WayPoint Mobility Pattern

A random WayPoint (RWP) mobility model is a commonly used mobility model in *ad hoc* networking research community. In our RWP model [see Fig. 2(a)], each target chooses uniformly at random a destination point or waypoint in the region of deployment. A target moves to this destination with a velocity v chosen uniformly at random in the interval $[v_{\min}; v_{\max}]$. In our simulations, we fixed $v_{\min}$ and $v_{\max}$ to 5.0 m·s^{-1} and 11.0 m·s^{-1}, respectively. When it reaches the destination, it remains static for a predefined time and then starts moving again according to the same rule.

B. Manhattan Grid Mobility Pattern

A Manhattan grid (MG) model [see Fig. 2(b)] uses a grid road topology and is suitable to model the movement in urban area, where the streets and roads are defined in an organized manner. In the MG model, the target moves in horizontal or vertical direction. At each intersection of a horizontal and vertical street, MG employs a probabilistic approach in selecting the targets' movements: the probability of going straight is 0.5, while going left or right is 0.25. If the target reaches one border of the deployment area, it goes in the opposite direction. With this model, each target moves with a constant velocity chosen uniformly at random in the interval $[v_{\min}; v_{\max}]$.

TABLE I
SIMULATION PARAMETERS BASED ON THE APPLICATION AND THE DRONE'S CAPABILITIES

Mobility pattern	RWP, MG, and RPG
Size of the deployment area	50 m × 50 m
Number of targets	20, 40, 60, 80, 100
Speed of moving targets	From 5.0 to 11.0 m·s^{-1}
Simulation time	50 s
Prediction time step	5 s
UAV camera's angle of view	90°
UAV camera's range	15 m

C. Reference Point Group Mobility Pattern

The reference point group (RPG) model [see Fig. 2(c)] is useful to simulate group behavior where each target belongs to a group. Every target follows a logical center (or group leader) that determines the group's motion behavior. The targets in a group are randomly distributed around the reference point. Different targets use their own mobility model and are then added to the reference point, which steers them in the direction of the group. At each instant, every target has a speed and direction that is derived by randomly deviating from that of the group leader. In our RPG model, the movement of the group leader follows the RWP mobility pattern. The movement of the group leader defines its own motion and also provides the general motion trend for the whole group. Each member of this group deviates from this general motion vector by some degree.

Worth noting here is that other mobility patterns have also been experimented, e.g., STEPS mobility [38] and Brownian notion [39]. Although their performance results were slightly different with respect to target tracking, they seem to show similar trends to the mobility patterns discussed above, which stems from the controlled nature of such patterns compared to RWP mobility with a high degree of randomness. Therefore, their results have not been included in Section VI.

VI. PERFORMANCE EVALUATION

This section will cover the performance of each proposed algorithm, based on each mobility pattern, while subjected to various simulation parameters. The simulation results are averaged over 20 runs and are obtained using MATLAB R2014b. The confidence interval error is set to be 95%.

A. Simulation Parameters

The simulation parameters are based on two major criteria—application and UAV. For instance, in the case of application-based, the velocity of a target is designed to match the average velocity of a car in a city. The parameters are summarized in Table I.

B. Evaluation Metrics

To evaluate the performance of the algorithms, we consider the following four metrics: *coverage ratio*, *number of mobile cameras* needed to ensure coverage, *distance traveled* by mobile

cameras, and *time complexity* of the coverage algorithms. They are defined as follows.

Coverage ratio: It measures the percentage of targets covered by at least one camera. This metric is measured with respect to the number of targets and time with a fixed number of targets (set to 100).

Number of drones: It measures the number of mobile cameras needed to cover the moving targets. This metric is also measured with respect to the number of targets and time with a fixed number of targets (set to 100).

Traveled distance: It measures the sum of distances traveled by all UAVs to cover targets. It is measured with respect to time with a fixed number of targets (set to 100).

Time complexity: It measures the average time taken (in seconds) by each simulation with respect to the number of targets, using a laptop Lenovo Y50 with single CPU.

C. Results and Analysis

In this section, we present the simulation results obtained from MATLAB for the three protocols (PFA, PIFA, and LIFA) presented in Section IV considering three mobility models (RWP, MG, and RPG). The results are averaged over 20 runs for each case. The error bar has 95% confidence interval.

Effect on coverage ratio: Fig. 3(a)–(c) shows the coverage ratio with respect to the number of targets. Regardless of the number of targets or mobility nodes, the average coverage ratio is always above 60%. However, noticeable differences exist among the algorithms due to their intrinsic mechanisms. PFA exhibits the best performance (above 90%) followed by PIFA and LIFA (between 70% and 80% for the RWP mobility model, 60% and 80% for the MG mobility model, and 70% and 90% for the RPG mobility model). LIFA works without any information on the targets' future locations and does not update the clustering to reposition mobile cameras, whereas PFA and PIFA use prediction and adapt the number of drones every 5th time step.

Fig. 3(d)–(f) shows considerable variations in the coverage ratio over time with a fixed number of targets. Under RWP and MG mobility patterns, LIFA covers more than 90% of the targets only to quickly drop and stabilize around 70% for RWP mobility and 60% for MG mobility. When the targets follow the RPG mobility model and stay grouped, the initial clustering stays valid thereafter. In this circumstance, LIFA performs similar to PFA (around 90%). On the other hand, PIFA updates the clustering and adapts the number of drones required every 5th time step. It also limits the drop in the coverage ratio while increasing the average coverage ratio. PFA also updates its coverage every 5th time step while trying to cover maximum of targets for the current time and future.

Effect on the number of drones required for tracking: Fig. 4(a)–(c) shows number of drones with respect to number of targets. As expected, LIFA is efficient in terms of number of drones, and these results are similar to those of [4]. LIFA does not consider prediction and does not update number of cameras in the field during simulation. As a result, only a constant few mobile cameras are required. PFA and PIFA, however, have stronger constraints and need to consider prediction. The MG model slightly increases the number of cameras needed for PFA and PIFA, while the RPG model drastically reduces the number of cameras needed. The MG model, by its grid topology, tends to fragment the targets over time, leading to more clusters by PFA and PIFA. RPG, on the other hand, simulates group behavior where each target belongs to a group and follows a logical center. The targets tend to stay together, keeping initial number of clusters low and constant. Hence, the number of cameras required in RPG is lesser.

Fig. 4(d)–(f) shows number of drones with respect to time for 100 targets. Under RWP and MG mobility patterns, number of drones is not constant over time for PFA and PIFA. Due to prediction and reclustering at every 5th time step, there is an unpredictable change in the number of drones over time. For LIFA, since the number of clusters remains constant so does the number of drones over time. For the RPG model, due to the group behavior, number of cameras required is significantly lesser than RWP and MG.

Effect on traveled distance: Fig. 5(a)–(c) represents the sum of traveled distances for all moving cameras with respect to time for the 100 targets. Traveled distance is computed every time step for LIFA and every five time steps for PIFA and PFA. Even though the number of drones is higher, the traveled distance is significantly lower for PFA and PIFA since they rely on prediction. Moreover, higher number of cameras for covering small clusters leads to a lower overhead in terms of the overall traveled distance. With MG mobility, the targets are more prone to scatter over time, hence increasing the traveled distance for PFA and LIFA. With RPG mobility, scattering is lesser since targets stick to respective groups. As a consequence, the traveled distance is also low compared to RWP and MG.

Effect on time complexity: Fig. 6(a)–(c) represents time complexity with respect to the number of targets. As expected, LIFA has the lowest cost in terms of time complexity regardless of the mobility model. By its design, LIFA runs FCM only at the beginning, and the CSCM is run every time step, whereas PFA and PIFA compute FCM every five time steps while considering predicted target locations. The difference between PFA and PIFA is that FCM is stopped for PIFA if the cluster coverage at t_{n+1} penalizes the coverage at t_n. This difference has a significant impact on the time complexity, especially as targets increase.

D. Results Summary

PFA: It shows excellent performance in coverage ratio and traveled distance. However, there is a higher overhead on number of drones required and time complexity.

PIFA: It shows decent performance in coverage, whereas excellent performance in terms of traveled distance. However, akin to PFA, it also has a high cost in terms of number of drones required and time complexity.

LIFA: It exhibits the least time complexity with a constant few drones. However, it suffers in coverage ratio and traveled distance, except in cases where the mobility pattern has group modeling or around few logical centers.

VII. Drone-Be-Gone: From Theory to Practice

To validate our proposed algorithms in practical environment, we have developed the DbeG testbed covered in this section.

A. Architecture

Here, we introduce the system architecture of our testbed named *DbeG* (see Fig. 7). It comprises two major modules,

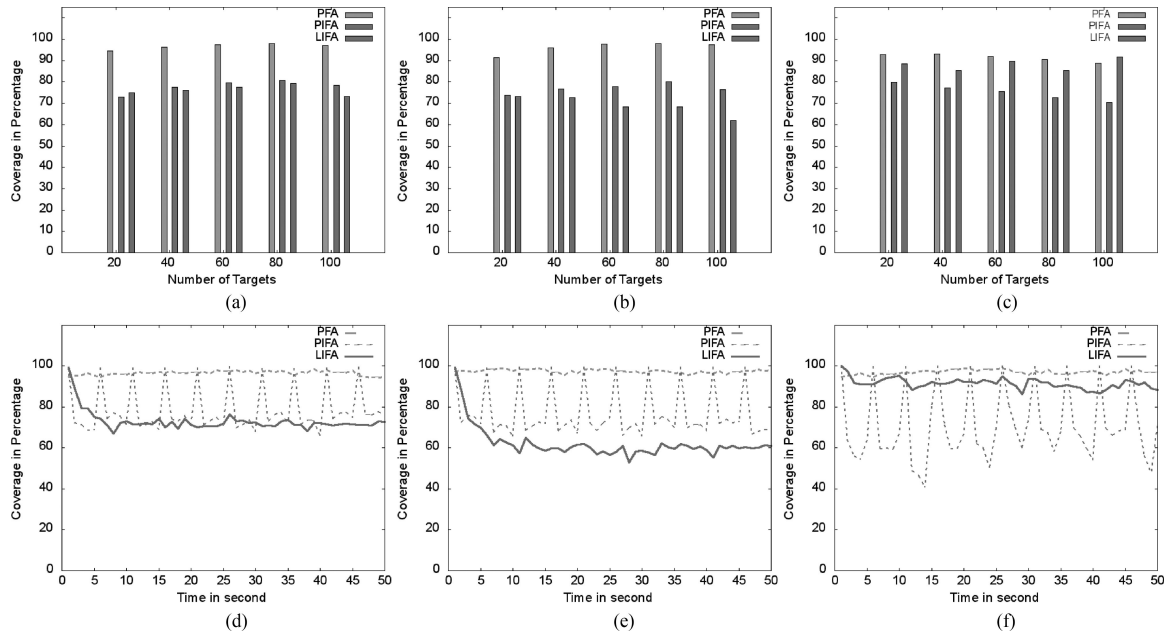

Fig. 3. Average coverage ratio with respect to number of targets (a)–(c) and time with 100 targets (d)–(f). (a) and (d) Random waypoint mobility. (b) and (e) Manhattan grid mobility. (c) and (f) Reference point group mobility.

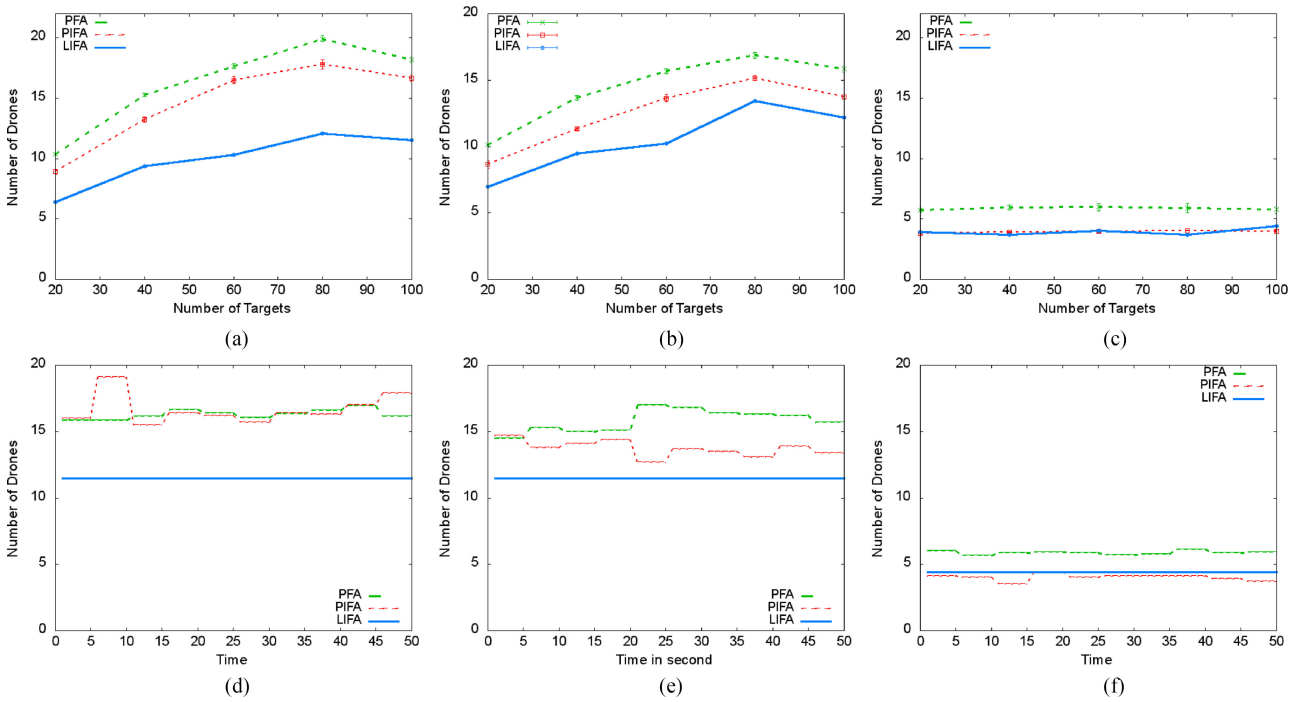

Fig. 4. Average number of mobile cameras required based on: number of targets (a)–(c) and time with 100 targets (d)–(f). (a) and (d) Random waypoint mobility. (b) and (e) Manhattan grid mobility. (c) and (f) Reference point group mobility.

central and *client*, along with a *multihomed* module. The submodules inside *central* and *client* modules are exclusive to them. In the case of *multihomed* module, the submodules can be integrated into either of the major modules depending on the system requirements.

Central module: This module is responsible for localizing UAVs and targets in 2-D. In our setup, the central module is run on a Lenovo Y50 and uses a master camera as the input for its localization component. We use an Axis 213 PTZ network camera, located directly above the testbed connected through an Ethernet cable to the laptop feeding it images at a frequency of 30 Hz. The central module also runs a *UAV and target localizer* component responsible for filtering noise, locating all UAVs and targets. The filtered and localized data are then fed to the adaptive tracker, which distinguishes and tracks each UAV.

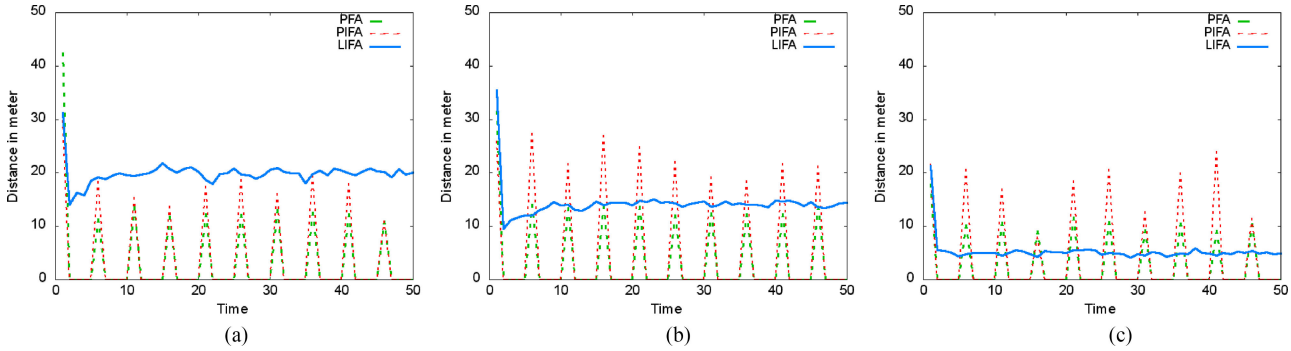

Fig. 5. Average traveled distance with respect to time for 100 targets. (a) and (d) Random waypoint mobility. (b) and (e) Manhattan grid mobility. (c) and (f) Reference point group mobility.

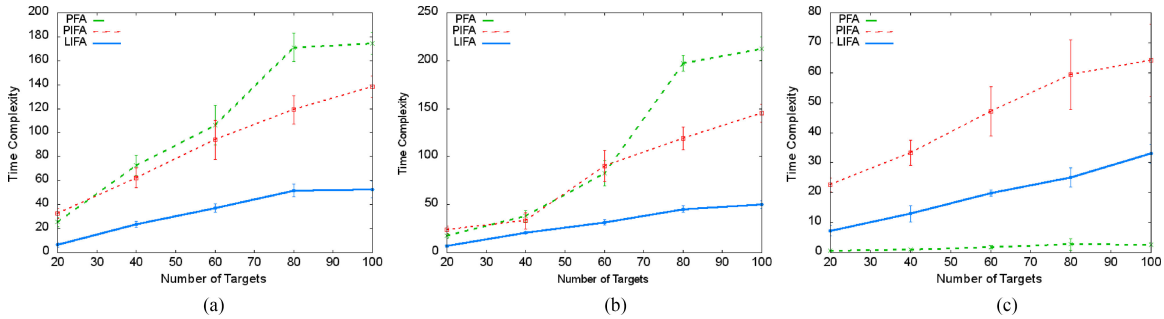

Fig. 6. Average time complexity with respect to the number of targets. (a) Random waypoint mobility. (b) Manhattan grid mobility. (c) Reference point group mobility.

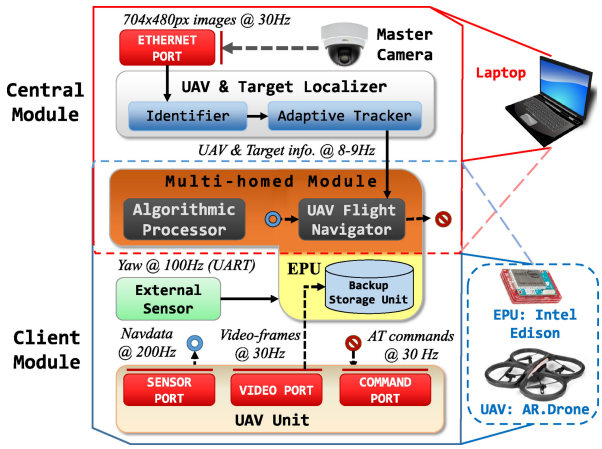

Fig. 7. Drone-be-gone architecture.

Client module: This module is the mobile component of *DbeG* and is composed of an EPU mounted on top of the UAV. We use AR.Drone as the UAV platform for *DbeG's* applications. Communication with AR.Drone takes place over Wi-Fi. For the EPU, we choose Intel Edison, which is a small computing device running Yocto Linux and powdered by Atom system-on-chip (SoC) dual-core CPU at 500 MHz and 1 GB RAM. MPU-6050 is used for obtaining accurate yaw angles. It is a low-cost external inertial measurement unit (IMU) sensor unsusceptible to magnetic field strength. The sensor combines a micro-electro-mechanical system three-axis gyroscope and three-axis accelerometer on the same silicon die along with an onboard digital motion process (DMP). A more complete description of the internal DMP algorithm can be found in [40]. Despite calibration, AR.Drone's sensor provides faulty yaw angles due to the magnetic distortions in our testbed's environment. The EPU is powered using a battery block and is hardwired with the external sensor using Arduino block. "Blocks" are electronic boards developed by SparkFun to utilize the 70-pin connector of Edison while maintaining the compatibility with miniature size of Edison.

Multihomed module: It is a special subset within *DbeG* whose submodules can belong to either modules; *central* or *client*. Flexibly housing its submodules in either module allows easy migration between a centralized to distributed platform. Its first submodule, *UAV flight navigator*, is responsible for autonomously maneuvering UAVs by taking in parameters from localizer (2-D coordinates of UAV) and IMU (yaw angles). Its second submodule, *algorithmic processor*, runs any computations for application testing on the testbed, e.g., coverage algorithms and mobility patterns.

B. DbeG Features

DbeG has the four main features described as follows, which are crucial in measuring the performance of any CPS application on the testbed.

1) Vision-Based 2-D Localization: As an initial implementation, we implement 2-D localization by hanging a network camera over the testbed. Although, Microsoft's Kinect sensor or ASUS Xtion Pro LIVE provides three-dimensional (3-D) information in the same setting, their maximum depth range is only 3.5 m [41]. For covering a wider area for better deployment of UAVs, we choose the Network Camera Axis 213

TABLE II
LOCALIZATION ACCURACY AGAINST GROUND TRUTH

Metric	ErrorX	ErrorY
Mean (cm)	4.13	3.41
Std. Dev. (cm)	1.30	1.83

PTZ. Our approach is feasible for applications bounded to a single plane such as search and rescue, path planning, and coverage but not applications requiring altitude adjustments. We use image-processing techniques and implement our own hybrid of intensity and contour detection for identifying the UAVs. We implement an *adaptive tracking wndow* (ATW) algorithm to our localization routine to differentiate multiple UAVs.

For ATW, initially a window, of dimensions 25×25 pixels, is placed around each UAV. The coordinate produced from intensity-contour detection is checked if it satisfies any of the window's x-boundary and y-boundary. If it satisfies UAV's x and y boundary conditions, up to three coordinates are recorded and a moving average filter is applied; thereby, the tracking starts. If the coordinate does not fall within any of the windows, the dimension of the tracking window is extended incrementally by 12 pixels along x or y depending on which boundary condition is unsatisfied. The window's maximum possible dimension can be 50×50 pixels. This adaptive nature allows the UAVs to fly up to a maximum speed of 4 m/s and still be tracked.

We calculate the error between localized coordinates of a UAV and the ground truth coordinates, which is the center of the UAV. The quantified results are presented in Table II. The error is less than 5 cm for x and y.

2) Autonomous Navigation of UAVs: Autonomously controlling multiple UAVs reliably is crucial for running applications on the testbed. We implemented a lightweight navigation routine for controlling a quadcopter. The submodule UAV flight navigator can either run on the *central* or *client* module since it belongs to the *multihomed* module. The submodule can control the quadcopter's flight parameters, maneuver it from one position to the next, and carry out any application-specific duties. A proportional-derivative controller controls the pitch and roll values of the UAV in order to guide it to its goal position.

The navigation script is tested in real indoor environment, with three drones covering four targets simultaneously. The proposed algorithm assigns targets to each drone to cover within their field-of-view (FoV). Drone-1 covers two targets since the two targets are adjacent to each other and fit within the FoV of drone-1. When the third target moves into the coverage area, drone-2 flies to cover it since it cannot be covered by drone-1. The last target elsewhere on the testbed is covered by drone-3 since drone-1 and drone-2 cannot cover the target along with their respective targets. The top-view perspective, each drone's perspective, and in-action view are shown in Fig. 8.

3) Simulation Environment: To facilitate the large-scale deployment and controlling targets mobility, we develop our own CPS simulation environment of the testbed called *testbed simulation with localization and autonomy* (*TeSLA*), with image-based models for targets and drones. *TeSLA* simulates *DbeG* in 2D and incorporates our implemented localization and autonomous navigation routines used in *DbeG*. However, *TeSLA*

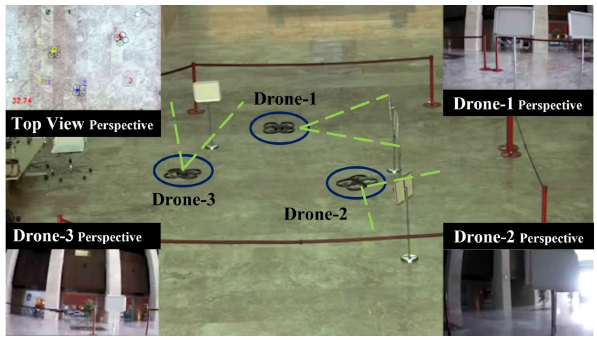

Fig. 8. Three drones (circled in blue) covering their respective targets. Green-dashed lines represent drone-FoV. Within each drones perspective, the white stands resemble the targets.

goes beyond just the integration of the aforementioned routines. Using *TeSLA*, we are able to observe the performance of our system when integrated with an external CPS application, such as coverage algorithms, on our testbed. The integration of localization, navigation, and an external CPS application in action provides us insights into the nature of our system and serves as an effective safety precursor. *TeSLA* is designed using OpenCV bindings for Python.

4) External Processing Unit: The addition of EPU enables the groundwork toward distributed control and/or processing. In order to achieve real-time processing with minimal latency, we choose on-board processing instead of cloud computing or servers. Also, EPUs provide extra processing power and added flexibility. We can install our own drivers, operating systems, integrate sensors, and overcome the typical closed nature characteristic of off-the-shelf quadcopters.

We invested efforts in setting up Arduino-based and Raspberry-Pi-based EPUs on AR.Drone 2.0. However, we use Intel Edison, an ultrasmall computing device powered by an Atom SoC, as our final EPU. It is a lighter module than the others and is computationally much stronger. We couple it with an IMU sensor MPU-6050, which is unsusceptible to magnetic field strength since AR.Drone's sensory yaw readings were sensitive to magnetic field strengths despite calibration.

To operate the SoC, a breakout board is required. The board used is a base block from SparkFun. The IMU sensor is attached to Edison with the help of an Arduino block. The Arduino block is programmed using a 5 V FTDI programming header. Communication between the block and Edison occurs over a selectable UART port. This system is powered using a battery block. We place the EPU on the top and close to the center of gravity of AR.Drone. This avoids affecting the balance and stability of the vehicle. The total weight of the quadcopter with 1500 mAh battery, indoor hull, and Intel Edison is 501 g. The setup of all EPUs is shown in Fig. 9.

VIII. CASE STUDY

The objective of the case study is to provide a practical view of the operation of our algorithms in an indoor testbed. We start with a description of the system overview followed by an experimental setup. Finally, we present our experimental results and relay some lessons learned upon moving from the theoretical view of the problem to the practical domain of implementing target coverage.

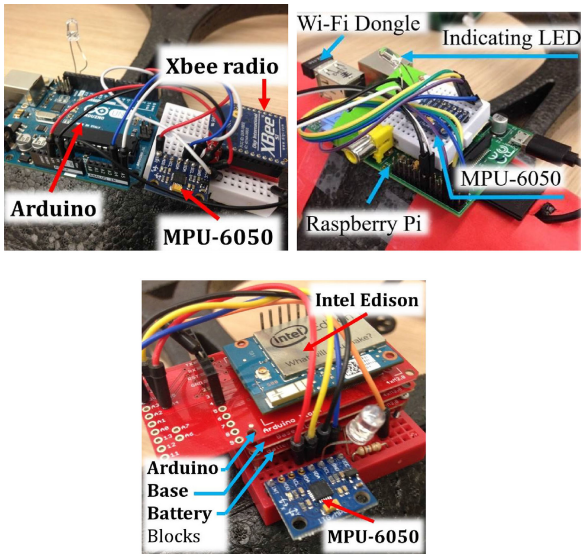

Fig. 9. Arduino microcontroller and two EPUs, Intel Edison and Raspberry Pi, mounted on AR.Drone.

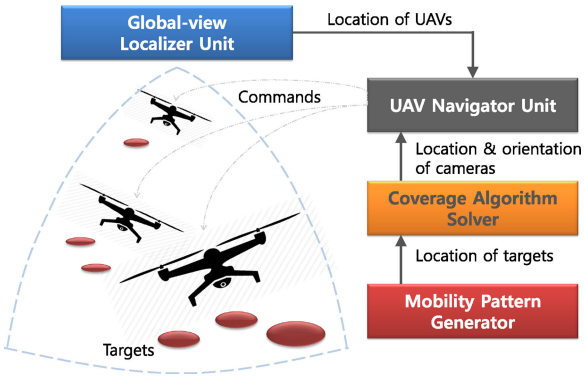

Fig. 10. Overview of system operation run on the testbed.

A. System Overview on the Testbed

Fig. 10 shows an overview of the operational system implemented on the testbed using UAVs as mobile sensors. It has four major components. The *global-view localizer unit* is responsible for overviewing the testbed and localizing the UAVs. The *mobility pattern generator* characterizes the targets mobility model based on one of the three models (RWP, MG, or RPG) and provides the information to the *coverage algorithm solver*. The solver then uses one of the three coverage algorithms (PFA, PIFA, or LIFA) to determine camera pose(s) [location(s) and orientation(s)]. The *UAV navigator unit* takes UAV locations from the localizer and camera pose information from the solver to navigate the UAVs autonomously to the destinations. The architecture is centralized in nature, i.e., all processes are run by a central entity.

B. Experimental Setup

The deployment area is shown in Fig. 11. We use an Axis 213 PTZ Network Camera connected through an Ethernet cable to the central entity, a Lenovo ThinkPad Y50. The camera overviews the testbed enabling vision-based indoor localization of UAVs using image processing techniques. Commercial off-the-shelf quadcopters such as the AR.Drone 2.0 are used as UAV

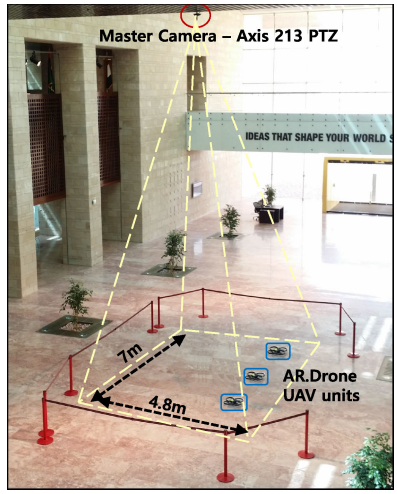

Fig. 11. Layout of testbed's deployment area showing master camera (red circle) and three UAVs (blue boxes).

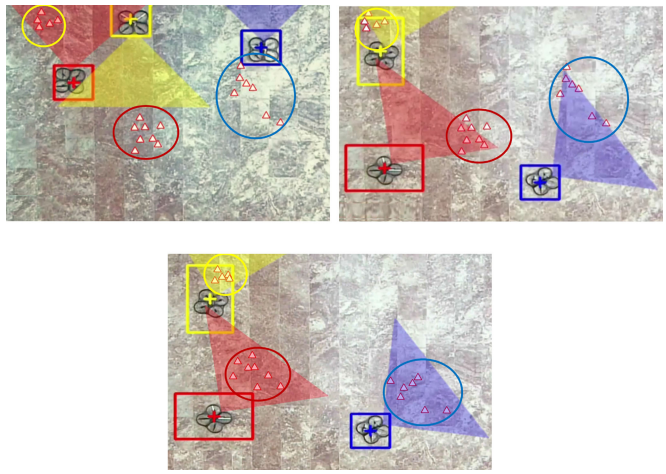

Fig. 12. Top-view perspective of three drones with their respective field-of-view (cones) and targets (circles) color-coded.

units. We equip each AR.Drone with an Intel Edison for better navigation purposes. The UAV is deployed in a 7 m × 4.8 m area, and communication with AR.Drone takes place over WiFi on an *ad hoc* network.

All required computations take place on the central entity. The *coverage solver* and *pattern generator* threads are run on a MATLAB process. The *localizer unit* is implemented using Python and OpenCV. The *UAV navigator* thread uses a Node.JS framework for command and control. A total of 20 targets are virtually created by the MATLAB script. Each UAV is tasked with going to its goal position sequentially. Once all the UAVs have reached their goals, the algorithm computes new goals for the UAVs.

C. Experimental Results

The range of the FoV for each UAV and group of targets is color-coded as per the tracking window placed on the UAV. Three instances of the experiment are shown in Fig. 12. PFA generates locations and orientations for the UAVs in order to cover their respective clusters. In the top-left image of Fig. 12,

all three groups are uncovered by their respective UAVs. Based on the output from PFA, the UAVs fly to cover their respective clusters, as shown in the top-right middle image of Fig. 12. The destination coordinates and orientation allow a partial coverage of the targets. Since PFA predicts the targets' new locations and takes that into consideration in order to generate UAV locations and orientations, the targets fall within the FoV of their respective UAVs in the next step (shown in the bottom image of Fig. 12). The video in [42] not only shows the PFA algorithm in action on the testbed, but it also shows the seamless integration between *TeSLA* and DbeG testbed, depicted by a similar performance of target coverage in both environments, which validates the results achieved by the simulator in actual practical environment.

IX. Conclusion and Future Work

In this paper, we studied the problem of positioning and orienting mobile cameras to cover mobile targets. We first implemented the following three different mobility patterns representing various mobility behavior of targets: RWP mobility, MG mobility, and RPG mobility pattern. Then, we proposed the following three different algorithms to follow the mobile targets: PFA, PIFA, and LIFA. We also implemented *DbeG*: an inexpensive, easy-to-deploy, general-purpose CPS testbed that integrates various components in order to implement indoor UAV-based applications. Finally, we demonstrated one of the coverage algorithms as a case study on our testbed. Thus, we learned that we can easily integrate applications into our testbed practically and in a cost-effective manner.

For future work, we plan to extrapolate localization to provide 3-D coordinates by replacing the network camera with a longer range depth sensing camera such as ZED stereo camera with a maximum depth range of 20 m with 1080p resolution. In terms of navigation, we intend to implement extended Kalman filter for better flight of the UAVs. We also intend to include a collision avoidance routine and better path-planning to the navigation routine.

Acknowledgment

The findings achieved herein are solely the responsibility of the authors.

References

[1] J. Markoff, "Helping the elderly stay at home with an army of caregiver drones." [Online]. Available: http://www.pressreader.com/thailand/bangkok-post/20151213/2830340535195 88

[2] A. Essameldin and K. A. Harras, "The hive: On-edge middleware solution for resource sharing in the Internet of Things," in *Proc. 3rd Workshop Experiences Design Implementation Smart Objects*, Oct. 2017, pp. 13–18.

[3] M. Ibrahim, M. Gruteser, K. A. Harras, and M. Youssef, "Over-the-air TV detection using mobile devices," in *Proc. 26th Int. Conf. Comput. Commun. Netw.*, 2017, pp. 1–9.

[4] A. Neishaboori, A. Saeed, K. Harras, and A. Mohamed, "Low complexity target coverage heuristics using mobile cameras," in *Proc. IEEE 11th Int. Conf. Mobile Ad Hoc Sens. Syst.*, Oct. 2014, pp. 217–221.

[5] D. S. Hochbaum and W. Maass, "Approximation schemes for covering and packing problems in image processing and VLSI," *J. ACM*, vol. 32, no. 1, pp. 130–136, Jan. 1985, doi: 10.1145/2455.214106.

[6] D. G. Costa and L. A. Guedes, "The coverage problem in video-based wireless sensor networks: A survey," *Sensors*, vol. 10, pp. 8215–8247, Sep. 2010.

[7] A. Gusrialdi, T. Hatanaka, and M. Fujita, "Coverage control for mobile networks with limited-range anisotropic sensors," in *Proc. 2008 47th IEEE Conf. Decis. Control*, Dec. 2008, pp. 4263–4268.

[8] E. Yildiz, K. Akkaya, E. Sisikoglu, and M. Sir, "Optimal camera placement for providing angular coverage in wireless video sensor networks," *IEEE Trans. Comput.*, vol. 63, no. 7, pp. 1812–1825, Jul. 2014.

[9] Y. Wang and G. Cao, "On full-view coverage in camera sensor networks," in *Proc. 2011 IEEE INFOCOM*, 2011, pp. 1781–1789.

[10] H. Khandani, H. Moradi, and J. Panah, "A real-time coverage and tracking algorithm for UAVs based on potential field," in *Proc. 2nd RSI/ISM Int. Conf. Robot. Mechatron.*, Oct. 2014, pp. 700–705.

[11] S. Kumar, T. H. Lai, and A. Arora, "Barrier coverage with wireless sensors," in *Proc. 11th Annu. Int. Conf. Mobile Comput. Netw.*, 2005, pp. 284–298.

[12] J. Urrutia, "Art gallery and illumination problems," in *Handbook of Computational Geometry*. Amsterdam, The Netherlands: Elsevier, 2000, pp. 973–1027.

[13] S. Eidenbenz, C. Stamm, and P. Widmayer, "Inapproximability results for guarding polygons and terrains," *Algorithmica*, vol. 31, no. 1, pp. 79–113, 2001, doi: 10.1007/s00453-001-0040-8.

[14] O. Cheong, A. Efrat, and S. Har-Peled, "Finding a guard that sees most and a shop that sells most," *Discrete Comput. Geom.*, vol. 37, no. 4, pp. 545–563, Jun. 2007.

[15] A. Efrat and S. Har-Peled, "Guarding galleries and terrains," *Inf. Process. Lett.*, vol. 100, no. 6, pp. 238–245, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020019006001359

[16] M. Younis and K. Akkaya, "Strategies and techniques for node placement in wireless sensor networks: A survey," *Ad Hoc Netw.*, vol. 6, no. 4, pp. 621–655, Jun. 2008.

[17] V. P. Munishwar and N. B. Abu-Ghazaleh, "Coverage algorithms for visual sensor networks," *ACM Trans. Sens. Netw.*, vol. 9, no. 4, pp. 1–34, Jul. 2013.

[18] J. Ai and A. A. Abouzeid, "Coverage by directional sensors in randomly deployed wireless sensor networks," *J. Combinatorial Optim.*, vol. 11, no. 1, pp. 21–41, 2006, doi: 10.1007/s10878-006-5975-x.

[19] Y. Cai, W. Lou, M. Li, and X.-Y. Li, "Target-oriented scheduling in directional sensor networks," in *Proc. IEEE INFOCOM 2007, 26th IEEE Int. Conf. Comput. Commun.*, May 2007, pp. 1550–1558.

[20] C. Soto, B. Song, and A. Roy-Chowdhury, "Distributed multi-target tracking in a self-configuring camera network," in *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, Jun. 2009, pp. 1486–1493.

[21] G. Fusco and H. Gupta, "Placement and orientation of rotating directional sensors," in *Proc. 7th Annu. IEEE Commun. Soc. Conf. Sens. Mesh Ad Hoc Commun. Netw.*, Jun. 2010, pp. 1–9.

[22] A. Saeed, A. Abdelkader, M. Khan, A. Neishaboori, K. A. Harras, and A. Mohamed, "Argus: Realistic target coverage by drones," in *Proc. 16th ACM/IEEE Int. Conf. Inf. Process. Sens. Netw.*, 2017, pp. 155–166.

[23] T. Furukawa, F. Bourgault, B. Lavis, and H. F. Durrant-Whyte, "Recursive Bayesian search-and-tracking using coordinated UAVs for lost targets," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2006, pp. 2521–2526.

[24] L. Sun, S. Baek, and D. Pack, *Distributed Probabilistic Search and Tracking of Agile Mobile Ground Targets Using a Network of Unmanned Aerial Vehicles*. Cham, Switzerland: Springer, 2014, pp. 301–319.

[25] B. Bethke, "Persistent vision-based search and track using multiple UAVs," Ph.D. dissertation, Dept. Aeronaut. Astronaut., Massachusetts Inst. Technol., Cambridge, MA, USA, 2007.

[26] P. Skoglar, U. Orguner, D. Trnqvist, and F. Gustafsson, "Road target search and tracking with gimballed vision sensor on an unmanned aerial vehicle," *Remote Sens.*, vol. 4, no. 7, pp. 2076–2111, 2012. [Online]. Available: http://www.mdpi.com/2072-4292/4/7/2076

[27] W. Meng, Z. He, R. Su, P. K. Yadav, R. Teo, and L. Xie, "Decentralized multi-UAV flight autonomy for moving convoys search and track," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 4, pp. 1480–1487, Jul. 2017.

[28] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-UAV testbed," *IEEE Robot. Autom. Mag.*, vol. 17, no. 3, pp. 56–65, Sep. 2010.

[29] S. Lupashin, A. Schollig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 1642–1648.

[30] M. Muller, S. Lupashin, and R. D'Andrea, "Quadrocopter ball juggling," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2011, pp. 5113–5120.

[31] M. Valenti *et al.*, "The MIT indoor multi-vehicle flight testbed," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 2758–2759.

[32] O. Lawlor, M. Moss, S. Kibler, C. Carson, S. Bond, and S. Bogosyan, "Search-and rescue robots for integrated research and education in cyber-physical systems," in *Proc. 7th IEEE Int. Conf. e-Learn. Ind. Electron.*, Nov. 2013, pp. 92–97.
[33] A. Saeed, A. Neishaboori, A. Mohamed, and K. Harras, "Up and away: A visually-controlled easy-to-deploy wireless UAV cyber-physical testbed," in *Proc. IEEE 10th Int. Conf. Wireless Mobile Comput. Netw. Commun.*, Oct. 2014, pp. 578–584.
[34] I. Van der Spek and M. Voorsluys, "Ar.drone autonomous control and position determination," Master's thesis, Dept. Elect. Eng., Delft Univ. Technol., Delft, The Netherlands, 2012.
[35] J. Jimenez Lugo and A. Zell, "Framework for autonomous onboard navigation with the ar.drone," in *Proc. Int. Conf. Unmanned Aircr. Syst.*, May 2013, pp. 575–583.
[36] J. C. Bezdek, *Pattern Recognition With Fuzzy Objective Function Algorithms*. Norwell, MA, USA: Kluwer, 1981.
[37] S. Ahmed, A. Mohamed, K. Harras, M. Kholief, and S. Mesbah, "Energy efficient path planning techniques for UAV-based systems with space discretization," in *Proc. IEEE Conf. Wireless Commun. Netw. Conf.*, Apr. 2016, pp. 1–6.
[38] A. D. Nguyen, P. Senac, V. Ramiro, and M. Diaz, "Steps—An approach for human mobility modeling," in *Proc. Int. Conf. Res. Netw.*, Valencia, May 2011, pp. 254–265.
[39] R. Groenevelt, E. Altman, and P. Nain, "Relaying in mobile ad hoc networks: The Brownian motion mobility model," *Wireless Netw.*, vol. 12, no. 5, pp. 561–571, Sep. 2006, doi: 10.1007/s11276-006-6535-0.
[40] W. Keal, "Motion determination," U.S. Patent 20 120 323 520 A1, Dec. 20, 2012. [Online]. Available: http://www.google.com/patents/US20120323520
[41] A. Holtermüller and J. Plödereder, "Tracking of persons with camera-fusion technology," Studienarbeit, Univ. Stuttgart, Stuttgart, Germany, 2012.
[42] K. Heurtefeux, M. Khan, S. Alam, A. Mohamed, and K. A. Harras, "Mobile target coverage and tracking on drone-be-gone," *Testbed Demonstration*, 2016. [Online]. Available: https://www.youtube.com/playlist?list=PLWpZgu4jg0HwezfPe2DRBpvzXiX9nE2f J

Mouhyemen Khan received the B.S. degree in electrical and computer engineering from Texas A&M University, College Station, TX, USA. Currently, he is working toward the Ph.D. degree at the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA, with a focus on autonomous systems of aerial robots and leveraging deep learning solutions.

He co-founded the Qatar Robotics Institute for Development, aimed for STEM development and robotics education in the region. From November 2014 to May 2016, he was a Research Assistant with Qatar University and Carnegie Mellon University, working on developing a cyber-physical system for aerial mobile units and deployment of coverage algorithms along with localization and autonomous navigation.

Karel Heurtefeux received the M.S. and Ph.D. degrees in computer networking and telecommunication from the University of Lyon, Lyon, France, in 2006, and 2009, respectively.

He was a Postdoctoral Researcher with the Verimag Laboratory, from 2010 to 2012, involved in the field of wireless sensor networks and self-stabilizing algorithms. In June 2012, he joined the Qatar Mobility Innovation Center, Doha, Qatar, as a Research Scientist to work on wireless body area networks, energy-constrained algorithms, and wireless communication protocols. From June 2015 to December 2015, he was with Qatar University and Carnegie Mellon University, as a Research Scientist working on real-time visual data processing and coverage optimization algorithms for micro-unmanned aerial vehicles. He is currently a Research Engineer with BMW, Munich, Germany, working on vehicle-to-grid communications. He is involved in the writing and implementation of international V2G standards for electric vehicle in the context of inductive charging. He has authored or co-authored more than 150 refereed journal and conference papers, textbook, and book chapters in reputed international journals and conferences. His research interests include wireless networking and edge computing for Internet of Things applications.

Amr Mohamed (S'00–M'06–SM'14) received the M.S. and Ph.D. degrees in electrical and computer engineering from the University of British Columbia, Vancouver, BC, Canada, in 2001 and 2006, respectively.

He was an Advisory IT Specialist with the IBM Innovation Centre, Vancouver, from 1998 to 2007, taking a leadership role in systems development for vertical industries. He is currently an Associate Professor with the College of Engineering, Qatar University, Doha, Qatar, and the Director for the Cisco Regional Academy. He has more than 25 years of experience in wireless networking research and industrial systems development.

Dr. Mohamed is serving as a Technical Editor for the *Journal of Internet Technology* and the *International Journal of Sensor Networks*. He has served as a Technical Program Committee (TPC) Co-Chair for workshops of IEEE WCNC16. He has served as a Co-Chair for technical symposia of international conferences, including Globecom16, Crowncom15, AICCSA14, IEEE WLN11, and IEEE ICT10. He has served on the Organization Committee of many other international conferences as a TPC member, including the IEEE ICC, GLOBECOM, WCNC, LCN, and PIMRC, and a Technical Reviewer for many international IEEE, ACM, Elsevier, Springer, and Wiley journals. He was the recipient of three awards from IBM Canada for his achievements and leadership and three Best Paper Awards, the latest from the IEEE/IFIP International Conference on New Technologies, Mobility, and Security 2015 in Paris, France.

Khaled A. Harras (S'05–M'09–SM'16) received the M.S. and Ph.D. degrees in computer science from the University of California at Santa Barbara, Santa Barbara, CA, USA.

He is currently an Associate Professor with Carnegie Melon University Qatar (CMUQ), Doha, Qatar, and also the Director of the computer science program. He is the founder and the Director with the Networking Systems Lab, CMUQ. For the past 14 years, he has been working on the areas of challenged and opportunistic networks, video sensor networks, unmanned air vehicles, ubiquitous and pervasive systems, mobile edge computation/networked architectures, and cyber security. He also has expertise in the domains of wireless and mobile networks measurement, building real systems and testbeds, and designing and implementing architectures and frameworks based on real data. He has more than 100 refereed publications in numerous international prestigious journals, conferences, and workshops, and 4 U.S. patents. He has supervised more than 30 different personnel including undergraduate and graduate students, postdoctoral researchers, and research engineers.

Dr. Harras is a Senior Member of the ACM. Along with his research group over the past few years, he was the recipient of the Best Computing Research Award in Qatar twice and two Best Paper Awards. To date, he has been involved in or managing research grants that amount to more than 3 million USD.

Mohammad Mehedi Hassan (M'12) received the Ph.D. degree in computer engineering from Kyung Hee University, Seoul, South Korea, in 2011.

He is currently an Associate Professor with the Information Systems Department, College of Computer and Information Sciences (CCIS), King Saud University (KSU), Riyadh, Saudi Arabia. He has authored or co-authored more than 100 research papers in journals and conferences of international repute. His research interests include cloud federation, multimedia cloud, sensor-cloud, Internet of Things, big data, mobile cloud, sensor network, publish/subscribe system, and recommender system.

Dr. Hassan was the Guest Editor of several international ISI-indexed journals. He was the recipient of the Best Paper Award from the CloudComp Conference in China in 2014 and the Excellence in Research Award from CCIS, KSU in 2015 and 2016.