H-DrunkWalk: Collaborative and Adaptive Navigation for Heterogeneous MAV Swarm

XINLEI CHEN and CARLOS RUIZ, Carnegie Mellon University

SIHAN ZENG, Tsinghua University

LIYAO GAO, Purdue University

AVEEK PUROHIT, Carnegie Mellon University

STEFANO CARPIN, UC Merced

PEI ZHANG, Carnegie Mellon University

Large-scale micro-aerial vehicle (MAV) swarms provide promising solutions for situational awareness in applications such as environmental monitoring, urban surveillance, search and rescue, and so on. However, these scenarios do not provide localization infrastructure and limit cost and size of on-board capabilities of individual nodes, which makes it challenging for nodes to autonomously navigate to suitable preassigned locations. In this article, we present *H-DrunkWalk*, a collaborative and adaptive technique for heterogeneous MAV swarm navigation in environments not formerly preconditioned for operation. Working with heterogeneous MAV swarm, the *H-DrunkWalk* achieves high accuracy through collaboration but still maintains a low cost of the entire swarm. The heterogeneous MAV swarm consists of two types of nodes: (1) basic MAVs with limited sensing, communication, computing capabilities and (2) advanced MAVs with premium sensing, communication, computing capabilities. The key focus behind this networked MAV swarm research is to (1) rely on collaboration to overcome limitations of individual nodes and efficiently achieve system-wide sensing objectives and (2) fully take advantage of advanced MAVs to help basic MAVs improve their performance. The evaluations based on real MAV testbed experiments and large-scale physical-feature-based simulations show that compared to the traditional non-collaborative and non-adaptive method (dead reckoning with map bias), our system achieves up to 6× reductions in location estimation errors, and as much as 3× improvements in navigation success rate under the given time and accuracy constraints. In addition, by comprehensively considering the environment, heterogeneous structure, and quality of location estimation, our *H-DrunkWalk* brings 2× performance improvement (on average) as that of a hardware upgrade.

CCS Concepts: • **Computer systems organization** → **Sensor networks**; **Robotic autonomy**; • **Computing methodologies** → **Multi-agent systems**; *Mobile agents*; *Intelligent agents*; **Cooperation and coordination**;

Additional Key Words and Phrases: Mobile sensor networks, aerial networks, swarm intelligence, MAV navigation, distributed artificial intelligence

Authors' addresses: X. Chen and A. Purohit, Carnegie Mellon University, NASA Research Park Carnegie Mellon University Silicon Valley P.O. Box 98 Moffett Field, CA 94035; emails: xinlei.chen@sv.cmu.edu, aveek.purohit@west.cmu.edu; S. Zeng, Tsinghua University, 30 Shuangqing Rd, Haidian District, Beijing, China; email: cengsh14@mails.tsinghua.edu.cn; L. Gao, Purdue University, 150 N University St, West Lafayette, IN 47907; email: gao463@purdue.edu; S. Carpin, UC Merced, 5200 N. Lake Road, Merced, CA 95343; email: scarpin@ucmerced.edu; P. Zhang, NASA Research Park Carnegie Mellon University Silicon Valley P.O. Box 98 Moffett Field, CA 94035; email: peizhang@cmu.edu.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2020 Association for Computing Machinery.

1550-4859/2020/04-ART20 \$15.00

<https://doi.org/10.1145/3382094>

ACM Reference format:

Xinlei Chen, Carlos Ruiz, Sihan Zeng, Liyao Gao, Aveek Purohit, Stefano Carpin, and Pei Zhang. 2020. H-DrunkWalk: Collaborative and Adaptive Navigation for Heterogeneous MAV Swarm. *ACM Trans. Sen. Netw.* 16, 2, Article 20 (April 2020), 27 pages.
<https://doi.org/10.1145/3382094>

1 INTRODUCTION

Situational awareness is essential in many hostile, dangerous, or inaccessible environments, such as environmental monitoring, surveillance, urban search and rescue, and so on [1–3]. Since it is impossible to manually deploy sensors in these dangerous scenarios, a promising solution is to autonomously navigate MAV swarms with sensors to a set of goal locations, in accordance with the needs of domain experts.

MAV swarms are composed of miniature aerial sensor nodes with sensing, computing, and communication capabilities [4, 5]. Initial MAV research focused on outdoor or highly instrumented environments that rely on external infrastructure to control individual devices [6, 7]. The reliance on sensing infrastructure implies requirements for a large deployment of support sensors covering all the locations that a MAV may visit. Thus, these approaches are only applicable in pre-surveyed locations. Recently, new navigation methods for MAV swarms that work in GPS denied areas achieve high navigation accuracy and do not rely on any localization infrastructure [8, 9]. These simultaneous localization and mapping (SLAM) methods require advanced sensors like depth camera and LIDAR, which are expensive, large, and heavy [10–14].

However, to be more cost effective and provide increased robustness through redundancy, the aforementioned scenarios require large-scale MAV swarms (hundreds or more nodes), which consist of low-cost, low-complexity mobile sensing nodes, as opposed to using a limited number of sophisticated robots. In addition, small lightweight mobile sensor nodes provide greater safety as the effects of their collisions with the objects or persons in the indoor environment are inconsequential. In such swarms, each MAV is only equipped with fundamental navigational sensors and costs around \$20, which are palm-size with the weight of 29g [4, 15]. We have proved that the performance of the entire swarm can be improved a lot through collaboration [16]. For such systems, upgrading on-board hardware (especially sensing capabilities) helps improve the performance but requires high cost due to the large scale. The heterogeneous MAV swarm is a promising solution, in which only some MAVs are equipped with better on-board sensing, computing and communication capabilities. Through collaboration, the system is able to achieve high accuracy through collaboration but still maintain a low cost of the entire swarm.

This article presents *H-DrunkWalk*, a technique for heterogeneous MAV swarm navigation in a collaborative and adaptive way in environments not formerly preconditioned for operation. The heterogeneous MAV swarm consists of two types of nodes: (1) basic MAVs with limited sensing, communication, computing capabilities and (2) advanced MAVs with premium sensing, communication, computing capabilities. The key focus behind this networked MAV swarm research is to (1) rely on collaboration to overcome limitations of individual nodes and efficiently achieve system-wide sensing objectives and (2) fully take advantage of advanced MAVs to help basic MAVs improve their performance. Note that *H-DrunkWalk* is an upgraded version of *Drunkwalk* and is able to work with both homogeneous and heterogeneous MAV swarms. With the homogeneous MAV swarm, the *H-DrunkWalk* acts the same as the *DrunkWalk* algorithm [16].

The *H-DrunkWalk* self-establishes a temporary infrastructure of a few landed MAV's acting as radio beacons. Using radio signatures from beacon nodes, the algorithm detects intersections in trajectories of exploring mobile MAV nodes. The algorithm combines noisy dead-reckoning measurements from multiple MAVs at the detected intersections to improve the accuracy of the MAVs'

location estimates. Most importantly, the algorithm adaptively plans trajectories of MAV nodes by considering multiple factors: the accuracy of sensing data, whether the nearby space is fully investigated, and the quality of location estimation. In this article, we consider the accuracy of the data from the optical flow and the magnetometer. The optical flow data is accumulated to get displacement of MAVs, while the magnetometer data is adopted to derive absolute orientation of MAVs. The localization accuracy of MAVs is decided by both displacement and absolute orientation. As a result, *H-DrunkWalk* directs MAVs' movement to improve location estimates in the exploration mode or directs MAVs to the goal location in the navigation mode. The adaptive strategy enables *H-DrunkWalk* to improve the location estimation accuracy and success rate of navigation under given time and accuracy constraints.

The main contributions of this article are:

- A collaborative location estimation algorithm that improves nodes' estimation accuracy at their trajectory intersection, which achieves up to 6× reductions in location estimation errors.
- An adaptive planning algorithm for navigation that comprehensively considers the environment, heterogeneous structure, and quality of location estimation. Under the given time and accuracy constraints, the system achieves up to 3× improvements in navigation success rate.
- A mode bias calculation (MBC) algorithm that utilizes advanced MAVs to help basic MAVs improve their performance by intelligently switching the mode of both types of MAVs under different conditions. The MBC brings 2× performance improvement (on average) as that of hardware upgrade.
- Evaluations based on real MAV testbed experiments and large-scale physical-feature-based simulations using radio signatures collected from the physical world and empirically determined sensor noise models.

Note that this article is a more extensive elaboration of our DrunkWalk paper in ACM SenSys 2015 [16]. In this article, we add a new MBC algorithm that intelligently switches the modes of all MAVs according to different conditions. Under the guidance of MBC, advanced MAVs are able to help basic MAVs improve their localization accuracy and navigation success rate, thus improving the performance of the entire MAV swarm.

The rest of this article is organized as follows. Section 2 gives a high-level overview of the architecture and operation of the system. Section 3 gives a detailed technical description of the various algorithms presented in the article. Section 4 evaluates and analyzes the system through extensive simulations and validates assumptions through MAV testbed experiments. In Section 6, we describe related work and discuss the state-of-the-art infrastructure-free navigation techniques in context of MAV swarm deployment. Finally, we draw conclusions and summarize our contributions in Section 7.

2 SYSTEM DESIGN

This section introduces the architecture and operational process of the proposed system. With very limited on-board capabilities (sensing, computing, and communication), we address the problem of how the heterogeneous MAV swarm collaboratively operates to autonomously navigate to desired locations in operating environments with no localization infrastructure support.

We assume there are two types of MAV nodes: (1) **basic MAV**, with limited sensing, communication, computing capabilities, and (2) **advanced MAV**, with premium sensing, communication, computing capabilities. In addition, the advanced MAV is also equipped with larger battery to

support the potential higher energy consumption from sensing, computing and communication. It is easy to extend the system to a heterogeneous MAV swarm with multiple types.

2.1 Operation and Architecture

The system first introduces a swarm of MAVs into the operating environments. We assume that a coarse map of the operating environments is available and can be utilized by domain experts to pre-determine suitable placement of MAVs. This is a valid assumption in most scenarios, as emergency response personnel have access to the rough floor-plans of buildings through city registries, and thanks to increased availability of indoor maps tailored to location-based services (e.g., indoor Google maps). It is noticed that the coarse map does not include the collapsed structures/rooms, inaccessible parts of the building due to disasters like earthquakes. The system adopts the Sensor-Fly V4 MAV node, which can detect obstacles with only the accelerometer in a real time way [15]. The obstacle information will be shared between MAVs to prevent collisions.

The proposed system has three major operational phases: pre-deployment, collaborative location estimation, and adaptive path planning (the latter two proceed in conjunction):

- **Pre-deployment:** To help improve the location estimation accuracy through collaborations between MAVs, the system autonomously establishes a transient infrastructure of stationary MAV nodes acting as wireless beacons. These nodes land upon being introduced into the area and remain stationary during the process. The major tasks of the stationary nodes are to enable mobile MAV nodes to obtain radio signatures or fingerprints of locations traversed on their paths. These nodes use a simple dispersion algorithm that lets them spread out in a new environment *without any estimation of their location* [17, 18].
- **Collaborative Location Estimation:** After pre-deployment, to achieve high accurate location estimation with limited on-board (sensing, computing and communication) capabilities, the system collaboratively estimates locations of MAV nodes. To realize this, the system first uses navigational sensors such as an optical flow velocity sensor and magnetometer (in our test MAV platform) to get rough estimates of the motion path of mobile nodes. Second, the system uses radio fingerprints, collected by mobile nodes from the self-established wireless beacons, to determine *snapshot points*, i.e., location points that were previously visited by other nodes or by itself. Finally, the system uses the snapshot points to combine location estimates from multiple nodes and collaboratively to improve location estimations of the entire swarm. It is noticed that in the entire operational process of mobile nodes, combining and improving location estimates only happen when they pass by snapshot points. During other times, each individual mobile node adopts on-board navigational sensors to get rough location estimates.
- **Adaptive Path Planning:** Having estimated locations, to achieve high quality path planning for the entire swarm, the system adaptively switches two modes for each MAV node that: (1) leads to subsequent goal positions (*navigation mode*) and (2) improves location estimation accuracy (*exploration mode*). This is because the quality of the planned path depends greatly on the accuracy of the initial location estimate of nodes. A bad location estimate will render any attempt to plan a deterministic path useless—*when the nodes do not know where they are, they cannot plan a correct path to their destination*. Our system thus considers the quality of location estimation, based on which the system adaptively plans node paths. Moreover, to improve the location estimation of the entire swarm, the system tends to switch advanced MAVs to *exploration mode* when nearby areas have not passed by any advanced MAVs. In *exploration mode*, the path planner commands nodes movement such that they increase the number of snapshot points and potentially improve location estimates,

Fig. 1. The figure shows the architecture of our navigation system. The mobile MAV nodes send dead-reckoning sensor data and radio signatures to the base. The base runs the *H-DrunkWalk* estimation and planning algorithm and issues movement commands to individual MAV nodes. In addition, according to the sensing capabilities, *H-DrunkWalk* assigns different mode switching bias to path planning.

since the quality of location estimate is likely to be low. In *navigation mode*, since the location estimates are likely to be more accurate, the planner directs them to their designated goal locations with map information.

Figure 1 shows the architecture of the system, which consists of **stationary MAV nodes** in green, **mobile MAV nodes** in orange and **the base** in blue. In the pre-deployment phase, through dispersion algorithm, the system deploys stationary MAV nodes that act as wireless beacons. Though both advanced MAVs and basic MAVs can be used as stationary MAV nodes, the system only adopts basic MAVs to save system resources. Mobile MAV nodes, which consist of both advanced MAVs and basic MAVs, explore and obtain dead-reckoning measurements from their on-board navigational sensors and radio RF-signatures from the stationary beacons. The mobile nodes relay this to the base. The base stores a database of known radio signatures (**signature database**) that is used to determine snapshot point in node paths and applies corrections to their dead-reckoning estimates through collaborations (**collaborative location estimate**). Then, considering the accuracy of sensing data, whether the nearby space is fully investigated, and the quality of location estimation, the **mode bias calculation (MBC)** module calculates the *adaptive mode switch (AMS)* value. The AMS value is used to switch mobile MAV nodes between two modes (**adaptive path planning**), which commands the subsequent movements of mobile MAV nodes (**control logic**).

2.2 Collaborative Location Estimate

The core idea behind our collaborative location estimate approach is to use relatively large number of sensors on mobile MAV nodes in the swarm to collaboratively reduce the error. This is achieved by detecting when nodes move over the same space in the environment and combining their individual location estimations at these points. Errors in dead reckoning measurements are mainly

Fig. 2. The figure shows the process of determining snapshot points. (a) Mobile MAV node 1 moves and obtains a radio signature from stationary MAV. This is entered into the Base Signature DB as new signature. (b) When mobile MAV node 2 visits the same location, its collected radio signature matches existing signature and a correction can be performed at the Base.

due to noise in inertial sensors that are independent across nodes and time [19]. Thus, combining estimates from multiple nodes and propagating corrections to them improves their location estimations. Figure 2 illustrates the on-line process of determining snapshot points from radio measurements.

2.2.1 Determining Snapshot Points. The collaborative location estimate requires a mobile MAV node to be able to determine when it visits a location previously visited by itself or by other nodes - a *snapshot*. The snapshot point provides the opportunity to combine estimations from multiple independent mobile nodes and improve location estimations.

The system determines a snapshot point using radio fingerprints collected by mobile nodes from the self-established beacon nodes. The radio fingerprints are collected in an *online* fashion, i.e., the nodes discover fingerprints as they explore the space. These fingerprints are sent to the Base and matched with a database of previously discovered signatures. If the signature matches an existing signature in the database (decided by a cosine distance and a threshold), then the point is classified as a snapshot point and a correction can be applied to the current location estimation. If the signature does not match any existing signatures, then it is added to the database as a new entry.

2.2.2 Combining Estimates at Snapshot Points. The snapshot points provide the opportunity to combine and improve location estimates, which must be performed carefully. The naive approach would be to take the average of all location estimates for a particular snapshot point. However, this approach does not consider the nature of the underlying distribution of noise in location estimations that often does not follow a normal distribution especially in indoor environments.

Combining estimates is a chicken and egg problem that requires a snapshot point to estimate and update its own location from visiting mobile nodes, and subsequently, use the updated location to correct the estimates of the visiting mobile nodes.

To achieve this, we employ a particle-filter-based approach. A particle filter [20] is a Bayesian estimation method to estimate system state based on multiple noisy sensor measurements. We use a particle filter to track the position and orientation of each mobile node. Similarly, we use a particle filter to track the position of each snapshot point as it is discovered and visited by the MAV nodes. Every visit to a snapshot point by a mobile node results in the mobile node *correcting* the estimation of the particles of the snapshot point, which in turn *corrects* the estimations of the particles of the mobile node. The various estimation algorithms are described in detail in Section 3.

2.3 Adaptive Path Planning Based on AMS Value

Similar to location estimate, planning paths is the second chicken and egg problem encountered in navigation. Better location estimates are needed by nodes to navigate to predetermined regions quickly. However, at the same time, achieving better location estimations may require nodes to take detours (to find snapshot points) costing time and energy. The planning component of our system seeks to make a suitable trade-off between these aspects of navigation. It is noticed that to improve the location estimation of the entire swarm, advanced MAVs should take more responsibility to search the unknown space for the team. This is because advanced MAVs are equipped with higher accuracy navigational sensors and more powerful CPUs, which potentially leads to better location estimates.

Besides, to reach the goal regions, we use an indoor layout with location of walls and doors of the environment. Such rough maps are generally available or easy to obtain in most application scenarios. However, the layout does not include the information of the position of obstacles, which is difficult to derive beforehand. To drive obstacle information with limited sensing capability in a real time way, the system adopts the SensorFly V4 MAV node and its virtual obstacle sensing method, which only relies on the accelerometer data [15]. The obstacle information will be shared between mobile MAVs to prevent collisions.

The map information enables us to bias the direction of node movement toward predetermined goal regions, if the current location of the node in the map can be reasonably determined. However, due to noisy sensors, the location of individual nodes cannot always be estimated correctly, which makes it difficult to consistently plan correct paths. The system attempts to solve this by operating in two modes:

- **Exploration:** In this mode, the mobile MAV node attempts to seek snapshot points that can potentially improve the location estimates of the mobile MAV node. This is executed (1) when the location estimate quality of either a basic MAV or an advanced MAV is low or (2) when an advanced MAV find that nearby areas have not fully searched by the swarm.
- **Navigation:** In this mode, given the estimated location from the collaborative location estimate algorithm, the mobile MAV node attempts to follow the direction of the bias from the graph. This is executed when location estimate quality of either a basic MAV or an advanced MAV is high.

It is easy to see that the performance of the navigation step depends on the outcome of exploration step. However, the exploration step requires extra use of resources that increase the time of navigation. Therefore, the *H-DrunkWalk* tries to balance between the time and navigation accuracy by adaptively switching between these two modes.

2.4 Mode Bias Calculation

To achieve accurate navigation for the entire swarm, the mode bias calculation module (MBC in Figure 1) guides mobile MAV nodes (both advanced MAVs and basic MAVs) switch between two modes in path planning with its output AMS. For homogeneous MAV swarm, as mentioned in

[16], due to the equivalent on-board resources of all the mobile nodes, MBC considers the quality of location estimates of each node as a reference for mode switching. For heterogeneous MAV swarm, mode switching is more complicated, since MAVs are equipped with different on-board capabilities, concerning sensing, computing, communication, and battery. Therefore, besides the quality of location estimation, MBC also considers the accuracy of sensing data and whether the nearby space is fully investigated.

The intuition behind our mode bias calculation approach for heterogeneous MAV swarm is that advanced MAVs take more responsibility to search the undiscovered space for the entire swarm, which can potentially improve the location estimates of basic MAVs. This is because advanced MAVs usually achieve high accurate location estimates with better on-board capabilities (especially sensing). The larger space advanced MAVs cover, the larger chance that basic MAVs improve their location estimate when they pass by snapshot points covered by advanced MAVs. As a result, even with low hardware configurations, basic MAVs can still achieve high performance on location estimate as well as node navigation.

3 ALGORITHM DESCRIPTION

This section provides a detailed description of the two phases of our proposed system: collaborative location estimation and adaptive path planning. First, this section introduces the Particle Filter background in Section 3.1, which is a major technology used in collaborative location estimation. Then, we describe how the location and orientation of each individual MAV in Section 3.2 and the positions of the snapshot points are estimated over time using a set of particle filters in Section 3.3. A separate particle filter is associated to each MAV in the team and each RF-signature being localized in space. Therefore, particles estimating the position and orientation of the MAVs include the components c_x, c_y, c_ϕ , whereas particles estimating the location of the signatures include components s_x, s_y for the position. As described in Section 2, the base exchanges information with the MAVs (commands and measurements) and maintains a database of known signatures (see Figure 1). Due to the limited on-board computational power on the MAV, our current implementation performs all computations in the base. Finally, in Section 3.4, we present how the system adaptively plans the paths of both advanced MAVs and basic MAVs to navigate and deploy quickly.

3.1 Particle Filter Background

A particle filter is a Bayesian estimation method using a finite number of elements (so-called *particles*) to represent a non-parametric probability density. It was introduced in the 1950s [21] and became popular in robotics in the past two decades [20].

As a specific implementation of a more general recursive Bayes filter under the Markov assumption, it requires assumption of availability of two probabilistic models, namely the state evolution model (often called motion model in mobile or robotic applications) and the measurement model. Assuming the unknown state to be estimated at time t is indicated by x_t , the state evolution model provides

$$p(x_t | x_{t-1}, u_t), \quad (1)$$

where u_t is the known command given to the system at time t . The measurement model, instead, is given by

$$p(z_t | x_t), \quad (2)$$

where z_t is the measurement at time t . Due to the Markov assumption, x_t is conditionally independent from x_k with $k < t - 1$ once x_{t-1} is known. Similarly, given x_t , the measurement z_t is conditionally independent from any other variable. Note that one does not need to commit to

specific distributions in Equations (1) and (2); e.g., they do not have to be Gaussian distributions. The generic algorithm to propagate a posterior using a particle filter is given in Algorithm 1, where we mostly follow the notation presented in Reference [20]. The algorithm starts with a set of M particles $\mathcal{X}$ estimating the posterior of x_{t-1} , i.e., the state x at time $t - 1$. Given the latest command u_t and measurement z_t , it produces a new set of M particles providing an updated posterior estimate for x at time t . The i th particle in $\mathcal{X}_t$, $x_t^{[i]}$, represents the i th possible hypothesis about the state at time t . Algorithm 1 shows the generic particle filter algorithm. The first *for* loop creates a new set of M particles sampling the motion model from the set of existing particles, while the second *for* loop implements the so-called *importance resampling*. The set of particles provides a discrete approximation for the posterior.

ALGORITHM 1: Generic particle filter algorithm

Data: $\mathcal{X}_{t-1}, u_t, z_t$
Result: $\mathcal{X}_t$

```

1  $\mathcal{X} \leftarrow \emptyset;$ 
2  $\mathcal{X}_t \leftarrow \emptyset;$ 
3 for  $i \leftarrow 1$  to  $M$  do
4    $x_t^{[i]} \leftarrow \text{sample} \sim p(x_t | x_{t-1}^{[i]}, u_t);$  % state evolution for all particles
5    $w_t^{[i]} \leftarrow p(z_t | x_t^{[i]});$  % calculate weights for all particles
6    $\mathcal{X} \leftarrow \mathcal{X} \cup \{x_t^{[i]}, w_t^{[i]}\};$ 
7 end
8 for  $i \leftarrow 1$  to  $M$  do
9   draw  $j$  with probability  $\propto w_t^{[j]}$ ; % importance resampling
10   $\mathcal{X}_t \leftarrow \mathcal{X}_t \cup \{x_t^{[j]}\};$ 
11 end
```

3.2 Location Estimation and Tracking for Individual MAVs

In this subsection, we show how the generic particle filter framework can be specialized to estimate the location of the MAVs. It is noticed that for location estimation, both advanced MAVs and basic MAVs adopt the same method. To reduce the computational complexity, rather than implementing a centralized particle filter jointly estimating the location of all the MAVs, we associate a particle filter to each MAV. Assuming there are N_M MAVs involved in the navigation task, the system then creates and updates N_M particle filters. Each filter is initialized with $M = 100$ particles uniformly distributed in the area. All computations take place on the base station. Note that the choice of M depends on both location estimation accuracy and computational complexity. A larger M brings higher accuracy but leads to higher computational complexity. The particle filter is initialized with $M = 100$ considering the trade-off between location estimation accuracy and computational complexity. More details of guidance on setting up the number of particles can be found in Reference [22].

3.2.1 Prediction from Motion Models. In the prediction step, a generative law is needed to implement the particle creation in line 4 of Algorithm 1. To this end, we use equations similar to the ones given in Reference [18]. Let the command at time t be $u_t = (v_t, \omega_t)$, where v_t is the translational velocity and ω_t is the rotational velocity. Note that the control system always generates commands in which only one of the two components is different from 0, i.e., the MAV either translates or rotates, but does not make both movements at the same time. Then, a new particle is

generated as

$$\begin{pmatrix} c_x \\ c_y \\ c_\phi \end{pmatrix}_t^{[i]} = \begin{pmatrix} c_x \\ c_y \\ c_\phi \end{pmatrix}_{t-1}^{[i]} + \delta t \begin{pmatrix} v_t \cos(c_{\phi_{t-1}}^{[i]}) \\ v_t \sin(c_{\phi_{t-1}}^{[i]}) \\ \omega_t \end{pmatrix}, \quad (3)$$

where δt is the time interval between two commands. The correctness of the equation follows from the assumption that only one of v_t and ω_t can be different from 0. Noise is added to the translational and rotational velocities, as per the empirically obtained actuation noise models $p(n_v)$ and $p(n_\omega)$ from our test MAV platform, but can be specified as per the specific sensor or MAV platform used. Thus, $v_t^{[i]}$ and $\omega_t^{[i]}$ are obtained as

$$v_t^{[i]} = v_t + n_v^{[i]}, \quad n_v^{[i]} \text{ is drawn from } p(n_v), \quad (4)$$

$$\omega_t^{[i]} = \omega_t + n_\omega^{[i]}, \quad n_\omega^{[i]} \text{ is drawn from } p(n_\omega), \quad (5)$$

where v_t and ω_t are the nominal commands. In our simulations, according to Reference [18], $p(n_v)$ and $p(n_\omega)$ are specified as normal distributions with $\mu = 0$ and σ is expressed as a percentage of the value of v_t or ω_t .

3.2.2 Correction from Measurements. The correction step relies on the weights assigned to the particles (line 5 in Algorithm 1). Each MAV is equipped with a magnetometer sensor returning a measurement for its heading. Moreover, RF-signature snapshot provides another measurement. These two measurements are asynchronous in the sense that, while the on-board heading sensor can be queried after each command is executed, signature matching occurs only when revisiting a location associated with a known signature. In the following, we therefore separately describe, how the two different weights are computed, given that they are generated and used (via resampling) in separate stages.

The heading measurement is straightforward to integrate. According to former experimental measures [23], the nominal heading returned by the sensor is affected by Gaussian noise with a known variance σ^2 ($\sigma = 40$ degrees to be precise). Therefore, for the heading weight, we set

$$p(z_t | x_t) = f_{\mathcal{N}, \sigma^2}(z_t - c_{\phi_t}^{[i]}), \quad (6)$$

where $f_{\mathcal{N}, \sigma^2}$ is the density probability of a Gaussian with 0 mean and variance σ^2 , and the argument $z_t - c_{\phi_t}^{[i]}$ is normalized to account for the 2π period. To be noticed, this step only corrects orientation estimates of individual MAVs, since there is no way to get position reference for correction due to lack of localization infrastructure support. The correction on MAV positions happens at the snapshot points, where other MAVs' location estimates are used as position reference. After all weights have been computed, resampling can take place as described in Algorithm 1.

3.2.3 Adding Particles Using Coarse Map. To address the particle depletion problem in the estimation process, we adopted an additional step by adding particles with coarse map. After the new set X_t has been created, we determine the location with the highest number of particles. Let v_d be this location, and let N be the set of neighbor nodes according to the coarse map. Then, the 25 particles with the lowest weight are discarded and replaced by an equal number of particles generated using a random distribution over the space associated with the nodes in N . The rationale behind this step is to generate particles to recover errors due to the erroneous determination that a transition from a room to the next effectively took place.

3.3 Location Estimate Improvement at Snapshot Points through Collaboration

This subsection describes how to improve location estimate of MAVs at snapshot points. We first introduce how we adopt a set of particle filters to estimate the spatial location of the snapshot points. Then, we illustrate how to improve the location estimate of MAVs in a collaborative way. For the MAVs case, we do not compute a centralized estimation, but we rather associate a filter with each snapshot point to be tracked. This estimation process has two main differences with the position and orientation estimation for the MAVs. First, the number of snapshot point locations to be estimated is not known upfront. So, new filters need to be created on-the-fly when a new signature to be localized is identified. Second, snapshot points do not move. Therefore the estimation process does not include a prediction step, only a correction step. As for the MAVs, each filter includes 100 particles.

3.3.1 Initialization from MAV Particles. As described in the previous subsection, a new snapshot point is generated when the known signatures database receives a query from one of the MAVs with a RF-signature that cannot be matched to any of the formerly discovered ones. In this case, a new entry in the database is created and a new particle filter is instantiated. The initial set of particles for this new filter is copied from the particles of the MAV that discovered the feature, while discarding the component related to heading, because it is irrelevant for the signature estimation process.

3.3.2 Correction from MAV Particle Filter. The correction process for snapshot points is substantially different from that of individual MAVs. In this case, rather than computing $p(z_t|x_t)$, we determine $w_t^{[i]}$ through a two steps process. (1) When a signature is measured, the first step is to communicate with the known signatures database to determine whether the signature is new or has been encountered already (either by the same MAVs or a different one). If the database determines the signature is new, then the MAV does not perform the second step and does not compute weights (however, the signature is stored in the database and a new particle filter is initialized from the MAV as described above; (2) On the contrary, if the database detects a snapshot point, the second step starts. First, on the database side, the particle filter estimating the position of the snapshot point being revisited is updated. Algorithm 1 is executed for the snapshot point particle filter, with the exception of line 4, because no prediction takes place. The weight for $w_t^{[i]}$ for the i th particle is computed as follows. First, the position of the MAV that generated the snapshot point is determined by taking the average of its particles. Note that this average is implicitly weighted, because through the resampling process, particles with higher weight will be included more often in the particle set (see line 9 in Algorithm 1). As a result, they will be counted multiple times when computing the average. Let $\bar{x}$ be the computed average position of the MAV generating the match, and let $s_{t-1}^{[i]}$ be the position of the i th particle in the snapshot point particle filter at time $t - 1$, and let $d_t^{[i]} = \|\bar{x} - s_{t-1}^{[i]}\|_2$ be the Euclidean distance between the expected position of the MAV and the particle. The weight of each particle at time t is then defined as

$$w_t^{[i]} = F_{d_t^{[i]}, \delta}(K) - F_{d_t^{[i]}, \delta}(-K), \quad (7)$$

where $F_{d_t^{[i]}, \delta}$ is the cumulative density function of a Gaussian distribution with mean $d_t^{[i]}$ and variance δ . It is noticed that F is not intended to be the CDF of the Euclidean distance. Large distance between the expected position of the MAV and the particle causes small weight of that particle. This formula is based on our experimental testbed showing that revisits are correctly detected when the displacement between the original and the new position is within K meters. The specific values for δ and K depend on the number of anchors and are further described in Section 4.5.3.

After the snapshot point particle filter has been updated, each particle in the MAV particle filter is assigned a weight as follows. A Gaussian Mixture Model (GMM) is created starting from the particles in the signature being matched. Such GMM is a bidimensional probability density function with the following equation:

$$f_{GMM}(x, y) = \frac{1}{M} \sum_{i=1}^M f_{N, \Sigma}^i(x, y), \quad (8)$$

where $f_{N, \Sigma}^i$ is a bidimensional Gaussian distribution with mean $\mu = [s_x^{[i]} \ s_y^{[i]}]^T$ and covariance matrix Σ (a diagonal matrix with value 2 on the main diagonal). Then, each particle is assigned the weight

$$w_t^{[i]} \leftarrow f_{GMM}(c_{x_t}^{[i]}, c_{y_t}^{[i]}). \quad (9)$$

After all weights have been computed, resampling can take place as described in Algorithm 1.

3.3.3 Database of Signatures. To help location estimation correction with snapshot points after each movement, the system maintains a database of signatures. A signature at a specific location is a set of RSSI values from different stationary nodes measured by the MAV and stored in a dictionary data structure. When the node arrives at a new location, it calculates the cosine similarity between the newly discovered signature and the signatures stored in the database. A pre-defined threshold T_{sig} is used to decide whether it is a new or known signature. The role of T_{sig} is to decide the spatial resolution of the snapshot point location. Small T_{sig} leads to high resolution, but more stationary MAV nodes are needed to provide higher dimensional signature for support.

3.4 Adaptive Path Planning

To navigate the swarm to the destination area quickly and accurately, the system adaptively plans the paths of advanced MAVs and basic MAVs nodes, respectively. Figure 3 shows a flowchart of the adaptive path planning algorithm. After each location estimate, the MBC calculates AMS value based on multiple factors, which switch MAV nodes to two different modes: (1) navigation mode, which directs nodes to subsequent goal positions with map information and (2) exploration mode, which searches more snapshot points with random walk to improve location estimation accuracy.

3.4.1 Coarse Map. The system uses the layout with location of walls and doors of the environment to extract a coarse map. The doors are usually selected as the destination where we navigate the MAVs. The coarse map makes very few assumptions about the quality of the map but provides a way to bias the motion of MAV nodes toward designated locations.

3.4.2 Mode Bias Calculation (MBC) for Heterogeneous MAV Swarms. To achieve accurate and quick navigation for the entire swarm, MBC considers the different characteristics of advanced MAVs and basic MAVs, as well as the quality of their location estimate to guide the mode switching with its output AMS value. As shown in Figure 4, if the sensing accuracy of the MAV is high, i.e., advanced MAVs, and its nearby space is not fully investigated, then the AMS value will be set to minimum value to lead to exploration mode. In this case, advanced MAVs help the entire swarm to estimate location with its high on-board capabilities, which help basic MAVs improve their location estimate when they pass by these snapshot points. In other cases, MBC sets AMS value based on the quality of location estimate at time t as

$$AMS_t = \frac{H_{\max} - H_t}{H_{\max}}, \quad (10)$$

where entropy H_t represents the quality of the particle filter location estimation and $H_{\max}$ is the max possible value of H_t .

Fig. 3. The figure shows the flowchart of the adaptive path planning algorithm. The planner adaptively changes between random walk and graph biased movement based on the the AMS value. The AMS is calculated by MBC, which considers multiple factors.

Fig. 4. The figure shows the flowchart of the mode bias calculation algorithm. MBC considers the different characteristics of advanced MAVs and basic MAVs, as well as the quality of their location estimate to guide the mode switching with its output AMS value.

The entropy of a random variable x can be defined as the expected information that the value of x carries. In the discrete case, it is given by

$$H(x) = E[-\log_2 p(x)], \quad (11)$$

which represents the number of bits required to encode using an optimal encoding, assuming that $p(x)$ is the probability of observing x . The entropy can therefore be used as an indication of the uncertainty of the estimate of a particle filter. The lower the entropy the better the certainty of the location estimate is, and vice versa. For the particle filter, we calculate the entropy [20] of the weights at time t as

$$H_t = - \sum_{i=1}^M w_t^{[i]} \log_2 w_t^{[i]}. \quad (12)$$

High H_t means high uncertainty and low quality of location estimation, which leads to low AMS_t value according to Equation (10). The intuition here is that with an incorrect estimation of current location, using the bias from the graph is likely to be incorrect. Therefore, MAVs are biased to exploration mode to search for potential snapshot points to improve its location estimation. However, low H_t switches MAVs to navigation mode to approach subsequent goal locations.

3.4.3 Exploration Mode. When the AWS value is low, the system seeks to primarily improve the location estimates with advanced MAVs or basic MAVs. This happens when (1) the nearby space is not fully investigated or (2) the quality of location estimation is low. This also results in cases where the MAV may get stuck and can potentially perform worse than a purely random deployment strategy.

With this in mind, the planner employs a random walk strategy to direct the motion of MAV nodes. With random walk, the likelihood of nodes discovering snapshot points increases and so does the likelihood of improving their location estimates. This is referred to as the exploration step.

3.4.4 Navigation Mode. Correspondingly, when the *AWS* value is high, the uncertainty of location estimates is low. The planner commands the nodes to follow the bias indicated by the coarse map. With a more accurate location, likelihood of nodes following the bias and then reaching the intended destination increases.

A key point in choosing the directional bias from the graph is that it is sampled based on the distribution of particles in the node's particle filter. For example, consider a node with 20 particles indicating its position as room A and therefore requiring the node to go north-west to exit the room, while 80 particles indicate the node is in room 2 and must move south. In this case, the planner samples the movement direction according to the distribution of particles over the nodes of the graph, i.e., the node has a 20% chance of being commanded to move north-west and a 80% chance of receiving a south command.

3.4.5 Collision Recovery Strategy. MAV platforms have very limited sensing capability and often do not employ sophisticated obstacle detection sensors. Proposed MAV platforms [4] rely on their low weight and often use collisions themselves to discover obstacles. However, a strategy is needed in dealing with collision to prevent MAV nodes from being stuck and enable them to back off from corners and crevices and seek out openings. This is especially useful when location estimates are inaccurate. The planner employs a random exponential back-off strategy, where nodes move in randomly chosen direction (uniformly from a discrete number of directions) for a time duration that increases exponentially with the number of recent collisions. This is implemented by keeping a counter for collisions in a certain time-window. The counter is decremented with time if no new collisions are encountered.

4 EVALUATION

In this section, we evaluate the performance of our system in location estimation and path planning for MAVs through physical-feature-based simulation and real experiments on a MAV testbed. The testbed is used to validate the system in a realistic setting, while the physical-feature-based simulation ensures the large-scale evaluation. Both simulation and real experiments are conducted in a building with multiple rooms connected with a hallway as shown in Figure 5. The MAVs start from the entrance and are navigated to different goal areas (rooms). We first evaluate the system performance with the real testbed for homogeneous and heterogeneous MAV swarm. Then, we test the navigation success rate with physical-feature-based simulations. Finally, we investigate how key factors influence the system performance.

For both testbed experiments and simulations, we compare our *H-DrunkWalk* algorithm to another online navigation strategy that requires no location infrastructure and minimal hardware (especially sensing) setting. In addition, our system also adopts this rudimental algorithm for individual MAV location estimation as shown in Figure 1, which helps to check the improvement from our collaborative and adaptive strategy. We briefly describe it below:

Dead-Reckoning with Map Bias (DRMB): Dead-reckoning with Map Bias is an infrastructure-free technique used to estimate a node's location in unknown environments [24]. This method uses measurements from motion sensors, optical flow and gyroscope, to estimate the change in position of the node. Having an estimate of location, we then use the map to bias the direction of the node's movement similar to *H-DrunkWalk*.

4.1 Testbed Experiment Setup

To validate our system in the realistic environment, we implement our algorithm on a server (the Base) and the SensorFly MAV testbed [4, 25]. The SensorFly platform used in our test has an 8-bit 16 Mhz AVR AtMega128rfa1 micro-controller, a three-axis accelerometer, a three-axis gyroscope,

Fig. 5. The figure shows the floor plan of six rooms with a hallway used for physical-feature-based simulation and real testbed experiments. The MAVs start from the entrance of the building and are navigated to different goal areas: near, medium, and far.

Fig. 6. The figure shows the average and standard deviation of location estimation errors at different flying duration heading for the near destination using *H-DrunkWalk* and DRMB from five experiments.

a two-axis optical flow velocity sensor, a three-axis magnetometer, an ultrasonic altitude sensor and a XBee radio [26]. The platform has a flight time of 6–10 minutes. The SensorFly nodes are capable of translational and rotational motion directed by on-board PID control algorithms utilizing feedback from the on-board sensors.

In our experiment setup, to improve the repeatability and get accurate ground truth, we attach MAVs on sticks of same height with wheels at the bottom to move. At every operation period, MAVs receive commands from the base and navigate autonomously according to these commands. In such way, each MAV only moves at the X and Y direction and always keeps same height. After each movement, an infrared-based distance measurement is used to get ground truth location of each MAV. In such way, we autonomously navigate 8 MAVs in a controlled way on a 4×28 m arena shown in Figure 5. Six nodes are allowed to disperse and deploy as beacons at initialization, while 2 nodes fly to seek out the destinations. The nodes are introduced to the rooms at the entrance and navigated to 3 kinds of goal destinations: near destination (room 2), medium destination (right door of room 4) and far destination (room 5). In addition, a laser range finder is used to track the location of the nodes as ground truth.

4.2 Testbed Experiment Results

Figure 6 compares the performance and robustness of *H-DrunkWalk* and DRMB at different phases of navigation duration. We plot the average and standard deviation of location estimation errors from five experiments at different percentages of navigation duration. It is noted that we stop the experiments at 600 seconds even if the drones do not arrive at the destination, since this is the typical flying time of SensorFly node. At the first 20% of the navigation duration, *H-DrunkWalk* performs similarly as DRMB due to lack of snapshot points to correct location estimation errors. After this initial period, *H-DrunkWalk*'s snapshot point correction maintained location estimation errors within the range of 1.5 to 2 m. In contrast, the error of DRMB kept accumulating to larger than 3 m. This is because multiple measurements at snapshot points can correct the location estimation errors in *H-DrunkWalk*. In addition, after 30% of the navigation time, the standard deviation of *H-DrunkWalk* location estimation is also 30%–60% smaller than DRMB approach. This shows *H-DrunkWalk* is more reliable than DRMB.

Figure 7 shows cumulative distribution function (CDF) of location estimation errors using *H-DrunkWalk* and DRMB from a typical experiment. In *H-DrunkWalk*, corrections from snapshot points help keep errors within 2.5 m. In comparison, location estimation error of DRMB remains

Fig. 7. The figure shows the cumulative distribution function (CDF) of location estimation errors to arrive at near destination using both *H-DrunkWalk* and DRMB from a typical run. Our *H-DrunkWalk* show advantages over the baseline, which comes from collaborative location estimation.

Fig. 8. The figure shows the location estimation error over time using *H-DrunkWalk* and DRMB to the medium destinations. Mobile nodes with *H-DrunkWalk* arrive at the destination earlier than those with DRMB due to their capability to limit the location estimation errors.

Fig. 9. The figure shows the location estimation error over time using *H-DrunkWalk* and DRMB to the far destinations. It is noted that we do not plot all the data for mobile node 1, since it fails to arrive at the destination before the battery was exhausted (600 seconds).

Fig. 10. The figure shows the cumulative distribution function (CDF) of location estimation errors using DRMB and *H-DrunkWalk* with three different settings from a typical run.

unbounded. More than 50% of the time, DRMB has more than 3 m error, compared to less than 1 m error for *H-DrunkWalk*.

To illustrate the performance of *H-DrunkWalk* and DRMB for medium and far destinations, we use Figures 8 and 9 to show single run experiment results. We further evaluate them through simulated results in Section 4.3. Note that besides the node using DRMB in the upper plot in Figure 9, the lines end when nodes reach their destination. The mobile node 1 failed to arrive at the far destination before 600 seconds when the battery was exhausted.

For both medium and far destinations, in the first 20 seconds, similar to navigation to near destination, *H-DrunkWalk* has similar location estimation errors with DRMB. After this initial period, adequate snapshot points help *H-DrunkWalk* to limit the location estimation errors while the error of DRMB keeps accumulating. This shows that multiple measurements at snapshot points in *H-DrunkWalk* do help limit location estimation errors. The higher location estimation accuracy from *H-DrunkWalk* leads to shorter (around 50%) navigation duration. It should be also noted that when using DRMB, mobile node 1 fails to arrive at the far destination before the battery died (600 seconds) while mobile node 2 arrives at the destination in around 170 seconds. This shows the unreliability of DRMB. However, both mobile node 1 and mobile node 2 with *H-DrunkWalk*

arrive at the medium and far destination within similar durations. This shows the stability of *H-DrunkWalk* with the help of snapshot points.

To check the performance improvement from our *H-DrunkWalk*'s two key parts, collaborative location estimation and adaptive path planning guided by MBC, we plot the CDF of location estimation error from the DRMB (blue square line) and three different settings of our *H-DrunkWalk* in Figure 10. The green diamond line shows the CDF of *H-DrunkWalk* with all basic MAVs, which is equivalent to DrunkWalk approach in Reference [16]. The improvement from DRMB to *H-DrunkWalk* with all basic MAVs illustrate the performance gain brought by the collaborative location estimation and homogeneous adaptive planning, which has been analyzed in Figure 7. The yellow star line shows the CDF of *H-DrunkWalk* with heterogeneous swarm (basic MAV and advanced MAV), but without heterogeneous adaptive path planning algorithm guided by MBC. Difference from green diamond to yellow star line is the hardware configuration (mostly sensing accuracy). Finally, we implement *H-DrunkWalk*, including the heterogeneous adaptive path planning algorithm guided by MBC, with heterogeneous swarm, which is presented in the red circle line. The improvement from the yellow star line to the red circle line is $\sim 2\times$ (on average) as the improvement from green diamond line to yellow star line. This shows that our heterogeneous adaptive path planning algorithm brings $\sim 2\times$ performance improvement as that of hardware upgrade. This is because that MBC fully takes advantage of advanced capabilities of advanced MAV in the swarm to limit the location estimation error for the entire swarm. This is achieved by considering multiple factors: (1) sensing accuracy of MAVs; (2) whether nearby space is fully discovered or not; (3) location estimation quality of MAVs.

4.3 Physical-feature-based Simulation Environment

We implemented a MAV simulation environment [27] for the SensorFly MAV indoor sensor swarm to evaluate our planning algorithms at large scale. The simulator includes a realistic physical arena, as well as sensor noise models, MAV mobility models and indoor radio signature collected from the testbed described earlier. The remainder of the evaluation section is based on this simulation environment.

For our evaluations, we configure the simulator as follows:

- **Arena**—We use a multi-room indoor scenario shown in Figure 5, where nodes are required to autonomously navigate to different goal areas. We collect the radio fingerprint from the real arena and feed them into the simulation platform to evaluate our system. This represents a typical indoor apartment scenario where such systems may be deployed in search and rescue applications. For more complex maps, we concatenate on portions of the map in Figure 5.
- **Node Sensors**—The sensor nodes in the simulation are modeled after the SensorFly [4] MAV platform, which is also used in our testbed experiments described in Section 4.1. Each node has a XBee 802.15.4 radio and Dead-Reckoning sensors—a gyroscope, an optical flow velocity sensor and an ultrasonic altitude measurement sensor. Noise models are obtained through empirical measurements on the testbed MAV platform.
- **Node Mobility**—The MAV nodes can turn by a commanded angle and move for a commanded time and velocity. We set the velocity to 1.0 m/s in accordance with the testbed MAV parameters. The velocity of course varies in accordance with the noisiness of the optical flow sensor, that provides feedback to each MAV's control algorithm.
- **Simulation Time-steps**—The simulation time-step is chosen as 1sec that enables nodes to cover a distance of 0.8 to 1.2 m in one simulation tick.

Fig. 11. The figure shows the navigation success rate under different destination accuracy constraints within 90 seconds for *H-DrunkWalk* and DRMB.

Fig. 12. The figure shows the success rate as a function of time limitation under 0.5 m destination accuracy constraint, using *H-DrunkWalk* and DRMB alone.

- **Radio**—The simulation supports estimating received signal strength (RSS) measurements between two nodes. The RSS is collected in the real scenario.
- **Destination**—We adopt the far destination (room 6) as the destination for simulation, since this is the hardest situation, which shows the baseline of the system performance and robustness.

In the default setting, all experiments were performed 25 times with 10 MAVs (6 stationary nodes and 4 mobile nodes) to evaluate both the performance and robustness of the system. We set the sensing noise of advanced MAVs as 10% for optical flow noise and 10° for magnetometer noise. We set the sensing noise of basic MAVs twice as that of advanced MAVs, i.e., 40% optical flow noise and 40° magnetometer noise. We run the simulation for a time period of 600 seconds (~10 minutes) corresponding to the typical battery life of current generation MAV nodes.

4.4 System Performance (Navigation)

The experiments in this section evaluate the success rate of navigation for MAVs under different destination constraints and time limitations. A successful navigation is achieved when the MAV can be navigated to the destination within the given accuracy and time limitation. For example, if the destination coordinate is (4 m, 5 m), the required accuracy is 1 m and time limit is 90 seconds, a successful navigation means that the mobile node can arrive within the range of 1 m from (4 m, 5 m) within 90 seconds.

Figure 11 illustrates the success rate of navigation for MAVs given different destination accuracy constraints within 90 seconds using *H-DrunkWalk* and DRMB. When the destination accuracy is strict (0.5 m), *H-DrunkWalk* achieves an acceptable success rate of around 40% while DRMB shows less than 5%. This means that, under very strict destination accuracy constraint, DRMB cannot achieve the accuracy within the time limitation, while *H-DrunkWalk* can still work with the help of snapshot points. For less constrained destination accuracy (1 to 2 m), *H-DrunkWalk* shows consistently 30%–40% higher success rate than DRMB. Furthermore, *H-DrunkWalk* achieves 100% success rate for even looser destination accuracy constraint (2.5 m) while DRMB can also get 84% success rate. This is expected, since above 2.5 m range is more than one half of the hallway width (4.0 m), where even with high location estimation errors, the mobile node can still arrive at the destinations simply by following the walls.

Figure 12 shows the navigation success rate with different time limitations for both *H-DrunkWalk* and DRMB under a 0.5 m destination accuracy constraint. Under a strict time limitation constraint (60 seconds), even *H-DrunkWalk* only achieved 8% success rate, since it is not able to get

Fig. 13. The figure shows location estimation error as a function of optical flow noise using *H-DrunkWalk* estimation and DRMB alone. The noise per sensor is modeled as a normal distribution with varying standard deviation. The plot shows that *H-DrunkWalk* is able to correct the DRMB error and maintain low standard deviation.

Fig. 14. The figure shows location estimation error as a function of magnetometer noise using *H-DrunkWalk* estimation and dead reckoning alone. The noise per sensor is modeled as a normal distribution with varying standard deviation. The plot shows that *H-DrunkWalk* is able to correct the DRMB error and maintain low standard deviation.

enough snapshot points to get accurate location estimation on the way to the destination. When the time limitations are released (120–300 seconds), *H-DrunkWalk* shows 30%–50% higher success rate than DRMB. This is because *H-DrunkWalk* has enough time to get snapshot points to correct the location estimation errors, while the error of DRMB keeps accumulating. After 300 seconds, *H-DrunkWalk* achieves 100% success rate, while the success rate of DRMB becomes stable yet still below 80%. This means that even with loose time constraints, *H-DrunkWalk* still gets more than 20% higher success rate than DRMB.

4.5 Influence of System Settings

The experiments in this section investigate how different system settings affect the location estimation errors on both *H-DrunkWalk* and DRMB. We check the variance on (1) navigational sensor noise (magnetometer, optical flow, noise ratio between advanced MAVs and basic MAVs), (2) number of MAV nodes (total MAV number, stationary MAV number, advanced MAV number), (3) radio fingerprint accuracy.

4.5.1 Navigational Sensor Noise. The noise in motion measurements due to navigational sensors is an important parameter in determining the eventual performance of the algorithm. Different MAV platforms and operating environments might have different amount of noise in their motion measurements, making it useful to analyze the performance of the algorithm for varying levels of sensor noise. For our simulations, in agreement with empirical measurements on our MAV platform, we model noise as a normal distribution with a standard deviation proportional to the sensor measurement [18, 23].

For the optical flow velocity sensor, a noise corresponding to a normal distribution with 0 mean and standard deviation of 40% (basic MAV) and 20% (advanced MAV) of the measured velocity value is added. For the magnetometer, a noise corresponding to a normal distribution with 0° mean and standard deviation of 40° (advanced MAV) and 20° (advanced MAV) is added. The resultant noise in DRMB location can be computed as per the motion update equation [18, 23]. We apply sensor noise to both DRMB and *H-DrunkWalk* estimates.

Figures 13 and 14 show how the noise of two navigational sensors affects location estimation error of DRMB and *H-DrunkWalk*. In Figure 13, when optical flow noise level increases from 0 to

Fig. 15. The figure shows the localization estimation error as a function of noise ratio between *Advanced* MAVs sensors and *Basic* MAVs sensors, using *H-DrunkWalk* and DRMB.

Fig. 16. The figure shows the localization estimation error of MAV swarms consisting of different numbers of *Advanced* MAVs. The total MAV number in the swarm is fixed as 10.

50%, the average location estimation error of *H-DrunkWalk* increases from 2.5–5.5 m while that of DRMB goes up very quickly to around 25 m. The error of *H-DrunkWalk* increase is limited due to the corrections from the multiple measurements at the same snapshot points. In Figure 14, a faster increasing trend (from 0.5 to 7.5 m) is observed with the magnetometer noise increase. This is because the magnetometer helps correct the orientation estimation as described in Section 3.2.2 while the optical flow is only used for prediction and not involved in any correction as described in Section 3.2.1. The error from the optical flow can be corrected at snapshot points, while the error from the magnetometer directly leads to location estimation error. Therefore, sensors that help correction step has stronger effect on location estimation error and needs to be carefully selected and tuned.

The noise level ratio of navigational sensors between advanced MAVs and basic MAVs affects location estimation of the entire swarm as shown in Figure 15. In this experiment, the sensor noise of basic MAVs is fixed as default settings, while we change the sensor noise of advanced MAVs. The x-axis represents the ratio of two different kinds of noise. The errors of both *H-DrunkWalk* and DRMB decrease when the sensors of advanced MAVs become more accurate. Specifically, from 1 to 0.2, *H-DrunkWalk* decreases more than 50% (from 3.3 to 1.5 m), while DRMB decreases 33% (from 7.5 to 5 m). This is because that *H-DrunkWalk* not only collaboratively estimates location through snapshot points but also considers multiple factors (especially different on-board hardware configuration) to adaptively plan the path of MAVs, which helps better collaboration to improve location estimation accuracy.

4.5.2 Number of MAV Nodes. Since our heterogeneous MAV swarm consists of stationary MAV and mobile MAV (advanced MAVs and basic MAVs), we check three numbers that affects the location estimation performance: the number of advanced MAVs, the number of stationary MAV nodes, and total number of the mobile MAVs.

To check influence of the number of advanced MAVs on system performance, we plot the location estimation error in the Figure 16. In this experiment, we fix the total number of mobile MAVs as 10. When the number of advanced MAVs increases, the number of basic MAVs decreases. First, the errors of both *H-DrunkWalk* and DRMB present descending trends when the number of advanced MAVs increases. The error reduction of *H-DrunkWalk* comes from both sensing accuracy improvement and collaboration, while the error reduction of DRMB only comes from sensing accuracy improvement. Second, the error of *H-DrunkWalk* decreases faster than that of DRMB. Specifically, from 0 advanced MAVs to 10 advanced MAVs, the error of *H-DrunkWalk* reduces 60% (from 3.75 to 1.5 m), while the error of DRMB reduces less than 20% (from 7.2 to 6.0 m).

Fig. 17. The figure shows the localization estimation error as a function of total number of MAV swarm. The noise levels for *advanced MAVs* and *baisc MAVs* are set as default and the portion of *advanced MAVs* in the swarm is set as 30%.

5.8 m). This illustrates that the MBC in *H-DrunkWalk* takes advantage of the high capabilities of advanced MAVs to help the entire swarm to reduce location estimation error. Finally, from 0 advanced MAVs to 10 advanced MAVs, the standard deviation of *H-DrunkWalk* errors decreases more than 40% (1.7 to 1 m), while DRMB does not show too much difference on standard deviation of errors. This shows the reliability of our *H-DrunkWalk* with heterogeneous MAV swarm. Both the error reduction and reliability improvement come from *H-DrunkWalk*'s ability to utilize advanced MAVs to reduce and limit location estimation error of basic MAVs.

We check how total number of MAVs in a swarm affect location estimation in Figure 17. The noise levels for advanced MAVs and basic MAVs are set as default and the portion of advanced MAVs in the swarm is set as 30%. We increase the total number of mobile MAVs from 10 to 100. With the total number of MAVs increases, average and standard deviation of error for DRMB does not show too much difference. In contrast, from 10 to 30, the average error for *H-DrunkWalk* decreases 30% (from 2.3 to 1.6 m) and the error of standard deviation for *H-DrunkWalk* decrease 15% (from 1.36 to 1.17 m). This is because that DRMB does not adopt collaboration between MAVs. As a result, increasing total number of the MAV swarm does not help reduce estimation error or improve stability. In contrast, our *H-DrunkWalk* not only adopts collaboration between MAVs but also assigning different roles to advanced MAVs and basic MAVs under different conditions, which helps to improve the system performance on accuracy and reliability. In addition, for *H-DrunkWalk*, the average and standard deviation of error do not change to much when total number of MAVs increase from 35 to 100. This is because that for the given arena shown in Figure 5, 30 mobile MAVs are enough to search collaboratively to limit the location estimation error.

Figure 18 shows the location estimation errors with different numbers of stationary MAV nodes using *H-DrunkWalk*. Since DRMB does not operate with stationary MAVs, we do not plot its location estimation error here. We can see obvious decreasing trend for both average and standard deviation in *H-DrunkWalk* when the number of stationary nodes increases. Specifically, 3.2 $\times$ reduction on average error and 2.8 $\times$ reduction on standard deviation. This shows that increasing the number of stationary nodes do help enhance performance.

4.5.3 Radio Fingerprint Accuracy. Location estimation error depends on the resolution to identify snapshot on the node paths. This is accomplished by using radio signature from stationary MAV nodes deployed at the beginning. When a radio signature collected by a node at a certain location is similar to a fingerprint in the database, the system classifies it as snapshot and performs the correction of location estimates. Thus, the performance of *H-DrunkWalk* depends on the resolution of the fingerprints matching, i.e., the area or distance within which two radio fingerprints

Fig. 18. The figure shows the location estimation error with varying number of stationary MAV nodes using *H-DrunkWalk*, which shows a decreasing trend when the number of stationary MAV nodes increases.

Fig. 19. The figure shows location estimation error as a function of signature matching area for *H-DrunkWalk*, which shows an increasing trend when signature matching area increases.

can be reliably classified as being at the same location. If the criteria is too loose, then incorrect points may be matched together, which leads to few snapshot points.

Figure 19 shows location estimation error as a function of signature matching area for *H-DrunkWalk* for 10 nodes. We do not plot the location estimation error of DRMB, since its operation does not require signature matching, which relies on stationary MAVs. We observe location estimation error improvement when signature matching spatial resolution increases ($\sim 2\times$ improvements from 6 m^2 to 1 m^2). In the case of MAV navigation, where the low-cost of nodes makes it possible to deploy a relatively large number of beacon nodes and attain high signature accuracy of around 1 m^2 , *H-DrunkWalk* provides a much larger reduction in error.

5 DISCUSSION

This section discusses several aspects of *H-DrunkWalk* algorithm: decentralized structure, communication, and database. Finally, we point out the potential new research extended from *H-DrunkWalk*.

Our *H-DrunkWalk* adopts a centralized framework, in which most complicated computation happens at the base. This is feasible in our indoor scenario, where MAVs cover small areas and are well connected. For large areas, decentralized framework will improve the efficiency but requires better computational resources on each individual MAV. An easy way is to adopt the decentralized framework from other works [28]. A possible solution is to offload the computing task to (1) the nearby stationary MAVs, in which broadcasting beacons does not consume too much computing resources and (2) advanced MAVs who are equipped with high computing capabilities. In such a way, the system improves the efficiency and still maintains a low cost of the entire swarm.

In our system, message overhead is composed of three parts: (1) Stationary MAVs broadcasting messages to flying MAVs for RSSI measurements takes 1 bytes for each message. Each stationary MAVs broadcast messages 10 times per second. (2) Flying MAVs report the RSSI values, heading information and some other status information to the base every second. The corresponding message size is 20 bytes for each flying MAV. (3) The base sends different command messages to different flying MAVs every second. The overhead of the command message takes 14 bytes. In real systems, due to collisions, the real overhead is often higher than the above value. Concerning the latency issue, the base will skip sending the command to the flying MAV if no message or delayed message is received from that MAV. The flying MAV keeps hovering to wait for another round command message if they get no message or delayed message from the base. This causes

efficiency due to latency issues with a centralized framework. In the future, we will design a decentralized framework as described in the previous paragraph.

Our *H-DrunkWalk* requires to scan the entire database to match the signature to decide whether the current location is a new discovered area or a covered area (a snapshot point). The size of the database grows over time until all the areas are investigated. Assume the size operation environment is S_o and the signature matching spatial resolution is r_s , we have maximum $\frac{S_o}{r_s}$ signatures in the database. If the system lands N_s stationary MAVs to work as beacons, then the maximum bytes needed to store signatures is $\frac{S_o}{r_s} * N_s$, since we utilize one byte to represent the RSSI values from one stationary MAV. It is very difficult to achieve very small r_s due to the indoor multi-path propagation effect. In addition, the number of stationary MAVs is not very large due to cost consideration. Therefore, as long as the size of the operation environment S_o is not extremely large, the database does not need too much space.

In the future, several potential research directions can be explored. First, a decentralized framework is important for large-scale heterogeneous MAV swarms, in which most complicated computation is completed by MAVs equipped with high computing capabilities. Since MAVs move frequently with high speed and most computation is time sensitive, it is challenging for basic MAVs to decide which advanced MAVs to offload the computing tasks [29, 30]. Second, the current *H-DrunkWalk* framework relies on radio to do localization. The framework can be extended to vision-based SLAM. For example, advanced MAVs can be equipped with cameras with high computing capabilities, while basic MAVs can keep the current setting up. In addition, for a large area, a large number of MAVs is required. Third, actuation mechanism can be considered to improve the sensing coverage quality for better environmental understanding [31–33]. Finally, the layout geometry of the operating environment also affects the localization and navigation performance. It is also interesting to explore how to utilize the prior knowledge of layout geometry to help improve the localization and navigation performance.

6 RELATED WORK

Works related to *H-DrunkWalk* mainly fall into three domains: sensor networks, robotics and mobile computing. We investigate these three domains in the following paragraphs of this section, respectively.

The sensor network domain has a number of works on deploying and navigating mobile sensors [34–37]. Howard et al. [38, 39] present techniques for mobile sensor network deployment in an unknown environment. Their approach constructs fields such that each node is repelled by both obstacles and by other nodes, enabling the network to spread itself throughout the environment. Similarly, Batalin et al. [40] present a deployment algorithm for robot teams without access to maps or location. The robots are assumed to be equipped with vision sensors and range finders and select a direction away from all their immediate sensed neighbors and move in that direction. The algorithm does not require communication between nodes but also does not allow nodes to be deployed at designated locations. The domain experts have no control over the emergent deployment locations of the nodes.

The problem addressed in this article can also be seen as an instance of the Simultaneous Localization and Mapping (SLAM) problem that has been extensively studied in the area of robotics [20, 41–43]. In fact, in the system, we described multiple MAVs try to localize themselves while at the same time trying to acquire a representation of the spatial distribution of the radio signatures. In recent years there has been copious research in SLAM using either method based on Kalman filters [44–47] or particle filters [48–51]. Both approaches, however, have been mostly applied to solve an instance of the SLAM problem where mobile agents are equipped with sensors returning

distances (e.g., laser range finders or sonars) or cameras (either monocular or stereo). Therefore, the ultimate objective of these solutions to the SLAM problem was to map physical entities located in the environment, like walls, obstacles, and so on. Methods based on Kalman filters are not applicable for the scenario we consider, because we are dealing with multimodal, nonparametric probability distributions. Robust navigation methods for MAV swarms that work in GPS denied areas are recently introduced [8, 9]. They can achieve high navigation accuracy and do not rely on any localization infrastructure. However, these simultaneous localization and mapping (SLAM) methods require advanced sensors like depth camera and LIDAR, which are expensive, large and heavy [10, 11].

Approaches based on explicit perception and processing of radio signals have been mostly aimed at implementing localization systems with the underlying assumption that radio signals were preliminarily collected off line to build so-called *map signals* [52–55]. A recent paper by Twigg et al. [56] discusses a system where a robot autonomously discovers the area within which connectivity with an assigned WiFi base station is ensured. Their solution, however, solves only the mapping side of the problem, because the robot is equipped with a laser range finder solving the localization problem. In other words, RSS readings are mapped to the physical space exploiting the availability of a different sensor providing reliable localization.

For people carrying mobile devices, SLAM-like approaches have recently been proposed that fuse WiFi-based RSS and motion sensor data to simultaneously build a sensor map of the environment and locate the user within this map. e.g., radio fingerprint maps [57–60], or organic landmark maps [61, 62]. These approaches focus on the location estimation part of an orthogonal problem, where the motion of users cannot be controlled and hence, does not involve motion planning or deployment. Purohit et al. [18] present a system for infrastructure-free single room *sweep* coverage with MAV sensor swarms. Their approach, however, does not involve the concept of location estimation and navigation and does not support navigating nodes to pre-assigned destinations.

To be noticed, this article is a more extensive elaboration of our *DrunkWalk* paper in ACM SenSys 2015 [16]. To the best of our knowledge, the *DrunkWalk* paper presented the first attempt to solve a SLAM problem using a swarm of MAVs that combines location estimation and adaptively planning to improve the success rate and accuracy of navigation, as shown in the first and second contribution in Section 1. The technique in the *DrunkWalk* paper made an assumption that all MAVs are equipped with same hardware and can be only applied on homogeneous MAV swarms, where all MAVs are equipped with same hardware. This article extends the technique so that it can be applied to heterogeneous MAV swarms, where MAVs are equipped with different hardware. As shown in the third contribution in Section 1, the elaboration is achieved by the new designed MBC algorithm, which utilizes advanced MAVs to help basic MAVs improve their performance by intelligently switching the mode of both types of MAVs under different conditions. To achieve accurate and quick navigation for the entire swarm, MBC considers the different characteristics of advanced MAVs and basic MAVs, as well as the quality of their location estimate to guide the mode switching, whose details can be found in Section 3.4.2.

7 CONCLUSION

This article presents *H-DrunkWalk*, a collaborative and adaptive system for heterogeneous MAV swarm navigation in environments not formerly preconditioned for operation in a quick and efficient way. The heterogeneity brings the possibility to achieve high accuracy through collaboration but still maintains a low cost of the entire swarm. The system relies on collaboration to overcome limitations of individual nodes and efficiently achieve system-wide sensing objectives and fully takes advantage of advanced MAVs to help basic MAVs improve their performance.

We comprehensively evaluate the system through large-scale simulations and real MAV testbed experiments. The results show that our system achieves up to $6\times$ reductions in location estimation errors and as much as $3\times$ improvements in navigation success rate under the given time and accuracy constraints. In addition, by comprehensively considering the environment, heterogeneous structure, and quality of location estimation, our *H-DrunkWalk* brings $2\times$ performance improvements (on average) as that of hardware upgrade.

REFERENCES

- [1] Xinlei Chen, Xiangxiang Xu, Xinyu Liu, Shijia Pan, Jiayou He, Hae Young Noh, Lin Zhang, and Pei Zhang. 2018. PGA: Physics guided and adaptive approach for mobile fine-grained air pollution estimation. In *Proceedings of the ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*. ACM, 1321–1330.
- [2] Xinyu Liu, Xinlei Chen, Xiangxiang Xu, Enhao Mai, Hae Young Noh, Pei Zhang, and Lin Zhang. 2017. Delay effect in mobile sensing system for urban air pollution monitoring. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. ACM, 73.
- [3] Xinlei Chen, Xiangxiang Xu, Xinyu Liu, Hae Young Noh, Lin Zhang, and Pei Zhang. 2016. Hap: Fine-grained dynamic air pollution map reconstruction by hybrid adaptive particle filter. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*. ACM, 336–337.
- [4] A. Purohit, Zheng Sun, F. Mokaya, and Pei Zhang. 2011. SensorFly: Controlled-mobile sensing platform for indoor emergency response applications. In *Proceeding of the 10th International Conference on Information Processing in Sensor Networks (IPSN'11)*. 223–234.
- [5] Robert J. Wood. 2008. The first takeoff of a biologically inspired at-scale robotic insect. *IEEE Trans. Robot.* 24, 2 (2008), 341–347. Retrieved from <http://cat.inist.fr/?aModele=afficheN&cpsidt=20276799>.
- [6] Karthik Dantu, Bryan Kate, Jason Waterman, Peter Bailis, and Matt Welsh. 2011. Programming micro-aerial vehicle swarms with Karma. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems (SenSys'11)*. ACM, New York, NY, 121–134. DOI : <http://dx.doi.org/10.1145/2070942.2070956>
- [7] Matthew Turpin, Nathan Michael, and Vijay Kumar. 2012. Trajectory design and control for aggressive formation flight with quadrotors. *Auton. Robots* 33, 1–2 (Feb. 2012), 143–156. DOI : <http://dx.doi.org/10.1007/s10514-012-9279-y>
- [8] Francisco J. Perez-Grau, Ricardo Ragel, Fernando Caballero, Antidio Viguria, and Anibal Ollero. 2018. An architecture for robust UAV navigation in GPS-denied areas. *J. Field Robot.* 35, 1 (2018), 121–145.
- [9] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D. Tardos. 2015. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robot.* 31, 5 (2015), 1147–1163.
- [10] Marco Dorigo, Dario Floreano, Luca Maria Gambardella, Francesco Mondada, Stefano Nolfi, Tarek Baaboura, Mauro Birattari, Michael Bonani, Manuele Brambilla, Arne Brutschy, et al. 2013. Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. *IEEE Robot. Autom. Mag.* 20, 4 (2013), 60–71.
- [11] Kartik Mohita, Michael Watterson, Yash Mulgaonkar, Sikang Liu, Chao Qu, Anurag Makineni, Kelsey Saulnier, Ke Sun, Alex Zhu, Jeffrey Delmerico, et al. 2018. Fast, autonomous flight in GPS-denied and cluttered environments. *J. Field Robot.* 35, 1 (2018), 101–120.
- [12] Carlos Ruiz, Xinlei Chen, Lin Zhang, and Pei Zhang. 2016. Collaborative localization and navigation in heterogeneous UAV swarms: Demo abstract. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*. ACM, 324–325.
- [13] Carlos Ruiz, Shijia Pan, Adeola Bannis, Xinlei Chen, Carlee Joe-Wong, Hae Young Noh, and Pei Zhang. 2018. IDrone: Robust drone identification through motion actuation feedback. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 2 (2018), 80.
- [14] Carlos Ruiz, Xinlei Chen, and Pei Zhang. 2017. Hybrid and adaptive drone identification through motion actuation and vision feature matching. In *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'17)*. IEEE, 327–328.
- [15] Xinlei Chen, Aavek Purohit, Shijia Pan, Carlos Ruiz, Jun Han, Zheng Sun, Frank Mokaya, Patric Tague, and Pei Zhang. 2017. Design experiences in minimalistic flying sensor node platform through sensorfly. *ACM Trans. Sensor Netw.* 13, 4 (2017), 33.
- [16] Xinlei Chen, Aavek Purohit, Carlos Ruiz Dominguez, Stefano Carpin, and Pei Zhang. 2015. Drunkwalk: Collaborative and adaptive planning for navigation of micro-aerial sensor swarms. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, 295–308.
- [17] Ryan Morlok and Maria Gini. 2007. Dispersing robots in an unknown environment. In *Distributed Autonomous Robotic Systems* 6. Springer, 253–262.

- [18] Aveek Purohit, Zheng Sun, and Pei Zhang. 2013. SugarMap: Location-less coverage for micro-aerial sensing swarms. In *Proceedings of the 12th International Conference on Information Processing in Sensor Networks (IPSN'13)*. ACM, New York, NY, 253–264. DOI: <http://dx.doi.org/10.1145/2461381.2461412>
- [19] I. Constandache, R. R. Choudhury, and I. Rhee. 2010. Toward mobile phone localization without war-driving. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'10)*. 1–9. DOI: <http://dx.doi.org/10.1109/INFCOM.2010.5462058>
- [20] S. Thrun, W. Burgard, and D. Fox. 2006. *Probabilistic Robotics*. MIT Press, 77–90.
- [21] A. Doucet, J. F. G. de Freitas, and N. J. Gordon (Eds.). 2001. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag.
- [22] M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. 2002. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Trans. Signal Process.* 50, 2 (2002), 174–188.
- [23] Zheng Sun, Aveek Purohit, Shijia Pan, Frank Mokaya, Raja Bose, and Pei Zhang. 2012. Polaris: Getting accurate indoor orientations for mobile devices using ubiquitous visual patterns on ceilings. In *Proceedings of the 12th Workshop on Mobile Computing Systems & Applications (HotMobile'12)*. ACM, New York, NY. DOI: <http://dx.doi.org/10.1145/2162081.2162101>
- [24] Johann Borenstein, Liqiang Feng, and H. R. Everett. 1996. *Navigating Mobile Robots: Systems and Techniques*. A. K. Peters, Ltd., Natick, MA.
- [25] Aveek Purohit, Pei Zhang, Brian M. Sadler, and Stefano Carpin. 2014. Deployment of swarms of micro-aerial vehicles: From theory to practice. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'14)*. IEEE, 5408–5413.
- [26] Digi International. 2009. XBee Multipoint RF Modules Datasheet, pp. 1–2. Retrieved from www.digi.com.
- [27] A. Purohit and Pei Zhang. 2011. Controlled-mobile sensing simulator for indoor fire monitoring. In *Proceedings of the 7th International Wireless Communications and Mobile Computing Conference (IWCMC'11)*. 1124–1129.
- [28] Boldizsár Balázs and Gábor Vásárhelyi. 2018. Coordinated dense aerial traffic with self-driving drones. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'18)*. IEEE, 6365–6372.
- [29] Xinlei Chen, Yulei Zhao, Yong Li, Xu Chen, Ning Ge, and Sheng Chen. 2018. Social trust aided D2D communications: Performance bound and implementation mechanism. *IEEE J. Select. Areas Commun.* 36, 7 (2018), 1593–1608.
- [30] Yu Liu, Xinlei Chen, Yong Niu, Bo Ai, Yong Li, and Depeng Jin. 2018. Mobility-aware transmission scheduling scheme for millimeter-wave cells. *IEEE Trans. Wireless Commun.* 17, 9 (2018), 5991–6004.
- [31] Susu Xu, Xinlei Chen, Xidong Pi, Carlee Joe-Wong, Pei Zhang, and Hae Young Noh. 2019. iLOCuS: Incentivizing vehicle mobility to optimize sensing distribution in crowd sensing. *IEEE Trans. Mobile Comput.* (2019). DOI: [10.1109/TMC.2019.2915838](https://doi.org/10.1109/TMC.2019.2915838)
- [32] Xinlei Chen, Susu Xu, Haohao Fu, Carlee Joe-Wong, Lin Zhang, Hae Young Noh, and Pei Zhang. 2019. ASC: Actuation system for city-wide crowdsensing with ride-sharing vehicular platform. In *Proceedings of the 4th Workshop on International Science of Smart City Operations and Platforms Engineering*. ACM, 19–24.
- [33] Susu Xu, Xinlei Chen, Xidong Pi, Carlee Joe-Wong, Pei Zhang, and Hae Young Noh. 2019. Vehicle dispatching for sensing coverage optimization in mobile crowdsensing systems. In *Proceedings of the 18th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'19)*. IEEE, 311–312.
- [34] Maxim A. Batalin and Gaurav S. Sukhatme. 2003. Coverage, exploration, and deployment by a mobile robot and communication network. In *Information Processing in Sensor Networks*. Springer, 376–391.
- [35] Maxim A. Batalin and Gaurav S. Sukhatme. 2004. Coverage, exploration and deployment by a mobile robot and communication network. *Telecommun. Syst.* 26, 2–4 (2004), 181–196.
- [36] Ting Liu, Christopher M. Sadler, Pei Zhang, and Margaret Martonosi. 2004. Implementing software on resource-constrained mobile sensors: Experiences with Impala and ZebraNet. In *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services*. ACM, 256–269.
- [37] Yong Wang, Pei Zhang, Ting Liu, Chris Sadler, and Margaret Martonosi. 2007. Movement data traces from princeton zebranet deployments. *CRAWDAD Database* 129 (2007).
- [38] Andrew Howard, Maja J. Matarić, and Gaurav S. Sukhatme. 2002. An incremental self-deployment algorithm for mobile sensor networks. *Auton. Robots* 13, 2 (Sept. 2002), 113–126. DOI: <http://dx.doi.org/10.1023/A:1019625207705>
- [39] Andrew Howard, Maja J. Matarić, and Gaurav S. Sukhatme. 2002. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Distributed Autonomous Robotic Systems 5*. Springer, 299–308.
- [40] Maxim A. Batalin and Gaurav S. Sukhatme. 2002. Spreading out: A local approach to multi-robot coverage. In *Distributed Autonomous Robotic Systems 5*. Springer, 373–382.
- [41] Wolfram Burgard, Cyrill Stachniss, Giorgio Grisetti, Bastian Steder, Rainer Kummerle, Christian Dornhege, Michael Ruhnke, Alexander Kleiner, and Juan D. Tardós. 2009. A comparison of SLAM algorithms based on a graph of relations. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'09)*. IEEE, 2089–2095.

- [42] Denis Chekhlov, Mark Pupilli, Walterio Mayol-Cuevas, and Andrew Calway. 2006. Real-time and robust monocular SLAM using predictive multi-resolution descriptors. In *Advances in Visual Computing*. Springer, 276–285.
- [43] Austin I. Eliazar and Ronald Parr. 2004. DP-SLAM 2.0. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'04)*, Vol. 2. IEEE, 1314–1320.
- [44] Tim Bailey, Juan Nieto, Jose Guivant, Michael Stevens, and Eduardo Nebot. 2006. Consistency of the EKF-SLAM algorithm. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 3562–3568.
- [45] G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. 2001. A solution to the simultaneous localisation and map building (SLAM) problem. *IEEE Trans. Robot. Autom.* 17, 3 (2001), 229–241.
- [46] Guoquan P. Huang, Anastasios I. Mourikis, and Stergios I. Roumeliotis. 2008. Analysis and improvement of the consistency of extended Kalman filter-based SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'08)*. IEEE, 473–479.
- [47] Shoudong Huang and Gamini Dissanayake. 2007. Convergence and consistency analysis for extended Kalman filter-based SLAM. *IEEE Trans. Robot.* 23, 5 (2007), 1036–1049.
- [48] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. 2005. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'05)*. IEEE, 2432–2437.
- [49] G. Grisetti, C. Stachniss, and W. Burgard. 2007. Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Trans. Robot.* 23, 1 (2007), 36–46.
- [50] Giorgio Grisetti, Gian Diego Tipaldi, Cyrill Stachniss, Wolfram Burgard, and Daniele Nardi. 2007. Fast and accurate SLAM with Rao-Blackwellized particle filters. *Robot. Auton. Syst.* 55, 1 (2007), 30–38.
- [51] Andrew Howard. 2006. Multi-robot simultaneous localization and mapping using particle filters. *Int. J. Robot. Res.* 25, 12 (2006), 1243–1256.
- [52] P. Bahl and V. Padmanabhan. 2000. RADAR: An in-building RF-based user location and tracking system. In *Proceedings of the IEEE International Conference on Computer Communications*. 775–784.
- [53] A. Ladd, K. Bekris, A. Rudys, D. Wallach, and L. Kavraki. 2004. On the feasibility of using wireless Ethernet for indoor localization. *IEEE Trans. Robot. Autom.* 20, 3 (2004), 555–559.
- [54] Moustafa Youssef and Ashok Agrawala. 2005. The Horus WLAN location determination system. In *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*. ACM, 205–218.
- [55] S. Zickler and M. Veloso. 2010. RSS-based relative localization and tethering for moving robots in unknown environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*. 5466–5471.
- [56] Jeffrey N. Twigg, Jonathan R. Fink, Paul L. Yu, and Brian M. Sadler. 2013. Efficient base station connectivity area discovery. *Int. J. Robot. Res.* 32, 12 (2013), 1398–1410.
- [57] Brian Ferris, Dieter Fox, and Neil Lawrence. 2007. WiFi-SLAM using gaussian process latent variable models. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 2480–2485. Retrieved from <http://dl.acm.org/citation.cfm?id=1625275.1625675>.
- [58] Mikkel Baun Kjærgaard. 2007. A taxonomy for radio location fingerprinting. In *Proceedings of the International Symposium on Location-and Context-awareness*. Springer, 139–156.
- [59] Guobin Shen, Zhuo Chen, Peichao Zhang, Thomas Moscibroda, and Yongguang Zhang. 2013. Walkie-Markie: Indoor pathway mapping made easy. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation (NSDI'13)*. USENIX Association, Berkeley, CA, 85–98. Retrieved from <http://dl.acm.org/citation.cfm?id=2482626.2482636>.
- [60] Jie Yin, Qiang Yang, and Lionel M. Ni. 2008. Learning adaptive temporal radio maps for signal-strength-based location estimation. *IEEE Trans. Mobile Comput.* 7, 7 (2008), 869–883.
- [61] He Wang, Souvik Sen, Ahmed Elgohary, Moustafa Farid, Moustafa Youssef, and Romit Roy Choudhury. 2012. No need to war-drive: Unsupervised indoor localization. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys'12)*. ACM, New York, NY, 197–210. DOI: <http://dx.doi.org/10.1145/2307636.2307655>
- [62] Aveek Purohit, Zheng Sun, Shijia Pan, and Pei Zhang. 2013. SugarTrail: Indoor navigation in retail environments without surveys and maps. In *Proceedings of the 10th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'13)*. IEEE, 300–308.

Received March 2018; revised January 2020; accepted January 2020